Statistics Under the BST Model

Tesi doctoral presentada al
Departament de Llenguatges i Sistemes Informàtics
de la Universitat Politècnica de Catalunya

per a optar al grau de
Doctor en Informàtica

per
Conrado Martínez Parra

sota la direcció del doctor
Rafael Casas Muñoz

Barcelona, 30 de Març de 1992

Aquesta tesi fou llegida el dia 24 d'Abril de 1992, davant el tribunal de tesi format per

– Dr. Josep Díaz	(President)
– Dr. Josep Grané	(Secretari)
– Dr. Philippe Flajolet	
– Dr. Carles Simó	
– Dr. Jean-Marc Steyaert	

Resumen

El presente trabajo está dedicado al análisis de algoritmos asumiendo que sus entradas están distribuídas según el modelo probabilístico BST (así denominado por su vinculación con los árboles binarios de búsqueda, *Binary Search Trees* en inglés) o según el modelo equilibrado, que generaliza al anterior. El conjunto de entradas de tales algoritmos es siempre alguna familia de árboles simplemente generados.

En ambos modelos, dos objetos (árboles, k-tuplas de árboles) del mismo tamaño no son necesariamente equiprobables, contrariamente a lo que se asume en la mayoría de análisis probabilísticos relativos a sistemas de manipulación simbólica, demostración automática de teoremas o compilación, donde la hipótesis de uniformidad es válida en gran número de situaciones.

Algunos ejemplos de familias de árboles que no están uniformemente distribuídos son los citados árboles binarios de búsqueda, los árboles k-d, los árboles m-arios de búsqueda, los *tries*, los *Patricia trees*, los *quadtrees*, etc.

El análisis probabilístico de algoritmos mediante el modelo BST y el modelo equilibrado conduce, por lo general, a ecuaciones diferenciales ordinarias y en derivadas parciales, en el camino hacia la solución; en cambio, en el análisis de algoritmos para árboles uniformemente distribuídos, hay que resolver problemas algebraicos y problemas de enumeración combinatoria.

La tesis se divide en cuatro partes. Las dos primeras contienen el material introductorio, conceptos básicos y definiciones, así como una breve descripción de las herramientas matemáticas empleadas: el método del operador simbólico y técnicas procedentes del análisis de variable compleja. En la parte II presentamos una colección de modelos de probabilidad para familias de árboles, describiendo sus contextos de aplicación natural, algunos resultados previos y la metodología a seguir en los correspondientes análisis de caso medio.

La parte III constituye la principal contribución de este trabajo. Comienza con un capítulo en el que se introduce la ocupación, una sencilla característica relacionada con el grado de completitud o equilibrado de los árboles. Analizamos dicha característica bajo diferentes hipótesis concernientes al modelo de probabilidad, con el fin de poder comparar los principales elementos distintivos de cada modelo y el tipo de problemas matemáticos asociados a cada uno de ellos.

En el capítulo 5 abordamos la definición de un conjunto de reglas que permitirán

obtener directamente las ecuaciones matemáticas que describen el comportamiento de un algoritmo a partir de la propia estructura del algoritmo, para una cierta clase de algoritmos.

El capítulo 6 se dedica al examen de una colección de algoritmos elementales y de predicados sobre pares de árboles, interesantes y sencillos, y que denominamos propiedades hereditarias. Dichos predicados admiten caracterizaciones recursivas muy similares. La formulación matemática del problema de evaluación de la probabilidad que una propiedad hereditaria se cumpla para un par de árboles de talla n, es estructuralmente idéntica para todas las propiedades hereditarias. Por otra parte, el comportamiento medio de los algoritmos que verifican si un par de árboles dado cumple una propiedad hereditaria, se puede describir mediante ecuaciones muy parecidas. El test de igualdad entre dos árboles dados es uno de tales algoritmos y recibe una atención mayor en un capítulo posterior.

Los dos siguientes capítulos contienen los análisis del tamaño medio de la intersección de un par de árboles y de la complejidad media del test de igualdad para pares de árboles binarios. En ambos análisis, el problema original se transforma en uno de resolución de una ecuación diferencial en derivadas parciales. El uso del método de Riemann, y la aplicación de técnicas de análisis asintótico proporciona los resultados deseados. Así, el tamaño medio de la intersección de un par de árboles binarios de tamaño n es $O(n^{2\sqrt{2}-2}/\sqrt{\log n})$ y el test de igualdad para pares de árboles binarios consume $O(\log n)$ pasos en promedio. Hacemos también un detallado análisis de la probabilidad que dos árboles, de tamaño n cada uno, sean iguales. Todos los análisis mencionados requieren el uso de resultados matemáticos específicos, de los cuales se hace un repaso en los preliminares.

Resum

El present treball està dedicat a l'anàlisi d'algorismes quan s'assumeix que les entrades están distribuïdes segons el model de probabilitat BST (així anomenat per la seva vinculació amb els arbres binaris de cerca, *Binary Search Trees* en anglès) o segons el model equilibrat que generalitza l'anterior. El conjunt d'entrades d'aquests algorismes sempre serà una família d'arbres simplement generats.

En ambdós models, dos objectes (arbres, k-tuples d'arbres) de la mateixa grandària no són necessàriament equiprobables, contràriament al que s'assumeix a la major part d'anàlisis probabilístiques relatives a sistemes de manipulació simbòlica, demostració automàtica de teoremes o compilació, on la hipòtesi d'uniformitat és vàlida en gran nombre de situacions.

Alguns examples de famílies d'arbres que no són uniformement distribuïts inclouen els ja esmentats arbres binaris de cerca, el arbres k-d, els arbres m-aris de cerca, els *tries*, els *Patricia trees*, els *quadtrees*, etc.

L'anàlisi probabilística d'algorismes mitjançant el model BST i el model equilibrat condueix gairebé sempre a equacions diferencials ordinàries i en derivades parcials, en el camí cap a la solució; mentre que en l'anàlisi d'algorismes per arbres uniformement distribuïts, s'han de resoldre problemes algebraics i d'enumeració combinatòria.

La tesi es divideix en quatre parts. Les dues primeres contenen el material introductori, conceptes bàsics i definicions, així com una breu descripció de les eines matemàtiques que hem emprat: el mètode de l'operador simbòlic i tècniques provinents de l'anàlisi de variable complexa. A la part II també es presenta una col·lecció de models de probabilitat per a famílies d'arbres, i hi descrivim els seus contextos d'aplicació natural, alguns resultats previs i la metodologia a seguir en els corresponents anàlisis de cas mitjà.

La part III constitueix la principal contribució d'aquest treball. Comença amb un capítol en el qual s'introdueix l'ocupació, una característica relacionada amb el grau de completitut o balancejat dels arbres. Analitzem aquesta característica sota diferents hipòtesis concernets al model de probabilitat, per tal de poder comparar els principals elements distintius de cada model i el tipus de problemes matemàtics associats a cada un dels models.

Al capítol 5 abordem la definició d'un conjunt de regles que permetran obtenir directament les equacions matemàtiques que descriuen el comportament mitjà d'un algorisme a partir de la pròpia estructura de l'algorisme, per a una certa classe d'algorismes.

El capítol 6 es dedica a l'examen d'una família d'algorismes i una família de predicats

sobre parells d'arbres, interesants i senzills, que anomenem propietats hereditàries. Aquestes propietats admeten caracteritzacions recursives molt similars. La formulació matemàtica del problema d'avaluar la probabilitat que una propietat hereditària es cumpleixi per un parell d'arbres de talla n, és estructuralment idèntica per totes les propietats hereditàries. D'altra banda, el comportament mitjà dels algorismes que verifiquen si un parell d'arbres donat satisfà una propietat hereditària, es pot descriure mitjançant equacions molt semblants. El test d'igualtat entre dos arbres donats és un d'aquests algorismes i rep una atenció més gran en un capítol posterior.

Els dos següents capítols contenen les análisis de la talla mitjana de la intersecció d'un parell d'arbres i de la complexitat mitjana del test d'igualtat per a parells d'arbres binaris. En totes dues anàlisis, el problema original es transforma en el de resoldre una equació diferencial en derivades parcials. L'ús del mètode de Riemann, així com l'aplicació de tècniques d'anàlisi asimptòtica proporciona els resultats desitjats. Per exemple, la talla mitjana de la intersecció d'un parell d'arbres binaris de talla n és $O(n^{2\sqrt{2}-2}/\sqrt{\log n})$ i el test d'igualtat per a parells d'arbres binaris consumeix $O(\log n)$ pasos en mitjana. Fem també una anàlisi detallada de la probabilitat que dos arbres, tots dos de talla n, siguin iguals. Totes les anàlisis mencionades demanen l'ús de resultats matemàtics molt específics, dels quals fem un repàs als preliminars.

Abstract

This thesis is devoted to the analysis of algorithms assuming that the input is distributed according to the BST probability model (thus named after the binary search trees) or to the balanced model that generalizes the former. The input set of such algorithms is some family of simply generated trees.

In those models, two objects of the same size are not equiprobable, contrary to what happens in most average-case analyses of algorithms for symbolic manipulation systems, automatic theorem proving or compiling, for which the assumption that all inputs are equiprobable is valid under most circumstances.

Some examples of families of trees that are not uniformly distributed are binary search trees, k-d-trees, m-ary search trees, digital tries, Patricia trees, quadtrees, etc.

The average-case analysis of algorithms under the BST and the balanced probability model almost always encounters ordinary and partial differential equations in the way towards the solution; while in the average-case analysis of algorithms dealing with uniformly distributed trees we have to solve algebraic and combinatorial enumeration problems.

The thesis is divided in four parts. The first two ones contain the introductory material, basic concepts and definitions, as well as a brief description of the mathematical tools that have been used: the symbolic operator method and complex analysis techniques. In Part II we also present a collection of probability models for families of trees, describing their natural contexts of application, known results and the methodology to be used in the corresponding average-case analyses.

Part III covers the main contributions of this work. It begins with a chapter introducing the occupancy, a characteristic that is closely related to the degree of balancing of trees. We analyze this characteristic under different assumptions concerning the probability model, in order to compare the main differences between the probability models and the kind of mathematical problems associated to each of them.

In Chapter 5 we define a set of translation rules that can be applied systematically to obtain the mathematical equations that describe the average behavior of an algorithm, directly from the structure of the algorithm, for a certain class of algorithms.

Chapter 6 is dedicated to examine a class of elementary algorithms and of simple predicates over pairs of trees, called hereditary properties, that are recursively characterized in similar terms. The mathematical formulation of the problem of evaluating the probability

that the property holds for pairs of size n, is structurally identical for all hereditary properties. On the other hand, the average behavior of the algorithms that check if a given pair of trees satisfies an hereditary property, are described by quite similar equations. The equality test is one of such algorithms and receives a more complete treatement in one of the following chapters.

The next two chapters are devoted to the analysis of the average of the intersection of a pair of trees and of the average-case complexity of the equality test for pairs of binary trees. In both analyses, the original problem translates into that of solving a partial differential equation. The use of Riemann's method as well the application of asymptotic analysis techniques provide the desired results. Thus, the average size of the intersection of two binary trees with total size n is $O(n^{2\sqrt{2}-2}/\sqrt{\log n})$ and the time spent by the equality test for a pair of binary trees is $O(\log n)$ on the average. There is also a detailed analysis of the probability of two binary trees being equal, if both have n nodes. All these analyses require the use of very specific mathematical results, that we overview in the preliminaries.

Acknowledgements

Many people deserves my gratitude, for they had contributed in many aspects and in many ways to the consecution of this work. I expect the ones I have missed will continue being as comprehensive as they have been till now.

I wish to express my indebtedness to Rafael Casas and Josep Díaz, for their constant advice and support; and to Ricardo Baeza-Yates, Philippe Flajolet, Josep Grané, Michèle Loday-Richaud, Hosam Mahmoud, Michael Paterson, Carles Simó and Jean-Marc Steyaert for their technical advice and useful suggestions.

I also would like to thank all the people at the Departament de Llenguatges i Sistemes Informàtics, for their companionship and moral support. I owe particular thanks to Felipe Cucker, who swapped his office with that of mine; the swapping benefited me with an adequate environment to write the thesis.

I also thank the material support received from the Departament de Llenguatges i Sistemes Informàtics (Software Department) of the Universitat Politècnica de Catalunya, from the project ALCOM I of the ESPRIT BRA Program of the EC (Contract no. 3075) that fully supported my research activities during the academic year 1990–1991 and partially during the academic years 1989–1990 and 1991–1992, and from the project ALCOM II (Contract no. 7141).

Finally, I dedicate this thesis to my family: Conrado, Carmen, Angel Carlos and Araceli. Thank you for your comprehension and love.

Contents

Part I

Introduction

The ultimate goal of the analysis of algorithms is to obtain useful information about the computational resources required by algorithms. In general, such information is given in a concise way, as a function of the size of the input to the algorithm.

The same techniques used to analyze algorithms are useful to analyze the behavior of characteristics of data structures. The forthcoming argument about the analysis of algorithms applies also to the analysis of characteristics of data structures. After all, the computational resources used by an algorithm can be viewed as a characteristic of its inputs. Hence, we will speak most of the times of either analysis of algorithms or analysis of characteristics of data structures, when we actually refer to both.

There are many reasons for doing the analysis of an algorithm; the most straightforward of them is to evaluate the suitability of the algorithm for applications and to compare the algorithm with others; another important reason is that the analysis helps to understand the algorithm better and may suggest improvements to it.

In order to analyze an algorithm we begin by defining the *complexity measure* we are interested in. Common complexity measures are time, storage, number of comparisons, number of records moved, number of disk accesses (in algorithms using external memory), etc. The most interesting ones are time and storage, but since these quantities are machine-dependent it is very usual to compute them for some abstract machine model. After that, a *size* function must be defined over the set of inputs of the algorithm. Most data structures carry a natural notion of size. For example, the size of an array is its dimension, the size of a string of symbols is its length, the size of a binary tree is the number of internal nodes it has, the size of a general tree is the total number of nodes (internal and external) in the tree, and so on.

Let us consider an algorithm $\mathcal{A}$ that operates on inputs in the set E. The size of an element $e \in E$ is denoted by $|e|$, and $E_n = \{\, e \in E \mid |e| = n \,\}$ is the set of elements in E that have size n. Let $\mu_{\mathcal{A}}$ be the complexity measure under study[1] (for instance, $\mu_{\mathcal{A}}(e)$ can be the number of elementary steps that algorithm $\mathcal{A}$ makes to process the input e). Notice that a given algorithm can behave in quite different ways even for inputs of the same size. For instance, the total running time of *insertion sort* is linear with respect to the number of items to be sorted, if these items are already sorted; but it is quadratic if the items are in reverse order (in fact, we expect this running time to be quadratic for random inputs) [Sed88].

[1]Subscript $\mathcal{A}$ in $\mu_{\mathcal{A}}$ will be dropped from now on, if there is no ambiguity.

Related to each complexity measure we can define the following quantities:

1. The *worst-case complexity* of algorithm $\mathcal{A}$ over E_n is

$$\mu^{WORST}(n) = \max\{\, \mu(e) \mid e \in E_n \,\}.$$

2. The *best-case complexity* of algorithm $\mathcal{A}$ over E_n is similarly defined by

$$\mu^{BEST}(n) = \min\{\, \mu(e) \mid e \in E_n \,\}.$$

Both complexities give indications concerning the extremal behavior of $\mathcal{A}$ when applied to instances of size n. Their determination usually requires the "construction" of the particular objects for which the algorithm achieves these extremal behaviors.

In many cases, the most valuable information is provided by average-case analysis. As an example of this, it is known that the *quicksort* algorithm requires $O(n \log n)$ operations to sort n items on the average, but it takes $O(n^2)$ operations in the worst case [Sed88].

In the average-case analysis of algorithms, there is a last step which concludes the modelization; it consists in the definition of a probability distribution over each subset of inputs of the same size. We call such family of probability distributions a *probability model*. We can consider the complexity measure as a random variable, and the average-case complexity can be defined as a mathematical expectation in the standard way. We have the following definition:

3. The *average-case complexity* of algorithm $\mathcal{A}$ over E_n is

$$\overline{\mu_{\mathcal{A}}(n)} = \mathrm{E}\{\, \mu_{\mathcal{A}}(e) \mid e \in E_n \,\},$$

where $\mathrm{E}\{\, X \,\}$ denotes the expectation of the random variable X.

The definition of the average-case complexity can be reexpressed as

$$\begin{aligned} \overline{\mu(n)} &= \sum_k k \,\Pr\{\, \mu(e) = k \mid e \in E_n \,\} = \\ &= \sum_{e \in E_n} \Pr(e)\mu(e), \end{aligned}$$

where $\Pr(\cdot)$ is the probability distribution over E_n.

Frequently, the average-case analysis is performed assuming that all elements of a given size are equally likely to happen. This probabilistic model is also known as the *uniform model*, and has proved useful in a wide variety of applications. Moreover, for many of these applications, the uniformity hypothesis leads to predictions that agree with the actual observed performances. For the uniform model the average-case complexity is reexpressed as follows:

$$\overline{\mu(n)} = \frac{1}{|E_n|} \sum_{e \in E_n} \mu(e), \tag{I.1}$$

where $|E_n|$ denotes the cardinality[2] of E_n. In this case, average-case analysis reduces to a problem of *combinatorial enumeration* (see Section 1.1).

It is worth to point out that many algorithms operate on sets of inputs E such that every E_n could be infinite. In general, this kind of algorithms can be analyzed without any additional difficulty, since each E_n can be partitioned into a finite number of disjoint classes, where each of these classes contains objects for which the algorithm behaves in the same way. For instance, in the comparison-based sorting algorithms what matters is not the actual value of the items to be sorted, but the relative order in which they are given. Once this partition has been established the analysis concentrates on the sets E_n^r of the representants of the classes. Worst-case, best-case and average-case complexities are defined accordingly, once we have defined the probability distributions over the E_n^r, in the last case.

There are several drawbacks in the computation of the average-case complexity of algorithms. The first objection to this kind of analysis is that most times the actual probability model is unknown or changes with time and/or from one application of the algorithm to another. When the analysis is carried on assuming a particular probability model, the significance of the obtained results arises as a natural question [Sed83, Fra85, Kar86].

On the other hand, and this is probably the most serious drawback, analyzing the average-case complexity of an algorithm is, in general, intrinsically more difficult than analyzing its worst-case complexity. This is the case even if the probability model is simple.

The study of the average behavior of algorithms under different hypotheses concerning the input distribution provides new information about the analyzed algorithms, since the average behavior of the algorithms depends on the assumed probability model. On the other

[2]We use the same notation to denote the size of an item and the cardinality of a set. Since it will be clear from the context whether x is an item or a set, the meaning of $|x|$ should also be clear.

hand, the study would enlight our knowledge of the probability models themselves and give us useful hints on the dependence of the analysis with respect to the probability model, since each probability model requires its own set of algebraic and analytic techniques to perform its corresponding analysis.

The recursive nature of programming schemes and of data structures is, whenever possible, exploited to describe the average behavior of algorithms, by means of recurrences or by means of functional equations over generating functions. Not surprisingly, many models of probability are recursively defined as well, and hence, the average-case analysis is reduced to the investigation of recursion models.

A possible approach consists in deriving the recurrences that describe the behavior of the algorithm and solving them. The recurrences are obtained from the recursive decomposition of the inputs into smaller components and the recursive nature of the algorithms. One then tries to solve the recurrences relying on classical techniques such as the calculus of finite differences or standard asymptotic analysis. Quite often, the recurrences are translated to functional equations over generating functions in order to solve the recurrences or to extract asymptotical information.

The application of the *symbolic operator method* is a more direct approach. In the symbolic operator method, the set-theoretic definitions of the data structures, and that of the complexity measure of a given instance in terms of smaller instances, are used to systematically derive functional equations over *generating functions*. The derivation of such equations is made by the application of translation rules that establish a "one-to-one" correspondence between algorithmic constructions and operators over generating functions. The coefficients of the generating functions represent the quantities that we need to perform the analysis.

Therefore, both approaches often reduce the original problem (analyzing an algorithm) to the same mathematical problem (extracting information from some given functional relations between generating functions).

If we are lucky enough, it will be easy to explicitly solve the recurrences or the functional equations over generating functions. The solution will provide us with explicit expressions for the average-case complexity as a function of the input size n. When this explicit expression is not in a closed form or cannot be easily interpreted and compared, standard asymptotic analysis can be applied to obtain the average-case complexity in terms of ordinary functions such as powers of n, logarithms, iterated logarithms, etc.

On the other hand, certain number of complex analysis techniques and, in particular, singularity analysis, have been developed in order to extract the asymptotic behavior of the coefficients of generating functions, even when the generating functions satisfy implicit functional equations that are not explicitly solvable. Complex analysis techniques have played an important rôle in the area, greatly simplifying analysis that are rather intricate.

The combination of the symbolic operator method and the complex analysis techniques has proved very successful for the analysis of algorithms where only some algorithmic constructions are allowed, and the inputs are some kind of combinatorial structures of recursive nature, such as trees and strings of symbols. An spectacular witness of the success of this combination is the construction of the computer system ΛΥΩ [FSZ91], that can automatically perform the average-case analysis of a non-trivial collection of algorithms.

These approaches are depicted in Figure I.1 [VF90].

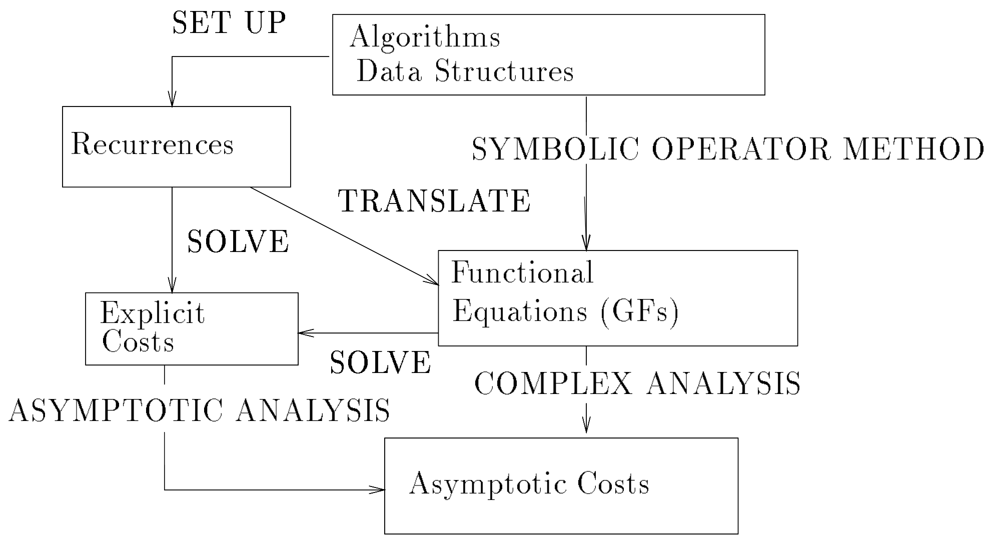


Figure I.1: *Average-case analysis.*

In this work we will concentrate on algorithms dealing with trees and on characteristics of trees. We have followed the methodological approach based on the use of the symbolic operator method and the use of complex analysis techniques. We are interested in doing the average-case analysis of algorithms, when the input trees are distributed according to the *BST probability model*. The name of this model comes from the fact that it corresponds to *binary search trees* built up from random permutations. It turns out that in this model, trees of the same size may not have the same probability; moreover, high probability is assigned to well balanced trees, whereas poorly balanced trees are quite unlikely. The BST model can be recursively defined and the characteristic type of functional equations associated to it are differential equations.

The BST probability model has been successfully applied to analyze the algorithms that dynamically maintain a dictionary implemented with a binary search tree. The model also arises in the study of heap ordered trees, used to implement priority queues, and in the study of multidimensional search algorithms that use k-d-trees [Knu73, Nie74, Rob82, Dev86, FP86, Fla88b].

Moreover, the BST model is a particular case of the probabilistic model for randomly built m-ary search trees. These trees are used for similar purposes as the BSTs, but with external storage devices. The average performance of the algorithms that dynamically maintain m-ary search trees and other average characteristics of these trees have received attention in the last years [Mah86, MP89, Mah92].

We have defined a generalization of the BST model, that we have called *balanced probability model*. The balanced probability model is defined over *simple families of trees* enjoys many of the attributes of the BST model. For trees in any simple family of trees, the intrinsic difficulty of doing the analysis using the balanced model is not greater than that of doing the same analysis using the BST model for binary trees.

The objective of this work is to investigate what kind of mathematical problems appear and what techniques should be used when we carry on the average-case analysis of algorithms, when the probability model is the BST model or the balanced model. The use of the BST probability model demands the development of specific mathematical tools for appropriately performing the average-case analysis. Consequently, most of the contributions of our work concern the techniques for obtaining the asymptotic average behavior of algorithms from the differential equations that characterize the BST model.

On the other hand, comparisons between the way the analysis is done for the BST model and for other models, and between the results of such analyses turns out to be relevant. For instance, the idea of "discriminative power" of a probability model arised as a consequence of our studies. We have observed that some algorithms that have the same qualitative average behavior for the uniform model have quite different average behavior if the BST model is assumed. Hence, we say that the BST model discriminates the average behavior of these algorithms while the uniform model does not. In other cases, two different algorithms exhibit the same behavior for both models: neither the uniform model nor the BST model discriminates the average behavior of the two algorithms. These phenomena help understanding the main distinctive characteristics of the algorithms and the probability models considered.

The first tool that we present for the average-case analysis under the BST and balanced models is a set of translation rules that map algorithmic constructions into functional equations over generating functions, in the same way it has been done for the uniform model [FS87]. This kind of complexity calculus is theoretically feasible, as long as we restrict ourselves analyzing algorithms using only certain algorithmic constructions. The set of translation rules, that we have proposed, provides a systematic procedure to directly obtain the equations that describe the average behavior of an algorithm from the structure of algorithm. The algorithmic constructions that we have considered are powerful enough to write many interesting algorithms. Hence, the limits on what can be analyzed and what cannot, substantially depend on our ability to extract information from the equations that describe the average behavior of algorithms. As an example of the application of the translation rules for the balanced model, we analyze a formal differentiation algorithm. The definition of a set of translation rules for the balanced model, useful for the analysis of recursive tree algorithms, constitutes an interesting application of the symbolic operator method.

To attain our goals we have chosen to study simple algorithms over pairs of trees, progressively going into more complex problems and trying to enlarge the collection of available techniques and results.

In this sense, we have found a collection of simple problems that are quite similar in many aspects. The unifying concept for those problems is that of *hereditary property*. An hereditary property is a predicate over pairs of trees that is true if and only if the property holds for the pair of left subtrees and the pair of right subtrees. The differences between hereditary properties are given by the way the property is defined for pairs containing at least an empty tree. Examples of such properties are the equality of trees, root occurrence (in the context of tree matching) and consistency (in the context of unification of first-order terms).

The most basic question about hereditary properties is that of evaluating the probability that a given pair of trees of size n verifies a given hereditary property. The similarities in the definition of hereditary properties lead to an essentially identical mathematical formulation of the problem of evaluating probabilities, for all the hereditary properties. Another interesting question is the relation between hereditary properties and algorithms. Even if we have only partial information on the probability of an hereditary property, we can use it to study algorithms whose average behavior ultimately depends on that probability. This is the case for the algorithm that tests equality between a pair of trees, for the one that checks whether a pair contains a direct ocurrence or for that determining if a tree (pattern) occurs inside another (text), etc. The equality test algorithm and the probability of two binary trees being equal have been studied more extensively and, in fact, initiated our interest about hereditary properties.

Besides the recursive scheme associated to hereditary properties, other recursive schemes have received our attention. Our first complete case of study is the size of the intersection of two binary trees. This characteristic exemplifies one of the simplest non-trivial recursion schemes over pairs of trees. Roughly speaking, the intersection of a pair of trees is the tree that results if one of the trees in the pair is superimposed to the other and then the nodes that do not belong to both are deleted. The complexity of the algorithm that computes the intersection of a given pair is proportional to the size of the intersection.

Our analysis of the intersection leads to a partial differential equation. As we have already said, this kind of functional relation relies on the nature of the probability model rather than on the characteristic we are analyzing. We solve the partial differential equation by means of Riemann's method. The next step in our analysis yields an asymptotic expansion of the solution around its singularity. The application of standard complex analysis techniques allows us to obtain an asymptotic estimation of the average size of the intersection of a pair of binary trees of size n: $\Theta(n^{2\sqrt{2}-2}/\sqrt{\log n})$.

As we said before, we have investigated the average-case complexity of the equality test between a pair of trees, as a function of the size of the pair. The equality test algorithm is still simple but differs substantially from our previous case of study. The equality test is based on the recursive application of the procedure to pairs of non-empty subtrees, but subsequent recursive calls are conditioned to the result of previous calls, while the computation of the intersection requires the unconditional recursive application of the procedure to all pairs of non-empty subtrees.

Once again, we have assumed that the input for the equality test is distributed according to the BST probability model. Like in the analysis of the intersection, the generating function associated to the average costs verifies a partial differential equation; but it is more difficult to solve. We carry on the analysis studying a simpler algorithm. The average-case complexity of this simple algorithm has an associated generating function that is proved to be asymptotically equivalent to the generating function of the equality test. Therefore, the average-case complexities of both algorithms are equivalent and turn out to be $\Theta(\log n)$.

One of the problems arising during the average-case analysis of the equality is the question of evaluating the probability that both members of the pair are equal, when each of the trees has size n. In order to prove that the equality test has logarithmic average-case complexity, we only need to show that this probability decreases exponentially as n gets large. A deeper study of the behavior of this probability has deserved a chapter in this thesis. It appears to be $O(\rho^{-n}\, n)$ where $\rho = 3.1408577\ldots$. The analysis involves the study of the analyticity of the solution of a second-order non-linear ordinary differential equation and of an asymptotic expansion of that solution around its dominant singularity.

The thesis is organized as follows: in Part II we present the basic concepts to be used in the remainder of thesis; the symbolic operator method and the related topic of combinatorial enumeration, and some common complex analysis techniques of wide application in combinatorics and analysis of algorithms. The last section exposes some basic concepts and identities of Bessel functions of the first kind that are needed in the next chapters.

Chapter 3 of Part II covers the definition of the BST and the balanced probability models. It also introduces the principle of extension of the models to pairs of trees and to k-tuples. It presents other common probability models for trees as well.

The main contributions of the thesis are contained in Part III. In Chapter 4 we present the occupancy, a characteristic over trees, related to the balancing of trees. The analysis of this characteristic under different probability models exemplifies the differences between the probability models as well as the steps to be followed and some of the available techniques, for each of the probability models. Part of this material appeared in *Average-case Analysis on Simple Families of Trees Using a Balanced Probability Model* by Casas, Díaz, and Martínez [CDM91a].

In Chapter 5 we establish a set of translation rules, whose systematic application yields an equation describing the average behavior of any algorithm when the inputs are distributed according to the balanced model, and as long as the algorithm only uses the considered operations over trees.

Chapter 6 introduces hereditary properties. In this chapter, we examine the problem of evaluating the probability that a given pair of trees satisfies an hereditary property. We also examine the relation between these properties, their probability and the algorithms that are closely related to them.

The rest of the chapters of Part III describes the concrete analyses that we have briefly mentioned before. Chapter 7 presents the analysis of the average size of intersection. Most of the material of this chapter was published in *On the Average Size of Intersection of Binary Trees* by Baeza-Yates, Casas, Díaz, and Martínez [BCDM92] and in [CDM91a]. Chapter 8 is on the average-case complexity of the equality test. The main results of this chapter were presented in *Average-case Analysis of Equality of Binary Trees under the BST Probability Model* by Martínez [Mar91]. Chapter 9 deals with the probability that two binary search trees are equal.

In the last part, we briefly sketch the main conclusions of this work and explore some ideas for future research. Moreover, this part contains a concise description of the *common subexpression problem* and the first steps of its average-case analysis for the BST probability model. This is an open research problem.

Background sources on the analysis of algorithms are *The Art of Computer Programming* [Knu68, Knu73] and the surveys of Sedgewick [Sed83], Flajolet [Fla88a] and Vitter and Flajolet [VF90].

Information about data structures and algorithms can be found in the books of Knuth [Knu68, Knu73], Aho, Hopcroft and Ullman [AHU76, AHU83], Sedgewick [Sed88], Cormen, Leiserson and Rivest [CLR90] and Gonnet and Baeza-Yates [GB91], among others.

Additional coverage of the topic of analysis of algorithms can be found in the books of Kemp [Kem84], Purdom and Brown [PB85] and the survey by Casas, Díaz and Martínez [CDM91b]. A recent book on the topic of average-case analysis of random tree structures, and therefore closely related to the topic of this thesis, is *Evolution of Random Search Trees* by Mahmoud [Mah92].

Combinatorial enumerations and the use of generating functions, which will be presented in the sequel, appear in the books of Comtet [Com74], Goulden and Jackson [GJ83], Stanley [Sta86], and Wilf [Wil90]. For the mathematical foundations of generating functions we refer the reader to the article *The idea of generating function* by Rota [Rot75]. For the general foundations on asymptotic methods we suggest the work of De Bruijn [DB58]. The survey of Bender [Ben74] illustrates many uses of asymptotic methods in enumeration problems.

Part II

Preliminaries

Chapter 1

Mathematical Preliminaries

1.1 Combinatorial Enumeration.

1.1.1 Generating Functions.

As we shall see, the basic tool in the analysis of data structures and algorithms used in this work is the *generating function* . Quoting Herbert Wilf [Wil90],

"A generating function is a clothesline on which we hang up a sequence of numbers for display."

Following the analogy, the clothes we will hang up in this work when doing average-case analysis are almost always expected values of the complexity measure for inputs of a given size.

Definition 1.1. *Let* $\{ a_k \}_{k\geq 0}$ *be any sequence of complex numbers. The* generating function *(g.f., for short) of the sequence* $\{ a_k \}_{k\geq 0}$ *is*

$$A(z) = \sum_{k\geq 0} a_k \, z^k. \tag{1.1}$$

The generating function of a sequence is often called ordinary generating function, to distinguish it from the exponential generating function, the Dirichlet generating function, etc.

Equation (1.1) associates a formal power series to a sequence of numbers. Thus, generating functions are elements of a Cauchy algebra, therefore the classical algebraic methods can be applied. On the other hand, the defined series are in most cases convergent and

Sequences	Generating functions
1. $c_n = a_n \pm b_n$	$C(z) = A(z) \pm B(z)$
2. $c_n = \sum_{k=0}^{n} a_k b_{n-k}$	$C(z) = A(z) \cdot B(z)$
3. $c_n = a_{n-1}$	$C(z) = zA(z)$
4. $c_n = a_{n+1}$	$C(z) = (A(z) - A(0))/z$
5. $c_n = na_n$	$C(z) = z\frac{d}{dz}A(z)$
6. $c_n = \frac{a_n}{n}$	$C(z) = \int_0^z (A(t) - A(0))\frac{dt}{t}$

Table 1.1: *Translation of operations over sequences to operations over g.f.'s.*

hence they can be treated by analytical methods. This duality makes the generating function a powerful tool. To apply analytical methods, the variable z of a generating function $F(z)$ is considered as a complex variable and the generating function as a complex function of z. Moreover, as we will see in next subsection, there is a natural way to associate generating functions to denumerable sets. This fact converts generating functions in helpful tools for counting applications.

The n-th coefficient of a formal power series $F(z)$ will be denoted by $[z^n]F(z)$ (also, $[z^n]F(z)$ denotes the n-th coefficient of the Taylor expansion of an analytic function $F(z)$ around $z = 0$). Thus, $F(z) = \sum_{n\geq 0} f_n\, z^n$ implies that $[z^n]F(z) = f_n$.

In Table 1.1 we present how elementary operations over sequences translate to operations over the corresponding generating functions.

Operations (1), (3) and (4) are known as *sum*, *backward shift* and *forward shift*; (2) is called *convolution* of sequences. *Differentiation* and *integration* are specified by (5) and (6).

1.1.2 Admissible Combinatorial Constructions.

Many set-theoretic operators of interest in combinatorial enumeration, such as *Cartesian product*, *powerset* or *Kleene closure*, translate into operators over the generating functions associated to the operands (see, for instance,[Fla88a]). We start defining a *class of combinatorial structures* .

Definition 1.2. *Given a finite or denumerable set $\mathcal{C}$ and a* size *function*[1] $|\cdot| : \mathcal{C} \mapsto \mathbb{N}$, *the pair* $< \mathcal{C}, |\cdot| >$ *is a* class of combinatorial structures *if and only if for each n, the set* $\mathcal{C}_n = \{ c \in \mathcal{C} \mid |c| = n \}$ *is finite.*

From now on, any given class of combinatorial structures will be denoted as the set in the pair, with some abuse of the notation.

Let c_n be the cardinality of $\mathcal{C}_n$. Then the *counting generating function* of $\mathcal{C}$ is

$$C(z) = \sum_{n \geq 0} c_n z^n. \tag{1.2}$$

An alternative interesting way to express (1.2) is

$$C(z) = \sum_{c \in \mathcal{C}} z^{|c|}. \tag{1.3}$$

Notice that we adhere to the notational convention that uses the same letter for the class, its generating function, and the elements of the induced sequence $\{ c_n \}_{n \geq 0}$ in different cases and fonts. We will also use lowercase roman letters to denote elements of the set in a class of combinatorial structures.

Any operation over k classes of combinatorial structures $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k$ producing a new class of combinatorial structures $\mathcal{A}$ is called *combinatorial construction.* The specification of a combinatorial construction describes how the elements in the set $\mathcal{A}$ relate to the elements of the sets $\mathcal{C}_1, \ldots, \mathcal{C}_k$ and how the size function of the resulting class relates to the size functions of the operands. For instance, the Cartesian product over classes $\mathcal{A}$ and $\mathcal{B}$ is defined as the class where the set is the usual Cartesian product of the corresponding sets, and the size of a pair (a, b) in $\mathcal{A} \times \mathcal{B}$ is the sum of the sizes of a and b. The most interesting combinatorial construction are the *admissible* combinatorial constructions:

Definition 1.3. *A combinatorial construction Φ is* admissible *if there exists an operator Ψ over generating functions such that*

$$\mathcal{A} = \Phi(\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k) \Rightarrow A(z) = \Psi(C_1(z), C_2(z), \ldots, C_k(z)),$$

where $A(z), C_1(z), \ldots, C_k(z)$ are the counting g.f.'s corresponding to the classes $\mathcal{A}, \mathcal{C}_1, \ldots, \mathcal{C}_k$.

A large collection of admissible combinatorial constructions has been investigated in the context of combinatorial enumeration. In Table 1.2, we summarize a set of admissible combinatorial constructions and their corresponding operators over generating functions.

[1] We shall use the usual infix notation for the size function with subscripts when necessary, like in $|\cdot|_{\mathcal{C}}$

Construction		Operator
Disjoint union	$\mathcal{C} = \mathcal{A} + \mathcal{B}$	$C(z) = A(z) + B(z)$
Cartesian product	$\mathcal{C} = \mathcal{A} \times \mathcal{B}$	$C(z) = A(z) \cdot B(z)$
Diagonal	$\mathcal{C} = \Delta(\mathcal{A} \times \mathcal{A})$	$C(z) = A(z^2)$
Kleene closure	$\mathcal{C} = \mathcal{A}^*$	$C(z) = \frac{1}{1-A(z)}$
Marking	$\mathcal{C} = \mu\mathcal{A}$	$C(z) = z\frac{d}{dz}A(z)$
Substitution	$\mathcal{C} = \mathcal{A}[\mathcal{B}]$	$C(z) = A(B(z))$

Table 1.2: *Admissible combinatorial constructions.*

Some of the constructions, such as *disjoint unions*, *Cartesian products* and *diagonals* are the standard ones in set theory; others like *composition* or *marking* arise when dealing with objects like trees, graphs and words, that are made up of *atomic units* or *elements*. We characterize formally some combinatorial constructions in the following definitions.

Definition 1.4. *A class $\mathcal{C}$ is the* disjoint union *of $\mathcal{A}$ and $\mathcal{B}$, denoted $\mathcal{C} = \mathcal{A} + \mathcal{B}$ if*

1. $\mathcal{C} = \mathcal{A} \cup \mathcal{B}$ *and* $\mathcal{A} \cap \mathcal{B} = \emptyset$.

2. $|x|_{\mathcal{C}} = |x|_{\mathcal{A}}$ *if* $x \in \mathcal{A}$ *and* $|x|_{\mathcal{C}} = |x|_{\mathcal{B}}$ *if* $x \in \mathcal{B}$.

Definition 1.5. *A class $\mathcal{C}$ is the* Cartesian product *of $\mathcal{A}$ and $\mathcal{B}$, denoted $\mathcal{C} = \mathcal{A} \times \mathcal{B}$ if*

1. $\mathcal{C} = \mathcal{A} \times \mathcal{B}$ *(in the set-theoretical sense).*

2. $|(a,b)|_{\mathcal{C}} = |a|_{\mathcal{A}} + |b|_{\mathcal{B}}, \qquad a \in \mathcal{A}, \quad b \in \mathcal{B}.$

Definition 1.6. *A class $\mathcal{C}$ is the* Kleene closure *of $\mathcal{A}$, denoted $\mathcal{C} = \mathcal{A}^*$ if*

$$\mathcal{C} = \{\,\lambda\,\} + \mathcal{A} + \mathcal{A} \times \mathcal{A} + \mathcal{A} \times \mathcal{A} \times \mathcal{A} + \cdots,$$

where λ is the empty structure (by definition $|\lambda| = 0$), and the size is defined consistently with the disjoint union and cartesian product definitions.

Definition 1.7. *A class $\mathcal{C}$ is the* diagonal *of another class $\mathcal{A}$, denoted $\mathcal{C} = \Delta(\mathcal{A} \times \mathcal{A})$ if*

1. $\mathcal{C} = \{\,(a,a) \mid a \in \mathcal{A}\,\}$.

2. $|(a,a)|_{\mathcal{C}} = 2|a|_{\mathcal{A}}, \qquad a \in \mathcal{A}$.

Definition 1.8. *A class $\mathcal{C}$ is the* composition *or* substitution *of two given classes $\mathcal{A}$ and $\mathcal{B}$, denoted $\mathcal{C} = \mathcal{A}[\mathcal{B}]$ if*

1. *$c \in \mathcal{C}$ if and only if there exists some $n \geq 0$, $a \in \mathcal{A}_n$ and $b_1, \ldots, b_n \in \mathcal{B}$ such that $c = (a, b_1, \ldots, b_n)$.*

2. *$|(a, b_1, b_2, \ldots, b_n)|_{\mathcal{C}} = |b_1|_{\mathcal{B}} + \cdots + |b_n|_{\mathcal{B}}$.*

Similar rules to those in Table 1.2 exist for generating functions of the form

$$F(z) = \sum_{c \in \mathcal{C}} f(c)\, z^{|c|},$$

when the class $\mathcal{C}$ is constructed from some other classes, and the function f is defined in terms of functions over the components of the elements in $\mathcal{C}$. For instance, if $\mathcal{C} = \mathcal{A} \times \mathcal{B}$ and if $f(c) = g(a)h(b)$ for $c = (a,b)$, then

$$F(z) = \sum_{(a,b) \in \mathcal{A} \times \mathcal{B}} g(a)h(b)\, z^{|a|+|b|} = \left(\sum_{a \in \mathcal{A}} g(a)\, z^{|a|}\right) \cdot \left(\sum_{b \in \mathcal{B}} h(b)\, z^{|b|}\right) = G(z)H(z).$$

More details and full proofs for the lemmas given in Table 1.2 can be found in the book of Goulden and Jackson [GJ83] and the surveys of Flajolet [Fla88a] and Vitter and Flajolet [VF90]. The last two references describe how these methods can be applied in average-case analysis of algorithms and data structures.

1.1.3 Two Expansion Theorems.

There are two interesting theorems about generating functions, that allow to recover exactly the n-th coefficient of a generating function under particular circumstances, and whose proof is purely algebraic, valid in the ring of formal power series.

The first one is the so-called general expansion theorem for rational generating functions.

Theorem 1.1. *If $R(z) = P(z)/Q(z)$, for some polynomials $P(z)$ and $Q(z)$, and $Q(z)$ has l distinct roots ρ_1, ρ_2,..., ρ_l of multiplicities d_1, d_2,..., d_l, and the degree of $P(z)$ is less than that of $Q(z)$ then*

$$[z^n]R(z) = f_1(n)\rho_1^n + f_2(n)\rho_2^n + \cdots + f_l(n)\rho_l^n, \qquad n \geq 0,$$

where each $f_k(n)$ is a polynomial of degree $d_k - 1$ and its leading coefficient a_k is

$$a_k = \frac{d_k\, P(1/\rho_k)(-\rho)^{d_k}}{Q^{(d_k)}(1/\rho_k)}.$$

A short proof of the theorem can be found in the book of Graham, Knuth and Patashnik [GKP89].

As we shall see in next section, there are other methods to get asymptotic estimates of the n-th coefficient of rational generating functions that are usually useful enough and easier to apply than this theorem.

The other theorems we consider are the Lagrange inversion formula and its variant, the Lagrange-Bürmann inversion formula (see, for example,[Com74, Wil90]). They can be applied when the generating functions under study satisfy a particular kind of implicit equation.

Theorem 1.2. (Lagrange Inversion Formula) *Let $\phi(u) = \sum_{n \geq 0} \phi_n\, z^n$ be a formal power series such that $\phi(0) = \phi_0 \neq 0$. Then the equation*

$$y(z) = z\, \phi(y(z))$$

has a unique formal power series solution that verifies

$$y(z) = \sum_{n>0} \frac{z^n}{n}\, [y^{n-1}]\phi^n(y).$$

The Lagrange-Bürmann inversion formula is a generalization of the Lagrange inversion formula. Although we give here an statement of the theorem that includes an analyticity hypothesis about the series g, the inversion formula can be stated without making such hypothesis, and proved in purely algebraic terms.

Theorem 1.3. (Lagrange-Bürmann Inversion Formula) *Let $\phi(u)$ be a power series as in Theorem 1.2 and $g(u)$ an analytic function around $u = 0$. Let $y(z) = \gamma + z\,\phi(y(z))$, for some arbitrary constant γ. Then,*

$$g(y(z)) = g(\gamma) + \sum_{n \geq 0} \frac{z^n}{n!} \left[\frac{d^{n-1}}{dy^{n-1}} \left(g'(y)\phi^n(y) \right) \right]_{y=\gamma}.$$

1.2 Complex Analysis Techniques.

1.2.1 Generalities.

The power of generating functions as a mathematical tool relies on its duality. They can be considered as formal power series, and most times they can also be considered as functions of complex variable in the complex plane, analytic in a disk around the origin.

Suppose we have the generating function $F(z)$ of a certain sequence. Since the power series of the function is analytic in the largest disk centered at the origin containing no *singularities*, our first step will be looking for the singularities of the function that are nearest to the origin. The distance from the origin to the nearest singularity is called *radius of convergence* of the power series and such singularity is called *dominant*.

The radius of convergence provides useful information about the behavior of the coefficients $f_n = [z^n]F(z)$, (see, for example [Tit39]):

Theorem 1.4. *Let ρ be the radius of convergence of the power series $F(z) = \sum_{n\geq 0} f_n\, z^n$. Then, for all $\epsilon > 0$,*

$$(1-\epsilon)^n \rho^{-n} <_{\text{i.o.}} f_n <_{\text{a.e.}} (1+\epsilon)^n \rho^{-n},$$

where $<_{\text{i.o.}}$ means that the inequality holds for an infinite number of values of n, whereas $<_{\text{a.e.}}$ means that the inequality holds for all values of n, except for a finite number of them.

But this information about the exponential growth of the f_n's is usually insufficient for our purposes and we look for information on the subexponential growth or, even better, an asymptotic equivalent of f_n. Next subsection briefly exposes some of the techniques, that using information about the nature of the dominant singularity of the function $F(z)$ and the behavior of the function near it[2], gather information about the asymptotic behavior of the coefficients. The basis of most of these techniques is Cauchy's integral formula (see [Tit39]):

Theorem 1.5. (Cauchy's Integral Formula) *Let $F(z)$ be an analytic function in an open domain D enclosing the origin, and let Γ be a simple closed curve entirely lying on D that also encloses the origin. Then the n-th coefficient of the Taylor series development of $F(z)$ around $z = 0$ is given by*

$$[z^n]F(z) = \frac{1}{2\pi i}\int_\Gamma \frac{F(z)}{z^{n+1}}\, dz.$$

[2]These techniques can be easily generalized to the case where there is more than one dominant singularity.

As an immediate corollary of Theorem 1.5 we have:

Corollary 1.1. *Let $F(z)$ be an analytic function and let ρ be the radius of convergence of its Taylor series development around the origin. Moreover, let $f_n = [z^n]F(z)$. Then, for any $0 < r < \rho$*

$$|f_n| \le \frac{M(r)}{r^n},$$

where $M(r) = max_{|z|=r} |F(z)|$. Furthermore, if $f_n \ge 0$ for all n, then $M(r) = |F(r)|$.

Finally, we state Cauchy's residue formula, which is at the core of some important techniques used in the analysis of algorithms and data structures.

Theorem 1.6. (Cauchy's Residue Formula) *Let z_0 be an isolated singularity of $F(z)$ and let C be a circle centered at z_0 such that $F(z)$ is analytic in C and its interior, except possibly at z_0. Then,*

$$\int_C F(z)\, dz = 2\pi i \,\mathrm{Res}[F(z); z = z_0],$$

where $\mathrm{Res}[F(z); z = z_0]$ denotes the residue of $F(z)$ at z_0.

In this subsection, as well as in the forthcoming, we have assumed that the reader is already familiar with concepts such as analyticity, convergence of a power series, singularity, residue, Laurent and Taylor series developments, etc. Some background references for these concepts and many more are the books by Titchmarsh [Tit39], Cartan [Car61], Henrici [Hen77] and Lang [Lan85]. The surveys of Flajolet [Fla88a], Flajolet and Odlyzko [FO90], and Vitter and Flajolet [VF90] cover most of the methods of complex analysis used in average-case analysis.

1.2.2 Singularity Analysis.

The first theorem that we will consider is Darboux's theorem (a proof of this theorem can be found in Henrici's *Applied and Computational Complex Analysis* [Hen77]). It can be applied in many situations where the dominant singularity is either a pole or is an algebraic singularity (branching point).

Theorem 1.7. (Darboux) *Let $F(z)$ be an analytic function in the open disk $|z| < \rho$ and assume that there is a unique singularity in the convergence circle at $z = \rho$. Furthermore, in a neighbourhood of $z = \rho$, $F(z)$ satisfies*

$$F(z) = \left(1 - \frac{z}{\rho}\right)^{-\beta} G(z) + H(z),$$

for some analytic functions $G(z)$ and $H(z)$ at $z = \rho$, with $G(\rho) \neq 0$, and for some real $\beta \notin \{0, -1, -2, -3, \dots\}$. Then,

$$[z^n]F(z) = f_n \sim \rho^{-n}\, n^{\beta-1}\, \frac{G(\rho)}{\Gamma(\beta)} \left(1 + O\left(\frac{1}{n}\right)\right),$$

where $\Gamma(z)$ denotes Euler's gamma function and $\sim$ denotes asymptotic equivalance.

An informal interpretation of this theorem is that it establishes the conditions that must be fulfilled to properly use a series development of $F(z)$ around $z = \rho$ and estimate the behavior of f_n from the first term of the series, using Newton's binomial theorem.

Another interesting collection of results are the so-called *transfer lemmas*, whose purpose is also to translate asymptotic information about a function around its dominant singularity to asymptotic information about the coefficients of the function. The application of a transfer lemma does not require conditions on the smoothness of the lower order terms in the asymptotic expansion, as Darboux's theorem does, but some other conditions should be satisfied.

In general, a transfer lemma states that if certain suitable conditions are satisfied, then asymptotic expansions valid for $z \to 1$, like

$$F(z) \sim G(z), \qquad F(z) = O(G(z)), \qquad F(z) = o(G(z)),$$

can be "transferred" to coefficients for $n \to \infty$

$$f_n \sim g_n, \qquad f_n = O(g_n), \qquad f_n = o(g_n).$$

If the dominant singularity of $F(z)$ is not at $z = 1$ an appropriate normalization can be performed. The suitable conditions that we have mentioned, are that $G(z)$ belongs to a restricted class of functions (that includes most common ones) and that the asymptotic expansion holds in a region that partially lies outside the convergence disk. This last condition usually requires analytic continuation of $F(z)$ to a region enclosing the convergence disk.

The asymptotic scale to which $G(z)$ must belong contains functions of the form:

$$G(z) = (1-z)^\alpha \left(\log \frac{1}{1-z}\right)^\gamma \left(\log\log \frac{1}{1-z}\right)^\delta,$$

for some constants α, γ and δ.

The proof of a transfer lemma is generally based on contour integration using Cauchy's formula (Theorem 1.5). Applications of transfer lemmas appear in the papers of

Odlyzko [Odl82] and Flajolet and Odlyzko [FO82, FO90]. Transfer lemmas are quite similar to *Tauberian theorems* for the kind of information that they provide, but there are important differences on the conditions that one should establish to apply them (see next subsection).

We finish giving a transfer lemma that we shall use in next chapters:

Theorem 1.8. *Assume that $F(z)$ is an analytic function in the domain $\Delta = \{\, z \mid |z| \leq 1+\eta, |\arg(z-1)| \geq \phi \,\}$, with $\eta > 0$ and $0 < \phi < \pi/2$. Moreover, as $z \to 1$ in Δ,*

$$F(z) = (1-z)^{\alpha}\left(\log\frac{1}{1-z}\right)^{\gamma}\left[\sum_{j=0}^{m-1} c_j\left(\log\frac{1}{1-z}\right)^{-j} + O\left(\left(\log\frac{1}{1-z}\right)^{-m}\right)\right]$$

for some $\alpha, \gamma \notin \{\, 0,1,2,\ldots \}$. Then as $n \to \infty$

$$f_n = \frac{n^{-\alpha-1}}{\Gamma(-\alpha)}\log^{\gamma} n\left[\sum_{j=0}^{m-1} c'_j \log^{-j} n + O(\log^{-m} n)\right].$$

In particular, $c_0 = c'_0$ and the other c'_j can also be explicitly computed from the c_j.

The theorem remains valid if any of α or γ is a positive integer, introducing the convention that $1/\Gamma(-\alpha)$ is null for nonnegative α. For a full proof of this theorem we refer the reader to the survey of Flajolet and Odlyzko [FO90].

1.2.3 Other Methods.

In this subsection, we make a brief review of other interesting complex analysis techniques.

Tauberian theorems infer estimates of the coefficients of a generating function provided we have an asymptotic expansion of the function valid when z tends to the dominant singularity *along the real axis.* This is a weaker requirement than that of transfer lemmas, for which an asymptotic expansion valid on a region partially lying outside the disk of convergence is needed. Nevertheless, side conditions on the coefficients, like positivity and monotonicity, have to be established *a priori*. For instance, Hardy-Littlewood-Karamata's theorem gives an estimation similar to the one in Theorem 1.8, if the coefficients f_n of $F(z)$ are positive and form a monotonic sequence [VF90]. For further inside on Tauberian theory and some of its combinatorial applications we refer the reader to the works of Titchmarsh [Tit39], Hardy [Har49], Feller [Fel68], Bender [Ben74] and Green and Knuth [GK82].

When the function we are studying is entire (has no singularities) or is exponentially growing near its singularity, we can use *saddle-point methods* [Hay56, HS68, OR85].

A somewhat different technique from those we have considered till now is based in *Mellin transform*. Given a real function $F(x)$ defined for all positive x, its Mellin transform, denoted by $F^*(s)$, is the complex-valued function given by:

$$F^*(s) = \int_0^\infty F(x)x^{s-1}\,dx.$$

The fundamental strip of the Mellin transform, denoted $<\alpha;\beta>$, is the strip $-\alpha < \Re(s) < -\beta$ such that $F^*(s)$ is defined. A very elementary property of the Mellin transform confers it most of its usefulness:

$$\left(\sum_k \lambda_k F(a_k x)\right)^* = \left(\sum_k \lambda_k a_k^{-s}\right) F^*(s).$$

If $F^*(s)$ is meromorphic, Cauchy's residue formula can be applied to evaluate the inversion formula of the Mellin transform and get

$$\begin{aligned} F(x) &= \frac{1}{2\pi \mathrm{i}}\int_{c-\mathrm{i}\infty}^{c+\mathrm{i}\infty} F^*(s)x^{-s}\,ds \\ &\sim \sum_\sigma \mathrm{Res}[F^*(s)x^{-s}; s=\sigma], \end{aligned}$$

where c is in the fundamental strip and the sum below extends to all poles of $F^*(s)$ in the semiplane $\Re(s) > c$. Mellin transform techniques are useful to analyze problems involving sums of arithmetic functions such as the ones appearing in analytic number theory. They are a primary tool in the analysis of tries, digital search trees, radix exchange sort, hashing, etc. [FRS85, VF90].

Standard real analysis techniques constitute another large family of methods to do asymptotics. Among these techniques, we shall mention Euler-Maclaurin summation formula, Laplace's method for sums and integrals and the bootstrapping techniques [DB58, Ben74, GK82].

1.3 Bessel Functions.

The aim of this section is to summarize some well known facts about Bessel functions of the first kind.

An extensive treatment of Bessel functions is *A Treatise on the Theory of Bessel Functions* of G.N. Watson [Wat44]. Another good source of summarized information about Bessel functions is Abramowitz and Stegun's *Handbook of Mathematical Functions* [AS64].

Besides the classical trigonometric, exponential, logarithmic and hyperbolic functions, *Bessel functions* are among the most important transcendental functions. The ν-th order Bessel function of the first kind, denoted by $J_\nu(z)$ is a solution of the following second-order linear differential equation:

$$z^2\frac{d^2w}{dz^2} + z\frac{dw}{dz} + (z^2-\nu^2)w = 0.$$

Each Bessel function of the first kind is an entire function when $\nu = 0, \pm1, \pm2, \dots$. Another solution to the differential equation above is $Y_\nu(z)$, the ν-th order Bessel function of the second kind. The linear combination of J_ν and Y_ν gives a fundamental system of solutions of the differential equation. Such a linear combination is called a *cylinder function.*

In our applications we shall deal mainly with J_0, and therefore we will particularize the few identities we need to the case $\nu = 0$. First of all, $J_0(z)$ admits the following Taylor series development around $z = 0$, which is valid in the whole complex plane:

$$J_0(z) = \sum_{n\geq 0}(-1)^n\frac{(z/2)^{2n}}{n!^2}. \tag{1.4}$$

Also, we have the following asymptotic expansion of $J_0(z)$ for large z. If $|z| \to \infty$ and $|\arg z| \leq \pi$ then

$$J_0(z) = \sqrt{\frac{2}{\pi z}}\left[\cos(z-\frac{\pi}{4}) + \exp(|\Im z|)O(|z|^{-1})\right]. \tag{1.5}$$

Another useful identity is the following integral representation of $J_0(z)$:

$$J_0(z) = \frac{1}{\pi}\int_0^\pi \exp(\mathrm{i}z\cos\theta)\,d\theta.$$

Chapter 2

Simple Families of Trees

This chapter covers the definition of *simple families of trees*, also known as *simply generated families of trees*. A formal definition and the introduction of some notation and concepts is inavoidable, as we concentrate on characteristics of trees and probabilistic models for trees.

The second section of the chapter gives a short introduction to the topic of enumeration of simple families of trees.

2.1 The Definition of Simple Families of Trees.

Let us recall that a *rooted tree* is a finite set of one or more elements, called nodes, such that there is a distinguished element, called the *root* of the tree, and the other elements can be partitioned in a finite number of disjoint subsets (the number of subsets can be 0), each one being a tree and called *subtrees* of the root node. A rooted tree is said to be an *ordered tree* if the relative order of the subtrees of each node counts [Knu68].

Trees are a very important data structure in computer science. They also appear implicitly in recursive schemes such as backtracking and in general, in procedures with multiple recursive calls.

The trees considered in computer science are almost always ordered trees; if the trees are explicitly built for representation of expressions, to store records, etc. the implementation imposes some order among the subtrees of each node; if they correspond to a sequence of recursive calls, these come, in some predictable way, one after another. Moreover, when trees are used as data structures their nodes store information. That information could be represented as labels on the nodes. For this reason, we shall consider both labelled and unlabelled trees.

For sources of information on the implementations and applications of tree structures, we address the reader to the books of Knuth [Knu68, Knu73], Aho, Hopcroft and Ullmann [AHU76], and Gonnet and Baeza-Yates [GB91], among others.

Given an ordered tree[1] T we will denote by root(T) the root of T, and by $T_1,\ldots,T_k$ each of its subtrees. Alternatively, we will use the Pascalish notation $T[1],\ldots,T[k]$ for the same purpose. A node with k subtrees is said to be of *degree* or *arity* k; nodes of degree 0 are called *leaves* or *external nodes*, whereas nodes of degree > 0 are called *internal nodes*. Given a tree T, deg(T) will denote the degree of its root.

Extended binary trees are non-empty finite sets of nodes where one of the nodes is the root and either there are no more nodes or they can be partitioned in two non-empty sets of nodes, each set of nodes being an extended binary tree. This definition is just a particularization of the general definition of trees to the case where each node has either exactly two subtrees or none. *Extended m-ary search trees* are analogously defined. Since an immediate isomorphism between binary trees and extended binary trees can be defined, we will omit the adjective *extended*, from now on.

The definition of simple families of trees implies that the size of any tree in those families is the number of nodes it contains. However, we will define the size of binary and m-ary trees to be the number of internal nodes in the tree, considering that leaves have null size. This is a very common convention, since for these families the total number of nodes in a tree and the number of internal nodes are mutually dependent. We will use the symbols $\mathcal{B}$ and $\mathcal{T}_m$ to denote the families of binary and m-ary trees, respectively.

Given a binary tree T that is not a leaf we call the subtrees of its root, the *left* and *right* subtree, and denote them by T^l and T^r, respectively.

The notion of *simply generated family of trees* or *simple family of trees* was introduced by Meir and Moon [MM78]. We shall adhere to the slight modification of the concept and notational conventions used in Steyaert's Ph.D. Thesis [Ste84].

Let S be a set of symbols and $\nu : S \longrightarrow \mathbb{N}$ an arity application defined on S.

[1]From now on, tree will mean ordered tree, unless otherwise stated.

Let $s(T_1, \dots, T_{\nu(s)})$ be the tree whose root is labelled s and its subtrees are denoted left-to-right : $T_1, \dots, T_{\nu(s)}$. The size of a tree $T = s(T_1, T_2, \dots, T_k)$ is

$$|T| = 1 + |T_1| + |T_2| + \cdots + |T_k|.$$

Definition 2.1. *The set of trees $\mathcal{F}$ defined recursively by*

$$\begin{cases} s \in \mathcal{F} & \text{if } \nu(s) = 0, \\ s(T_1, \dots, T_{\nu(s)}) \in \mathcal{F} & \text{if } T_1, \dots, T_{\nu(s)} \in \mathcal{F}. \end{cases}$$

is said to be a simple family of trees, generated by S, *if*

$$\exists M \in \mathbb{N} \ s.t. \ \forall n \quad |\nu^{-1}(n)| \leq M, \tag{2.1}$$

where $\nu^{-1}(n)$ denotes the set of symbols of arity *n.*

If we denote the set $\{s(T_1, \dots, T_k) \mid s \in \nu^{-1}(k) \wedge T_1, \dots, T_k \in \mathcal{F}\}$ by $S_k(\mathcal{F}, \mathcal{F}, \dots, \mathcal{F})$ then we can express the family $\mathcal{F}$ of trees, generated by S, in the following handy way:

$$\mathcal{F} = S_0 + S_1(\mathcal{F}) + S_2(\mathcal{F}, \mathcal{F}) + \cdots + S_j \underbrace{(\mathcal{F}, \dots, \mathcal{F})}_{j} + \cdots$$

This representation of simple families of trees is called *formal series of trees.*

2.2 Enumeration of Simple Families of Trees.

The classical characterization of simple families of trees is given in the next proposition.

Proposition 2.1. *The counting generating function $F(z) = \sum_{T \in \mathcal{F}} z^{|T|}$ of any simple family of trees $\mathcal{F}$ satisfies a functional relation of the type*

$$F(z) = z\phi(F(z)),$$

where $\phi(u)$ is a power series in u.

Condition (2.1) guarantees that the number of trees of size n in any simple family of trees is finite. Let $\phi_n = |\nu^{-1}(n)|$ be the number of symbols in S of arity n. The *characteristic series* of the family of trees generated by S is defined by

$$\phi(u) = \sum_{n \geq 0} \phi_n \, u^n.$$

Using Definition 2.1, it is straightforward to check that the counting generating function of the set $\mathcal{F}$, $F(z)$, satisfies $F(z) = z\phi(F(z))$.

Since we are now interested in the enumeration of simple families of trees, Lagrange inversion formula can prove to be useful. Fpr example, consider the family $\mathcal{B}$ of binary trees, generated by $S = \{\circ, \Box\}$, with $\nu(\circ) = 2$ and $\nu(\Box) = 0$. The characteristic series of binary trees is $\phi(u) = 1 + u^2$ and if we call $B(z)$ the counting generating function of $\mathcal{B}$ we have

$$B(z) = z(1 + B^2(z)).$$

Notice that in this derivation, we define the size of a tree as the total number of nodes in the tree. Using Lagrange inversion formula yields

$$[z^n]B(z) = \frac{1}{n}[u^{n-1}](1+u^2)^n,$$

and from Newton's binomial theorem

$$[z^n]B(z) = \begin{cases} \frac{1}{2k+1}\binom{2k+1}{k}, & \text{if } n = 2k+1, \\ 0, & \text{otherwise.} \end{cases}$$

The number of nodes n in a binary tree is always odd and, if $n = 2k+1$ then k is the number of internal nodes of the tree. Therefore, the number of binary trees with n internal nodes, b_n, is given by:

$$b_n = \frac{\binom{2n}{n}}{n+1}.$$

The numbers b_n are called *Catalan numbers* after the French mathematician Eugène Catalan, who wrote an influential paper about them in 1838.

It was easy to solve the enumeration problem for the family of binary trees from its statement to the final answer, but for many other simple families of trees the way is not so easy or the answer is not in a convenient "closed form". However, we are still able to compute an asymptotic estimation of the number f_n of trees of size n for any simple family $\mathcal{F}$.

The following theorem, due to Meir and Moon, gives the answer [MM78].

Theorem 2.1. (Meir, Moon) *Let $\phi(u)$ be an analytic function in the disk $|u| < \rho \leq \infty$ and let $F(z)$ denote the solution of $F(z) = z\phi(F(z))$ in a vicinity of $z = 0$. For simplicity, assume that $\gcd\{\, n \mid \phi_n \neq 0 \,\} = 1$. Then, $[z^n]F(z)$ is asymptotically equivalent to*

$$f_n = [z^n]F(z) \sim \alpha\rho^{-n}n^{-3/2}\left(1 + O\left(\frac{1}{n}\right)\right),$$

where

$$\alpha = \sqrt{\frac{\phi(\tau)}{2\pi\phi''(\tau)}}, \qquad \rho = \frac{\tau}{\phi(\tau)}$$

and τ is the root of smallest modulus of the equation $\phi(u) - u\phi'(u) = 0$.

The proof of this theorem combines the implicit function theorem and Darboux's theorem [MM78]. Similar enumerating results have been obtained by other authors (see, for example, the survey of Bender [Ben74], where a collection of such results appears).

Chapter 3

The BST and the Balanced Probability Models

This chapter deals with the probability models that we will assume for the average-case analysis of algorithms over trees.

The first probability model we discuss is the *Binary Search Tree probability model*, BST probability model for short. Later in this chapter, we present the *balanced probability model*, a generalization of the BST model to arbitrary simple families of trees.

We shall see that the BST probability model tends to assign high probability to well balanced trees, while it assigns low probability to poorly balanced trees of the same size. This is a characteristic that is preserved in the extension of the model. An important characteristic of the model is the fact that the corresponding average-case analysis yield first-order ordinary differential equations. This fact contrasts with the algebraic equations obtained when the analysis is done over a uniform distribution of the inputs.

When the BST and the balanced models are extended to pairs of trees, the characteristic equations change from ordinary to partial differential equations in two variables. This change implies a modification of the mathematical techniques needed to complete the average-case analysis.

The last section of this chapter presents other two probability models, the *mST model* and the *digital search tree model*.

We expect to give in this chapter a brief overview of several recursion models that appear around tree data structures, and to illustrate our point on the interest that can have the study of the relationships between probability models, their mathematical analysis and the dependences between algorithmic schemes and probability models.

3.1 The BST Probability Model.

The BST probability model is the one corresponding to randomly built *binary search trees*. Binary search trees and their balanced variants are mainly used to implement dictionary operations, i.e. insertions, deletions and queries of items in a set of items [Knu73, AHU76, Sed88, CLR90, GB91]. A great amount of work has been done on statistics for binary search trees; most of this work relates to the average-case analysis of algorithms associated with the manipulation of this particular data structure. Some other works use the BST model for computing characteristics of binary trees. Devroye proved that the average height of binary trees under the BST model is asymptotically $O(\log\ n)$ [Dev86]. In his proof, he used techniques rather different from the ones we use in this work. This expected value of the height differs from the average height of binary trees under the uniform model, which is $O(\sqrt{n})$ [FO82]. Other known result is about the internal path length of trees. The average internal path length for binary search trees of size n is $O(n \log n)$ and for uniformly distributed trees is $O(n\sqrt{n})$. The differences lie on the fact that the BST model assigns high probability to the more balanced binary trees, and relatively low probability to the skewer trees of the same size.

There are other data structures for which the BST probability model applies as well : heap-ordered trees [Fla88b], used for the implementation of priority queues, and k-d-trees [Ben75, BF79, FP86], for range multidimensional search.

Recall that a binary search tree is a data structure that consists in a binary tree whose nodes are labeled in increasing order from left to right. Binary search trees have the following recursive definition [VF90]:

Definition 3.1. *Given a sequence of n keys $S = (k_1, k_2, \ldots, k_n)$ where the keys belong to a totally ordered set, we define recursively the* binary search tree *of S as,*

$$BST(S) = \begin{cases} k_1(BST(S_l), BST(S_g)), & \text{if } |S| \geq 1, \\ \square, & \text{otherwise,} \end{cases}$$

where $|S|$ denotes the number of keys in S, S_l and S_g denote the subsequences of S formed respectively by the elements of S which are less than k_1 and greater than k_1, and $\square$ denotes *the empty binary tree.*

In the model of probability associated with binary search trees, each sequence S is obtained by consecutively sampling at random n elements from a real interval, or equivalently, as far as only relative ordering concerns, the elements form a random permutation of size n. In other words, we consider that the sequences are permutations of the set $\{1, \ldots, n\}$ and that all sequences of size n have the same probability $1/n!$.

Let $N(T)$ denote the number of sequences S of size n that generate the same binary search tree $T = BST(S)$. It has been shown by Knuth [Knu73, exercise 5 of 6.2.2] that we can compute $N(T)$ from the following recursive equation:

$$N(T) = \begin{cases} 1, & \text{if } T = \square \\ N(T^l) \cdot N(T^r) \cdot \dfrac{(|T|-1)!}{|T^l|! \cdot |T^r|!}, & \text{otherwise.} \end{cases}$$

If we denote by $\Pr(T)$ the probability of tree T, then $\Pr(T) = N(T)/|T|!$ and we have the following recursive definition of the BST probability model:

Definition 3.2. *The probability* $\Pr(T)$ *of a binary tree* T *in the BST probability model is*

$$\Pr(T) = \begin{cases} 1, & \textit{if } T = \square \\ \dfrac{\Pr(T^l) \cdot \Pr(T^r)}{1 + |T^l| + |T^r|}, & \textit{otherwise.} \end{cases}$$

The recursive manner in which we express this probability model is very handy to simplify some proofs about average behaviour of binary search trees.

For example, consider the *internal path lenght* of binary trees. The internal path lenght (IPL, for short) of a tree is the sum of the lenghts of the paths from the root to each internal node. This quantity is directly related to the performance of insertions, successful and unsuccessful queries, etc. If we denote $\mathrm{ipl}(T)$ the internal path lenght of the tree T we have

$$\mathrm{ipl}(T) = \begin{cases} \mathrm{ipl}(T^l) + \mathrm{ipl}(T^r) + |T| - 1, & \text{if } T \neq \square, \\ 0, & \text{otherwise.} \end{cases}$$

In order to analyze the average value of the IPL for trees of size n, say $\overline{ipl(n)}$, we introduce the generating function

$$\mathrm{Ipl}(z) = \sum_{T \in \mathcal{B}} \mathrm{ipl}(T)\, \Pr(T)\, z^{|T|},$$

so $[z^n]\mathrm{Ipl}(z) = \overline{ipl(n)}$.

Using the recursive decomposition of the characteristic, of the probability model and of the binary trees we get

$$\begin{aligned}
\mathrm{Ipl}(z) &= \sum_{T\in\mathcal{B}-\square} \left(\mathrm{ipl}(T^l)+\mathrm{ipl}(T^r)+|T^l|+|T^r|\right)\ \mathrm{Pr}(T)\, z^{|T|} = \\
&= \sum_{T\in\mathcal{B}-\square} \left(\mathrm{ipl}(T^l)+\mathrm{ipl}(T^r)+|T^l|+|T^r|\right)\ \frac{\mathrm{Pr}(T^l)\,\mathrm{Pr}(T^r)}{|T^l|+|T^r|+1}\, z^{|T^l|+|T^r|+1}.
\end{aligned}$$

By symmetry,

$$\mathrm{Ipl}(z) = 2\sum_{T\in\mathcal{B}-\square} \left(\mathrm{ipl}(T^l)+|T^l|\right)\ \frac{\mathrm{Pr}(T^l)\,\mathrm{Pr}(T^r)}{|T^l|+|T^r|+1}\, z^{|T^l|+|T^r|+1}.$$

Differentiating with respect to z,

$$\begin{aligned}
\frac{d}{dz}\mathrm{Ipl}(z) &= 2\sum_{T\in\mathcal{B}-\square} \left(\mathrm{ipl}(T^l)+|T^l|\right)\ \mathrm{Pr}(T^l)\,\mathrm{Pr}(T^r)\, z^{|T^l|+|T^r|} = \\
&= 2\sum_{T\in\mathcal{B}-\square} \mathrm{ipl}(T^l)\ \mathrm{Pr}(T^l)\,\mathrm{Pr}(T^r)\, z^{|T^l|+|T^r|} + 2\sum_{T\in\mathcal{B}-\square} |T^l|\ \mathrm{Pr}(T^l)\,\mathrm{Pr}(T^r)\, z^{|T^l|+|T^r|} = \\
&= 2\sum_{T_1,T_2\in\mathcal{B}} \mathrm{ipl}(T_1)\ \mathrm{Pr}(T_1)\,\mathrm{Pr}(T_2)\, z^{|T_1|+|T_2|} + 2\sum_{T_1,T_2\in\mathcal{B}} |T_1|\ \mathrm{Pr}(T_1)\,\mathrm{Pr}(T_2)\, z^{|T_1|+|T_2|}.
\end{aligned}$$

Since T_1 and T_2 are independent in the summations above, and $\sum_{T\in\mathcal{B}}\mathrm{Pr}(T)\, z^{|T|} = (1-z)^{-1}$, one has

$$\sum_{T_1,T_2\in\mathcal{B}} \mathrm{ipl}(T_1)\ \mathrm{Pr}(T_1)\,\mathrm{Pr}(T_2)\, z^{|T_1|+|T_2|} = \mathrm{Ipl}(z)\cdot\frac{1}{1-z}.$$

The generating function $\sum_{T\in\mathcal{B}}|T|\ \mathrm{Pr}(T)\, z^{|T|-1}$ is the formal derivative of $\sum_{T\in\mathcal{B}}\mathrm{Pr}(T)\, z^{|T|}$. Hence,

$$\sum_{T_1,T_2\in\mathcal{B}} |T_1|\ \mathrm{Pr}(T_1)\,\mathrm{Pr}(T_2)\, z^{|T_1|+|T_2|} = z\frac{d}{dz}\left(\frac{1}{1-z}\right)\cdot\frac{1}{1-z} = \frac{z}{(1-z)^3}.$$

Finally, we have

$$\frac{d}{dz}\mathrm{Ipl}(z) = \frac{2\mathrm{Ipl}(z)}{1-z} + \frac{2z}{(1-z)^3}, \qquad \mathrm{Ipl}(0) = 0.$$

Solving this differential equation and extracting the n-th coefficient of $\mathrm{Ipl}(z)$ yields the known result [Knu73]

$$\overline{ipl(n)} = 2(nH_n - 2n + H_n),$$

where H_n denotes the n-th harmonic number.

A useful way to view the BST model is as an "urn" model: Suppose we have defined the BST distribution for binary trees of sizes 0 to $n-1$. In order to construct a binary tree T with n internal nodes, select the size of its left subtree, say i, from $0, \ldots, n-1$, at random. Then pick a tree T_1 of size i with probability $\Pr(T_1)$, pick another tree T_2 of size $n-1-i$ with probability $\Pr(T_2)$ and set T_1 and T_2 as the left and right subtrees of the tree T. This protocol , also gives a recursive definition of the BST distribution equivalent to Definition 3.2, and it is equivalent to the classical characterization of random binary search trees, which states that the size of the left subtree (or of the right subtree) of a tree of size n is a random discrete variable X taking values in the range $[0, \ldots, n-1]$ and such that $\Pr(X = i) = 1/n, \quad i = 0, \ldots, n-1$.

More details on the average-case analysis for this model will be given in Sections 3.3 and 3.4 of this chapter. In Chapter 4 we analyze the average behavior of the occupancy for the balanced probability model; the analysis is a just generalization of what we would do for the BST model.

3.2 Extension of the BST Model to Pairs of Trees.

There are many algorithms that operate over pairs of trees (for example, equality, unification, tree matching, etc.) and therefore, we need to extend the BST probability model to pairs of binary trees, in order to analyze such algorithms.

This could be done in several reasonable ways, but we have tried to do such extension in the simplest fashion and to keep most of the BST model attributes.

One important attribute of the BST probability model (and of other models) is that the probability of a tree can be recursively defined in terms of the probabilities of its subtrees and some factor involving their sizes. Our extensions should mantain this property; for instance, the probability of a pair should be defined in terms of the probabilities of its components.

In order to define the extension of the BST probability model to pairs, we confront two situations:

- If each of the trees is drawn *independently* one of the other, the probability of the pair is just

$$\Pr_{\text{indp}}(T_1, T_2) = \Pr(T_1) \cdot \Pr(T_2).$$

Notice that in this case for all n and m,

$$\sum_{|T_1|=n,|T_2|=m} \Pr_{\mathrm{indp}}(T_1, T_2) = 1.$$

- If we consider pairs of binary trees (T_1, T_2) such that $|T_1| + |T_2| = n$, where n is the size of the input and assume that all the $n+1$ possible partitions of n into $n = i + j$ (being $|T_1| = i$ and $|T_2| = j$) are equally likely, we come up with a definition of a probability distribution over each of the subsets of pairs of binary trees of total size n. This probability model will be called *extended BST model*[1].

Definition 3.3. *The probability* $\Pr(T_1, T_2)$ *of the pair of binary trees* (T_1, T_2) *in the extended BST probability model is*

$$\Pr(T_1, T_2) = \frac{\Pr(T_1) \cdot \Pr(T_2)}{1 + |T_1| + |T_2|} = \Pr(\circ(T_1, T_2)),$$

where $\Pr(T)$ *denotes the probability of the single tree* T *in the BST model.*

Notice that the probability of a pair coincides with the probability of the tree formed by a root and whose subtrees are the components of the pair; a similar principle is used in the extension of the balanced model to Cartesian products of simple families of trees (see Section 3.3).

Also, the extended BST model is well defined since

$$\sum_{|T_1|+|T_2|=n} \Pr(T_1, T_2) = 1.$$

An inherent characteristic of the average-case analysis under the extended BST model is that the functional relations that appear are partial differential equations. Given a characteristic $f(T_1, T_2)$, we introduce the generating function $F(z)$

$$F(z) = \sum_{(T_1,T_2)\in\mathcal{B}^2} \Pr(T_1, T_2)\, f(T_1, T_2)\, z^{|T_1|+|T_2|} \tag{3.1}$$

to have that $[z^n]F(z)$ is

$$[z^n]F(z) = \sum_{|(T_1,T_2)|=n} \Pr(T_1, T_2)\, f(T_1, T_2).$$

[1]If there is no confusion, we speak also of *BST model* when is actually meant extended BST model.

Since the summation in (3.1) is to be splitted to obtain some functional relation satisfied by $F(z)$, we must introduce the bivariate generating function corresponding to the probability model for pairs whose members are independently drawn

$$F(x, y) = \sum_{(T_1,T_2)\in\mathcal{B}^2} \Pr(T_1)\ \Pr(T_2)\, f(T_1, T_2)\, x^{|T_1|}\, y^{|T_2|} \tag{3.2}$$

and then relate $F(z)$ and $F(x, y)$. Using the definition of the extended BST model, one has

$$F(z) = \frac{1}{z}\int_0^z F(t, t)\, dt.$$

The extension of the BST model to pairs of trees is the realm of partial differential equations in two variables. Since the "splitting" of the summation in (3.2) is done for both T_1 and T_2, differentiation with respect to x and y should be done and the resulting partial differential equations are of second order. Deriving asymptotic estimations of the coefficients of $[z^n]F(z)$ is a main subject of this thesis (see Chapters 7 and 8). The first stage of the analysis under the BST model, where the recursive decomposition of algorithms and data structures is exploited to derive equations describing the average behavior of the algorithms, is discussed in the next section, and in Chapters 5 and 6.

3.3 The Balanced Probability Model.

The balanced probability model was defined in an attempt to generalize the BST probability model to other families of trees (with nodes of arbitrary degree, labels, etc.). At the same time, we wished that the model kept most of the properties of the BST model. For instance, the balanced probability model admits a recursive splitting, i.e., the probability of a given non-empty tree can be expressed in terms of the probabilities of its subtrees. As in the BST model, well balanced trees are common in the balanced probability model, whereas poorly balanced trees and linear lists are rare.

In fact, the balanced model is not a concrete model, but a set of rules or principles that allow the systematic definition of a probability model over any given simple family of trees. Thus, it would we more appropriate to talk about "...the balanced model for the family $\mathcal{F}$..." or "...average-case analysis under a balanced probability model...". Nevertheless, we will use the name in a more informal way, just as we do with the term "uniform model".

Other models for families of trees with internal nodes of degree > 2 and that generalize binary search trees are the *quadtree model* and the *m-ary search tree model*. Quadtrees are specially useful for geometric data storage and range search, but the degree of internal nodes in quadtrees must be some power of 2. The m-ary search trees were defined as efficient data structures for the dictionary problem using external storage and to improve balancing, by putting $m-1$ keys per node (see Subsection 3.5.1). The probability models for these two generalizations do not coincide with the balanced probability model for the corresponding families. It should be emphasized that average-case analysis under the balanced probability model is easier to do than under any of the former models; and there is not an increase of the difficulty with respect to the analysis with the BST probability model. The same set of techniques that are used to carry out the analysis under the BST model can be used for the analysis under the balanced model.

The balanced probability model is also closely related to the so-called *increasing trees* [BFS92]; increasing trees are the generalization of the idea of heap ordered trees to trees with nodes of arbitrary degree.

A rather curious phenomenon is that the balanced model does not depend neither on the labels of the nodes nor on the number of distinct labels given to nodes of the same degree.

In order to define the balanced probability model over a simple family of trees $\mathcal{F}$, generated by the symbol set S, we assume that we are given probability distributions over each subset of symbols of the same arity. Let $p(s)$ be the probability of the symbol $s \in S$. We impose that

$$\forall k \geq 0 \quad \sum_{s \in \nu^{-1}(k)} p(s) = 1, \qquad \text{if } \nu^{-1}(k) \neq \emptyset.$$

We are now ready to define the balanced probability model over $\mathcal{F}$, by firstly introducing a weight measure w over $\mathcal{F}$.

Definition 3.4. *The weight measure $w(T)$ of a tree $T \in \mathcal{F}$ is recursively defined by*

$$w(T) = \begin{cases} p(s), & \text{if } T = s \text{ and } \nu(s) = 0, \\ p(s) \cdot w(T_1) \cdots w(T_k) \cdot \frac{1}{|T|}, & \text{if } T = s(T_1, \ldots, T_k) \text{ and } \nu(s) = k. \end{cases}$$

And in order to obtain a probability model over $\mathcal{F}$, we define the probability of a tree T as follows:

Definition 3.5. *The probability* $\Pr(T)$ *of a tree* T *in the balanced model is*

$$\Pr(T) = \frac{w(T)}{\sum_{|t|=|T|} w(t)}.$$

It can be readily verified that $\Pr(\cdot)$ satisfies Kolmogorov's axioms. Notice that $\Pr(T)$ varies depending on how balanced the tree is, being maximum for complete trees and minimum for linear list-like trees (see Figure 3.1).

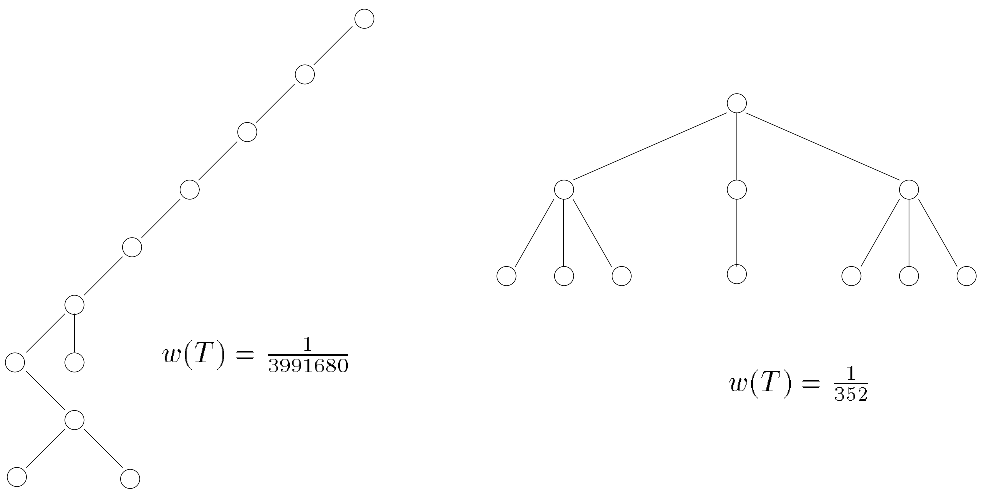


Figure 3.1: *Two particular examples of the weight measure for ternary trees of the same size (leaves have been omitted).*

The definitions above can be extended to k-tuples of trees from a simple family $\mathcal{F}$.

Definition 3.6. *The weight* $w(T_1, T_2, \ldots, T_k)$ *of a* k*-tuple of trees* $T_1,\ldots,T_k$ *in* $\mathcal{F}$ *is*

$$w(T_1, \ldots, T_k) = w(T_1) \cdots w(T_k).$$

The probability measure over k-tuples $(T_1, \ldots, T_k)$ of size $|T_1| + \cdots + |T_k| = n$ is defined in the same way it has been done before:

Definition 3.7. *The probability* $\Pr(T_1, T_2, \ldots, T_k)$ *of a* k*-tuple of trees* $T_1, T_2, \ldots, T_k$ *in* $\mathcal{F}$ *is*

$$\Pr(T_1, \ldots, T_k) = \frac{w(T_1, \ldots, T_k)}{\sum_{|t_1|+\cdots+|t_k|=n} w(t_1, \ldots, t_k)}.$$

Also, we can define a probability model over the Cartesian product $\mathcal{F}_1 \times \cdots \times \mathcal{F}_k$, for $\mathcal{F}_1, \ldots, \mathcal{F}_k$ simple families of trees, in the same way.

Given a family $\mathcal{F}$ let its *weight characteristic series* be

$$\phi(u) = \sum_{s \in S} p(s) \cdot u^{\nu(s)} = \sum_{n \in \mathrm{Im}\nu} u^n.$$

A given degree n belongs to the set $\mathrm{Im}\nu$ if there is a least a symbol s in S such that $\nu(s) = n$.

Let $W(z) = \sum_{T \in \mathcal{F}} w(T)\, z^{|T|}$. Then the n-th coefficient of $W(z)$ is the normalizing constant needed to get the probability distribution for trees of size n (see Definition 3.5). We will call $W(z)$ the *weight generating function* of the family $\mathcal{F}$.

Using Definitions 2.1 and 3.4 and depending on the convention for the size of a leaf, we get the following functional relations for $W(z)$.

If we impose $|s| = 0$ for all 0-ary symbols s, we get,

$$\frac{dW}{dz} = \phi(W) - 1, \qquad W(0) = 1,$$

whereas if $|s| = 1$ for $s \in \nu^{-1}(0)$, then $W(z)$ satisfies,

$$\frac{dW}{dz} = \phi(W), \qquad W(0) = 0.$$

Note that the weight characteristic series $\phi(u)$ does not depend on the symbols of S. Moreover, it does not depend on the number of symbols of a given arity, but on the existence or not of symbols of the given arity, and consequently, $W(z)$ has the same properties.

Some interesting characteristic series and weight generating functions are given in Table 3.1, where in the three first families we have assumed that the size of a tree is the number of internal nodes (leaves have null size) and in the last two families each node, of whatever arity, contributes to the total size.

In the particular case of binary trees, we have that $\Pr(T) = w(T)$, since $[z^n]W(z) = 1$ for all $n \geq 0$. Therefore, the definitions of $\Pr(T)$ in the BST model and the balanced model coincide, as claimed.

Notice also that for pairs of binary trees, Definition 3.7 implies

$$\Pr(T_1, T_2) = \frac{\Pr(T_1) \cdot \Pr(T_2)}{|T_1| + |T_2| + 1} = \Pr(\circ(T_1, T_2)),$$

so the extension of the BST model to pairs of binary trees is also a particular case of the balanced model.

Family	$\phi(u)$	$W(z)$
Linear lists (unary trees)	$1+u$	e^z
Binary trees ($\mathcal{B}$)	$1+u^2$	$\frac{1}{1-z}$
m-ary trees, $m>1$ ($\mathcal{T}_m$)	$1+u^m$	$(1-(m-1)z)^{-1/(m-1)}$
Motzkin trees (unary-binary trees)	$1+u+u^2$	$\frac{\sqrt{3}}{2}\tan\left(\frac{\sqrt{3}}{2}z+\frac{\pi}{6}\right)-\frac{1}{2}$
General trees	$\frac{1}{1-u}$	$1-\sqrt{1-2z}$

Table 3.1: *Weight characteristic series and generating functions.*

3.4 Average-case Analysis under the Balanced Model.

When the characteristic under study is an *inductive valuation* and we want to carry out the analysis of that characteristic for the balanced model, the functional relations over generating functions are ordinary linear differential equation, as for the BST model.

Any inductive valuation $f(T)$ can be recursively expressed as

$$f(T) = \sum_{t \subset T} f(t) + m(|T|),$$

where $t \subset T$ denotes that t is a subtree of the root of T, and $m(n)$ is any numerical function whose domain are the positive integers.

The steps to be followed are the same as for the BST model. We introduce a generating function for the sequence of expected values of the characteristic, and translate the relationships between the expected values to a functional equation for this generating function. Differentiation must be done because we ought to get rid of the $|T|^{-1}$ factor in the recursive definition of $w(T)$ so as to relate the generating function with itself.

Let

$$F(z) = \sum_{T \in \mathcal{F}} w(T)\, f(T)\, z^{|T|}.$$

Then, the expected value of $f(T)$ over trees of size n is

$$\overline{f(n)} = \frac{[z^n]F(z)}{[z^n]W(z)}.$$

If the size measure over the family $\mathcal{F}$ is defined assigning null size to 0-ary nodes, then

$$\begin{aligned}F(z) &= \sum_{s\in\nu^{-1}(0)} m(0)\,p(s) + \sum_{\substack{k>0\\k\in\mathrm{Im}\,\nu}} \sum_{s\in\nu^{-1}(k)} \sum_{T_1,\ldots,T_k\in\mathcal{F}} (m(|T_1|+\cdots+1)+f(T_1)+\cdots+ \\ &+ f(T_k))\,p(s)\,w(T_1)\cdots w(T_k)\frac{z^{1+|T_1|+\cdots+|T_k|}}{1+|T_1|+\cdots+|T_k|}.\end{aligned}$$

Differentiating the formula above with respect to z, some additional manipulations yield

$$\begin{aligned}\frac{dF}{dz} = \sum_{\substack{k>0\\k\in\mathrm{Im}\,\nu}} &\left[\sum_{T_1,\ldots,T_k\in\mathcal{F}} w(T_1)\cdots w(T_k)m(\cdots)\,z^{|T_1|+\cdots+|T_k|} + k\,F(z)\,W^{k-1}(z)\right] = \\ &= \left(\sum_{n\geq 0} m(n+1)\,z^n \sum_{\substack{k>0\\k\in\mathrm{Im}\,\nu}} \sum_{|T_1|+\cdots+|T_k|=n} w(T_1)\cdots w(T_k)\right) + F(z)\,\phi'(W(z)) = \\ &= \left(\sum_{n\geq 0} m(n+1)\,z^n\,[z^n](\phi(W(z))-1)\right) + F(z)\,\phi'(W(z)) = \\ &= \left(\sum_{n\geq 0} m(n+1)\,z^n\,[z^n]W'(z)\right) + F(z)\,\phi'(W(z)) = \\ &= \frac{dM}{dz} + F(z)\,\phi'(W(z)) \qquad (3.3)\end{aligned}$$

if we define $M(z) = \sum_{n\geq 0} m(n)\,w_n\,z^n$ and $w_n = [z^n]W(z)$. The initial condition for the differential equation is $L(0) = M(0)$.

If the balanced model is defined over a family where the size of a tree is the total number of nodes (leaves and internal nodes), the differential equation relating the generating function $F(z)$ with itself and $M(z)$ and $W(z)$, is again the same and, with the same initial condition.

Finally, the average-case analysis over pairs of trees in the balanced model is conducted in the same way as for the BST model. For statistics over m-tuples of trees with $m > 2$, a generating function on m variables ought to be defined and the functional relation over generating functions is, in general, a partial differential equation of the m-th order in m variables (see Chapter 5).

3.5 Other Probability Models.

In this section we consider other two interesting models: the *mST probability model* and the *digital search tree probability model* (DST model, for short).

The mST probability model is a generalization of the BST probability model to m-ary trees. Like the BST and the balanced model, it assigns high probability to well balanced trees. This model corresponds to *m-ary search trees* and as we will see has many peculiarities, since the size of such trees should be defined as the number of keys that a tree holds, not the number of nodes it has.

The DST probability model corresponds to randomly built *digital search trees*. While binary search trees and m-ary search trees are built by comparison of the keys on a given sequence, the construction of a DST relies on the decomposition of keys into their smaller units (for instance, the bit representation of the keys, if the keys are integers).

3.5.1 The mST Probability Model.

The m-ary search tree is a natural generalization of the binary search tree, where each node holds up to $m-1$ keys, and internal nodes have degree $m \geq 2$. The m-ary search trees were first introduced by Muntz and Uzgalis [MU71], and balanced variants of them were suggested thereafter.

The insertion on an m-ary search tree goes as follows: if the root node is not yet completely filled with $m-1$ keys, put the new key in the root, maintaining the increasing order between the keys already there. If the root has been completely filled the new key is inserted in the i-th subtree if it is greater than the key stored at the root at position $i-1$ and less than the key at position i. A new node is created if the subtree was empty. The procedure is applied recursively in this way until the new key gets inserted.

As for binary search trees, we can define m-ary search trees as follows :

Definition 3.8. *Let S be a sequence of n keys $S = (k_1, k_2, \dots, k_n)$ where the k_i belong to a totally ordered set. Furthermore, if $n \geq m-1$, let $S' = (k'_1, k'_2, \dots, k'_{m-1})$ be the ordered sequence of the first $m-1$ keys in S ($k'_1 < k'_2 < \cdots < k'_{m-1}$). Then the* m-ary search tree *of*

S is

$$mST(S) = \begin{cases} \boxed{\mathit{0}}, & \textit{if } n = 0 \\ \boxed{\mathit{1}}, & \textit{if } n = 1 \\ \boxed{\mathit{2}}, & \textit{if } n = 2 \\ \cdots & \\ \boxed{\text{m-}\mathit{1}}\,(mST(S_1), mST(S_2), \ldots, mST(S_m)), & \textit{if } n \geq m-1. \end{cases}$$

where the symbol $\boxed{i}$ represents a node holding the first $0 \leq i \leq m-1$ keys of S in increasing order, and S_j is the sequence of keys in S such that are greater than k'_{j-1} and less than k'_j (if $j = 1$ then S_1 contains the keys that are less than k'_1 and if $j = m$ then S_m contains the keys that are greater than k'_{m-1}).

This definition of the m-ary search trees is rather intricate but shows clearly the following fact : the m-ary search trees are isomorphic (in shape) to the simple family of trees generated by $S = \{\boxed{0}, \boxed{1}, \ldots, \boxed{\text{m-1}}\}$ (S denotes now the set of symbols generating the family), and where the arity function is $\nu(\boxed{\text{m-1}}) = m$ and $\nu = 0$ for all other symbols in S. Now, the symbols $\boxed{i}$ have no relation with keys. Also, the size measure over this family should be defined as the sum of the sizes of all the nodes, with $\left|\boxed{0}\right| = 0$, $\left|\boxed{1}\right| = 1, \ldots,$ $\left|\boxed{\text{m-1}}\right| = m-1$.

The common assumption for the definition of a probability model for m-ary search trees is that the $n!$ distinct relative orderings of the n keys stored at the tree are equally likely. Notice that the size of an m-ary search tree ought to be the number of keys in the tree, and not the number of nodes. Under the previous assumption it turns out that the probability of a tree of size $0 \leq i \leq m-1$ is 1 (there is only one m-ary tree of size i if $0 \leq i \leq m-1$), and if the tree has size $n \geq m-1$ then the probability that its i-th subtree has size n_i is

$$\Pr(|T_i| = n_i \,|\, |T| = n) = \binom{n}{m-1}^{-1}, \qquad 1 \leq i \leq m, \quad 0 \leq n_i \leq n-m+1, \quad m \geq 2$$

So the mST probability model can be defined in an analogous way to our definition of the BST probability model:

Definition 3.9. *The probability $\Pr(T)$ of a m-ary search tree in the mST probability model is*

$$\Pr(T) = \begin{cases} 1, & \textit{if } 0 \leq |T| \leq m-1, \\ \Pr(T_1) \cdots \Pr(T_m) \Big/ \binom{|T|}{m-1}, & \textit{if } T = \boxed{\text{m-}\mathit{1}}\,(T_1, \ldots, T_m). \end{cases}$$

The average-case analysis of algorithms over m-ary search trees introduces, in general, $(m-1)$-th order ordinary differential equations. Many interesting characteristics lead to linear differential equations and the corresponding homogeneous differential equation is an Euler equation whose solutions can be explicitly computed.

The methodology to follow in the first steps is the familiar one. A generating function of the characteristic $f(T)$ is introduced

$$F(z) = \sum_{T \in \mathcal{T}_m} f(T)\ \Pr(T)\, z^{|T|}$$

and then recurrences for $f(T)$ and Definition 3.9 are exploited to get a functional relation satisfied by $F(z)$.

One of the characteristics of importance in m-ary search trees is the number of nodes needed to host the n keys. Its expected value is

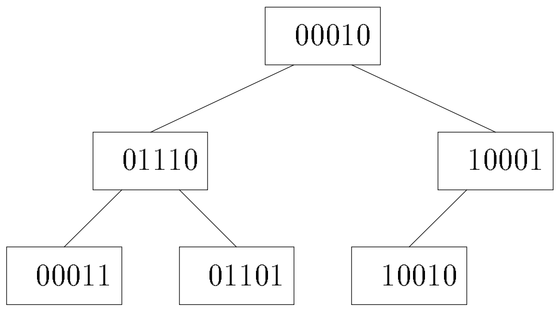
$$\frac{1}{2(H_m - 1)} n - \frac{1}{m-1} + O(n^{\alpha - 1})$$

where $\alpha = \alpha(m)$ is a rather complicated function that never exceeds 2, and is less than 1.5 for $m \le 26$; and H_m is the m-th harmonic number [Knu73, Bae87, MP89]. The average internal path length is $\sim 1/(H_m - 1) n \ln n$ [Mah86], and the average height tends in probability to $\gamma \ln n$, where $\gamma = \gamma(m)$ is a decreasing function that for large m approaches $1/\ln m$ [Dev90].

3.5.2 The DST Probability Model.

Digital search trees (DSTs for short) are built based on the decomposition of the keys in symbols and not on the relative order between them. For simplicity, assume that the keys consist in bit strings. Therefore, it makes sense to refer to the j-th bit of a key numbering the bits from left-to-right. Each node holds a key, and the keys stored at the left subtree of any node share a common prefix with that node followed by 0, whereas the keys at the right subtree share the same common prefix followed by 1. The length of this prefix is the level at which the node appears (see Figure 3.2).

When a key is to be retrieved or the spot to insert a new key is needed, the corresponding algorithm follows a path, branching to left or to right when at level $j-1$, according as whether the j-th bit is 0 or 1. In many implementations, it is convenient to require all keys to be of the same length; but the method also works for varying length keys, if none of the keys is a proper prefix of another. DSTs were firstly introduced by Coffman and Eve [CE70].



$$S = \{\, 00010, 01110, 01101, 10001, 00011, 10010 \,\}$$

Figure 3.2: *A digital search tree.*

Additional references for both the implementation and performance analysis are the books of Knuth [Knu73], Gonnet and Baeza-Yates [GB91] and Mahmoud [Mah92], and the survey by Flajolet and Sedgewick [FS86].

Before discussing the appropriate random model for DSTs, it should be emphasized that the order of insertion of the keys is relevant.

The simplest model for analyzing these data structures is the Bernoulli model (from now on, called DST model). That model assumes that keys are infinite random strings of bits, independently generated. This is equivalent to consider that the keys are real numbers independently sampled from the unit interval. Therefore, the probability that the left subtree of a tree with n nodes has size n_1 (and the right subtree $n-1-n_1$ nodes) is

$$\frac{1}{2^{n-1}}\binom{n-1}{n_1},$$

the probability that n_1 of the $n-1$ keys to be inserted start with a 0 and thus get inserted at the left subtree [FS86].

Definition 3.10. *The probability* $\Pr(T)$ *of a binary tree* T *in the DST probability model is*

$$\Pr(T) = \begin{cases} 1, & \text{if } T = \square \\ \dfrac{\Pr(T^l)\cdot\Pr(T^r)}{2^{|T|-1}}\dbinom{|T^l|+|T^r|}{|T^l|}, & \text{otherwise.} \end{cases}$$

Some significative results for this model concern the average number of comparisons in successful and unsuccessful search in a DST of size n (both are $\sim log_2 n + O(1)$ [KN73, Knu73, FS86]); and the average height of a DST ($\sim log_2 n$ [Pit85]).

A data structure that is closely related to DSTs, is the *trie* or *radix search trie*. This data structure has two types of nodes; the internal nodes contain pointers to other nodes, while the external nodes do actually store the keys. In contrast to DSTs, the shape of a trie does not depend on the order of insertion of the keys, but a trie storing n keys can have more than n internal nodes. There is a variant of the trie, called *Patricia tree*, where exactly $n-1$ internal nodes (and n external ones) are needed to store n keys. Although we will not define their corresponding probability models, we remark that there are many resemblances with the DSTs. The average-case analysis is performed for these data structures in a similar way.

The average-case analysis in the DST model has some significative differences with respect to the previously considered models, and leads to difference-differential equations.

In the average-case analysis under the DST model, we use generating functions, but they are exponential g.f.'s instead of ordinary ones. For instance,

$$V(z) = \sum_{T \in \mathcal{B}} v(T) \Pr(T) \frac{z^{|T|}}{|T|!}$$

where $v(T)$ is the measure we are interested in.

If $v(T)$ is an inductive valuation, the functional relation for $V(z)$ is

$$V'(z) = 2V(z/2) + U(z),$$

using the model given by Definition 3.10 and the definition of inductive valuation, and where $U(z)$ is a generating function corresponding to the term that depends only on the size in the definition of the inductive valuation. The next step considers $Y(z) = e^{-z}V(z)$ and $X(z) = e^{-z}U(z)$; translating the equation above, one gets

$$Y'(z) + Y(z) = 2Y(z/2) + X'(z) + X(z),$$

that corresponds to a non-linear recurrence for the Y_n. The relation between the coefficients $v_n = [z^n/n!]V(z)$ and $y_n = [z^n/n!]Y(z)$ is given by

$$v_n = \sum_k \binom{n}{k} y_k \tag{3.4}$$

There are many instances were y_n is explicitly known or there is a good estimation, and relates to the quantities

$$Q_n = \prod_{1 \le i \le n} (1 - 2^{-i}),$$

that arise in the theory of partitions. Classical identities (for instance, Euler's formulæ) from this theory, Rice's method and the Mellin transform techniques are then used to evaluate asymptotically (3.4).

Part III

Some Elementary Statistics over Trees Using the BST Probability Model

Part III includes most of the contributions and results of this work. Chapter 4 introduces *occupancy*, a very simple characteristic over m-ary trees. The occupancy of a tree is closely related with its degree of balancing. We analyze its average behavior under the uniform model and the probability models described in Chapter 3. This yields a basis for the comparison of these probability models and allows us to point out some of the distinctive features of each model. Moreover, the analyses provide an example of the application of some of the techniques discussed in Part II, and show the nature of the problems and the available tools associated to the analysis under the different probability models. In this way, we fix some of the ideas already given in Chapter 3.

Our approach to the study of the BST model begins with the application of the symbolic operator method to derive a set of algebraic techniques for the average-case analysis under the BST model. Following similar developments for the analysis under the uniform model, in Chapter 5 we propose a set of rules that allow the systematic translation of the algorithmic constructions used in recursive tree algorithms to equations over generating functions. These equations over generating functions describe the average behavior of the analyzed algorithm, when the input trees are distributed according to the balanced model.

The development of such a set of translation rules permits a better comprehension of the type of problems that arise in the second stage of the analysis, and the relationships with some interesting recursion schemes. For instance, the *simultaneous descent* construction over pairs of trees is one of the simplest non-trivial recursion schemes. Another basic recursion scheme that we have studied is based on the *conditional iteration* construction in combination with a recursion scheme satisfied by certain binary predicates that we call *hereditary properties*. The resulting recursion scheme is the one appearing in the algorithms that check if a given pair verifies a given hereditary property. Examples of these algorithms are the equality test, the direct occurrences check and the sequential pattern-matching algorithm. The probability that a given pair of trees of size n satisfies an hereditary property is a question that arises in the average-case analysis of all these algorithms. We cope with the problems associated to hereditary properties in a general setting and enumerate some particular instances of these problems in Chapter 6.

The first complete analysis of this work corresponds to the average size of the intersection, in Chapter 7. The intersection of a pair of trees stems from the application of the simultaneous descent scheme. The average-case analysis of the size of the intersection begins with the derivation of a partial differential equation that describes the expected size of the intersection. We find an explicit solution of the partial differential equation by means of Riemann's method. The solution is expressed in terms of Bessel functions. A subsequent asymptotic analysis and the use of standard complex analysis techniques renders an asymptotic estimate of the average size of the intersection under the balanced model. The chapter examines also the application of the translation rules of Chapter 5 to the analysis of the intersection algorithm.

Turning back to hereditary properties, we analyze in Chapter 8 the equality test algorithm and the probability that two binary trees are equal. The analysis of the performance of the equality test algorithm follows the general framework given in Chapter 6. The partial differential equation associated to the equality test is more complex than the one of the intersection. We use an argument of asymptotical equivalence to compute the average-case complexity of the equality test. Our previous analysis of the intersection proves to be helpful for the analysis of the equality test, since the same techniques used in the analysis of the intersection are used to study an algorithm closely related to the equality test. It turns out that the average-case complexities of the equality test and of the other algorithm are equivalent. This asymptotical equivalence follows from the asymptotical equivalence of the corresponding Riemann-Green functions.

Finally, we address the other basic question arising in the context of the equality between binary trees in Chapter 9. This basic question is that of evaluating the probability that two trees of size n each are equal. One of the facts that we establish in the chapter is the exponentially decreasing behavior of this probability for large n.

Chapter 4

Average Occupancy of Trees

We begin this part on elementary statistics analyzing a simple characteristic over trees, called *occupancy*. The occupancy of a tree is closely related to the degree of balancing of the tree. We analyze the average behavior of the occupancy under different probability models. This gives us the opportunity to compare these probability models and discuss their main distinctive features. Furthermore, the analyses exemplify the techniques involved in the average-case analysis when each of the probability models is assumed. We will analyze average behavior of occupancy under the uniform model, the mST model and DST model and the balanced model.

As far as the author knows, occupancy has never been defined before, although its definition is quite easy and natural. This is not an strange circumstance since occupancy has little algorithmic significance.

By occupancy of a tree we mean the sum of the ratios between the number of internal nodes and the maximum number of nodes at a given level. This characteristic can be defined for any simple family of trees with bounded arity, but it is not straightforward to do it, unless we restrict to the case where internal nodes have some fixed arity m. It is not a serious restriction since all the probabilistic models that we have considered, except the balanced model, are only defined over m-ary trees.

The occupancy of an m-ary tree T is defined as

$$\operatorname{occ}(T) = \sum_{k\geq 0} \frac{W_k(T)}{m^k},$$

where $W_k(T)$ is the number of internal nodes of T at level k, also known as *width* of T at level k. The definition can be transformed to the recursive expression:

$$\mathrm{occ}(T) = \begin{cases} 1 + \frac{1}{m}(\mathrm{occ}(T_1) + \cdots + \mathrm{occ}(T_m)), & \text{if } T \neq \square, \\ 0, & \text{otherwise.} \end{cases}$$

Let $\mathrm{occ}_{MIN}(n)$ and $\mathrm{occ}_{MAX}(n)$ denote the minimum occupancy and maximum occupancy achieved by m-ary trees of size n. These extremal values correspond to linear trees and complete trees, respectively. No much effort is needed to get,

$$\begin{aligned} \mathrm{occ}_{MIN}(n) &= \frac{m}{m-1}\left(1 - \frac{1}{m^n}\right) \sim \frac{m}{m-1}, \\ \mathrm{occ}_{MAX}(n) &= \lfloor \log_m n \rfloor + 1 + \frac{\ell}{m^{d+1}}, \qquad n = \tfrac{m^{d+1}-1}{m-1} + \ell, 0 \leq \ell < m^{d+1}, \\ & \qquad\qquad\qquad\qquad\qquad\quad d = \lfloor \log_m n \rfloor. \end{aligned}$$

4.1 Average Occupancy under the Uniform Model.

We start computing the average occupancy $\overline{occ(n)}$, for m-ary trees of size n supposing that all of them are equiprobable. Let

$$\mathrm{Occ}(z) = \sum_{T \in \mathcal{T}_m} \mathrm{occ}(T)\, z^{|T|}.$$

The average occupancy is therefore,

$$\overline{occ(n)} = \frac{[z^n]\mathrm{Occ}(z)}{t_{n,m}},$$

where $t_{n,m}$ stands for the number of m-ary trees of n internal nodes. From the definition of $\mathrm{occ}(T)$ is not difficult to verify that,

$$\mathrm{Occ}(z) = \frac{zF^m(z)}{1 - zF^{m-1}(z)} = F^2(z) - F(z),$$

with $F(z) = \sum_{n \geq 0} t_{n,m}\, z^n = 1 + zF^m(z)$.

Applying Lagrange-Bürmann inversion formula (see Subsection 1.1.3), we have

$$\begin{aligned} [z^n]\mathrm{Occ}(z) &= [z^n] \sum_{n \geq 1} \frac{z^n}{n!} \left[\frac{d^{n-1}}{dy^{n-1}} \left((2y-1)y^{mn}\right)_{y=1} \right] = \\ &= \binom{mn}{n} \cdot \left(2\frac{mn+1}{(mn-n+2)(mn-n+1)} - \frac{1}{mn-n+1} \right). \end{aligned}$$

And using the same inversion formula, it turns out that,

$$[z^n]F(z) = t_{n,m} = \binom{mn}{n} \cdot \frac{1}{mn-n+1}.$$

Hence, we have the following theorem.

Theorem 4.1. *The average occupancy for uniformly distributed m-ary trees of size n is*

$$\overline{occ(n)} = 2\frac{mn+1}{(m-1)n+2} - 1 \sim \frac{m+1}{m-1} + O\left(\frac{1}{n}\right).$$

The theorem above shows that the average value of the occupancy for trees of size n is of the same order of magnitude as the minimal value, that corresponds to linear trees,

$$\overline{occ(n)} \sim \frac{m+1}{m} \cdot \mathrm{occ}_{MIN}(n).$$

4.2 Average Occupancy under the Balanced Model.

If we use the balanced probability model defined in Section 3.3 for m-ary trees, the average occupancy is now given by

$$\overline{occ(n)} = \sum_{|T|=n} \mathrm{occ}(T)\ \Pr(T) = \frac{[z^n]\sum_{T\in\mathcal{T}_m} \mathrm{occ}(T)\ w(T)\ z^{|T|}}{[z^n]W(z)},$$

where $W(z)$ is the weighting generating function of the family $\mathcal{T}_m$.

Denoting by $\mathrm{Occ}(z)$ the generating function associated to the occupancy, the relationships for the occupancy translate into a first-order ordinary differential equation,

$$\frac{d}{dz}\mathrm{Occ} - \frac{1}{1-(m-1)z}\mathrm{Occ} = \frac{1}{(1-(m-1)z)^{m/(m-1)}}, \quad \mathrm{Occ}(0) = 0,$$

whose solution is,

$$\mathrm{Occ}(z) = \frac{1}{m-1}W(z)\ln\left(\frac{1}{1-(m-1)z}\right).$$

Using the transfer lemma given in Theorem 1.8 (Subsection 1.2.2), $[z^n]\mathrm{Occ}(z)$ is asymptotically,

$$[z^n]\mathrm{Occ}(z) \sim \frac{1}{(m-1)^{n+1}} \cdot n^{-\frac{m-2}{m-1}} \cdot \ln n \cdot \frac{1}{\Gamma(1/(m-1))}\left(1 + O\left(\frac{1}{n}\right)\right).$$

The n-th coefficient of $W(z)$ is easily computed by means of Darboux's theorem.

The asymptotic behavior of $\overline{occ(n)}$ is thus derived and we conclude the following theorem.

Theorem 4.2. *The average occupancy for m-ary trees, under the balanced probability model is*

$$\overline{occ(n)} \sim \frac{1}{m-1}\ln n.$$

The average value of the occupancy under the balanced model behaves now as the maximum value, achieved by complete trees,

$$\overline{occ(n)} \sim \frac{1}{m-1} \ln m \cdot \mathrm{occ}_{MAX}(n).$$

4.3 Average Occupancy under the mST Model.

Remember that in the mST model the size of a tree is the number of keys it contains, not its number of nodes. Recall also that in a mST each node holds up to $m-1$ keys and no node has a non-empty subtree unless it holds $m-1$ keys.

To properly define occupancy over the family of mSTs, we introduce the following definition of the width at level k:

$$W_k(T) = \begin{cases} 0, & \text{if } |T| < m-1, \\ W_{k-1}(T_1) + \cdots + W_{k-1}(T_m) & \text{if } T = \boxed{\text{m-1}}\,(T_1, \ldots, T_m), \end{cases}$$

when $k > 0$, and

$$W_0(T) = \begin{cases} 0, & \text{if } |T| < m-1, \\ 1, & \text{otherwise.} \end{cases}$$

The minimum occupancy is achieved by a tree with $\lceil \frac{n}{m-1} \rceil$ nodes linearly arranged:

$$\mathrm{occ}_{MIN}(n) = \sum_{0 \le k \le \lfloor n/(m-1) \rfloor} \frac{1}{m^k} = \frac{m - \left(\frac{1}{m}\right)^{\lfloor n/(m-1) \rfloor}}{m-1} \sim \frac{m}{m-1}$$

and the maximum by a perfectly balanced tree:

$$\mathrm{occ}_{MAX}(n) \le \lceil \log_m(n+1) \rceil,$$

where the bound is tight if $n = m^k - 1$ for some k.

As expected, the generating function

$$\mathrm{Occ}(z) = \sum_{T \in \mathcal{T}_m} \mathrm{occ}(T) \Pr(T) z^{|T|}$$

satisfies an m-th order ordinary differential equation (see Subsection 3.5.1):

$$\frac{d^{m-1}}{dz^{m-1}} \mathrm{Occ}(z) = \frac{(m-1)!}{(1-z)^m} + (m-1)!(1-z)^{-(m-1)} \mathrm{Occ}(z), \tag{4.1}$$

with $\mathrm{Occ}^{(i)}(0) = 0$ for $i = 0, \ldots, m-2$ and $\mathrm{Occ}^{(m-1)}(0) = (m-1)!$.

The homogeneous equation corresponding to (4.1) is an Euler equation, so its solution is of the type:

$$\mathrm{Occ}_h(z) = \sum_{1\le i\le m-1} A_i(1-z)^{-s_i},$$

where the s_i's are the solutions of $(s+m-2)^{\underline{m-1}} = (m-1)!$ and the A_i's are constants depending only on m and the initial conditions. Here, $x^{\underline{n}}$ denotes the n-th falling power of x, $x^{\underline{n}} = x(x-1)(x-2)\cdots(x-n+1)$ [GKP89]. Except for $s=1$, the other $m-2$ solutions are negative or complex.

To obtain a particular solution we assume that

$$\mathrm{Occ}_p(z) = A\,\frac{u(z)}{1-z}$$

for some function $u(z)$ and a constant A. Application of Liebniz's rule for the m-th derivative of a product and the initial conditions of the problem show that such a particular solution is

$$\mathrm{Occ}_p(z) = \frac{1}{H_{m-1}}\frac{\ln\left(\frac{1}{1-z}\right)}{1-z},$$

where H_n denotes the n-th harmonic number. Therefore, the following theorem is satisfied:

Theorem 4.3. *The average occupancy for m-ary trees, under the mST probability model is*

$$\overline{occ(n)} \sim \frac{\ln m}{H_{m-1}}\log_m n + O(1).$$

As for the balanced model, the average occupancy for m-ary search trees of size n is proportional to the occupancy of the complete trees of size n,

$$\overline{occ(n)} \sim \frac{\ln m}{H_{m-1}}\mathrm{occ}_{MAX}(n).$$

4.4 Average Occupancy under the DST Model.

The digital search tree model was only defined for binary trees in Subsection 3.5.2. We perform therefore the analysis only for this family of trees and take $m=2$ in our definition of the occupancy. The generalization of the model to m-ary trees and of the forthcoming results are immediate. A difference with the previous sections will be the use of an exponential generating function for the sequence of expected values $\{\overline{occ(n)}\}_{n\ge 0}$:

$$\mathrm{Occ}(z) = \sum_{T\in\mathcal{B}} \Pr(T)\mathrm{occ}(T)\frac{z^{|T|}}{T!}.$$

From the definition of occ(T) and the definition of $\Pr(T)$ in the DST probability model, we get that $\mathrm{Occ}(z)$ verifies a difference-differential equation:

$$\frac{d}{dz}\mathrm{Occ} = e^z + e^{z/2}\,\mathrm{Occ}(z/2).$$

Let $\Phi(z) = \sum_{n\geq 0} \Phi_n\, z^n/n! = e^{-z}\mathrm{Occ}(z)$. Then

$$\overline{occ(n)} = \sum_{0\leq k\leq n} \binom{n}{k} \Phi_k,$$

and

$$\frac{d\Phi}{dz} + \Phi = 1 + \Phi(z/2).$$

Since $\Phi(z)$ is an exponential generating function, we have

$$\Phi_{n+1} = -\Phi_n\left(1 - \frac{1}{2^n}\right) = (-1)^n \prod_{1\leq j\leq n}\left(1 - \frac{1}{2^j}\right).$$

Some standard manipulations, including the use of Euler formulæ [Knu68] yield

$$\overline{occ(n)} = -\sum_{j\geq 0}\left(\sum_{l\geq 0} a_{l+1}\, 2^{-j\cdot l}\right)\left(\exp(n/2^j) - 1\right),$$

where

$$a_i = \frac{(-1)^{i+1}\, 2^{-\binom{i}{2}}}{Q_{i-1}},$$

the quantities Q_n are

$$Q_n = \prod_{1\leq i\leq n} (1 - 2^{-i}),$$

and

$$\sum_{l\geq 1} \frac{a_{l+1}}{1 - 2^{-l}} = -\sum_{l\geq 1} \frac{1}{2^l - 1} = -\alpha = 1.6066\ldots$$

The last step involves the approximation of $\overline{occ(n)}$ by means of a continuous function and the application of the Mellin transform techniques (see Subsection 1.2.3). Let

$$\Theta(x) = -\sum_{j\geq 0}\left(\sum_{l\geq 0} a_{l+1}\, 2^{-j\cdot l}\right)\left(\exp(x/2^j) - 1\right).$$

Hence, $\overline{occ(n)} = \Theta(n)$. And from Cauchy's residue formula (1.6)

$$\Theta(x) = -\sum_{-\beta\leq\Re\sigma\leq d} \mathrm{Res}[\Theta^*(s)x^{-s}; s = \sigma] + O(x^{-d}), \qquad x \to \infty, \tag{4.2}$$

where $\Theta^*(s)$ denotes the Mellin transform of the function $\Theta(x)$. Here,

$$\Theta^*(s) = -\sum_{l\geq 0} a_{l+1}\frac{\Gamma(s)}{1-2^{s-l}}$$

and the fundamental strip is $< -1; 0 >$.

Then (4.2) yields

$$\begin{aligned}\Theta(x) &= \operatorname{Res}\left[\frac{\Gamma(s)x^{-s}}{1-2^s}; s=0\right] + \sum_{k\neq 0}\operatorname{Res}\left[\frac{\Gamma(s)x^{-s}}{1-2^s}; s=\frac{2\pi \mathrm{i}}{\ln 2}k\right] + \\ &+ \sum_{l\geq 1} a_{l+1}\operatorname{Res}\left[\frac{\Gamma(s)x^{-s}}{1-2^{s-l}}; s=0\right].\end{aligned}$$

Since

$$\begin{aligned}\operatorname{Res}\left[\frac{\Gamma(s)x^{-s}}{1-2^s}; s=0\right] &= \log_2 x + \frac{\gamma}{\ln 2}, \\ \operatorname{Res}\left[\frac{\Gamma(s)x^{-s}}{1-2^s}; s=\frac{2\pi \mathrm{i}}{\ln 2}\right] &= -\frac{1}{\ln 2}\Gamma\left(\frac{2\pi \mathrm{i}}{\ln 2}k\right)e^{-2\pi \mathrm{i}k\log_2 x}, \\ \operatorname{Res}\left[\frac{\Gamma(s)x^{-s}}{1-2^{s-l}}; s=0\right] &= \frac{1}{1-2^{-l}},\end{aligned}$$

we obtain:

Theorem 4.4. *The average occupancy for binary trees, under the DST probability model is*

$$\overline{occ(n)} = \log_2 n + \frac{\gamma}{\ln 2} - \alpha - \delta_0(\log_2 n)$$

where $\gamma = 0.57721\ldots$ is Euler's constant, $\alpha = \sum_{i>0}(2^i-1)^{-1} = 1.6066\ldots$ and

$$\delta_0(x) = \frac{1}{\ln 2}\sum_{k\neq 0}\Gamma\left(\frac{2\pi i}{\ln 2}k\right)e^{-2k\pi i x}$$

is a periodic function of mean value 0 and small amplitude ($\leq 10^{-5}$).

The properties of the function $\delta_0(x)$ have been considered in the analysis of approximate counting [Fla85] and are quite similar to the properties of another function that appears in the analysis of radix exchange sort [Knu73].

As in the last two analysis, the average occupancy is of the same order of magnitude as the maximum occupancy. This is because the DST model also assigns high probability to well balanced trees and low probability to poorly balanced trees.

4.5 Conclusions.

Even this simple example captures some of the particularities of the average-case analysis for each of the considered models; and reflects the differences that should be expected between the corresponding results.

First of all, the average-case analysis under the balanced model involves first-order differential equations over generating functions. The average-case analysis under the uniform model leads to combinatorial problems and the functional relations for the generating functions are of algebraic nature.

The results summarized in the theorems at the end of each section allow us to conclude that the expectation of occupancy is determined by the high probability of complete and nearly complete trees for the balanced model, whereas corresponds to the occupancy of the linear trees if the uniform model is assumed. Similarly, the mST and the DST probability models assign high probability to well balanced trees, and the average occupancy for these models behaves like the upper bound for the occupancy, a bound that is achieved by complete trees.

Chapter 5

Recursive Tree Algorithms and the Balanced Model

A lot of work has been done on the mechanization of the analysis of algorithms and data structures and there have been several approaches to the subject. One of the approaches to the mechanization of the analysis is that of Flajolet and Steyaert. Their approach consists in the translation of the structure of programs to equations over generating functions that describe the average-case complexity of the programs. To do so, they provide a system of rules that map each elementary algorithmic construction and each data type construction into an operation over the generating functions. The generating functions are associated to the sequences of the cumulated complexities (*complexity descriptors*) and the counting generating functions, since the analysis is done under the uniform model. The next stage of their approach is to recover asymptotic estimates of the average-case complexity of the algorithm from the equations over generating functions, using complex analysis techniques. We have already described this methodology in Part I. These ideas have been applied in the design and implementation of the automatic algorithm analyzer $\Lambda\Upsilon\Omega$ [FSZ89, FSZ91]. $\Lambda\Upsilon\Omega$ consists in the conjunction of: 1) an algebraic analyzer that automatically computes the equations over generating functions from the algorithm and data structures specification; 2) an analytic analyzer, that tries to solve the equations and then applies some techniques of complex analysis to extract asymptotic information from the explicit form of the generating functions.

The system of rules mapping programs to a set of relations satisfied by complexity descriptors constitutes the algebraic part of a complexity calculus. The programs that can be analyzed in this way are, *a priori*, all those that can be expressed using the algorithmic and data type constructions considered, since the system of rules is complete with respect to the "programming language". As long as the algorithmic constructions guarantee the termination of the algorithms that they express, the question of undecidability does not arise, at least in principle, and there is no theoretical limit on the possibility of analyzing the algorithm behavior. However, there are practical limitations to what can be analyzed: the "size" and complexity of the programs to be analyzed, the complexity of the functional equations, etc.

A basic reference on the subject of automatic analysis is the paper *A Complexity Calculus for Recursive Tree Algorithms* [FS87]. Flajolet and Steyaert concentrate on the analysis of algorithms dealing with trees and give a set of rules to translate the operations of a language called *PL-tree* into equations over generating functions. PL-tree is a restricted programming language over tree structures that has conditionals, iteration and recursive descent constructions. The language has enough expressiveness to include formal differentiation, tree matching, reduction/simplification algorithms, etc.

Once we restrict our interest to algorithms dealing with trees, the development of a complexity calculus for the balanced model is an immediate objective. Our purpose in this chapter is to devise a system of rules for the systematic translation of programs to functional relations over complexity descriptors. We will point out the similarities with the uniform case and the particular characteristics of the system of translation rules for the balanced model that we propose.

5.1 Translation Rules.

We shall use the programming language PL-tree as described in the paper of Flajolet and Steyaert [FS87]. We will introduce each construction and the corresponding translation rule at the same time.

A generating function whose n-th coefficient is the expected value of the complexity measure of an algorithm for inputs of size n is often called *complexity descriptor*.

The complexity descriptors that we use are of the type (see Sections 3.3 and 3.4)

$$A(x_1, x_2, \ldots, x_m) = \sum_{T_1,\ldots,T_m} w_1(T_1)\cdots w_m(T_m)\, a(T_1,\ldots,T_m)\, x_1^{|T_1|}\, x_2^{|T_2|}\cdots x_m^{|T_m|},$$

in order to analyze the average behavior of an algorithm $\mathcal{A}$ whose input is an m-tuple $(T_1, \ldots, T_m)$ in $\mathcal{F}_1 \times \cdots \times \mathcal{F}_m$ and whose time complexity is denoted by $a(T_1, \ldots, T_m)$. Each $\mathcal{F}_i$ is a simple family of trees with weight measure $w_i(\cdot)$, characteristic series $\phi_i(u)$, weight generating function $W_i(z)$, etc.[1] Then, the coefficient

$$[x_1^{n_1} x_2^{n_2} \cdots x_m^{n_m}] A(x_1, \ldots, x_m)$$

divided by $[z^{n_1}]W_1(z)[z^{n_2}]W_2(z) \cdots [z^{n_m}]W_m(z)$ is the expected value of the complexity of the algorithm for inputs $(T_1, \ldots, T_m)$ such that $|T_i| = n_i, \quad 1 \leq i \leq m$. We will often use $A(z)$ instead of $A(z, z, \ldots, z)$ to denote the complexity descriptor needed for the average-case analysis over the set of inputs of total size $n = |T_1| + \cdots + |T_m|$.

Also, by $A(x_1, \ldots, x_m \,|\, Q)$ we mean the complexity descriptor associated to the algorithm $\mathcal{A}$ but over the set of inputs that satisfy the m-ary predicate Q. We shall refer to this type of descriptor as a *conditional complexity descriptor.*

There is a full correspondence between some rules in the complexity calculus for the uniform model and that for the balanced model. The rules are syntactically identical if the corresponding construction does not access the subtrees (recursive descent) of the input trees:

1. If $Q = Q_1 \vee Q_2$ and $Q_1 \wedge Q_2 =$ `false` then

$$A(x_1, \ldots, x_m \,|\, Q) = A(x_1, \ldots, x_m \,|\, Q_1) + A(x_1, \ldots, x_m \,|\, Q_2).$$

2. If the algorithm $\mathcal{A}$ actually operates on $p < m$ trees, say $\mathcal{A}(T_1, \ldots, T_m) \equiv \mathcal{B}(T_1, \ldots, T_p)$ then

$$A(x_1, \ldots, x_m) = B(x_1, \ldots, x_p) \prod_{p<i\leq m} W_i(x_i).$$

3. Composition: If $\mathcal{A}(T_1, \ldots, T_m) \equiv \mathcal{B}(T_1, \ldots, T_m); \mathcal{C}(T_1, \ldots, T_m)$, where the semicolon denotes the usual sequential composition of procedures, then, for any predicate Q, we have

$$A(x_1, \ldots, x_m \,|\, Q) = B(x_1, \ldots, x_m \,|\, Q) + C(x_1, \ldots, x_m \,|\, Q).$$

4. Conditional: For the conditional construction

```
A(T_1, ..., T_m)  ≡  if Q(T_1, ..., T_m) then B(T_1, ..., T_m)
                     else C(T_1, ..., T_m)
                     endif;
```

[1]We will omit subscripts in the arity function $\nu_i(s)$ and the probabilities $p_i(s)$ associated to S_i, to simplify the notation.

the translation rule states that

$$A(x_1,\ldots,x_m) = QT(x_1,\ldots,x_m) + B(x_1,\ldots,x_m \mid Q) + C(x_1,\ldots,x_m \mid \neg Q),$$

where $QT(x_1,\ldots,x_m)$ is the complexity descriptor associated to the algorithm testing if $Q(T_1,\ldots,T_m)$ holds or does not.

The differences between the system of translation rules for the balanced model and the system of rules for the uniform model lie on the translation rules for the constructions involving subtree descent. Interestingly enough, there are clear relations between the "shapes" adopted by functional equations for each one of the systems of rules. We give some clarifying examples in the form of proposition.

Proposition 5.1.

1. *Subtree Descent: Let $\mathcal{A}(T) \equiv \mathcal{B}(T[i_1], T[i_2], \ldots, T[i_p])$, for some distinct $i_1,\ldots,i_p$ and where $T[i]$ denotes the i-th subtree of T. Then*

$$\begin{aligned} \frac{dA}{dz} &= B(z)\psi(W(z)), \\ \psi(u) &= \sum_{\substack{n\in \mathrm{Im}\,\nu \\ n\geq p}} u^{n-p}. \end{aligned}$$

2. *Iteration: Let*

$$\begin{aligned} \mathcal{A}(T_1,\ldots,T_m) \;\equiv\;& \textbf{for } (i_1,\ldots,i_m) \textbf{ with } Q(i_1,\ldots,i_m, root(T_1),\ldots,root(T_m)) \textbf{ do} \\ & \quad \mathcal{B}(T_1[i_1],\ldots,T_m[i_m]) \\ & \textbf{endfor}; \end{aligned}$$

where each i_j takes successively the values in $\{1,\ldots,deg(T_j)\}$. The procedure $\mathcal{B}$ is applied to each m-tuple $(T_1[i_1],\ldots,T_m[i_m])$ if the predicate Q, depending on $i_1,\ldots,i_m$ and the roots of the m input trees, is true. We assume that the test of the predicate Q takes constant time. The relation between the complexity descriptors is then

$$\frac{\partial^m}{\partial x_1\cdots\partial x_m}A(x_1,\ldots,x_m) = B(x_1,\ldots,x_m)\cdot F_Q(W_1(x_1),\ldots,W_m(x_m)) + \prod_{1\leq i\leq m}\phi_i'(W_i(x_i)).$$

where

$$F_Q(z_1,\ldots,z_m) = \sum_{\substack{Q(i_1,\ldots,i_m,s_1,\ldots,s_m) \\ s_j\in S_j, 1\leq i_j\leq \nu(s_j)}} p(s_1)\cdots p(s_m)\, z_1^{\nu(s_1)-1}\cdots z_m^{\nu(s_m)-1},$$

3. *Distributive Descent: Let*

```
$\mathcal{A}(T_1,\ldots,T_m) \equiv$ for $(i_1,\ldots,i_m)$ do
                                   $\mathcal{B}(T_1[i_1],\ldots,T_m[i_m])$
                               endfor;
```

This construction is a particularization of the previous, when no condition is imposed neither on the roots of the input trees nor on the indexes i_j. Then,

$$\frac{\partial^m A}{\partial x_1\cdots\partial x_m} = B(x_1,\ldots,x_m)\prod_{1\le i\le m}\phi_i'(W_i(x_i)).$$

4. *Simultaneous Descent: Let*

```
$\mathcal{A}(T_1,\ldots,T_m) \equiv$ for $i := 1$ to deg$(T_1)$ do $\mathcal{B}(T_1[i],\ldots,T_m[i])$ endfor;
```

where $T_1,\ldots,T_m$ belong to the family $\mathcal{F}$ and all of them have the same root symbol. This construction consecutively applies the procedure $\mathcal{B}$ to the m-tuple of first subtrees of the m input trees, then to the m-tuple of second subtrees, etc. Then

$$\frac{\partial^m}{\partial x_1\cdots\partial x_m}A(x_1,\ldots,x_m \mid Eqroot) = B(x_1,\ldots,x_m)\,\psi'\left(\prod_{1\le i\le m}W(x_i)\right).$$

where the m-ary predicate Eqroot is true if and only if its m arguments have the same root, and

$$\psi(u) = \sum_{s\in S-\nu^{-1}(0)} p^m(s)\cdot u^{\nu(s)}$$

Notice that if S has only one kind of symbol for each possible arity or m=1 then $\psi(u) = \phi(u) - 1$.

5. *Partial Descent: Let*

```
$\mathcal{A}(T_1,T_2) \equiv$ for $i := 1$ to deg$(T_1)$ do $\mathcal{B}(T_1,T_2[i])$ endfor;
```

The construction applies the procedure $\mathcal{B}$ to each of the pairs consisting of the first input tree and each of the subtrees of the root of the second input tree. If T_2 belongs to the family $\mathcal{F}$ then we have

$$\frac{\partial A}{\partial y} = B(x,y)\phi'(W(y)).$$

6. *Conditional Iteration:* *Let*

```
$\mathcal{A}(T_1,\ldots,T_m)$ ≡ for $i := 1$ to $\deg(T_1)$ while $Q(T_1[i],\ldots,T_m[i])$ do
                                  $\mathcal{B}(T_1[i],\ldots,T_m[i])$
                              endfor;
```

The construction applies consecutively the procedure $\mathcal{B}$ to m-tuples of subtrees of the m input trees, like in simultaneous descent, but this process halts before visiting the m-tuple of i-th subtrees if that tuple does not satisfy the m-ary predicate Q. Then, the complexity descriptor associated to $\mathcal{A}$ satisfies

$$\frac{\partial^m}{\partial x_1\cdots\partial x_m}A(x_1,\ldots,x_m\,|\,Eqroot) =$$
$$\Big(B(x_1,\ldots,x_m\,|\,Q)+QT(x_1,\ldots,x_m)\Big)\,\frac{\psi\left(\prod_{1\le i\le m}W(x_i)\right)-\psi(W(x_1,\ldots,x_m\,|\,Q))}{\prod_{1\le i\le m}W(x_i)-W(x_1,\ldots,x_m\,|\,Q)}.$$

where $B(x_1,\ldots,x_m\,|\,Q)$ is the conditional complexity descriptor of the procedure $\mathcal{B}$ over inputs satisfying the predicate Q, $QT(x_1,\ldots,x_m)$ is the complexity descriptor of the procedure that tests if predicate Q holds and

$$\begin{aligned}\psi(u) &= \sum_{s\in S-\nu^{-1}(0)} p^m(s)\cdot u^{\nu(s)},\\ W(x_1,\ldots,x_m\,|\,Q) &= \sum_{Q(T_1,\ldots,T_m)} w_1(T_1)\cdots w_m(T_m)\,x_1^{|T_1|}\cdots x_m^{|T_m|}.\end{aligned}$$

The proofs of the formulas in the proposition above are similar to our derivation of equation (3.3) in Section 3.4, but the complexity measures, complexity descriptors, etc. have m arguments instead of only one, and the manipulation gets m times more cumbersome.

For example, for the simultaneous descent construction one has:

$$a(T_1,\ldots,T_m) = b(T_1[1],\ldots,T_m[1]) + b(T_1[2],\ldots,T_m[2]) + \cdots + b(T_1[k],\ldots,T_m[k])$$

with $k=\deg(T_1)=\cdots=\deg(T_m)$, and $a(\cdot)$ and $b(\cdot)$ denoting the complexity of algorithms $\mathcal{A}$ and $\mathcal{B}$ on input $(T_1,\ldots,T_m)$, respectively.

The complexity descriptor $A(x_1,\ldots,x_m\,|\,\text{Eqroot})$ is, by definition,

$$A(x_1,\ldots,x_m\,|\,\text{Eqroot}) = \sum_{\substack{T_1,\ldots,T_m\in\mathcal{F}\\ \text{root}(T_1)=\cdots=\text{root}(T_m)}} a(T_1,\ldots,T_m)\,w(T_1)\cdots w(T_m)\,x_1^{|T_1|}\cdots x_m^{|T_m|}. \quad (5.1)$$

Let $S^{>} = S - \nu^{-1}(0)$ be the set of symbols of non-null arity. Moreover, to simplify notation in the following derivation, we will use k for $\nu(s)$ and the convention that the index i runs from 1 to m and the index j runs from 1 to $k = \nu(s)$. For instance, the summation index $T_i[j] \in \mathcal{F}$ indicates that the sum extends for

$$T_1[1], T_1[2], \ldots, T_1[k], \ldots, T_m[1], \ldots, T_m[k] \in \mathcal{F}.$$

The decomposition of $\mathcal{F}$, the definition of $w(\cdot)$ and the condition that all the roots must be equal allow us to rewrite equation (5.1) as

$$A(x_1, \ldots, x_m \,|\, \text{Eqroot}) =$$
$$= \sum_{s \in S^{>}} p^m(s) \sum_{\substack{T_i = s(T_i[1], \ldots, T_i[k]) \\ T_i[j] \in \mathcal{F}}} a(T_1, \ldots, T_m)\, w(T_1[1]) \cdots w(T_m[k])\, \frac{x_1^{|T_1[1]| + \cdots + |T_1[k]| + 1}}{|T_1[1]| + \cdots + |T_1[k]| + 1} \cdots .$$

Differentiating w.r.t. $x_1, x_2, \ldots, x_m$ yields

$$\frac{\partial^m}{\partial x_1 \cdots \partial x_m} A(x_1, \ldots, x_m \,|\, \text{Eqroot}) =$$
$$= \sum_{s \in S^{>}} p^m(s) \sum_{\substack{T_i = s(T_i[1], \ldots, T_i[k]) \\ T_i[j] \in \mathcal{F}}} a(T_1, \ldots, T_m)\, w(T_1[1]) \cdots w(T_m[k])\, x_1^{|T_1[1]| + \cdots + |T_1[k]|} \cdots .$$

And using the recursive definition of $a(\cdot)$ in terms of $b(\cdot)$

$$\frac{\partial^m}{\partial x_1 \cdots \partial x_m} A(x_1, \ldots, x_m \,|\, \text{Eqroot}) =$$
$$= \sum_{s \in S^{>}} p^m(s) \sum_{T_i[j] \in \mathcal{F}} \Big[\cdots + b(T_1[j], \ldots, T_m[j]) + \cdots \Big]\, w(T_1[1]) \cdots w(T_m[k])\, x_1^{|T_1[1]| + \cdots + |T_1[k]|} \cdots .$$

By symmetry,

$$\frac{\partial^m}{\partial x_1 \cdots \partial x_m} A(x_1, \ldots, x_m \,|\, \text{Eqroot}) =$$
$$= \sum_{s \in S^{>}} k \cdot p^m(s) \sum_{T_i[j] \in \mathcal{F}} b(T_1[1], \ldots, T_m[1])\, w(T_1[1]) \cdots w(T_m[k])\, x_1^{|T_1[1]| + \cdots + |T_1[k]|} \cdots .$$

Hence,

$$\frac{\partial^m}{\partial x_1 \cdots \partial x_m} A(x_1, \ldots, x_m \,|\, \text{Eqroot}) =$$
$$= \sum_{s \in S^{>}} k \cdot p^m(s) \cdot B(x_1, \ldots, x_m) \cdot \prod_{1 \le i \le m} W^{k-1}(x_i).$$

And introducing $\psi(u) = \sum_{s \in S - \nu^{-1}(0)} p^m(s) \cdot u^{\nu(s)}$,

$$\frac{\partial^m}{\partial x_1 \cdots \partial x_m} A(x_1, \ldots, x_m \,|\, \text{Eqroot}) = B(x_1, \ldots, x_m) \cdot \psi' \left(\prod_{1 \le i \le m} W(x_i) \right).$$

We remark that the PL-tree language only allows boolean functions and procedures, and does not allow assignment of variables. Programs written in PL-tree can output symbols in a write-only file. We use **return** to return boolean values and strings of symbols (trees), for convenience, but the reader must be aware that "returned" trees cannot be used to guide the subsequent behavior of the program that makes the call.

5.2 Formal Differentiation.

This section illustrates our point on the usefulness and limitations of the translation rules for the balanced model, analyzing a formal differentiation algorithm. Given a term containing constants, variables and operators and represented by a labelled tree, the algorithm outputs the tree that represents the derivative of the input term.

Since formal differentiation algorithms operate over single trees, the algorithmic constructions lead to ordinary differential equations over the complexity descriptors.

We will follow the presentation of the paper of Flajolet and Steyaert [FS87], where they analyzed formal differentiation algorithms for the uniform model. The same concepts that appeared in that paper are relevant in the average-case analysis with the balanced model. In particular, one should introduce the concept of *header*, the additional symbols needed to express the derivative of an expression. The derivative is built using the header, the subexpressions of the original expression and the derivatives of these subexpressions. We must also take into account how many copies of the subexpressions need to be done when differentiating. For any given symbol $s \in S$, where S is the symbol set generating the family $\mathcal{F}$, we denote by $h(s)$ the size of the header corresponding to s and by $\alpha_i(s)$ the number of copies of T_i that have to be done when differentiating a tree T with s at the root. Furthermore, let $\alpha(s) = \alpha_1(s) + \cdots + \alpha_k(s)$, for a symbol s of arity k.

For instance, let $S = \{\, x, \mathrm{cnst}, \sqrt{\ }, \uparrow^{-1}, \sin, \cos, \log, +, -, *, / \,\}$, each symbol having the obvious meaning. Each node with a label *cnst* has an additional field with the value of the constant. Let $d(T)$ or dT denote the derivative of the expression represented by T. Then

$$\begin{aligned}
d(\log(T)) &= /(dT, T), \qquad h(\log) = 1, \quad \alpha_1(\log) = 1, \\
d(\cos(T)) &= *(dT, -(\sin(T))), \qquad h(\cos) = 3, \quad \alpha_1(\cos) = 1, \\
&\cdots \\
d(/(T_1, T_2)) &= /(-(*(dT_1, T_2), *(T_1, dT_2)), *(T_2, T_2)), \qquad h(/) = 5, \quad \alpha_1(/) = 1, \quad \alpha_2(/) = 3.
\end{aligned}$$

```
function Diff($T$ : Expr) returns Expr is
1       case root($T$) of
3           x     :  return 1 /* returns a 'cnst' with value '1' */
4           cnst  :  return 0
5           $\sqrt{}$     :  return /(Diff(T[1]),*(2,$\sqrt{}$(Copy(T[1]))))
            ...
6           *     :  return +(*(Diff(T[1]),Copy(T[2])),*(Copy(T[1]),Diff(T[2])))
            ...
7       endcase
endfunction;
```

Algorithm 5.1: *Formal differentiation.*

The differentiation algorithm consists in a conditional structure that checks which differentiation rule should be applied, examining the symbol at the root, and then generates the derivative of the input expression recursively applying the formal differentiation procedure, generating the header and making as many copies of the subexpressions as needed (see Algorithm 5.1).

If we denote diff(T) the complexity of the differentiation algorithm for the input tree T and by Diff(z) its complexity descriptor we have

$$\begin{aligned}\mathrm{Diff}(z) &= \sum_{T\in\mathcal{F}} w(T)\,\mathrm{diff}(T)\,z^{|T|} = \\ &= \mathrm{Test}(z) + \mathrm{Gen}(z\,|\,\mathrm{root}=s_1) + \cdots + \mathrm{Gen}(z\,|\,\mathrm{root}=s_l),\end{aligned}$$

where $\mathrm{Test}(z) = W(z)$ is the complexity descriptor associated with the conditional structure checking which of the symbols in S is the root of the expression; and $\mathrm{Gen}(z\,|\,\mathrm{root}=s_i)$ is the complexity descriptor of the procedure that actually generates the derivative of a tree T when the root symbol is s_i. To generate such derivative one must generate a header with $h(s_i)$ symbols (Gen_header), make $\alpha_j(s_i)$ copies of the j-th subtree of T, for $j = 1, \ldots, \nu(s_i)$ (Copy_subexpr) and finally differentiate each of the $\nu(s_i)$ subtrees of T (Diff_subexpr). Hence,

$$\begin{aligned}\mathrm{Gen}(z\,|\,\mathrm{root}=s) &= \mathrm{Gen\_header}(z\,|\,\mathrm{root}=s) + \mathrm{Copy\_subexpr}(z\,|\,\mathrm{root}=s) + \\ &+ \mathrm{Diff\_subexpr}(z\,|\,\mathrm{root}=s).\end{aligned}$$

The generation of the header does not actually depend on T except for its root, and the generation of each symbol of the header requires constant (unit) time, so

$$\frac{d}{dz}\mathrm{Gen\_header}(z \mid \mathrm{root} = s) = h(s)\, p(s)\, W^{\nu(s)}(z).$$

For the copies of the subtrees, we have a simultaneous descent, but the cost of the body is different in each iteration, since $\alpha_i(s)$ copies are made when visiting the i-th subtree, and furthermore we impose that the root of the input tree T is the symbol s. Therefore,

$$\frac{d}{dz}\mathrm{Copy\_subexpr}(z \mid \mathrm{root} = s) = \alpha(s)\, p(s)\, W^{\nu(s)-1}(z)\mathrm{Copy}(z),$$

where $\mathrm{Copy}(z)$ is the complexity descriptor associated with the procedure 'copy'. The differentiation of the subexpressions is also a simultaneous descent construction, but the root is again s, so

$$\frac{d}{dz}\mathrm{Diff\_subexpr}(z \mid \mathrm{root} = s) = p(s)\, \nu(s)\, W^{\nu(s)-1}(z)\, \mathrm{Diff}(z).$$

The procedure 'copy' makes a simultaneous descent to copy each subtree and puts a root above the copies, or simply returns a leaf if a leaf must be copied. To simplify, we assume that the cost of testing if the input tree is a leaf, returning a leaf if it is the case and putting a root above the copies of subtrees is unitary. The translation rules yield

$$\frac{d}{dz}\mathrm{Copy}(z) = \frac{d}{dz}\mathrm{Const\_ops}(z) + \mathrm{Copy}(z)\phi'(W(z)),$$

and

$$\mathrm{Const\_ops}(z) = W(z).$$

Therefore,

$$\mathrm{Copy}(z) = zW'(z).$$

Putting everything together,

$$\frac{d}{dz}\mathrm{Diff}(z) = W'(z) + H(W(z)) + zW'(z)A(W(z)) + \mathrm{Diff}(z)\phi'(W(z)), \tag{5.2}$$

where $\phi(\cdot)$ is the characteristic series of the family, and $H(u)$ and $A(u)$ are polynomials defined as follows:

$$\begin{aligned} H(u) &= \sum_{s \in S} p(s)\, h(s)\, u^{\nu(s)}, \\ A(u) &= \sum_{s \in S} p(s)\, \alpha(s)\, u^{\nu(s)-1}. \end{aligned}$$

Furthermore, since the size of an expression is the total number of symbols, the initial condition for equation (5.2) is $\mathrm{Diff}(0) = 0$.

Hence, we can express $\mathrm{Diff}(z)$ as

$$\mathrm{Diff}(z) = W'(z) \int_0^z \frac{W'(t) + H(W(t)) + tW'(t)A(W(t))}{W'(t)}\, dt.$$

For our particular example, the only allowed degrees for nodes are 0, 1 and 2. That implies that H and A are polynomials of second and first degree respectively, and that the weighting generating function for the family is

$$W(z) = \frac{\sqrt{3}}{2} \tan\left(\frac{\sqrt{3}}{2} z + \frac{\pi}{6}\right) - \frac{1}{2}.$$

The asymptotic expansion of $\mathrm{Diff}(z)$ around the dominant singularity $z = 2\pi\sqrt{3}/9$ and the application of standard complex analysis techniques yields

$$\frac{[z^n]\mathrm{Diff}(z)}{[z^n]W(z)} = 2\gamma_A \frac{\sqrt{3}}{3} n \ln n \left(1 + O\left(\frac{1}{\ln n}\right)\right),$$

where γ_A is the coefficient of u in $A(u)$. The other coefficient of $A(u)$ and the coefficients of $H(u)$ appear in the lower order terms of the formula above. As in the uniform model, the average-case complexity of the formal differentiation algorithm is strongly influenced by the task of producing the copies of subexpressions, and it is between the worst case ($O(n^2)$) and the best case ($O(n)$) complexities; however, the average-case complexity of this kind of algorithms for the uniform model is $O(n\sqrt{n})$.

5.3 Conclusions.

The translation rules that we have proposed allow the systematic description of the average behavior of algorithms by means of functional equations, for a large class of algorithms dealing with trees and under the balanced probability model. The first stage of the average-case analysis of such algorithms can be done by mechanical application of the rules (see Section 5.1) and it is attainable by automatic symbolic computation methods.

For the second stage of the analysis, that of extracting asymptotic estimates of the average behavior of the algorithm from the functional equations that describe it, we conclude that it is necessary to develop techniques to cope with ordinary and partial differential equations, since this is the type of equations that characterize the balanced model.

An interesting question that arises in the analysis of algorithms for the balanced model is the evaluation of the probability (or weight) of subsets of pairs of trees satisfying a predicate Q and the evaluation of the conditional complexity descriptors $A(x_1, \ldots, x_m \,|\, Q)$, when the predicate Q is itself recursively defined.

In next chapter, we examine a family of recursively characterized binary predicates, the common mathematical problems that appear because of the common characterization of the predicates, the evaluation of complexity descriptors conditional to one of these predicates and some of the contexts where these problems appear: equality testing, unification and pattern-matching.

Chapter 6

Hereditary Properties.

The purpose of the previous chapter was to establish some algebraic techniques that reduced the average-case analysis of algorithms under the BST and balanced models to the study of generating functions that satisfy differential equations.

From now on, our objective is the application of these techniques to the analysis of concrete algorithms. We begin analyzing very simple algorithms, with the purpose of progressively going into more complex algorithms.

This simplicity criterion has leaded us to consider algorithms that perform a simultaneous traversal of a pair of trees. Such algorithms compare at each step, and according to some given criterion, a pair of nodes that appear at the same place in both trees.

As we will soon see, this type of elementary algorithms appear as integral components of more complex procedures of substantial relevance.

We begin with the study of a family of algorithms whose main characteristic is to verify that given pairs of trees satisfy a property, that belongs to a class that we call *hereditary properties*. Hereditary properties are predicates over pairs of trees that are recursively characterized by the fact that the property holds for a given pair of non-empty trees if and only the pairs of left subtrees and the pair of right subtrees also verify the property (see Section 6.1). The rest of the chapter is devoted to study the hereditary properties. The similarities of the recursive formulation of hereditary properties reflect in a unified mathematical formulation of the problems where these predicates get involved.

We choose to study hereditary properties, because its simplicity and because they arise in different contexts of tree manipulation: equality (see Chapter 8), unification, tree matching. The problem of evaluating the probability that a pair of a given size satisfies one of this properties is a challenging mathematical problem and is a quantity of main importance to understand the average performance of the algorithms that are related in some way to that property (see Chapters 8 and 9).

On the other hand, the algorithms that check if given pairs verify an hereditary property have the same structure, and therefore their average behavior is described by structurally identical equations over generating functions. The probability of the property, as a function of the size, gets involved in the equations for the average behavior of these algorithms.

The equations for algorithms that are based upon conditional and conditional iteration constructions also have some similarities with the equations of the algorithms checking hereditary properties. On the contrary, the equation for the average-case complexity of the intersection algorithm is structurally different from the equations of the former type (see Chapter 7). One of the reasons for such difference is that the intersection algorithm is an application of simultaneous (unconditional) descent, whilst the hereditary property checkers are representative of the algorithms based in conditional iteration.

6.1 The Recursive Characterization of Hereditary Properties.

Consider the balanced probability model restricted to families of labelled binary trees, that is, $S = \mathcal{L} + \mathcal{M}$, where $\mathcal{L}$ is the set of 0-ary symbols and $\mathcal{M}$ the set of binary symbols (see Section 3.3). We shall use $\mathcal{B}$ to denote the family of trees generated by S although we used this symbol only for the family with $S = \{\square, \circ\}$ (unlabelled binary trees). Trees in $\mathcal{L}$ will be said to be empty or to be leaves, and the size of a tree is defined as the number of its internal nodes.

The probability model for these families of binary trees is recursively defined by

$$\Pr(T) = \begin{cases} p(s), & \text{if } T = s \in \mathcal{L}, \\ p(s)\dfrac{\Pr(T_1)\cdot\Pr(T_2)}{|T|}, & \text{if } T = s(T_1, T_2), s \in \mathcal{M}. \end{cases}$$

Recall that for any family such that $\phi(u) = 1 + u^2$ and the size of trees is the number of internal nodes, the weight measure coincides with the probability.

A binary predicate h over pairs of trees is said to be an hereditary property when a pair satisfies the property if and only if the pair of left subtrees satisfies the property and the pair of right subtrees also satisfies the property. Therefore, the additional definition of the predicate for pairs where at least one of the trees is a leaf fully characterizes the predicate. More formally, we have the following definition.

Definition 6.1. *A binary predicate $h : \mathcal{B} \times \mathcal{B} \to \{true, false\}$ is an* hereditary property *if for any pair of non-empty trees (T_1, T_2) verifies*

$$h(T_1, T_2) \iff h(T_1^l, T_2^l) \wedge h(T_1^r, T_2^r) \wedge r(root(T_1), root(T_2)), \tag{6.1}$$

for some binary predicate $r : \mathcal{M} \times \mathcal{M} \to \{true, false\}$ not identically false.

Most times, our ultimate goal will be to evaluate the probability that a pair (T_1, T_2) of sizes m and n satisfies h, when the trees are distributed according to the BST model. This evaluation is analogous to the problem of counting how many pairs of trees of sizes m and n do satisfy the predicate h. Notice that the predicate r must not be identically false for the definition to have any interest.

The differences between hereditary properties rely on the definition of the property when at least one of the trees is a leaf, and on the definition of the predicate r corresponding to each property.

The algorithms that check if a given pair belongs to the subset satisfying an hereditary property h follow the scheme given in Algorithm 6.1. We call such algorithms *hereditary property checkers*. A direct consequence is that their average-case complexity depends on the probability that a pair of trees verifies the property.

The evaluation of the predicate $r(s_1, s_2)$, made by function R, requires constant time for any binary symbols s_1 and s_2. Since $W(z) = (1-z)^{-1}$ for binary trees, the complexity descriptor of an hereditary property checker can be expressed as

$$\begin{aligned}\frac{\partial^2 H}{\partial x \partial y} &= \frac{1}{(1-x)^2(1-y)^2} + \\ &+ \quad \frac{\partial^2}{\partial x \partial y} \text{basis\_H}(x, y \,|\, T_1 \in \mathcal{L} \vee T_2 \in \mathcal{L} \vee \neg r(\text{root}(T_1), \text{root}(T_2))) + \\ &+ \quad H(x,y)\left(\frac{1}{(1-x)(1-y)} + W(x,y \,|\, h)\right),\end{aligned} \tag{6.2}$$

where

$$W(x, y \,|\, h) = \sum_{h(T_1,T_2) \text{ is true}} \Pr(T_1)\Pr(T_2)\, x^{|T_1|} y^{|T_2|},$$

```
function H($T_1$,$T_2$ : BinTree) returns Boolean is
1       if $T_1$ or $T_2$ is a leaf or $\neg$R(root($T_1$),root($T_2$)) then
2          ...
3       else
4          if H($T_1^l$,$T_2^r$) then
5             return H($T_1^r$,$T_2^r$)
6          else return false
7          endif
8       endif
endfunction;
```

Algorithm 6.1: *Algorithmic scheme for hereditary property checkers.*

basis_H is the complexity descriptor of the procedure that is performed if the test of line 1 is satisfied, we have considered negligible the cost of line 6 and we have considered the test in line 1 to have unit cost.

Notice that the coefficient $[x^m y^n]W(x, y \mid h)$ is the probability that a pair of trees of sizes m and n, respectively, satisfies property h, when the trees are independently chosen and distributed according to the balanced model.

Renaming $Q(x, y) = W(x, y \mid h)$ and using the definition of hereditary properties, it turns out that $Q(x, y)$ satisfies the non-linear second-order partial differential equation:

$$\frac{\partial^2 Q}{\partial x \partial y} = \mu\, Q^2(x, y), \tag{6.3}$$

where

$$\mu = \sum_{\substack{s_1, s_2 \in \mathcal{M} \\ r(s_1, s_2)}} p(s_1)p(s_2).$$

We call Equation (6.3) the *characteristic equation* of the hereditary properties. The initial conditions depend on the way that h is defined for the induction basis, that is, for pairs in $\mathcal{L} \times \mathcal{L} + (\mathcal{B} - \mathcal{L}) \times \mathcal{L} + \mathcal{L} \times (\mathcal{B} - \mathcal{L})$. In particular, the initial conditions may depend on the the probability assigned to each leaf.

If $S = \{\,\Box, \circ\,\}$ then $\mu = 1$ and equation (6.3) reduces to

$$\frac{\partial^2 Q}{\partial x \partial y} = Q^2(x, y).$$

6.2 A Short Catalogue of Hereditary Properties.

6.2.1 Equality of Binary Trees.

The first hereditary property we consider is the equality of a pair of binary trees. Clearly, the equality predicate verifies (6.1) and for the basis, when one of the members of the pair is empty, $h(t, t')$ is true only if both t and t' belong to $\mathcal{L}$ and $t = t'$. The predicate r is true if and only if both binary symbols are the same. The algorithm that checks the equality between binary trees receives a more complete treatment in Chapter 8, whereas the evaluation of the probability that two binary trees are equal is the subject of Chapter 9.

6.2.2 Unification of First-Order Terms.

In the context of unification two interesting hereditary properties appear. The problem of unification of two first-order terms consists in producing the most general substitution of the variables in the input terms that makes them equal, or detecting that no such a substitution exists. For example, the terms $f(x, g(z, x))$ and $f(h(w), g(t, h(w)))$ can be unified by means of the general substitution $\{x \leftarrow h(w), z \leftarrow t\}$[1]. On the other hand, $f(y)$ and $g(x)$ are not unifiable, since f and g are not variables and hence, no substitution of variables would make the terms equal. Neither are $f(g(x))$ and $f(x)$, since the substitution $x \leftarrow g(x)$ would introduce an infinite recursion.

Unification is a very important operation in symbolic computation, arising in areas such as automatic theorem proving, the design of logic programming languages and natural language parsers, machine learning systems, etc. [KB70, CL73, Mil78, Col82]. For additional information on the unification problem and its basic terminology, the interested reader could refer to the survey of Knight [Kni89]. A lot of unification algorithms have been proposed: Herbrand-Robinson [Rob71], Paterson-Wegman [PW78], Martelli-Montanari [MM82], etc. The average-case analysis of these algorithms is quite difficult, even when the terms are assumed to be equally likely [CDS89, ACF$^+$91]. For the average-case analysis of unification algorithms is common to modellize terms as trees with the appropriate labels at the nodes.

[1]We use the first lower case letters $a, \ldots, h$ to denote operators, while $s, t, \ldots, z$ will denote variables.

From now on, we will talk about trees and not about terms. The properties (and the families characterized by these properties), that we have mentioned earlier, are relevant to understand the causes of failure in unification and to analyze the average behavior of different unification algorithms.

One of the causes of failure is the apparition of a *direct occurrence*. Two trees are said to have a direct occurrence if and only if there is a leaf with variable x at some place of one of the trees, while the same place in the other tree is the root of a subtree that contains a leaf x. An example of failure because of a direct occurrence was already given: $f(g(x))$ and $f(x)$ are not unifiable since x appears in the tree $f(x)$ and at the corresponding place of the other tree there is the subtree $g(x)$ that contains x. Figure 6.1 also exemplifies the direct occurrence problem.

A pair of trees is *consistent* if it does not contain any direct occurrence. The family of consistent pairs of trees properly contains the family of unifiable trees [ACF+91], since there are other causes of failure in the unification process. And it turns out that a pair of non-empty trees is consistent if and only if the pair of left subtrees and the pair of right subtrees are both consistent. To isolate the phenomenon of consistency from other causes of failure, we define the predicate r to be true for any two binary symbols. The basis of the relation is that a pair (T, x) (conversely, (x, T)) with $x \in \mathcal{L}$ is consistent if and only if T also belongs to $\mathcal{L}$ or it is a binary tree generated by $S = \mathcal{M} + (\mathcal{L} - x)$, i.e., does not contain x.

Obviously, the probability of the set of consistent pairs of trees of a given size is also relevant in the average-case analysis of the direct occurrences check.

The other interesting property appearing in the context of unification is also related to a cause of failure: *clashes*. A pair of trees is said to clash if and only if there is some internal node f in one of the trees and at the corresponding place of the other tree there is another internal node g, such that $f \neq g$. A pair of trees that clashes cannot be unified. For instance, $b(f(u, v), g(z, y))$ and $b(g(u, v), g(z, y))$ are not unifiable since these trees clash: the operator f appears at the first tree and g appears in the second tree at the same place.

A pair of non-empty trees does not clash if the pair of left subtrees does not clash, the pair of right subtrees does not clash and the roots are equal. By the definition, any pair where one of the members is a leaf does not clash. Therefore, non-clash is an hereditary property and we have fully characterized it: the predicate r is the equality between symbols and $h(T, x) = h(x, T)$ is true for any tree T if $x \in \mathcal{L}$.

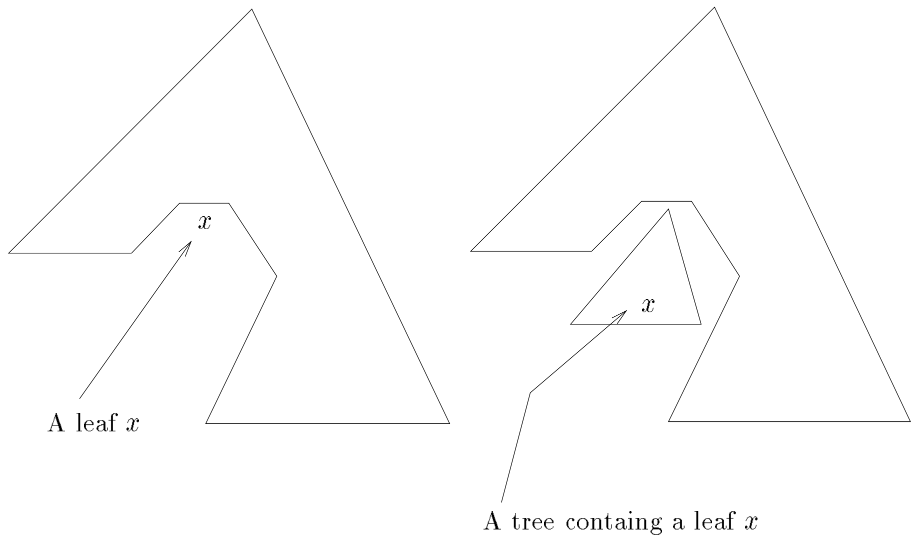


Figure 6.1: *A pair of trees containing a direct occurrence.*

6.2.3 Pattern Matching in Trees.

This subsection covers the problem of *tree matching* [KMR72, HO82, SF83]. Tree matching occurs as an important element in the design of intepreters for functional languages, automatic implementation of abstract data types, code optimization, symbolic computation, structured program editors, etc.

The input to a tree matching algorithm consists of two trees T and P called *text* and *pattern*, respectively. The pattern P is a tree with an additional type of leaves; the new type of leaf is denoted by $*$ and called *wildcard*. More precisely, the texts are trees in a simple family generated by a symbol set S, whereas the patterns belong to the simple family generated by $S' = \{*\} + S$, for some $* \notin S$ and with $\nu(*) = 0$.

A pattern P is said to *occur* at a place of the text T if the subtree rooted at that place and P match node by node, except for the wildcards of P that can "match" a whole subtree (see Figure 6.2). The output of the algorithm should be one or all the places where the pattern P occurs in the text T, if P occurs at all. On the other hand, an occurrence is called *root occurrence* of P in T if the place where P occurs is precisely the root of T. For the sake of simplicity, we shall consider that texts contain only one type of binary node.

The predicate "P occurs at the root of T" satisfies (6.1), provided we take into

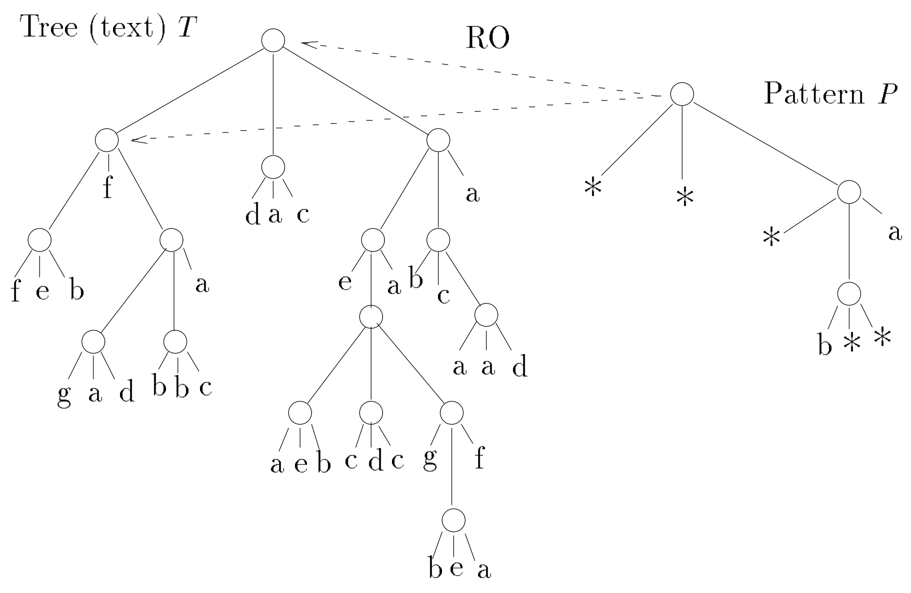


Figure 6.2: *The pattern P occurs at some places of T (dashed lines). The ocurrence that is marked with RO is a root occurrence.*

account the slight modification introduced by the fact that T and P do not belong to the same family of trees. A distinctive characteristic of root occurrence is that it is not symmetrically defined for pairs where one of the members is a leaf. If the text is a leaf and the pattern is not, then the pattern cannot occur at the root of the text. If the pattern is any 0-ary symbol $\neq *$, that is, belongs to S, then the text must be the same symbol to match the pattern. But if the pattern is $*$ then the text will always match the pattern.

Several tree matching algorithms have been proposed, the sequential tree matching algorithm being the simplest. The sequential tree-matching algorithm makes a preorder traversal of the tree, checking if the pattern occurs at the root of each subtree of the text. To determine if the pattern occurs at a given tree (root occurrence) a procedure following the scheme in Algorithm 6.1 is used. Therefore, the probability that a pattern P occurs at the root of the text T appears as part of the analysis and directly influences the average-case complexity of the sequential tree-matching algorithm; eventually, the average-case complexity of other tree matching algorithms may also depend on the probability of root occurrence of a pattern in a text.

6.3 The Characteristic Equation for Hereditary Properties.

In this section, we examine the particularization of equation (6.3) for each of the hereditary properties that we have considered in the previous section. Recall that (6.3) states that the generating function $Q(x,y)$ verifies

$$\frac{\partial^2 Q}{\partial x \partial y} = \mu\, Q^2(x,y),$$

where the coefficient $[x^m y^n]Q(x,y)$ is the probability that a pair of trees of sizes m and n satisfies a certain hereditary property and $\mu = \sum_{m(s_1,s_2)} p(s_1)\, p(s_2)$.

Equality: In the case of equality, the roots of non-empty trees must be equal; therefore, $\mu = \sum_{s \in \mathcal{M}} p^2(s)$. A pair containing a leaf is equal if and only if both members of the pair are the same leaf, yielding as initial conditions $Q(z,0) = Q(0,z) = \lambda = \sum_{s \in \mathcal{L}} p^2(s)$. The partial differential equation corresponding to equality is then

$$\begin{aligned} \frac{\partial^2 Q}{\partial x \partial y} &= \mu\, Q^2(x,y), \\ Q(z,0) &= Q(0,z) = \lambda. \end{aligned}$$

The case of $S = \{\,\square, \circ\,\}$, $\lambda = \mu = 1$ was the first one we have studied. It is not difficult to see that $[x^i y^j]Q(x,y) = 0$ if $i \neq j$, so $Q(x,y) = R(x \cdot y)$. The generating function $R(z)$ is studied in Chapter 9.

Direct Occurrences (in Unification): Since the predicate r is identically true for all pairs of binary operators we have $\mu = 1$. On the other hand, the problem is meaningless if $k = |\mathcal{L}| < 2$. Moreover, assume that $p(s) = 1/k$ for all leaves $s \in \mathcal{L}$. Then, making $\epsilon = 1 - 1/k$,

$$\begin{aligned} \frac{\partial^2 Q}{\partial x \partial y} &= Q^2(x,y), \\ Q(z,0) &= Q(0,z) = 1 + \frac{\epsilon^2\, z}{1 - \epsilon z}. \end{aligned}$$

Clash (in Unification): This phenomenon directly depends on $\mathcal{M}$, the set of binary symbols, since a pair of non-empty trees does not clash if the roots are identical. The phenomenon is trivial unless $|\mathcal{M}| \geq 2$. Hence, $\mu = \sum_{s \in \mathcal{M}} p^2(s) < 1$. For pairs containing at least a leaf, a clash cannot occur: consequently, the initial conditions are $Q(z,0) = Q(0,z) =$

$(1-z)^{-1}$. The partial differential equation for the probability of non-clash is then

$$\begin{aligned}\frac{\partial^2 Q}{\partial x \partial y} &= \mu\, Q^2(x,y),\\ Q(z,0) &= Q(0,z) = \frac{1}{1-z}.\end{aligned}$$

Root Occurrence (in Tree Matching): We have assumed already that there is only one possible binary operator, giving $\mu = 1$. Due to the asymmetry on the definition of root occurrence, we take the pattern as the first argument of the predicate and the text as the second one, by convention.

The probability of the 0-ary symbols will differ in the family of texts and the family of patterns, so we will use $p(\cdot)$ and $p'(\cdot)$ to denote these probabilities. Recall that the set of leaves in the family of patterns is the same as the set of leaves in the family of text trees, $\mathcal{L}$, plus the additional wildcard $*$. Letting $\lambda = \sum_{s\in\mathcal{L}} p(s)p'(s)$ and $p_* = p'(*)$ one has

$$\begin{aligned}\frac{\partial^2 Q}{\partial x \partial y} &= Q^2(x,y),\\ Q(x,0) &= \lambda + p_*,\\ Q(0,y) &= \lambda + \frac{p_*}{1-y}.\end{aligned}$$

The concept of hereditary property can easily be extended to m-ary predicates; the generalization would yield non-linear partial differential equations of the type

$$\frac{\partial^m Q}{\partial x_1 \ldots \partial x_m} = \mu\, Q^m.$$

For our purposes of presentation of the phenomenon of hereditary properties and its mathematical characterization, binary predicates are simpler, aesthetically more pleasant, and what is more important, fully capture the essence of the problem.

The simplest equations (presumably) are those of the equality and the non-clash, if we consider their initial conditions. It is likely that we cannot draw any information about the singularities of the function $Q(x,y)$ from the initial conditions alone, although there are some suggestive observations. For instance, there is some empirical evidence that for non-clash, $Q(x,y)$ is analytic in the domain $|x| < 1, |y| < 1$. On the other hand, for the consistency relation, empirical evidence indicates that $Q(x,y)$ is analytic on $|x| < C, |y| < C$ for some

$C > 1$ ($C = \epsilon^{-1}$?), and thus the probability that a pair of binary trees of size n is consistent (does not have a direct occurrence) would exponentially decrease with n. On the other hand, in the case of equality (with $\mu = \lambda = 1$), $Q(x, y)$ is analytic in the region $|x \cdot y| < \rho = 3.1408\ldots$ (see Chapter 9), but it is difficult to figure out the relation between the initial conditions and the domain of convergence.

In connection with the complexity calculus, information on the order of growth (or decrease) of the probability of an hereditary property can be of much value, as in the analysis of the equality test algorithm (see Chapter 8). Thus, a result similar to Theorem 8.2 holds for the direct occurrences check algorithm. Theorem 8.2 states that the average-case complexity of the equality test is $\Theta(\log n)$ and that would also be the average-case complexity of the direct occurrences check, provided that we were able to show that the probability that a pair of trees of size n is consistent is exponentially small.

Chapter 7

Average Size of Intersection

This chapter investigates a simple characteristic over pairs of binary trees, the size of the intersection of the pair, when the pairs are distributed according to the BST probability model. We will see that the functional equation that arises in this analysis is a partial differential equation.

The intersection of pair of binary trees is the tree that remains if we stack one above the other and put a leaf whenever two leaves or a leaf and an internal node are on top one of the other; and an internal node if both trees have an internal node at the corresponding place. When each pair of trees with total size n is equiprobable, the average size of the intersection is $O(1)$. But if we assume the BST probability model the average size of the intersection is $O(n^{2\sqrt{2}-2}/\sqrt{\log n})$.

We chose to study the average size of the intersection of two binary trees because of its simplicity. Although this is a very elementary problem, it reflects the structure of the problems dealing with a pair of trees and the extended version of the BST probability model.

The computation of the intersection appears in a natural way in the analysis of a number of algorithms; for example in processes involving tree matching [SF83] or unification [CDS89]. For instance, the intersection of binary trees is exactly the kernel of the *tree shuffle* algorithm described in Choppy, Kaplan and Soria [CKS89]. The time complexity of computing the intersection of any pair of binary trees is twice the size of the intersection of the two trees plus one.

The rest of this chapter is structured as follows. In Section 7.1 we obtain and solve a partial differential equation which defines the probabilistic generating function associated with the size of the intersection of binary trees. In Section 7.2 we derive exact expressions for

```
function Inters($T_1$,$T_2$ : BinTree) returns BinTree is
1        if $T_1$ or $T_2$ is $\square$ then
2            return $\square$
3        else
4            return $\circ$(Inters($T_1^l$,$T_2^l$),Inters($T_1^r$,$T_2^r$))
5        endif
endfunction;
```

Algorithm 7.1: *The intersection algorithm.*

the n-th coefficient of the function obtained in the previous section. In the following Section we deduce the main result of the chapter, which is the asymptotic behaviour of this coefficient. Section 7.4 discusses the generalization of the previous results to m-ary trees, if the balanced probability model is assumed.

An appendix on Riemann's method for the solution of partial differential equation ends the chapter.

7.1 Average Size of the Intersection of Two Binary Trees.

Let $\mathcal{B}$ be the set of all binary trees, and let $\circ$ denote any internal node, as usual. Given trees $T_1, T_2 \in \mathcal{B}$ we wish to compute the average size of the intersection of the two trees, where the intersection of T_1 and T_2, denoted $(T_1 \cap T_2)$, is given by:

$$(T_1 \cap T_2) = \begin{cases} \square, & \text{if } T_1 \text{ or } T_2 \text{ is } \square, \\ \circ((T_1^l \cap T_2^l), (T_1^r \cap T_2^r)), & \text{otherwise.} \end{cases}$$

The intersection is computed by the intersection algorithm in the obvious way (see Algorithm 7.1 and Figure 7.1).

We shall define the size of the intersection of trees T_1 and T_2 by

$$s(T_1, T_2) = \begin{cases} 0, & \text{if } T_1 \text{ or } T_2 \text{ is } \square, \\ 1 + s(T_1^l, T_2^l) + s(T_1^r, T_2^r), & \text{otherwise.} \end{cases} \tag{7.1}$$

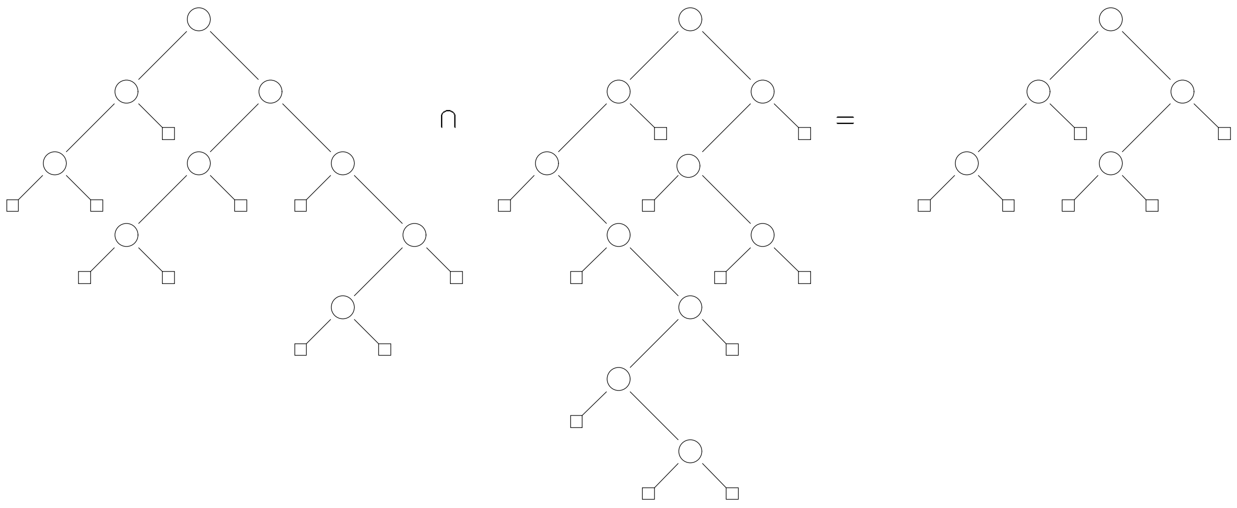


Figure 7.1: *An example of the intersection of two trees.*

We wish to compute the average value of $s(T_1, T_2)$ over all the pairs (T_1, T_2) with $|T_1| + |T_2| = n$. Let $\overline{s(n)}$ denote this average value, then we get

$$\overline{s(n)} = \sum_{|T_1|+|T_2|=n} s(T_1, T_2) \cdot \Pr(T_1, T_2).$$

Following the standard technique (see Section 3.1 and Chapter 5) let us define the generating function:

$$S(z) = \sum_{(T_1,T_2)\in\mathcal{B}^2} s(T_1, T_2) \cdot \Pr(T_1, T_2) \cdot z^{|T_1|+|T_2|},$$

and the bivariate generating function for the random independence model

$$S(x, y) = \sum_{(T_1,T_2)\in\mathcal{B}^2} s(T_1, T_2) \Pr(T_1) \Pr(T_2) x^{|T_1|} y^{|T_2|}. \tag{7.2}$$

Therefore, we have to evaluate

$$\overline{s(n)} = [z^n] S(z),$$

and the relation between $S(x, y)$ and $S(z)$ is given by

$$S(z) = \frac{1}{z} \int_0^z S(t, t)\, dt. \tag{7.3}$$

We use the following decomposition of the Cartesian product of binary trees

$$\mathcal{B}^2 = (\square, \square) + \square \times (\mathcal{B} - \square) + (\mathcal{B} - \square) \times \square + (\mathcal{B} - \square)^2. \tag{7.4}$$

From equation (7.2) and using (7.4), (7.1) and Definition 3.2, we get the following partial differential equation

$$\frac{\partial^2 S(x,y)}{\partial x \partial y} = \frac{1}{(1-x)^2(1-y)^2} + \frac{2S(x,y)}{(1-x)(1-y)}, \tag{7.5}$$

subject to the boundary conditions: for all x and y, $S(x,0) = 0$ and $S(0,y) = 0$. These boundary conditions are given by the consideration of the intersection of an arbitrary tree and a leaf and the intersection of a leaf and an arbitrary tree, respectively.

The generating function $S(x,y)$ can be reexpressed as

$$S(x,y) = S_h(x,y) - \frac{1}{(1-x)(1-y)}, \tag{7.6}$$

where $-(1-x)^{-1} \cdot (1-y)^{-1}$ is a particular solution of (7.5) and $S_h(x,y)$ satisfies the homogeneous equation

$$\frac{\partial^2 S_h}{\partial x \partial y} = \frac{2S_h}{(1-x)(1-y)}, \tag{7.7}$$

with boundary conditions $S_h(x,0) = 1/(1-x)$ and $S_h(0,y) = 1/(1-y)$.

Making the change of variables

$$\begin{cases} X = -\sqrt{2}\ln(1-x) \\ Y = -\sqrt{2}\ln(1-y) \end{cases}$$

and setting

$$G(X,Y) = S_h(1 - e^{-X/\sqrt{2}}, 1 - e^{-Y/\sqrt{2}}), \tag{7.8}$$

we finally obtain the hyperbolic differential equation

$$\frac{\partial^2 G}{\partial X \partial Y} = G, \tag{7.9}$$

subject to boundary conditions $G(X,0) = e^{X/\sqrt{2}}$, $G(0,Y) = e^{Y/\sqrt{2}}$.

This partial differential equation can be solved by the method of Riemann (see the appendix at the end of this chapter) to yield

$$\begin{aligned} G(X,Y) &= \frac{1}{\sqrt{2}} \int_0^X e^{t/\sqrt{2}} J_0\left(2\mathrm{i}\sqrt{(X-t)Y}\right) dt + \\ &\quad \frac{1}{\sqrt{2}} \int_0^Y e^{t/\sqrt{2}} J_0\left(2\mathrm{i}\sqrt{(Y-t)X}\right) dt + J_0(2\mathrm{i}\sqrt{XY}), \end{aligned} \tag{7.10}$$

where $J_0(\cdot)$ denotes the Bessel function of the first kind of order 0.

7.2 Exact Developments.

In this section, we state two theorems concerning the exact average size of the intersection of a pair of trees of size n. The exact values are not in a closed form and involve sums of binomial and Stirling numbers of the first kind. A first application of standard asymptotic methods shows that $\overline{s(n)} = O(n^{2\sqrt{2}-2})$. Nevertheless, we will get better asymptotic estimates from the solution (7.10) of the partial differential equation in next section, and hence we will not give further details about the way this asymptotic estimation can be computed from the exact value of $\overline{s(n)}$. The exact developments were obtained by more classical methods, setting up recurrences and solving them.

The coefficients of the generating function

$$S(x,y) = \sum_{n,m\geq 0} s_{n,m} x^n y^m$$

satisfy the following recurrence

$$(m+1)(n+1)s_{n+1,m+1} = (n+1)(m+1) + 2 \sum_{\substack{0\leq i\leq n\\ 0\leq j\leq m}} s_{i,j}, \qquad s_{k,0} = s_{0,k} = 0, \quad k \geq 0,$$

which can be obtained from the differential equation (7.5). The recurrence can be simplified and transformed into

$$s_{n,m} = \frac{1}{n\cdot m}\left[(1-nm+n+m)s_{n-1,m-1} + n(m-1)s_{n,m-1} + (n-1)m s_{n-1,m}\right], \qquad n,m > 0$$

R. Baeza-Yates worked out this recurrence and was able to prove the following result [BCDM92]:

Theorem 7.1. *The expected size of the intersection of two independently chosen random binary trees of sizes n and m, under the BST model, is given by the formula*

$$[x^n y^m]S(x,y) = \frac{1}{n!m!} \sum_{j=0}^{\min(n,m)} 2^j \left(\begin{bmatrix} n \\ j \end{bmatrix} \sum_{k=j}^{m} \begin{bmatrix} m \\ k \end{bmatrix} + \begin{bmatrix} m \\ j \end{bmatrix} \sum_{k=j}^{n} \begin{bmatrix} n \\ k \end{bmatrix} - \begin{bmatrix} m \\ j \end{bmatrix} \begin{bmatrix} n \\ j \end{bmatrix} \right) - 1,$$

where, as usual, the notation $\begin{bmatrix} n \\ k \end{bmatrix}$ denotes the Stirling numbers of first kind [Knu68].

On the other hand, we know by (7.3) that

$$[z^n]S(z) = \frac{1}{n+1} \cdot [z^n]S(z,z) \tag{7.11}$$

and using the previous Theorem 7.1 one also gets the following result

Theorem 7.2. *Under the BST model, the average size of the intersection of two trees T_1 and T_2 such that $|T_1| + |T_2| = n$, is given by the formula*

$$\overline{s(n)} = \frac{1}{(n+1)!}\left(2\sum_{k=0}^{n}\begin{bmatrix} n \\ k \end{bmatrix}\sum_{j=0}^{\lfloor k/2 \rfloor}\binom{k}{j}2^j - \sum_{j=0}^{\lceil n/2 \rceil}\begin{bmatrix} n \\ 2j \end{bmatrix}\binom{2j}{j}2^j\right) - 1.$$

7.3 Asymptotic Results.

We are interested in obtaining asymptotics to the $[z^n]S(z)$. Using (7.11) together with (7.6) we obtain an expression for the average value of $s(T_1, T_2)$

$$\overline{s(n)} = \frac{1}{n+1}[z^n]S_h(z,z) - 1. \tag{7.12}$$

To obtain an asymptotic value for $[z^n]S_h(z,z)$ we need first the following technical lemma,

Lemma 7.1. *The asymptotic behavior of $G(Z,Z)$ is given by*

$$G(Z,Z) \sim 3J_0(2iZ) + \sqrt{2}\cdot\frac{d}{dZ}J_0(2iZ).$$

Proof. Let $G(Z,Z) = A(Z) + J_0(2\text{i}Z)$ with

$$A(Z) = \sqrt{2}\int_0^Z e^{t/\sqrt{2}} J_0\left(2\text{i}\sqrt{(Z-t)Z}\right)dt. \tag{7.13}$$

Let us recall the series expansion of $J_0(x)$ (1.4)

$$J_0(x) = \sum_{k\geq 0}\frac{(-1)^k}{(k!)^2}\left(\frac{x}{2}\right)^{2k}.$$

Then, substituting in (7.13)

$$A(Z) = \sqrt{2}\int_0^Z e^{t/\sqrt{2}}\left(\sum_{k\geq 0}\frac{(-1)^k}{(k!)^2}\cdot(-(Z-t)Z)^k\right)dt$$

If we define

$$\Phi_k(Z) = \int_0^Z e^{t/\sqrt{2}}(Z-t)^k dt = (\sqrt{2})^{k+1}\cdot k!\cdot\sum_{j>k}\frac{1}{j!}\left(\frac{Z}{\sqrt{2}}\right)^j, \tag{7.14}$$

we get

$$A(Z) = \sqrt{2} \cdot \sum_{k \geq 0} \frac{Z^k}{(k!)^2} \cdot \Phi_k(Z) = 2 \sum_{k \geq 0} \frac{(Z\sqrt{2})^k}{k!} \left(\sum_{j>k} \frac{\left(\frac{Z}{\sqrt{2}}\right)^j}{j!} \right),$$

where (7.14) can be proved by induction on k.

Let us consider the coefficient $a_n = [Z^n]A(Z)$. In order to evaluate the asymptotic behaviour of a_n, we shall distinguish three different cases:

if $n = 0$ then $a_0 = 0$,

if $n = 2p + 1$, then

$$a_{2p+1} = \frac{2}{(\sqrt{2})^{2p+1}(2p+1)!} \cdot \sum_{k=0}^{p} \binom{2p+1}{k} 2^k,$$

if $n = 2p$, then

$$a_{2p} = \frac{2}{2^p(2p)!} \cdot \sum_{k=0}^{p-1} \binom{2p}{k} 2^k.$$

The value of each one of the summations involved in these expressions tends to concentrate in its last term; therefore, we write them in the following form,

$$\begin{aligned} a_{2p+1} &= c'_p \cdot \frac{2\sqrt{2}}{(p!)^2(p+1)} \\ a_{2p} &= c''_p \cdot \frac{2}{(p!)^2}. \end{aligned}$$

with

$$\begin{aligned} c'_p &= \sum_{j=1}^{p+1} \frac{(p+1)!p!}{(p+1-j)!(p+j)!} 2^{-j} \\ c''_p &= \sum_{j=1}^{p} \frac{(p!)^2}{(p-j)!(p+j)!} 2^{-j} \end{aligned}$$

Let us consider c''_p first. Since

$$\forall j, \qquad 1 - \frac{j^2}{p} < \frac{(p!)^2}{(p-j)!(p+j)!} < 1,$$

we have

$$\sum_{j=1}^{p} \left(1 - \frac{j^2}{p}\right) 2^{-j} < c''_p < \sum_{j=1}^{p} 2^{-j}.$$

Therefore as $p \to \infty$, $c''_p \to 1$. The same argument holds for c'_p. So we conclude that $A(Z)$ is asymptotically equivalent to

$$\begin{aligned} A(Z) &= \sum_{p>0} a_{2p}\, Z^{2p} + \sum_{p\geq 0} a_{2p+1}\, Z^{2p+1} = \\ &= 2\sum_{p>0} c''_p \frac{Z^{2p}}{(p!)^2} + 2\sqrt{2}\sum_{p\geq 0} c'_p \frac{Z^{2p+1}}{p!(p+1)!} = \\ &\sim 2J_0(2\mathrm{i}Z) + \sqrt{2}\frac{d}{dZ}J_0(2\mathrm{i}Z) \end{aligned}$$

Hence,

$$G(Z,Z) \sim 3J_0(2\mathrm{i}Z) + \sqrt{2}\cdot\frac{d}{dZ}J_0(2\mathrm{i}Z).$$

■

Now, we are in position to state the asymptotic behavior of $[z^n]S_h(z,z)$.

Lemma 7.2. *The asymptotic behavior of* $[z^n]S_h(z,z)$ *is given by*

$$[z^n]S_h(z,z) \sim c_1 \cdot [z^n]J_0(-2\sqrt{2}\cdot i\cdot \ln(1-z)),$$

where $c_1 = 3 + 2\sqrt{2}$.

Proof. The statement of the lemma is obtained from Lemma 7.1 by making the change of variable $Z = -\sqrt{2}\ln(1-z)$ and taking into account that $[z^n]J'_0(2\mathrm{i}Z)|_{Z=-a\ln(1-bz)}$ is asymptotically twice $[z^n]J_0(2\mathrm{i}Z)|_{Z=-a\ln(1-bz)}$ for any constants $a>0$, $b>0$. Let f_n and g_n be $[z^n]F(z) = [z^n]J_0(2\mathrm{i}Z)|_{Z=-a\ln(1-bz)}$ and $[z^n]G(z) = [z^n]J'_0(2\mathrm{i}Z)|_{Z=-a\ln(1-bz)}$, respectively. Since

$$\frac{dF}{dz} = G(z)\frac{dZ}{dz},$$

we have

$$\begin{aligned} G(z) &= \frac{1-bz}{ab}\sum_{n\geq 0}(n+1)f_{n+1}\, z^n = \\ &= \frac{1}{ab}\sum_{n\geq 0}[(n+1)f_{n+1} - bnf_n]\, z^n. \end{aligned}$$

Using Lemma 7.3, one obtains the asymptotic behavior of f_n

$$f_n \sim A\frac{n^{2a-1}b^n}{\sqrt{\ln n}},$$

for some constant A depending on a and b.

Now,

$$\begin{aligned}(n+1)f_{n+1} - bnf_n &\sim \frac{Ab^{n+1}n^{2a}}{\sqrt{\ln n}}\left[\frac{(1+1/n)^{2a}}{\sqrt{\frac{\ln(n+1)}{\ln n}}} - 1\right] = \\ &= 2a\frac{Ab^{n+1}n^{2a-1}}{\sqrt{\ln n}}(1+O\left(\frac{1}{\ln n}\right) = \\ &= 2abf_n.\end{aligned}$$

Finally, combining this last result with the relation between f_n and g_n yields

$$g_n = \frac{1}{ab}[(n+1)f_{n+1} - bnf_n] \sim 2f_n,$$

as claimed. ■

An asymptotic estimation of the n-th coefficient of $J_0(-2\mathrm{i}\sqrt{2}\ln(1-z))$ is given in the following lemma.

Lemma 7.3. *The n-th coefficient of $J_0(-i\,\alpha\ln(1-\beta z))$ for any positive real numbers α and β, is asymptotically given by*

$$[z^n]J_0(-i\alpha\ln(1-\beta z)) = \beta^n\frac{\sqrt{2}}{2\Gamma(\alpha)\sqrt{\pi\alpha}}\frac{n^{\alpha-1}}{\sqrt{\ln n}}\left(1+O\left(\frac{1}{\ln n}\right)\right). \tag{7.15}$$

Proof. The asymptotic expansion of $J_0(z)$ (see Section 1.3, equation (1.5)) can be used to estimate the asymptotic behavior of the n-th coefficient of $J_0(-\mathrm{i}\alpha\ln(1-\beta z))$ for any real positive numbers α and β. If $z \to \beta^{-1}$ then $|\ln(1-\beta z)|$ tends to infinity and $|\arg(-\mathrm{i}\alpha\ln(1-\beta z)| < \pi$ provided that we are sufficiently close to β^{-1} and $|\arg(1-\beta z)| > 0$. Therefore,

$$J_0(-\mathrm{i}\alpha\ln(1-\beta z)) = \frac{1}{2}\left(\frac{2}{\pi\alpha\ln(1/(1-\beta z))}\right)^{1/2}\left[(1-\beta z)^{-\alpha} - \mathrm{i}(1-\beta z)^{\alpha} + O\left(\frac{|1-\beta z|^{-\alpha}}{|\ln(1/(1-\beta z))|}\right)\right],$$

where we have used the exponential form of cos and

$$\begin{aligned}|\Im(-\mathrm{i}\alpha\ln(1-\beta z))| &= \alpha\left|\Re\left(\ln\left(\frac{1}{1-\beta z}\right)\right)\right| = \\ &= \alpha\left|\ln\left(\left|\frac{1}{1-\beta z}\right|\right)\right| = \alpha\ln\left(\left|\frac{1}{1-\beta z}\right|\right),\end{aligned}$$

since $\alpha > 0$ and we may assume that $|1-\beta z| < 1$.

Finally, using the transfer lemma given in Theorem 1.8 gives[1] the stated asymptotic estimate for the n-th coefficient of $J_0(-\mathrm{i}\alpha\ln(1-\beta z))$.

The asymptotic behavior of the n-th coefficient of $J_0(-\mathrm{i}\alpha\ln(1-z))$ can be alternatively derived using Laplace's method and the integral representation of $J_0(z)$ (see Section 1.3). ■

Applying the previous lemma with $\alpha = 2\sqrt{2}$ and $\beta = 1$, we obtain:

$$[z^n]J_0\left(-2\mathrm{i}\sqrt{2}\ln(1-z)\right) = c_2 \cdot \frac{n^{2\sqrt{2}-1}}{\sqrt{\ln n}}\left(1+O\left(\frac{1}{\ln n}\right)\right), \tag{7.16}$$

where the value of the constant c_2 is given by

$$c_2 = \frac{1}{2^{5/4}\sqrt{\pi}\,\Gamma(2\sqrt{2})} = 0.1381288\ldots$$

Combining Lemma 7.2 together with (7.12) and (7.16) yields the following theorem:

Theorem 7.3. *Under the extended BST model, the average size of the intersection of two trees behaves asymptotically as*

$$\overline{s(n)} = c \cdot \frac{n^{2\sqrt{2}-2}}{\sqrt{\ln n}} \cdot \left(1+O\left(\frac{1}{\ln n}\right)\right),$$

with $c = c_1c_2 = 0.8050738\ldots$

Again, it should be emphasized, that under the uniform model, the average size of the intersection of two trees is $1.5 \cdot \left(1+O\left(\frac{1}{n}\right)\right)$, which is quite a different result from the one we just obtained. This constant average behavior for the size of the intersection can be explained in rather intuitive terms. The probability that a pair of trees of size n consists in a leaf and a tree of size n is about 0.5 for uniformly distributed pairs. The size of the intersection of such pairs is null. Analogously, the probability that a pair of size n is of the type $({}_{\square}\overset{\circ}{\wedge}{}_{\square}, T)$ or $(T, {}_{\square}\overset{\circ}{\wedge}{}_{\square})$ is approximately 0.125, and the size of the intersection of such pairs is 1, etc.

As said in the introduction, it also follows from Theorem 7.3 that under the BST model the average-case complexity of the intersection algorithm is

$$2c \cdot \frac{n^{2\sqrt{2}-2}}{\sqrt{\ln n}} \cdot \left(1+O\left(\frac{1}{\ln n}\right)\right),$$

while under the uniform distribution the average-case complexity of the intersection algorithm is $4 \cdot (1+O\left(\frac{1}{n}\right))$ [CKS89].

[1] If α is a positive integer a slight modification of the transfer lemma must be used instead.

7.4 Average Size of Intersection under the Balanced Model.

The analysis of the average size of the intersection of two m-ary trees distributed accordingly to the balanced probability model is done in a similar way to that of the case of binary trees.

Given two m-ary trees T', T'' its intersection is the m-ary tree defined by:

$$T' \cap T'' = \begin{cases} \square, & \text{if } T' = \square \text{ or } T'' = \square, \\ \circ((T_1' \cap T_1''), \ldots, (T_m' \cap T_m'')), & \text{if } T' = \circ(T_1', \ldots, T_m') \text{ and } T'' = \circ(T_1'', \ldots, T_m''). \end{cases}$$

Let $s(T', T'')$ denote the size of the intersection of T' and T''. Then $s(T', T'')$ is null if any of the trees is a leaf; otherwise, it is the sum of the sizes of the intersections of the subtrees.

The average value of $s(T', T'')$ over all the pairs $(T', T'') \in \mathcal{T}_m^2$ with $|T'| + |T''| = n$ is given by the quotient of the n-th coefficients of two generating functions (see for example, Section 3.4):

$$\begin{aligned} \overline{s(n)} &= \sum_{|T'|+|T''|=n} s(T', T'') \cdot \Pr(T', T'') = \\ &= \frac{\sum_{|T'|+|T''|=n} s(T', T'') \cdot w(T', T'')}{\sum_{|T'|+|T''|=n} w(T') \cdot w(T'')} = \frac{[z^n] S(z, z)}{[z^n] W^2(z)}, \end{aligned}$$

where

$$S(x, y) = \sum_{(T', T'') \in \mathcal{T}_m^2} s(T', T'') w(T') w(T'') x^{|T'|} y^{|T''|} \tag{7.17}$$

and

$$W(z) = \sum_{T \in \mathcal{T}_m} w(T) z^{|T|} = (1 - (m-1)z)^{-1/(m-1)}. \tag{7.18}$$

Recall that $w(\cdot)$ denotes the weight measure of the object considered, and that $W(z)$ is the weighting generating function of the family of trees (in this case, the family of m-ary trees).

The asymptotic behavior of the n-th coefficient $(n \to \infty)$ of $W^2(z)$ can be derived by means of Darboux's theorem (see Subsection 1.2.2) that yields the following value:

$$[z^n] W^2(z) \sim (m-1)^n \cdot n^{-(m-3)/(m-1)} \cdot \frac{1}{\Gamma(2/(m-1))} \cdot (1 + O(1/n)). \tag{7.19}$$

Differentiating (7.17) and using the definitions of $w(T)$ (Section 3.3) and $s(T', T'')$, together with (7.18), we get the following hyperbolic partial differential equation

$$\frac{\partial^2 S(x,y)}{\partial x \partial y} = [W(x) \cdot W(y)]^m + [W(x) \cdot W(y)]^{m-1} \cdot m \cdot S(x,y), \tag{7.20}$$

subject to the boundary conditions: for all x and y, $S(x,0) = 0$ and $S(0,y) = 0$.

This partial differential equation admits a particular solution and the general solution for the corresponding homogeneous equation can be obtained by means of the Riemann's method. The asymptotic behavior of the solution of the homogeneous equation can be derived in much the same way as in Lemma 7.2 and turns out to be asymptotically equivalent to

$$c \cdot [z^n] J_0 \left(-2\frac{\sqrt{m}}{m-1} \mathrm{i} \ln(1-(m-1)z) \right),$$

with $c = ((\sqrt{m}+1)/(m-1))^2$. Finally, the asymptotic behavior of $\overline{s(n)}$ is gathered using Lemma 7.3. These steps lead us to the following theorem:

Theorem 7.4. *Under the balanced probability model, the average size of the intersection of two m-ary trees behaves asymptotically as*

$$\overline{s(n)} \sim c \cdot \frac{n^{(2\sqrt{m}-2)/(m-1)}}{\sqrt{\ln n}} \cdot \left(1 + O\left(\frac{1}{\ln n}\right)\right),$$

where the constant c has value

$$c = \frac{m + 2\sqrt{m} + 1}{2\sqrt{\pi} m^{1/4} (m-1)^{3/2}} \cdot \frac{\Gamma(\frac{2}{m-1})}{\Gamma(\frac{2\sqrt{m}}{m-1})}.$$

Notice that Theorem 7.3 is just a particular case of Theorem 7.4, setting $m = 2$.

Before finishing, let us see how the translation rules of Chapter 5 can be applied to the intersection algorithm for pairs of m-ary trees. We define the size of a tree as the number of internal nodes that it contains, and the size of a pair as the sum of the sizes, as usual. The algorithm consists in a simple test followed by a simultaneous descent construction that is performed when none of the members of the pair is a leaf, like in Algorithm 7.1, but repeatedly for the first, second, third, ...pair. The simultaneous descent loop computes the intersection of each of the m pairs of subtrees and the results are attached to a root. The resulting tree is the intersection of the pair. This part, the simultaneous descent and the construction of the result form a procedure that we will call Int_loop. Therefore, the complexity descriptor Intsc(x, y) of the intersection algorithm is given by

$$\text{Intsc}(x,y) = \text{Test}(x,y) + \text{Return}(x,y \,|\, T_1 = \Box \vee T_2 = \Box) + \text{Int\_loop}(x,y \,|\, T_1 \neq \Box \wedge T_2 \neq \Box).$$

The test and the instruction returning a leaf have constant complexity, and hence,

$$\begin{aligned} \text{Test}(x,y) &= W(x)W(y), \\ \text{Return}(x,y \,|\, T_1 = \Box \vee T_2 = \Box) &= 1 + W(x) + W(y), \end{aligned}$$

taking into account in the last equation that $|\Box| = 0$.

Since the input is a pair of m-ary trees, we know that the simultaneous descent is performed only if both trees in the pair have the same root and have subtrees. Within the loop, we attach each newly computed intersection of the corresponding pairs of subtrees to a root. As each edge between the root and a subtree can be added in constant time, we can write

$$\text{Int\_loop}(x,y \,|\, T_1 \neq \Box \wedge T_2 \neq \Box) = m(W(x)-1)(W(y)-1) + \text{Desc}(x,y),$$

where Desc(x, y) is the complexity descriptor associated to the simultaneous descent. Applying rule 4 of Proposition 5.1 and as non-empty trees have the same root (there is only one type of internal node, of degree m)

$$\frac{\partial^2}{\partial x \partial y}\text{Desc}(x,y \,|\, \text{Eqroot}) = \text{Intsc}(x,y)\psi'(W(x)W(y)).$$

For the family of m-ary trees, $\psi(u) = \phi(u) - 1 = u^m$, so we have the following functional equation for Intsc(x, y)

$$\frac{\partial^2 \text{Intsc}}{\partial x \partial y} = W'(x)W'(y)(1+m) + m \cdot \text{Intsc}(x,y) \cdot W^{m-1}(x)W^{m-1}(y).$$

The initial conditions of the partial differential equation above as well as $W(z)$ depend on the definition of size. As we take $|\Box| = 0$, $W(z) = (1-(m-1)z)^{-1/(m-1)}$ and the equation reads

$$\begin{aligned} \frac{\partial^2 \text{Intsc}}{\partial x \partial y} &= (1+m)(W(x)W(y))^m + m \cdot \text{Intsc}(x,y) \cdot (W(x)W(y))^{m-1}, \\ \text{Intsc}(z,0) &= \text{Intsc}(0,z) = W(z). \end{aligned}$$

This partial differential equation is almost identical to (7.20), except for the fact that (7.20) corresponds to the size of intersection and the last one corresponds to the complexity of the algorithm computing the intersection. The same methods used to obtain the average behavior of the size of the intersection could be applied to get the average-case complexity of the intersection algorithm.

Appendix A Riemann's Method.

Riemann's method was devised in 1860 to obtain solutions of second-order linear partial differential equations (see for instance [Rie53],[CH62, ch. 5],[Cop75, ch. 5]). The method is applied for the analysis of the average size of the intersection (in this chapter) and for the analysis of the average-case complexity of the equality test (Chapter 8).

Riemann's method can be applied to equations of the form

$$au_{xx} + 2hu_{xy} + bu_{yy} + 2gu_x + 2fu_y + cu = F(x, y),$$

where $u = u(x, y)$, subscripts denote partial differentiation with respect to the indicated variable(s) as usual and a,b,h,g,f and c are functions of x and y alone.

We will begin introducing the notion of adjoint linear operator and of Riemann-Green function. Let the linear operator L be

$$L[u] = au_{xx} + 2hu_{xy} + bu_{yy} + 2gu_x + 2fu_y + cu.$$

Then, there exists a unique linear operator L^* called the adjoint of L, and a function v, called *Riemann-Green function* of L, such that $vL[u] - uL^*[v]$ is a divergence, i.e.,

$$vL[u] - uL^*[v] = \frac{\partial H}{\partial x} + \frac{\partial K}{\partial y}.$$

The adjoint operator L^* of L can be calculated using Green's theorem.

Theorem 7.5. (Green) *Let D be some domain closed by a regular closed curve C. If u and v are continuous in the closure of D, both u_x and v_y exist and are bounded in D, and the double integrals of u_x and v_y over D exist, then*

$$\int\!\!\int_D (u_x + v_y)\,dx\,dy = \int_C (lu + mv)\,ds$$

where (l, m) are the direction cosines of the outward normal to C.

Since L^* does not depend on any particular domain, we can use a domain D consisting of a rectangle to apply Green's theorem, and one obtains

$$L^*[v] \quad = \quad (av)_{xx} + 2(hv)_{xy} + (bv)_{yy} - 2(gv)_x - 2(fv)_y + cv \qquad (7.21)$$

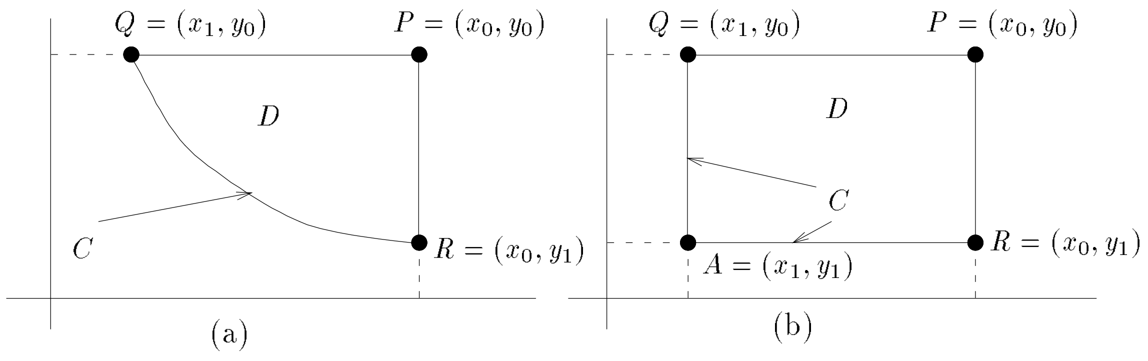


Figure 7.2: *The domain of dependence for the solution of an hyperbolic partial differential equation. (a) C is non-degenerate; (b) C is degenerate.*

$$H = avu_x - u(av)_x + hvu_y - u(hv)_y + 2guv, \quad (7.22)$$

$$K = hvu_x - u(hv)_x + bvu_y - u(bv)_y + 2fuv. \quad (7.23)$$

in order that L^* is unique, and $vL[u] - uL^*[v]$ is a divergence.

Riemann's method can be applied to solve the so-called Cauchy problem for linear equations of hyperbolic type. A linear second-order partial differential equation is said to be hyperbolic in a domain D if and only if $ab - h^2 < 0$ in D. There is a canonical form into which any hyperbolic partial differential equation can be transformed using *characteristic* variables:

$$2u_{xy} + 2gu_x + 2fu_y + cu = F(x, y),$$

so $L^*[v] = 2v_{xy} - 2(gv)_x - 2(fv)_y + cv$, $H = vu_y - uv_y + 2gv$ and $K = vu_x - uv_x + 2fuv$. As we shall deal with partial differential equations where $a = b = 0$ and h is some constant, all of them are of hyperbolic type. The equations we will find are already in canonical form, except that we have to multiply by 2 both sides of these equations.

Let C be some regular duly-inclined arc for which u,u_x and u_y are known, i.e., Cauchy data is given for C. The arc can degenerate to a pair of characteristics; this is the case in our applications of the method. Let the characteristics $x = x_0$ and $y = y_0$ through $P = (x_0, y_0)$ cut C at $Q = (x_1, y_0)$ and $R = (x_0, y_1)$, and let D be the domain bounded by PQ,PR and C (see Figure 7.2(a)).

Using Green's theorem and equations (7.21) to (7.23) it can be shown that, if $L^*[v] = 0$ then

$$\begin{aligned} u(x_0, y_0) &= \frac{1}{2}u(x_1, y_0)v(x_1, y_0; x_0, y_0) + \frac{1}{2}u(x_0, y_1)v(x_0, y_1; x_0, y_0) + \\ &+ \frac{1}{2}\int_{QR}(K\,dx - H\,dy) + \frac{1}{2}\iint_D v(x, y; x_0, y_0)F(x, y)\,dxdy, \end{aligned} \tag{7.24}$$

where the Riemann-Green function $v(x, y; x_0, y_0)$ also verifies

$$\begin{aligned} v_x &= fv \qquad \text{on } y = y_0, \\ v_y &= gv \qquad \text{on } x = x_0 \end{aligned}$$

and

$$v(x_0, y_0; x_0, y_0) = 1.$$

In our application of the method, the arc C is degenerated and consists in the parallel segments to x and y-axis, cutting at $A = (x_1, y_1)$, so $Q = (x_1, y_0)$ and $R = (x_0, y_1)$ (see Figure 7.2(b)).

Using the identity

$$v_z u - u_z v - 2euv = (uv)_z - 2v(u_z + eu),$$

where z is any of x and y, and $e = e(x, y)$ is any function, and integrating the first term of (7.24), we have

$$\begin{aligned} u(x_0, y_0) &= u(x_1, y_1)v(x_1, y_1; x_0, y_0) + \\ &+ \int_{y_1}^{y_0} v(u_y + fu)\,dy + \\ &+ \int_{x_1}^{x_0} v(u_x + gu)\,dx + \frac{1}{2}\iint_D v(x, y; x_0, y_0)F(x, y)\,dxdy. \end{aligned} \tag{7.25}$$

The partial differential equation (7.9) arising in the analysis of the average size of the intersection is

$$L[G] \equiv 2G_{xy} - 2G = 0,$$

provided we multiply both sides by 2.

Then $F = f = g = 0$ and $c = -2$, yielding $L^* = L$. On the other hand, Cauchy data is known along the x-axis and y-axis, i.e. $x_1 = y_1 = 0$. Therefore, the functions given initially are $G(x, 0)$, $G(0, y)$, and

$$\left.\frac{\partial G}{\partial x}\right|_{y=0}, \qquad \left.\frac{\partial G}{\partial y}\right|_{x=0}.$$

The last two functions can be computed from the first two, if $G(x,y)$ is analytic at $(x,y)=(0,0)$, since

$$\begin{aligned}\left.\frac{\partial G}{\partial x}\right|_{y=0} &= \left(\sum_{i,j\geq 0} i\, g_{i,j}\, x^{i-1} y^j\right)_{y=0} = \\ &= \sum_{i\geq 0}(i+1)\, g_{i+1,0}\, x^i = \\ &= \frac{d}{dx}\sum_{i\geq 0} g_{i,0}\, x^i = \frac{d}{dx}G(x,0),\end{aligned}$$

and analogously $G_y(0,y) = d/dy\, G(0,y)$.

In order to obtain the Riemann-Green function, we try a Hadamard series development for it,

$$v(x,y;x_0,y_0) = \sum_{j\geq 0} \frac{v_j \Gamma^j}{j!^2},$$

where v_j are functions to be determined and $\Gamma = (x-x_0)(y-y_0)$.

Imposing the conditions that v should satisfy, we obtain that $v_j = 1$, for all $j \geq 0$. Hence,

$$v(x,y;x_0,y_0) = J_0\left(2\mathrm{i}\sqrt{(x-x_0)(y-y_0)}\right).$$

Changing dummy variables under integration signs by t, and x_0,y_0 by x,y, equation (7.25) becomes,

$$\begin{aligned}G(x,y) &= \int_0^y J_0\left(2\mathrm{i}\sqrt{x(y-t)}\right) \left.\frac{\partial G}{\partial y}\right|_{x=0,y=t} dt + \\ &+ \int_0^x J_0\left(2\mathrm{i}\sqrt{(x-t)y}\right) \left.\frac{\partial G}{\partial x}\right|_{y=0,x=t} dt + \\ &+ G(0,0) J_0(2\mathrm{i}\sqrt{xy}).\end{aligned}$$

Substituting $G_y(0,t)$ and $G_x(t,0)$ by the particular values corresponding to the intersection, we obtain the explicit solution of the homogeneous equation (7.9), given in (7.10).

Chapter 8

Average-case Complexity of the Equality Test

The equality test of trees is an important algorithm in its own, but it is also the core of many other algorithms for trees, for example simplification [CFS90]; and it is closely related to other algorithms such as tree matching [SF83] and unification [CDS89].

It is well known that it takes $O(1)$ steps on average to decide if the trees in the given pair of total size n are equal or not, when the uniform probability model for the input is assumed. In other words, it takes a constant number of steps on the average to decide if they are equal or not, no matter what is the size of the given trees.

In this chapter we present the analysis of the average-case complexity of the canonical algorithm performing the equality test, under the BST probability model.

As in the previous chapter, this analysis includes the solution of a partial differential equation. The partial differential equation describing the average behavior of this algorithm follows from the set of rules of Chapter 5 and more specifically from the considerations we have made in Chapter 6. This equation is but a member of a broader class of equations: algorithms that check hereditary properties have average behaviors that are described by means of similar equations.

An interesting way to see the difference between the intersection algorithm and the equality test is that equality test checks an hereditary property, while the intersection algorithm is closely related to the class of algorithms that check "disjunctive hereditary properties": by disjunctive hereditary property we mean a binary predicate that holds for a given pair if it holds for the pair of left subtrees *or* it holds for the pair of right subtrees, but not for both.

In spite to solve the partial differential equation that describes the average behavior of the equality test, we solve a simpler equation and show the asymptotic equivalence between the solutions of the simpler equation and of the original equation. Using that indirect mechanism, we prove that the average-case complexity of testing equality of a pair of binary trees with total size n is $\Theta(\log n)$ under the BST distribution. This result and that of Chapter 7 clearly contrast with those obtained using the uniform model, that predicts qualitatively identical average behaviors for equality test and the intersection algorithm (constant). They are of different orders of magnitude if the BST probability model is assumed instead.

Section 8.1 is devoted to the formulation of the problem. The application of the translation rules given in Section 5.1 reduces the analysis to the study of a partial differential equation. The difficulty to solve it leads us to seek lower and upper bounds for the average-case complexity of the algorithm. In the same section, we provide a non-trivial upper bound.

In Section 8.2 our efforts are directed towards finding a lower bound. We introduce and analyze a related algorithm that yields a logarithmic lower bound.

In Section 8.3 we deal again with the initial problem and we show that the average-case complexity of the equality test is of the same order of magnitude as the lower bound already known. We get this last result in a rather subtle way: we show that the Riemann-Green function of the partial differential equation associated to the equality test is asymptotically equivalent to the Riemman-Green function of the partial differential equation associated to the problem that provides the lower bound. Hence, their solutions are asymptotically equivalent and the result is proved.

8.1 Testing Equality of Binary Trees.

In order to test the equality of two binary trees, one proceeds as follows : if one of the given trees is a leaf determine with a unique comparison if the other is also a leaf; if both are not leaves, recursively test if their left subtrees are equal; if so, continue testing equality of the right subtrees and then return `true` or `false` depending on the last test. Otherwise, return `false` (see Algorithm 8.1). The result is that a simultaneous preorder traversal of the pair of trees is carried on until a difference is found or the traversal is finished.

```
function Equal?($T_1, T_2$ : BinTree) returns Boolean is
1       if $T_1$ or $T_2$ is $\square$ then
2          return true if both are $\square$, false otherwise
3       else
4          if Equal?($T_1^l, T_2^l$) then
5             return Equal?($T_1^r, T_2^r$)
6          else
7             return false
8          endif
9       endif
endfunction;
```

Algorithm 8.1: *Equality test.*

This proccess follows the algorithmic scheme given in Algorithm 6.1 and therefore the average-case complexity of the equality test is can be described by means of (6.2).

We will denote by $e(T_1, T_2)$ the number of steps that Algorithm 8.1 performs on input (T_1, T_2), and the average of $e(T_1, T_2)$ over all pairs (T_1, T_2) of joint size $|T_1| + |T_2| = n$ will be denoted $\overline{e(n)}$. Hence,

$$\overline{e(n)} = \sum_{|T_1|+|T_2|=n} \Pr(T_1, T_2) \cdot e(T_1, T_2). \tag{8.1}$$

The complexity descriptor $E(z)$ will be used for the average complexity of the equality test over pairs of size n

$$E(z) = \sum_{T_1, T_2 \in \mathcal{B}} \Pr(T_1, T_2)\, e(T_1, T_2)\, z^{|T_1|+|T_2|} \tag{8.2}$$

while $E(x, y)$ is the complexity descriptor corresponding to the average of $e(T_1, T_2)$ over pairs of sizes m and n

$$E(x, y) = \sum_{T_1, T_2 \in \mathcal{B}} \Pr(T_1) \Pr(T_2)\, e(T_1, T_2)\, x^{|T_1|}\, y^{|T_2|}. \tag{8.3}$$

Therefore, $\overline{e(n)} = [z^n]E(z)$ and the two generating functions are related by

$$E(z) = \frac{1}{z} \int_0^z E(t, t)\, dt. \tag{8.4}$$

Notice that we can use $\Pr(\cdot)$ in the complexity descriptors instead of the weight $w(\cdot)$, since both measures coincide for binary trees.

Particularizing Equation (6.2) for the case of the equality test and renaming

$$W(x, y \mid \text{Equality}) = Q(x, y),$$

we have

$$\frac{\partial^2 E}{\partial x \partial y} = \frac{1}{(1-x)^2(1-y)^2} + E(x,y)\left(\frac{1}{(1-x)(1-y)} + Q(x,y)\right). \tag{8.5}$$

Since we are considering binary trees, the size of leaves is null and then the initial conditions of $E(x, y)$ are,

$$\forall x \qquad E(x,0) = \frac{1}{1-x}, \tag{8.6}$$

$$\forall y \qquad E(0,y) = \frac{1}{1-y}. \tag{8.7}$$

We have already discussed about the generating function $Q(x, y)$ in Chapter 6. For binary trees, we have $\mu = \lambda = 1$, and $Q(x, y)$ satisfies $Q_{xy} = Q^2$. Also, and this was already mentioned, $Q(x \cdot y)$ depends, in fact, on the product $x \cdot y$. Defining

$$R(z) = \sum_{T \in \mathcal{B}} \Pr^2(T)\, z^{|T|} \tag{8.8}$$

we have that $Q(x, y) = R(x \cdot y)$. $R(z)$ satisfies the second order ordinary differential equation:

$$z\frac{d^2 R}{dz^2} + \frac{dR}{dz} - R^2(z) = 0, \tag{8.9}$$

with $R(0) = R'(0) = 1$.

Equation (8.9) seems very difficult to be solved explicitly, mainly because its non-linearity. Moreover, $R(z)$ has a finite movable singularity that depends on the initial conditions (see Chapter 9).

In order to obtain the answer for the average-case complexity of the equality, we will devise an upper and a lower bound for it. Our purpose is to squeeze the average-case complexity of the equality test between two non-trivial bounds (i.e., neither its worst-case nor its best-case complexity are of help), giving some approximate idea of its order of magnitude. This approach has been very sucessful since we have been able to prove that the average-case complexity of the equality test actually reaches the lower bound.

A first non-trivial upper bound can be obtained in the following way; we replace in the previous algorithm the conditional execution of the equality test over right subtrees (lines 4–8) by

```
function BackboneEqual?($T_1, T_2$ : BinTree) returns Boolean is
    if $T_1$ or $T_2$ is $\square$ then
        return true if both are $\square$, false otherwise
    else
        return BackboneEqual?($T_1^l, T_2^l$)
    endif
endfunction;
```

Algorithm 8.2: *Backbones equality test.*

return Equal?(T_1^l, T_2^l) **and** Equal?(T_1^r, T_2^r)

This version works correctly but in an inefficient way, since it tests the equality of the right subtrees even when left subtrees are not equal, just as an algorithm testing a "disjunctive hereditary property" ought to do.

The average-case complexity of this dully version coincides with the average-case complexity of the algorithm of intersection of binary trees (since symmetrically tests equality in both pairs of subtrees, when the trees in the original pair are not empty) and then

$$\overline{e(n)} = O(n^{2\sqrt{2}-2}/\sqrt{\log n}),$$

under the assumption that input is distributed according to the BST probability model (see Theorem 7.3).

Recall that both tests (that defined in Algorithm 8.1 and the dully proposed version) have the same worst-case complexity and constant average-case complexity when the probability model is the uniform one. On the contrary, these algorithms have qualitatively different average behavior for the BST model, as we shall see.

8.2 A Lower Bound for Equality.

In the previous section, we found an upper bound to $\overline{e(n)}$. In this section we are going to develop a lower bound, and in this way get a first estimation for $\overline{e(n)}$.

Algorithm 8.2 tests if the leftmost branches, also called *backbones* [Cul85], of two given trees are equal or not. For any given pair of binary trees, Algorithm 8.2 never makes more steps than the equality test algorithm with the same pair of trees as input.

Let $l(T_1, T_2)$ denote the number of steps it takes to decide if T_1 and T_2 have the same backbone. Then $l(T_1, T_2) \leq e(T_1, T_2)$, for all pairs (T_1, T_2), and similarly, the average number of steps done by algorithm `BackboneEqual?` on input of size n, say $\overline{l(n)}$, verifies $\overline{l(n)} \leq \overline{e(n)}$.

Straighforward application of the translation rules of Section 5.1 yield

$$\frac{\partial^2 L}{\partial x \partial y} = \frac{1}{(1-x)^2(1-y)^2} + \frac{L(x,y)}{(1-x)(1-y)}, \tag{8.10}$$

subjected to the initial conditions,

$$\forall x \qquad L(x,0) = \frac{1}{1-x}, \tag{8.11}$$

$$\forall y \qquad L(0,y) = \frac{1}{1-y}. \tag{8.12}$$

where $L(x,y)$ is the complexity descriptor associated to $l(T_1, T_2)$. As usual,

$$\overline{l(n)} = \frac{[z^n]L(z,z)}{n+1}.$$

To solve this equation we proceed as in the analysis of intersection looking for a particular solution. It turns out that

$$\frac{1}{2}\frac{\ln\left(\frac{1}{1-x}\right) + \ln\left(\frac{1}{1-x}\right)}{(1-x)(1-y)}$$

is such a particular solution. Therefore, if we denote $L_h(x,y)$ the solution of the homogeneous equation we have

$$\overline{l(n)} = \frac{1}{n+1}[z^n]L_h(z,z) + H_{n+1} - 1 \tag{8.13}$$

where $H_n \sim \ln n + \gamma + O(1/n)$ denotes the n-th harmonic number.

The homogeneous partial differential equation can be solved by means of Riemann's method. To do so, we use a change of variables to get the homogeneous partial differential equation $2G_{xy} - 2G = 0$ and then apply the method in the "standard" way (see the appendix A at the end of Chapter 7). Finally we obtain the following technical lemma:

Lemma 8.1. *Let $L_h(x,y)$ be the solution of the homogeneous equation corresponding to (8.10). Then*

$$[z^n]L_h(z,z) \sim 3 \cdot [z^n]J_0\left(-2i\ln(1-z)\right),$$

The proof of this lemma is analogous to that of Lemma 7.2.

Using Lemma 7.3 and Lemma 8.1 we get the asymptotic behavior of the coefficients of $L_h(z, z)$. Combining this asymptotic behavior with equation (8.13) we get Theorem 8.1.

Theorem 8.1. *The average-case complexity of the algorithm that tests if the leftmost branches of a pair of binary trees of size n are equal is asymptotically*

$$\begin{aligned} \overline{l(n)} &\sim H_{n+1} - 1 + \frac{3}{2\sqrt{\pi \ln n}} + O(\ln^{-3/2} n) \\ &\sim \ln n + O(1). \end{aligned}$$

8.3 The Average-case Complexity of Equality Testing.

The question that now arises is how far is $\overline{e(n)}$ from the bounds we have obtained in previous sections:

$$\ln n + O(1) \le \overline{e(n)} \le c\, n^{2\sqrt{2}-2}/\sqrt{\ln n}.$$

Is it of the same order of magnitude as any of them?

Let us consider the length of the backbone of binary trees distributed accordingly to the BST probability model. Its expectation is $O(\log n)$, whereas its standard deviation is $O(\sqrt{\log n})$. We conclude that the probability of two trees of size n having the same backbone is small, namely of order $O(1/\sqrt{\log n})$. This fact strongly suggests that $\overline{e(n)} = O(\overline{l(n)})$.

A closer look at equations (8.5) and (8.10) reveals that they are very similar, where the unique difference is the term $Q(x, y)$ in the factor multiplying the unknown function. Moreover, $E(x, y)$ and $L(x, y)$ have Taylor series development only in the region $\{\, (x, y) \mid |x| < 1, |y| < 1 \,\}$, while the domain of convergence of $Q(x, y)$ is larger (see Chapter 9). Therefore, if the behavior of $\overline{e(n)}$ is determined by the singularities of $E(x, y)$, then $Q(x, y)$ can be considered just as a "perturbation" that can be neglected.

This observation leaded us to consider whether there is a relationship between the solutions of both equations, and more precisely how their respective Riemann-Green functions are related.

The Riemann-Green functions are, respectively, the solutions of the homogeneous equations,

$$\frac{\partial^2 \hat{v}}{\partial x \partial y} = \hat{v} \left(\frac{1}{(1-x)(1-y)} + R(x \cdot y) \right) \tag{8.14}$$

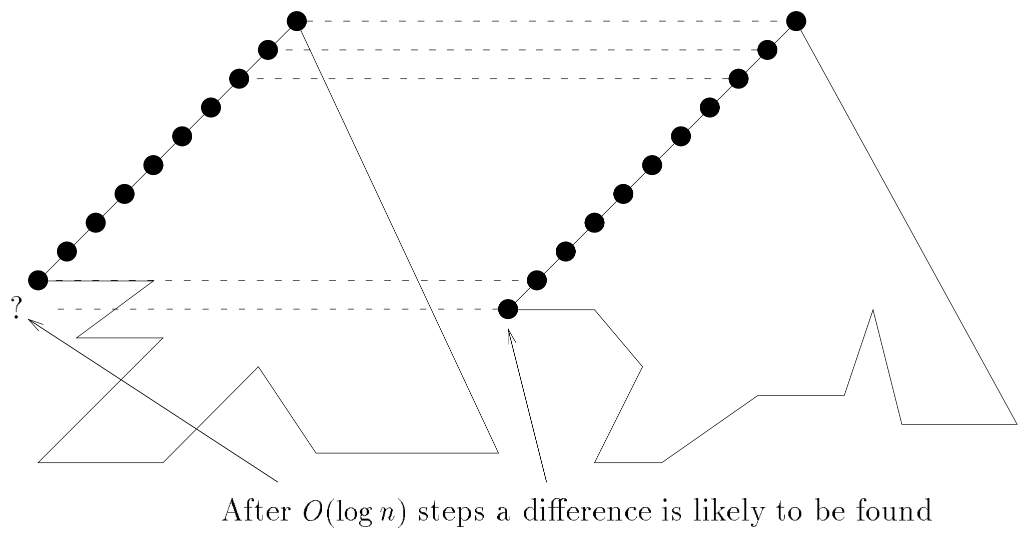


Figure 8.1: *Testing equality of leftmost branches.*

and

$$\frac{\partial^2 v}{\partial x \partial y} = \frac{v}{(1-x)(1-y)}, \tag{8.15}$$

where both functions share the same initial conditions:

$$\begin{aligned} \left.\frac{\partial v}{\partial x}\right|_{y=0} &= \left.\frac{\partial \hat{v}}{\partial x}\right|_{y=0} = \left.\frac{\partial v}{\partial y}\right|_{x=0} = \left.\frac{\partial \hat{v}}{\partial y}\right|_{x=0} = 0, \\ v(0,0;0,0) &= \hat{v}(0,0;0,0) = 1. \end{aligned} \tag{8.16}$$

Therefore, using equation (7.25) we can express $E(x,y)$ in the following way

$$\begin{aligned} E(x,y) &= E(0,0)\,\hat{v}(0,0;x,y) + \\ &+ \int_0^y \hat{v}(0,t;x,y)\left.\frac{\partial E}{\partial y}\right|_{\substack{x=0\\y=t}} dt + \int_0^x \hat{v}(t,0;x,y)\left.\frac{\partial E}{\partial x}\right|_{\substack{x=t\\y=0}} dt + \\ &+ \int_0^x \int_0^y \hat{v}(s,t;x,y)\frac{1}{(1-s)^2(1-t)^2}\,ds\,dt, \end{aligned} \tag{8.17}$$

and in the same way we can express $L(x,y)$ as

$$\begin{aligned} L(x,y) &= L(0,0)\,v(0,0;x,y) + \\ &+ \int_0^y v(0,t;x,y)\left.\frac{\partial L}{\partial y}\right|_{\substack{x=0\\y=t}} dt + \int_0^x v(t,0;x,y)\left.\frac{\partial L}{\partial x}\right|_{\substack{x=t\\y=0}} dt + \\ &+ \int_0^x \int_0^y v(s,t;x,y)\frac{1}{(1-s)^2(1-t)^2}\,ds\,dt. \end{aligned} \tag{8.18}$$

It should be pointed out that both E and L satisfy hyperbolic partial differential equations and that those equations are self-adjoint. In particular, their respective Riemann-Green functions have the following "symmetry" property:

$$\begin{aligned} v(x,y;x_0,y_0) &= v(x_0,y_0;x,y), \\ \hat{v}(x,y;x_0,y_0) &= \hat{v}(x_0,y_0;x,y). \end{aligned}$$

If we make a change of variables

$$\begin{cases} X &= \ln\left(\frac{1}{1-x}\right) \\ Y &= \ln\left(\frac{1}{1-y}\right) \end{cases},$$

the partial differential equations satisfied by E and L are correspondingly transformed, and the new Riemann-Green functions are characterized by simpler equations. The following result establishes the relation between $V(X,Y;X_0,Y_0) = v(x,y;x_0,y_0)$ and $\hat{V}(X,Y;X_0,Y_0) = \hat{v}(x,y;x_0,y_0)$.

Lemma 8.2. *Let $V(X,Y;X_0,Y_0) = v(x,y;x_0,y_0)$ and $\hat{V}(X,Y;X_0,Y_0) = \hat{v}(x,y;x_0,y_0)$. Then as $X \to \infty, Y \to \infty$ along the real axis and if X_0 and Y_0 are real valued, we have,*

$$\hat{V}(X,Y;X_0,Y_0) \sim V(X,Y;X_0,Y_0)$$

The lemma holds also if we interchange X with X_0 and Y with Y_0, by the symmetry of V and $\hat{V}$ w.r.t. the pairs $(X,Y);(X_0,Y_0)$.

Proof. In order to obtain the Riemann-Green functions V and $\hat{V}$ we assume that both have Hadamard series expansions.

Making the change of variables described above

$$\begin{cases} X &= \ln\left(\frac{1}{1-x}\right) \\ Y &= \ln\left(\frac{1}{1-y}\right) \end{cases},$$

equations (8.14,8.15) are transformed into

$$\frac{\partial^2 \hat{V}}{\partial X \partial Y} = \hat{V}\left[1 + R((1-e^{-X})\cdot(1-e^{-Y}))e^{-(X+Y)}\right], \tag{8.19}$$

$$\frac{\partial^2 V}{\partial X \partial Y} = V. \tag{8.20}$$

Imposing conditions (8.16) we find that the coefficients of the Hadamard series development of V are $V_n = 1$, for every $n \geq 0$, and we obtain

$$V(X,Y;X_0,Y_0) = J_0\left(2\mathrm{i}\sqrt{(X-X_0)(Y-Y_0)}\right)$$

We shall use the following formula to relate V_n and $\hat{V}_n$ [Cop75]:

$$v_n = -\frac{n\,v_0}{r^n}\int_0^r \frac{s^{n-1}}{v_0} L[v_{n-1}]\,ds, \quad n > 0, \tag{8.21}$$

where v_n are the coefficients in the Hadamard series expansion of the Riemann-Green function v of an arbitrary linear differential operator L, $x = x_0 + r\sin\theta$, $\quad y = y_0 + r\cos\theta$, and L is given as a function of $\xi = x_0 + s\sin\theta$ and $\eta = y_0 + s\cos\theta$ instead of x and y.

Using (8.21) to determine the coefficients $\hat{V}_n$ in the Hadamard series development of $\hat{V}$ we get

$$\begin{aligned}\hat{V}_0 &= 1\\ \hat{V}_n &= -\frac{n}{r^n}\Big[\int_0^r s^{n-1}\left(\frac{\partial^2\hat{V}_{n-1}}{\partial\xi\partial\eta} - \hat{V}_{n-1}\right)ds - \\ &- \int_0^r s^{n-1}\,\hat{V}_{n-1}\,h(\xi,\eta)\,ds\Big],\end{aligned} \tag{8.22}$$

where $h(X,Y) = R((1-e^{-X})(1-e^{-Y}))e^{-(X+Y)}$ and the we have expressed (X,Y) in polar coordinates: $X = X_0 + r\sin\theta$, $\quad Y = Y_0 + r\cos\theta$. The variables ξ and η are also transformed to polar coordinates: $\xi = X_0 + s\sin\theta$ and $\eta = Y_0 + s\cos\theta$.

Now, we will sketch the induction proof for Lemma 8.2.

For $n = 0$, we have $\hat{V}_0 = V_0 = 1$. Assume that, for all $j < n$, if $X \to \infty$ and $Y \to \infty$, we have $\hat{V}_j \sim V_j$. Then, from (8.22) we obtain

$$\hat{V}_n \sim V_n + \frac{n}{r^n}\int_0^r s^{n-1}\,h(\xi,\eta)\,ds. \tag{8.23}$$

In Chapter 9 we examine in depth the behavior of $R(z)$; it is not difficult to prove that $R(|z|)$ is a monotonic increasing function and that it is analytic in a disk of radius larger than 1. These two facts are crucial to finish the proof; since the modulus of h is bounded, we can use the Mean Value Theorem to evaluate the integral in (8.23).

By hypothesis, both X_0 and Y_0 are real and both X and Y tend to ∞ along the real axis. Hence, r and θ are also real. Then, from (8.23) one gets

$$\hat{V}_n \sim V_n + \frac{nK}{a^n r^n}e^{-(X_0+Y_0)}\int_0^r t^{n-1}e^{-t}\,dt, \tag{8.24}$$

where K is a constant and $a = \sin\theta + \cos\theta$. Since $X \to \infty$ and $Y \to \infty$, it follows that $r \to \infty$ and the equation above is

$$\hat{V}_n \sim V_n + \frac{n!K}{a^n r^n} e^{-(X_0+Y_0)} \sim V_n. \tag{8.25}$$

Hence, Lemma 8.2 follows. ■

Undoing the change of variables, and applying Lemma 8.2 to the partial differential equations (8.17,8.18) we have

$$E(x,y) \sim L(x,y),$$

when $x \to 1$ and $y \to 1$ along the real axis. To translate this asymptotic equivalence of functions to asymptotic equivalence of coefficients one must show that suitable Tauberian conditions hold since we have only proved that the asymptotic equivalence is valid when both x and y tend to 1 along the real axis. In particular, we can show that the Hardy-Littlewood-Karamata theorem can be applied because we have

$$E(z,z) \sim 3J_0(-2\mathrm{i}\ln(1-z)) + \frac{\ln\left(\frac{1}{1-z}\right)}{(1-z)^2}, \qquad z \to 1^-.$$

using once again the asymptotic expansion of $J_0(z)$ for large z, and the fact that all $[z^n]E(z,z)$ are positive and form a monotonic sequence.

It is now immediate to state our main theorem,

Theorem 8.2. *Under the BST probability model, the average-case complexity time of the equality test for pairs of binary trees of size n behaves as*

$$\overline{e(n)} = \Theta(\log n). \tag{8.26}$$

We point out that if we assume that the input is uniformly distributed the average-case complexity of the equality test algorithm is $O(1)$ as for the intersection algorithm; this fact shows that the BST probability model predicts qualitative differences between the complexity of algorithms where the former does not. On the other hand, the uniform and the BST probability models do not discriminate between the average-case complexities of the equality test and of the backbones equality test, since both algorithms have the same average-case complexity for each one of these models. In particular, the average-case complexities are logarithmic for the BST model and constant for the uniform model. This is because the probability of two binary trees being equal is exponentially decreasing with the size of the trees in both models, although at different rates.

Chapter 9

On the Probability of Two BSTs Being Equal

In Chapter 8 we have analyzed the average-case complexity of the canonical algorithm for the equality test between binary trees. The analysis is carried on without explicitly solving the equation associated to the binary predicate of equality, that is one of the so-called hereditary properties (see Chapter 6). In this chapter we address the problem of evaluating the probability that two binary search trees are equal, when the trees are independently drawn and have n internal nodes each.

9.1 The Problem.

Recall that the problem of evaluating the probability that two trees of size n each are equal, reduces to the evaluation of the following bivariate generating function (see Chapter 6):

$$Q(x, y) = \sum_{T \in \mathcal{B}} \Pr{}^2(T)\, x^{|T|}\, y^{|T|}.$$

Since $Q(x, y) = R(x \cdot y)$ we can alternatively evaluate

$$R(z) = \sum_{T \in \mathcal{B}} \Pr{}^2(T)\, z^{|T|} = \sum_{n \geq 0} r_n\, z^n.$$

Here, r_n denotes the probability that two BSTs, of size n each, are equal.

It turns out that this problem is by no means easy to solve, since the power series $R(z)$ is the solution of the non-linear second-order differential equation

$$zy'' + y' - y^2 = 0, \tag{9.1}$$

with initial conditions

$$y(0) = y'(0) = 1. \tag{9.2}$$

The singularities of the solution of (9.1) are not fixed (if it has any at all), as they depend on the initial conditions and on the differential equation itself [Inc56]. This differential equation is a particular case of the Emden-Fowler equation (see [Bel53, Zwi89]), but neither the singularities nor asymptotic expansions of the solution have been studied for (9.1). From now on, we will denote by $\hat{R}(z)$ the unique analytic solution of (9.1),(9.2); so, $R(z)$ is a valid power series expansion for $\hat{R}$ inside the convergence disk of $R(z)$.

Although the problem of evaluating $R(z)$ seems very difficult, recall that the analysis of equality test made in Chapter 8 was still possible, because we only needed to show that the function $R(z)$ is analytic in a disk of radius $\rho > 1$.

The proof of the the last fact and other elementary facts about $R(z)$ is based upon the recurrence obeyed by the coefficients r_n of the power series $R(z)$:

$$r_{n+1} = \frac{\sum\limits_{0 \le k \le n} r_k r_{n-k}}{(n+1)^2}, \qquad r_0 = 1.$$

Our aim is to derive the asymptotic behavior of $r_n = [z^n]R(z)$ for large n. To obtain this objective, we will use the technique of *subtracted singularities* and try to apply one of the transfer lemmas (see Subsection 1.2.2). In particular, the two conditions for the application of a transfer lemma are that we have an asymptotic expansion of the solution of (9.1),(9.2) around the singularity with smallest modulus; and that such an expansion involves functions belonging to a certain restricted class including algebraic and logarithmic functions. Moreover, the asymptotic expansion should hold for some domain of the complex plane enclosing the disk of convergence, and therefore, requires the analytic continuation of $R(z)$ outside this disk (see Subsection 1.2.2).

9.2 The Domain of Analyticity of $R(z)$.

We begin enumerating three easy-to-prove propositions about the power series $R(z)$.

Proposition 9.1.

1. *$R(z)$ is analytic in a disk of finite radius $1 < \rho < \infty$.*

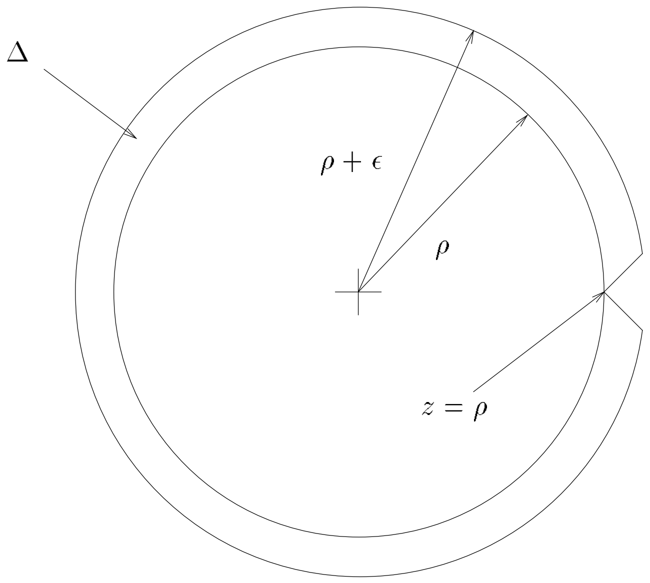


Figure 9.1: *The domain of analyticity of $\hat{R}(z)$ encloses the disk $|z| < \rho$.*

2. *$R(|z|)$ is a monotonic increasing function and for every z in the convergence disk, $|R(z)| \leq R(|z|)$; $R(z)$ has a singularity at the real positive axis at $z = \rho$.*

3. *$R(z)$ and all its derivatives become infinite at the singularity at ρ.*

Proof. In order to prove (1) it suffices to show by induction that

$$\forall n \geq 1, \qquad c_1 n(3\sqrt{3})^{-n} \leq r_n \leq c_2 n(\sqrt{6})^{-n}$$

for some constants c_1 and c_2. Fact (2) follows from the positivity of the r_n. Finally, (3) is proved by reductio ad absurdum, showing that $R(z)$ would not be singular otherwise. ■

The following lemma establishes that the domain of analyticity of $\hat{R}$ properly encloses the disk $|z| < \rho$ (see Figure 9.1).

Lemma 9.1. *Assume that $z = \rho_\theta e^{i\theta}$ is a singularity of $\hat{R}(z)$, for some $\rho_\theta > 0$ and $\theta \neq 0$. Then, $\rho_\theta > \rho$.*

Proof. The proof of this lemma is based upon a continuity argument due to Michèle Loday-Richaud. Suppose that $z = \rho_\theta e^{i\theta}$ is a singularity of $\hat{R}(z)$ in the direction given by $\theta \neq 0$. For any $x \geq 0$, let $v(x) = x\hat{R}'(x)$. Therefore,

$$\hat{R}(z) = 1 + \int_0^z \frac{v(\psi)}{\psi}\, d\psi.$$

Furthermore, let $w(x) = |v(x\, e^{\mathrm{i}\theta})|$ and let $u(x) = |\hat{R}(x\, e^{\mathrm{i}\theta})|$. Neither $w(x)$ nor $u(x)$ are bounded in a neighbourhood of ρ_θ, by reductio ad absurdum. By Proposition 9.1 (2), if x is close enough to the origin, then $w(x) < v(x)$ and $u(x) < \hat{R}(x)$. Fix x_1, such that $0 < x_1 < \rho$, $w(x_1) < v(x_1)$ and $u(x_1) < \hat{R}(x_1)$. Since v, w, u and $\hat{R}$ are continuous functions, there must be x_2 such that $0 < x_2 < x_1 < \rho$ and $w(x_1) < v(x_2)$ and $u(x_1) < \hat{R}(x_2)$.

Let $t \in [0, \rho - x_2) \cap [0, \rho_\theta - x_1)$, where x_1 and x_2 are as before. We prove that $w(x_1 + t) < v(x_2 + t)$ for any such t. Assume that t_1 is the minimum value such that $w(x_1 + t_1) = v(x_2 + t_1)$. Since $w(x_1 + t) \leq v(x_2 + t)$, we have $w'(x_1 + t_1) \geq v'(x_2 + t_1)$. We get then,

$$\hat{R}^2(x_2 + t_1) = v'(x_2 + t_1) \leq w'(x_1 + t_1) \leq u^2(x_1 + t_1),$$

where the last inequality is given by the fact that $|w'| \leq |v'|$, whenever w is differentiable. Hence,

$$0 \leq u(x_1 + t_1) \leq u(x_1) + \int_0^{t_1} \frac{w(x_1 + \psi)}{x_1 + \psi}\, d\psi < \hat{R}(x_2) + \int_0^{t_1} \frac{v(x_2 + \psi)}{x_2 + \psi}\, d\psi = \hat{R}(x_2 + t_1),$$

yielding the contradiction. Therefore, $w(x_1 + t) < v(x_2 + t)$ for any $t \in [0, \rho - x_2) \cap [0, \rho_\theta - x_1)$.

The last argument and the fact that $w(x)$ is not bounded in a neighbourhood of ρ_θ, yield that

$$\rho_\theta \geq \rho + x_1 - x_2,$$

and the lemma follows. ■

9.3 A Family of Solutions around the Singularity.

The asymptotic expansion for $\hat{R}(z)$ can be obtained solving the differential equation (9.1) in a neighborhood of $z = \rho$ (see example 5 in Section 4.3 of the book of Bender and Orszag [BO78], for instance). Making the change of variables $Z = \rho - z$, we obtain

$$(\rho - Z)y'' - y' - y^2 = 0. \tag{9.3}$$

Local analysis shows that the leading singular behavior of $y(Z)$ that becomes infinite at $Z = 0$ is

$$\hat{R}(z) \sim \frac{A}{Z^2}, \qquad Z \to 0,$$

for $A = 6\rho$. However, this leading behavior does not mean that $\hat{R}(z)$ has a pole at $z = \rho$. If there is a pole at $z = \rho$ then $\hat{R}(z) = y(Z) = Z^{-2}F(Z)$ where $F(Z)$ is analytic in a neighborhood of $Z = 0$. But $F(Z)$ is not analytic in a neighborhood of $Z = 0$ and the singularity is not a pole, because the assumption that

$$y(Z) = Z^{-2}\left(a_0 + a_1 Z + a_2 Z^2 + a_3 Z^3 + \cdots\right)$$

leads to a contradiction.

Trying instead a series including terms of the type $Z^m \ln^n Z$ [BO78] no contradiction arises and we can conclude that

$$Y(Z) = d_{-2,0}\, Z^{-2} + d_{-1,0}\, Z^{-1} + \sum_{m,n\geq 0} d_{m,n}\, Z^m \log^n Z \tag{9.4}$$

is a family of solutions of (9.3), depending on two arbitrary constants ρ and h, since $d_{m,n} = d_{m,n}(h,\rho)$ for all $m \geq -2, n \geq 0$. We shall consider only those functions such that ρ is actually the radius of convergence of $R(z)$ and use $Y_h(Z)$ to denote the dependence on h.

Substituting (9.4) into (9.3) and equating terms in $Z^m \log^n Z$, for $m \geq -2, n \geq 0$ we can set up the following recurrences for the coefficients $d_{m,n}$ (we omit some of them):

$$\begin{aligned}
d_{-2,0} &= 6\rho, \\
d_{-1,0} &= -12/5, \\
\ldots & \\
d_{4,0} &= h, \\
\ldots & \\
\rho(n+2)(n+1)d_{m+2,n+2} &= \rho\Big((m+1)(m+2)d_{m+2,n} + (2m+3)(n+1)d_{m+2,n+1}\Big) + \\
&+ (m+1)^2 d_{m+1,n} - (2m+2)(n+1)d_{m+1,n+1} \\
&+ (n+2)(n+1)d_{m+1,n+2} \\
&+ 2d_{-1,0}d_{m+1,n} + 2d_{-2,0}d_{m+2,n} + \sum_{\substack{0\leq i\leq m \\ 0\leq j\leq n}} d_{i,j}\, d_{m-i,n-j}.
\end{aligned}$$

From the recurrences above is straightforward to prove the following lemma, by induction on m:

Lemma 9.2. *For all* $m \geq -2, \quad n \geq 0, \qquad d_{m,n} = 0, \quad$ *if* $n > \left\lfloor \frac{m+2}{6} \right\rfloor$.

From now on, we shall denote $\varphi(m) := \left\lfloor \frac{m+2}{6} \right\rfloor$, since this integer function will appear quite often in the rest of the chapter.

A second inductive proof shows that the following lemma holds.

Lemma 9.3. *Foa all $m \geq -2, \quad n \geq 0$,*

$$d_{m,n} = \frac{(-1)^n}{\rho^{m+1}} K_{m,n}\left(h\rho^5\right), \qquad m \geq -2, n \geq 0$$

where $K_{m,n}(x)$ are polynomials of degree $\varphi(m) - n$ and their coefficients are constants that do not depend on h nor on ρ. Moreover, $K_{4,0}(x) = x$ and $d_{4,0} = h$.

Our recurrences for $d_{m,n}$ translate to recurrences for the coefficients of the polynomials $K_{m,n}(x)$, and Lemma 9.2 implies that $K_{m,n}(x) = 0$ whenever $n > \varphi(m)$.

To finish our collection of lemmas about the polynomials $K_{m,n}$, we state:

Lemma 9.4. *For all $m > 0, n \geq 0$ and for any $\zeta \geq e^{1/6} \approx 1.18136\ldots$,*

$$|K_{m,n}(0)| \leq \frac{\zeta^m}{m^n\, n!}.$$

Proof. We shall only give an sketch of the proof of the lemma. It can be proved by induction on m and n. In fact, the right way to carry on the proof is to use the following ordering for the pairs (m,n): $(m,n) < (m',n')$ if $m < m'$ or, $m = m'$ and $n > n'$. Then induction is done over whole "columns", i.e. from $n = \varphi(m)$ to $n = 0$, and then over m. To simplify notation, let us denote $K_{m,n} = K_{m,n}(0)$.

Assume that the lemma is true for all m such that $0 < m \leq m_0$ and $m_0 > 3$, and for all $n \leq \varphi(m_0)$. Let $m = m_0 - 1$. Since $K_{m+2,n}(0) = 0$ if $n > \varphi(m+2)$, we need only to prove that the lemma is also true for $n \leq \varphi(m+2)$. We start with the following recurrence (valid for all $m + 2 > 4$),

$$\begin{aligned} K_{m+2,n} &= \frac{1}{m^2 - 3m - 10}\Big[(2m+3)(n+1)K_{m+2,n+1} + (n+2)(n+1)K_{m+2,n+2} + \\ &+ ((m+2-1)^2 + 2K_{-1,0})K_{m+1,n} - (2m+2)(n+1)K_{m+1,n+1} + \\ &+ (n+2)(n+1)K_{m+1,n+2} + \sum_{\substack{0\leq i\leq m \\ 0\leq j\leq n}} K_{i,j}K_{m-i,n-j}\Big]. \end{aligned}$$

Applying the induction hypothesis and substituting $K_{0,0}$ and $K_{-1,0}$ by their known values, we have

$$|K_{m+2,n}| \leq \frac{\zeta^{m+2}}{(m+2)^n\, n!}\left[\frac{2m+3}{(m^2+3m-10)(m+2)} + \frac{1}{(m^2+3m-10)(m+2)^2} + \right.$$

$$+ \quad \frac{1}{\zeta}\left(\frac{m+2}{m+1}\right)^n \left(\frac{m^2+2m-19/5}{m^2+3m-10} + \frac{2m+2}{(m^2+3m-10)(m+1)} + \frac{1}{(m^2+3m-10)(m+1)^2}\right) +$$
$$+ \quad \frac{1}{\zeta^2}\frac{(m^2+2m)^n}{m^2+3m-10}\sum_{0<i<m} i^{-n}(m-i)^{-n} + \frac{14}{25(m^2+3m-10)\zeta^2}\left(\frac{m+2}{m}\right)^n\Bigg].$$

Additional manipulations show that the expression inside the square brackets is ≤ 1 for any $m > 0$ and any n such that $0 \leq n \leq \varphi(m+2)$, if $\zeta \geq e^{1/6}$. ■

We remark that the bound provided by this lemma is not tight.

Once we are equipped with the last three lemmas we are ready to prove that $Y_0(Z)$ is analytic in a punctured disk around $Z = 0$ of radius $\delta = e^{-1/6}\rho < \rho$ and cut along the real axis for $Z > 0$. Recall that we already know from Proposition 9.1 that $\rho \geq \sqrt{6}$ and hence $\delta > 2.0$.

Lemma 9.5. *The function*

$$Y_0(Z) = d_{-2,0}\, Z^{-2} + d_{-1,0}\, Z^{-1} + \sum_{m,n\geq 0} d_{m,n}\, Z^m \log^n Z$$

is analytic in the domain $D = \{\, Z \,|\, 0 < |Z| < e^{-1/6}\rho, 0 < |\arg(Z)| \leq \pi \,\}$.

Proof. By definition, the functions

$$\frac{d_{-2,0}}{Z^2}, \quad \frac{d_{-1,0}}{Z}, \quad \text{and } d_{m,n} Z^m \log^n Z$$

are analytic in D. Hence, we ought to prove that

$$\sum_{m,n\geq 0} d_{m,n}\, Z^m \log^n Z$$

is uniformly convergent for any Z in D. Lemmas 9.3 and 9.4 provide the adequate majorization for $|d_{m,n}|$; for $\zeta = \exp(1/6)$, we have

$$|d_{m,n}| \leq \rho^{-(m+1)}\frac{\zeta^m}{m^n\, n!}.$$

Let Z_0 be any complex number inside D. By Lemma 9.2, $d_{m,n} = 0$ whenever $n > \varphi(m)$. If $|Z_0| < 1$ then $|Z_0 \log Z_0| \leq |Z_0|^{1/2}(1+\pi)$ and we can bound the absolute value of each of the summands by

$$|d_{m,n}\, Z_0^m\, \log^n Z_0| \leq \rho^{-(m+1)}\frac{\zeta^m}{m^n\, n!}|Z_0|^{m-n/2}(1+\pi)^{n/2}.$$

But as $m - n/2 \geq m/2$, we get

$$|d_{m,n}\, Z_0^m\, \log^n Z_0| \leq \rho^{-(m+1)} \frac{\zeta^m}{m^n\, n!} |Z_0|^{m/2} (1+\pi)^{n/2},$$

from where it follows that

$$\sum_{m \geq 0} \sum_{n=0}^{n=\varphi(m)} \rho^{-m} \frac{\zeta^m}{m^n\, n!} |Z_0|^{m/2} (1+\pi)^{n/2}$$

is convergent. Analogously, if $1 \leq |Z_0| < e^{-1/6}\rho$, $|Z_0 \log Z_0| \leq |Z_0|(\log \rho + \pi)$ and a similar bound whose sum is convergent can be found.

This proves that $\sum_{m,n \geq 0} d_{m,n} Z^m \log^n Z$ uniformly converges and hence that $Y_0(Z)$ is analytic in D. ■

9.4 Asymptotic Behavior of the n-th Coefficient of $R(z)$.

Up to this point we have already shown that $\hat{R}(z)$ is analytic in a domain properly enclosing the disk of convergence of $R(z)$. Also, that there is a family $Y(Z)$ of solutions of the differential equation around the singularity at ρ and that $Y_0(Z)$ is analytic in a region of radius $e^{-1/6}\rho$ centered at ρ. The next step would be proving that there is some domain where one of the $Y_h(Z)$'s is a valid asymptotic expansion of $\hat{R}(z)$. Nevertheless, we have been not able to do so and therefore we introduce the following conjecture

Conjecture 9.1. *For some z such that both $R(z)$ and $Y_0(\rho - z)$ are analytic*

$$\begin{aligned} Y_0(\rho - z) &= R(z), \\ -Y_0'(\rho - z) &= R'(z). \end{aligned}$$

There is overwhelming empirical evidence that the conjecture actually holds. For instance, numerical computations of ρ using Hadamard's expression of the convergence radius ($\rho^{-1} = \lim_{n\to\infty} r_n^{(1/n)}$) and the ones done assuming the conjecture are coincident in all significative digits, giving $\rho = 3.140857701\ldots$. On the other hand, approximations of $R(z)$ by $Y_0(z)$, along the real axis, are good. Finally, it can be shown that for any h such that $|h\,\rho^5| < 1$, then $Y_h(z)$ is analytic on a domain $D_h = \{\, Z \,|\, 0 < |Z| < \delta, 0 < |\arg(Z) \leq \pi \,\}$, with $\delta \approx 0.35$. If we require that $Y_h(Z_0) = R(\rho - Z_0)$ and $-Y_h'(Z_0) = R'(\rho - Z_0)$ for some $Z_0 \in D_h$, and solve these equations for an unknown h, our numerical calculations show that

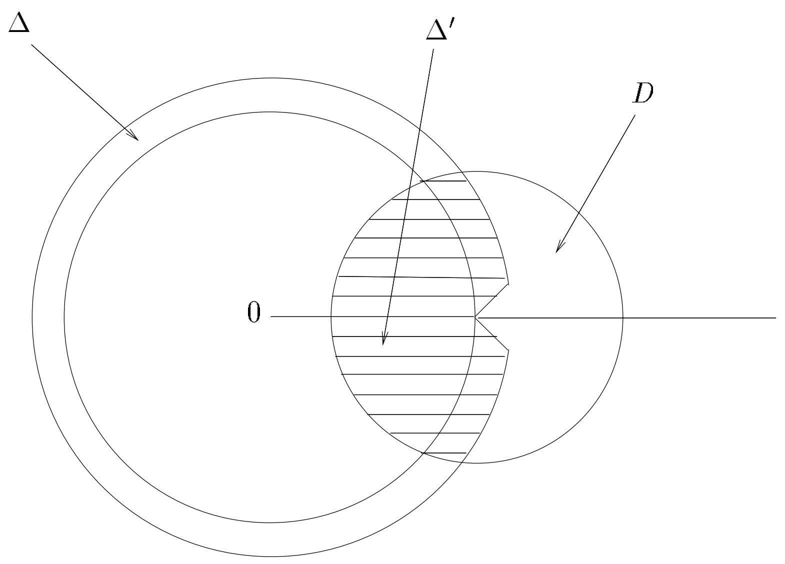


Figure 9.2: *Domains of analyticity of $\hat{R}(z)$ and $Y_0(Z)$.*

$|h| \to 0$ as we increase the accuracy of the approximations of $Y_h(Z_0), Y_h'(Z_0), R(\rho - Z_0)$ and $R'(\rho - Z_0)$.

If the conjecture is true, the rest of the argumentation is simple. By the existence and uniqueness theorem for analytic solutions of ordinary differential equations, $\hat{R}(z)$ and $Y_0(Z)$ must coincide in a neighbourhood of the point where we conjecture the functions are equal. By the analytic continuation principle, it can be stated that both functions are equal in $\Delta' = D \cap \Delta$, where Δ is the domain of analyticity of $\hat{R}(z)$ (see Figure 9.2).

Therefore, all conditions for the application of a transfer lemma are fulfilled since $\hat{R}(z)$ is analytic in the domain Δ' and as $z \to \rho$ inside Δ' we have

$$\begin{aligned}\hat{R}(z) &= \frac{d_{-2,0}}{(\rho - z)^2} + \frac{d_{-1,0}}{(\rho - z)} + \\ &+ \sum_{m \geq 0} \sum_{0 \leq n \leq m} d_{m,n} (\rho - z)^m \log^n (\rho - z).\end{aligned}$$

We summarize our results in the following claim.

Claim 9.1. *The probability that two binary search trees are equal, if they are independently drawn and have size n each is, provided that Conjecture 9.1 holds,*

$$r_n \sim 6n\rho^{-n-1} - 12/5\rho^{-n-1} + \rho^{-n-1} \sum_{\substack{j \geq 0 \\ 0 \leq k \leq \varphi(j)}} d'_{j,k} n^{-j-1} \log^{k-1} n,$$

where $\rho = 3.140857701\ldots$, *and the constants* $d'_{j,k}$ are

$$d'_{j,k} \sim K_{j,k}(0)(-1)^{k+1}\log^k \rho$$

Notice that the leading behavior of the asymptotic estimate of r_n does not depend on h.

Part IV

Conclusions and Open Problems

This part contains the description and the first steps of the analysis of the *common subexpression* problem.We obtain an explicit expression for the generating function associated to the size of the compacted tree, and leave open the estimation of asymptotic behavior of the coefficients of that generating function.

We also summarize the main conclusions of the thesis, discussing the results obtained and its significance.

IV.1 Analysis of the Common Subexpression Problem.

This section presents a problem that is quite different from those problems that we have examined in thesis. The problem of *common subexpression*, also known as *tree compaction* problem, is an interesting representant of a different family of problems. The canonical algorithm that performs the tree compaction is one of the so-called *bottom-up* algorithms [CFS90]. We are particularly interested in the evaluation of the average size of the compacted tree, as a function of the size of the original tree.

Any tree can be represented in a one-to-one correspondence by a direct acyclic graph (DAG, for short), where common subtrees are factored and represented only once. In the implementation, several pointers will point to the root of any common subtree, and the resulting structure is a DAG (see Figure IV.3). The compaction process can be done in linear time, and is routinely used in Lisp systems, computer algebra and symbolic manipulation systems and compilers (mainly in the register allocation, code generation and optimization steps).

As the size of the DAG will be most of the times less than the size the original tree, the expected savings in storage is the quantity that we are interested in analyzing. The analysis of the size of the DAG corresponding to a tree of size n was done for the uniform model and related variants by Flajolet, Sippala, and Steyaert [FSS90].

A similar approach to that followed in the mentioned paper, works for the BST model, but algebraic equations are replaced by differential ones, and the "classification" on sizes that works for the uniform model seems no longer possible for the BST model.

Let $K(T)$ be the size of the DAG obtained when tree T is compacted. Then the average size of a DAG coming from a tree of size n is

$$\overline{K(n)} = \sum_{|T|=n} \Pr(T)\, K(T),$$

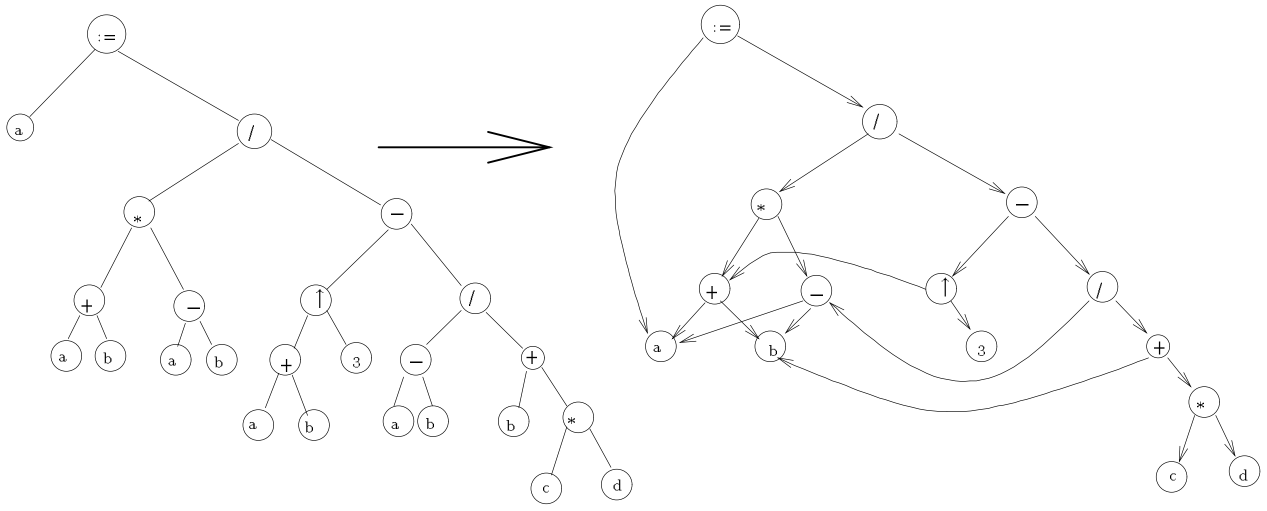


Figure IV.3: *Compacted representation (DAG) of a labelled binary tree.*

where $\Pr(T)$ denotes the probability of the tree T in the BST model.

The characteristic $K(T)$ turns out to be

$$K(T) = \sum_{\substack{u \subset T \\ u \neq}} 1,$$

where $u \subset T$ means that u is a subtree of T and $\Box$ denotes the empty tree; therefore, every subtree u of T contributes by 1 to $K(T)$ no matter the number of times it appears.

Hence, if we define the generating function $K(z)$ for the sequence $\{\overline{K(n)}\}_{n \geq 0}$ as usual, then we have

$$\begin{aligned} K(z) &= \sum_{n \geq 0} \overline{K(n)}\, z^n = \sum_{T \in \mathcal{B}} \Pr(T)\, K(T)\, z^{|T|} = \\ &= \sum_{T \in \mathcal{B}} \Pr(T) \left(\sum_{\Box \neq u \subset T} 1 \right) z^{|T|} \\ &= \sum_{(u,T) \in \mathcal{F}} \Pr(T)\, z^{|T|}, \end{aligned}$$

with $\mathcal{F} = \{(u, T) \,|\, T \in \mathcal{B}, \Box \neq u \subset T\}$.

The set $\mathcal{F}$ can be expressed as

$$\mathcal{F} \quad = \quad \bigcup_{u \in \mathcal{B} - \Box} \{u\} \times \mathcal{A}_u,$$

$$\mathcal{A}_u = \{T \in B \mid u \subset T\} = u + \circ(\mathcal{A}_u, \mathcal{B} - \mathcal{A}_u) + \circ(\mathcal{B} - \mathcal{A}_u, \mathcal{A}_u) + \circ(\mathcal{A}_u, \mathcal{A}_u).$$

And hence

$$\begin{aligned} K(z) &= \sum_{u \in \mathcal{B} - \Box} A_u(z), \\ A_u(z) &= \sum_{T \in \mathcal{A}_u} \Pr(T)\, z^{|T|}. \end{aligned}$$

Notice that the last equations are valid for any probability model.

Using the recursive decomposition of $\mathcal{A}_u$ as well as the definition of $\Pr(T)$ we can easily derive the following differential equation for $A_u(z)$

$$\frac{dA_u}{dz} = |u|\, Pr(u)\, z^{|u|-1} + \frac{2A_u(z)}{1-z} - A_u^2(z), \tag{IV.1}$$

subject to initial conditions $A_u(0) = 0$, $A'_u(0) = 0$ if $|u| > 1$ and $A'_u(0) = 1$ if $|u| = 1$.

Let $y(z) \equiv A_u(z)$, $n = |u| - 1$ and $\lambda = (n+1)\Pr(u)$. Then (IV.1) is

$$y' = \frac{2y}{1-z} - y^2 + \lambda z^n,$$

which is a Ricatti differential equation [Inc56]. If we take

$$y(z) = \frac{w'(z)}{w(z)} + \frac{1}{1-z},$$

the original differential equation is transformed into

$$w'' - \lambda\, z^n\, w = 0,$$

whose solution is (see, for eample [AS64, (9.1.51)])

$$w(z) = \sqrt{z}\, \mathcal{C}_\nu \left(2\mathrm{i}\sqrt{\lambda}\, \frac{z^{n/2+1}}{n+2} \right)$$

where $\nu = \frac{1}{n+2}$ and $\mathcal{C}_\nu$ is a linear combination of the Bessel functions J_ν and Y_ν (a cylinder function).

Hence, $K(z)$ can be expressed as

$$K(z) = \frac{d}{dz} \sum_{u \in \mathcal{B} - \Box} \ln \left(\frac{z^{1/2}}{1-z}\, \mathcal{C}_\nu \left(2\mathrm{i}\sqrt{\lambda}\, \frac{z^{n/2+1}}{n+2} \right) \right).$$

Notice that the order of the cylinder functions in $K(z)$ depends only on the size of u, but their arguments depend also on $\Pr(u)$.

IV.2 Conclusions.

The recursive nature of the BST model allows the average-case analysis of algorithms under this model, by means of the symbolic operator method. The resulting equations over generating functions for the BST model are differential equations. For the case of algorithms dealing with pairs of trees the equations are partial differential. Specific mathematical tools, such as the Riemann's method, are needed in order to obtain the asymptotic estimates of the coefficients of the generating functions that satisfy those partial differential equations.

We have explored some simple recursion schemes over pairs of trees. We have been able to perform the complete analysis of the average size of the intersection of trees, and of the average-case complexity of the equality test algorithm. The average size of the intersection of a pair of trees of size n turns out to be $\Theta(n^{2\sqrt{2}-2}/\sqrt{\log n})$. The average-case complexity of the equality test is $\Theta(\log n)$. These results contrast with the results corresponding to the uniform model, that are in both cases $O(1)$. We say that the BST model is able to discriminate between the equality test algorithm and the intersection algorithm, because their average-case complexities are qualitatively different. This is not the case for the uniform model.

Closely related to the problem of equality is the problem of evaluating the probability that two binary trees of the size n are equal. It appears to be $O(\rho^{-n}\, n)$. We have also examined other problems associated to hereditary properties, that are characterized by the same kind of differential equations.

A lot of work has still to be done for a better comprehension of the BST model and the average-case analysis under the BST model. However, much of the mathematical problems that arise in these studies are rather difficult, as this thesis shows.

A possible line of research is the development of complex analysis techniques to cope with multivariate complex functions. Almost all of the complex analysis techniques in use in the area of average-case analysis deal with functions of a single variable. For instance, much is known about the nature of the singularities and the asymptotic behavior of the Taylor coefficients of functions satisfying linear ordinary differential equations. A lot of information can be extracted without even solving the differential equations. Similar results for partial differential equations would be of great interest for the average-case analysis under the BST model, where this type of equations arise naturally.

Besides the BST model, it would be interesting to study other probability models, such as the mST, DST or quadtree model. Furthermore, these models could be generalized to arbitrary families of trees and to pairs of trees. Just to give an example, the extension of the DST model to pairs of trees is quite similar to the extension of the BST model to pairs, and the type of equations that appear in the average-case analysis under the extended DST model are, in general, partial difference-differential equations.

Bibliography

[ACF$^+$91] L. Albert, R. Casas, F. Fages, A. Torrecillas, and P. Zimmermann. Average case analysis of unification algorithms. In C. Choffrut and M. Jantzen, editors, *Proc. of the 8th Symposium on Theoretical Aspects of Computer Science*, pages 196–213. Lecture Notes in Computer Science, Springer-Verlag, 1991.

[AHU76] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, MA, 1976.

[AHU83] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, Reading, MA, 1983.

[AS64] M. Abramowitz and I.A. Stegun, editors. *Handbook of Mathematical Functions*. Dover Publ., New York, 1964.

[Bae87] R. Baeza-Yates. Some average measures in m-ary search trees. *Information Processing Letters*, 25:375–381, Jul 1987.

[BCDM92] R. Baeza-Yates, R. Casas, J. Díaz, and C. Martínez. On the average size of the intersection of binary trees. *SIAM J. on Computing*, 21(1):24–32, Feb 1992.

[Bel53] J. Bellman. *Stability Theory of Differential Equations*, chapter 7. MacGraw-Hill, 1953.

[Ben74] E.A. Bender. Asymptotic methods in enumeration. *SIAM Review*, 16(4):485–515, Oct 1974.

[Ben75] J.L. Bentley. Multidimensional binary search trees used for associative searching. *Comm. of the ACM*, 18(9):509–517, Sep 1975.

[BF79] J.L. Bentley and J.H. Friedman. Data structures for range searching. *ACM Comp. Surveys*, 11(4):397–409, Dec 1979.

[BFS92] F. Bergeron, Ph. Flajolet, and B. Salvy. Varieties of increasing trees. In J.C. Raoult, editor, *Colloquium in Trees, in Algebra and Programming.* Springer-Verlag, Lecture Notes in Computer Science, 1992. (To be published).

[BO78] C.M. Bender and S.A. Orszag. *Advanced Mathematical Methods for Scientists and Engineers.* McGraw-Hill, 1978.

[Car61] H. Cartan. *Théorie Élémentaire des Fonctions Analytiques d'une oú plusieurs variables complexes.* Hermann, Paris, 1961.

[CDM91a] R. Casas, J. Díaz, and C. Martínez. Average-case analysis on simple families of trees using a balanced probability model. In M. Delest, G. Jacob, and P. Leroux, editors, *3rd Formal Power Series*, pages 133–143, Bourdeaux, May 1991. LaBRI. (Also to be published in Theoretical Computer Science).

[CDM91b] R. Casas, J. Díaz, and C. Martínez. Statistics on random trees. In J. Leach, B. Monien, and M. Rodríguez-Artalejo, editors, *18th. Int. Colloquium on Automata, Languages and Programming*, volume 510, pages 186–203. Springer-Verlag, Lecture Notes in Computer Science, Jul 1991.

[CDS89] R. Casas, J. Díaz, and J.-M. Steyaert. Average-case analysis of Robinson's unification algorithm with two different variables. *Information Processing Letters*, 31:227–232, Jun 1989.

[CE70] E. Coffman, Jr. and J. Eve. File structures using hashing functions. *Comm. of the ACM*, 13:427–432 and p.436, 1970.

[CFS90] R. Casas, M.I. Fernández-Camacho, and J.-M. Steyaert. Algebraic simplification in computer algebra: An analysis of bottom-up algorithms. *Theoretical Computer Science*, 74:273–298, 1990.

[CH62] R. Courant and D. Hilbert. *Methods of Mathematical Physics*, volume 2. Inter-Science J. Wiley, 1962.

[CKS89] C. Choppy, S. Kaplan, and M. Soria. Complexity analysis of term-rewriting systems. *Theoretical Computer Science*, 67(2):261–282, 1989.

[CL73] C.L. Chang and R.C. Lee. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, New York, 1973.

[CLR90] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms.* The MIT Press, Cambridge, MA, 1990.

[Col82] A. Colmerauer. Prolog ii, manuel de référence et modèle théorique. Technical report, Groupe d'Intelligence Artificielle, Univ. d'Aix-Marseille II, Mar 1982.

[Com74] L. Comtet. *Advanced Combinatorics.* Reidel, D., Dordrecht, 1974.

[Cop75] E.T. Copson. *Partial Differential Equations.* Cambridge University Press, Cambridge, 1975.

[Cul85] J. Culberson. The effect of updates in binary search trees. In *ACM Symposium on the Theory of Computation*, pages 205–212, 1985.

[DB58] N.G. De Bruijn. *Asymptotic Methods in Analysis.* Dover Publ., New York, 1958.

[Dev86] L. Devroye. A note on the height of binary search trees. *J.ACM*, 33(3):489–498, Jul 1986.

[Dev90] L. Devroye. On the height of random m-ary search trees. *Random Structures and Algorithms*, 1(2):191–203, 1990.

[Fel68] W. Feller. *An Introduction to Probability Theory and its Applications, vol. I.* J. Wiley, New York, 1968.

[Fla85] Ph. Flajolet. Approximate counting: A detailed analysis. *BIT*, 25:113–114, 1985.

[Fla88a] Ph. Flajolet. Mathematical methods in the analysis of algorithms and data structures. In Egon Börger, editor, *Trends in Theoretical Computer Science*, chapter 6, pages 225–304. Computer Science Press, 1988.

[Fla88b] Ph. Flajolet. Random tree models in the analysis of algorithms. In *Performance'87*, pages 171–187, 1988.

[FO82] Ph. Flajolet and A.M. Odlyzko. The average height of binary trees and other simple trees. *JCSS*, 25(2):171–213, Oct 1982.

[FO90] Ph. Flajolet and A.M. Odlyzko. Singularity analysis of generating functions. *SIAM J. Disc. Math.*, 3(2):216–240, May 1990.

[FP86] Ph. Flajolet and C. Puech. Partial match retrieval of multidimensional data. *J. ACM*, 33(2):371–407, Apr 1986.

[Fra85] J. Franco. Sensitivity of probabilistic results on algorithms for np-complete problems to input distributions. *SIGACT News*, 17(1):40–59, 1985.

[FRS85] Ph. Flajolet, M. Régnier, and R. Sedgewick. Some uses of the mellin integral transform in the analysis of algorithms. In A. Apostolico and Z. Galil, editors, *Combinatorial Algorithms on Words*, volume 12 of *NATO Advanced Science Institute Series*, pages 241–254. Springer-Verlag, 1985.

[FS86] Ph. Flajolet and R. Sedgewick. Digital search trees revisited. *SIAM J. on Computing*, 15(3):748–767, Aug 1986.

[FS87] Ph. Flajolet and J.-M. Steyaert. A complexity calculus for recursive tree algorithms. *Mathematical Systems Theory*, 19:301–331, 1987.

[FSS90] Ph. Flajolet, P. Sipala, and J.-M. Steyaert. Analytic variations on the common subexpression problem. In M.S. Paterson, editor, *17th. Int. Colloquium on Automata, Languages and Programming*, volume 20, pages 220–234. Springer-Verlag, Lecture Notes in Computer Science, 1990.

[FSZ89] Ph. Flajolet, B. Salvy, and P. Zimmermann. Lambda-upsilon-omega the 1989 cookbook. Technical Report 1073, INRIA-Rocquencourt, Aug 1989.

[FSZ91] Ph. Flajolet, B. Salvy, and P. Zimmermann. Automatic average-case analysis of algorithms. *Theoretical Computer Science, Series A*, 79(1):37–109, Feb 1991.

[GB91] G.H. Gonnet and R. Baeza-Yates. *Handbook of Algorithms and Data Structures - In Pascal and C.* Addison-Wesley, Wokingham, UK, 2nd edition, 1991.

[GJ83] I. Goulden and D. Jackson. *Combinatorial Enumerations.* J. Wiley, New York, 1983.

[GK82] D. Greene and D.E. Knuth. *Mathematics for the Analysis of Algorithms.* Birkhauser, Boston, 2nd edition, 1982.

[GKP89] R.L. Graham, D.E. Knuth, and O. Patashnik. *Concrete Mathematics.* Addison-Wesley, Reading, MA, 1989.

[Har49] G.H. Hardy. *Divergent Series.* Oxford University Press, London, 1949.

[Hay56] W.K. Hayman. A generalization of Stirling's formula. *J. reine Angew. Math.*, 196:67–95, 1956.

[Hen77] P. Henrici. *Applied and Computational Complex Analysis*, volume 2. Wiley Interscience, New York, 1977.

[HO82] C.M. Hoffmann and M. J. O'Donnell. Pattern matching in trees. *J. ACM*, 29(1):68–95, Jan 1982.

[HS68] B. Harris and L. Schoenfeld. Asymptotic expansions for the coefficients of analytic functions. *Illinois J. Math.*, 12:264–277, 1968.

[Inc56] E.L. Ince. *Ordinary Differential Equations.* Dover Publ., New York, 1956.

[Kar86] R. Karp. Combinatorics, complexity and randomness. *Comm. of the ACM*, 29(2):98–119, Feb 1986.

[KB70] D.E. Knuth and P. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–397. Pergamon Press, 1970.

[Kem84] R. Kemp. *Fundementals of the Average-Case Analysis of Particular Algorithms.* Series in Computer Science. J. Wiley, New York, 1984.

[KMR72] R.M. Karp, R.E. Miller, and A. L. Rosenberg. Rapid identification of repeated patterns in strings, trees and arrays. In *ACM Symposium on the Theory of Computation*, pages 125–136, 1972.

[KN73] A. Konheim and D. Newman. A note on growing binary trees. *Discrete Mathematics*, 4:57–63, 1973.

[Kni89] K. Knight. Unification: a multidisciplinary survey. *ACM Comp. Surveys*, 21:93–124, 1989.

[Knu68] D.E. Knuth. *The Art of Computer Programming: Fundamental Algorithms*, volume 1. Addison-Wesley, Reading, MA, 1968.

[Knu73] D.E. Knuth. *The Art of Computer Programming: Sorting and Searching*, volume 3. Addison-Wesley, Reading, MA, 1973.

[Lan85] S. Lang. *Complex Analysis.* Springer-Verlag, New York, 2nd edition, 1985.

[Mah86] H.M. Mahmoud. On the average internal path length of m-ary search trees. *Acta Informatica*, 23:111–117, 1986.

[Mah92] H.M. Mahmoud. *Evolution of Random Search Trees.* Wiley Interscience, New York, 1992.

[Mar91] C. Martínez. Average-case analysis of equality of binary trees under the BST probability model. In L. Budach, editor, *Fundamentals of Computation Theory*, volume 529, pages 350–359. Springer-Verlag, Lecture Notes in Computer Science, 1991.

[Mil78] R. Milner. A theory of type polymorphism in programming. *JCSS*, 17:348–375, 1978.

[MM78] A. Meir and J.W. Moon. On the altitude of nodes in random trees. *Canad. J. Math*, 30(5):997–1015, 1978.

[MM82] A. Martelli and U. Montanari. An efficient unification algorithm. *IEEE Trans. Programming Lang. Systems*, 4:258–282, 1982.

[MP89] H.M. Mahmoud and B. Pittel. Analysis of the space of search trees under the random insertion algorithm. *Journal of Algorithms*, 10:52–75, 1989.

[MU71] R. Muntz and R. Uzgalis. Dynamic storage allocation for binary search trees in a two-level memory. In *Proc. of Princeton Conf. on Information Sciences and Systems*, volume 4, pages 345–349, Princeton University, NJ, 1971. Department of Electrical Engineering.

[Nie74] J. Nievergelt. Binary search trees and file organization. *ACM Comp. Surveys*, 6(3):195–207, Sep 1974.

[Odl82] A.M. Odlyzko. Periodic oscillations of coefficients of power series that satisfy functional equations. *Adv. Math.*, 44:50–63, 1982.

[OR85] A.M. Odlyzko and L.B. Richmond. Asymptotic expansions for the coefficients of analytic generating functions. *Aequationes Math.*, 28:50–63, 1985.

[PB85] P. Purdom and C. Brown. *The Analysis of Algorithms.* Holt, Rinehart and Winston, New York, 1985.

[Pit85] B. Pittel. Asymptotical growth of a class of random trees. *Annals of Probability*, 13:414–427, 1985.

[PW78] M.S. Paterson and M.N. Wegman. Linear unification. *JCSS*, 16:158–167, 1978.

[Rie53] B. Riemann. *Gesammelte Mathematische Werke*, pages 156–178. Dover Press, New York, 1953. (Reprint).

[Rob71] J.A. Robinson. Computational logic: the unification computation. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 6. Elsevier, New York, 1971.

[Rob82] J.M. Robson. The asymptotic behaviour of the height of binary search trees. *Australian Computer Science Communications*, 4(1):88–98, 1982.

[Rot75] G.-C. Rota. The idea of generating function. In *Finite Operator Calculus*, chapter 3. Academic Press, 1975.

[Sed83] R. Sedgewick. Mathematical analysis of combinatorial algorithms. In *Probability Theory and Computer Science*, pages 124–205. Academic Press, 1983.

[Sed88] R. Sedgewick. *Algorithms.* Addison-Wesley, Reading, MA, 2nd edition, 1988.

[SF83] J.-M. Steyaert and Ph. Flajolet. Patterns and pattern-matching in trees: an analysis. *Information and Control*, 58(1-3):19–58, 1983.

[Sta86] R.P. Stanley. *Enumerative Combinatorics*, volume 1 of *Mathematics Series.* Wadsworth & Brooks/Cole, 1986.

[Ste84] J.-M. Steyaert. *Structure et Complexité des Algorithmes.* PhD thesis, Université Paris 7, Apr 1984.

[Tit39] E.C. Titchmarsh. *The Theory of Functions.* Oxford University Press, Oxford, 1939.

[VF90] J.S. Vitter and Ph. Flajolet. Average-case analysis of algorithms and data structures. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A, chapter 9. North-Holland, 1990.

[Wat44] G.N. Watson. *A Treatise on the Theory of Bessel Functions.* Cambridge University Press, Cambridge, 2nd edition, 1944.

[Wil90] H. Wilf. *Generatingfunctionology.* Academic Press, San Diego, CA, 1990.

[Zwi89] D. Zwillinger. *Handbook of Differential Equations.* Academic Press, San Diego, CA, 1989.

List of Tables

List of Figures