Análisis de Algoritmos: Teoría y Aplicaciones

Conrado Martínez

Univ. Politècnica de Catalunya

Universidad de Zaragoza
Junio 2008

Introducción

El Análisis de Algoritmos tiene como objetivo describir de manera muy precisa y detallada la eficiencia de algoritmos, en particular, en situaciones "típicas".

- Los escenarios de caso peor pueden ser muy infrecuentes $\Longrightarrow$ el análisis de caso peor pierde relevancia y capacidad predictiva
- La notación asintótica oculta demasiados detalles $\Longrightarrow$ perdemos capacidad comparativa entre algoritmos asintóticamente equivalentes o muy próximos
- Queremos conocer la eficiencia en casos típicos o mejor aún la distribución probabilística
- Para algoritmos aleatorizados, el análisis probabilístico es la herramienta adecuada

Introducción

El Análisis de Algoritmos tiene como objetivo describir de manera muy precisa y detallada la eficiencia de algoritmos, en particular, en situaciones "típicas".

- Los escenarios de caso peor pueden ser muy infrecuentes $\Longrightarrow$ el análisis de caso peor pierde relevancia y capacidad predictiva
- La notación asintótica oculta demasiados detalles $\Longrightarrow$ perdemos capacidad comparativa entre algoritmos asintóticamente equivalentes o muy próximos
- Queremos conocer la eficiencia en casos típicos o mejor aún la distribución probabilística
- Para algoritmos aleatorizados, el análisis probabilístico es la herramienta adecuada

Introducción

El Análisis de Algoritmos tiene como objetivo describir de manera muy precisa y detallada la eficiencia de algoritmos, en particular, en situaciones "típicas".

- Los escenarios de caso peor pueden ser muy infrecuentes $\Longrightarrow$ el análisis de caso peor pierde relevancia y capacidad predictiva
- La notación asintótica oculta demasiados detalles $\Longrightarrow$ perdemos capacidad comparativa entre algoritmos asintóticamente equivalentes o muy próximos
- Queremos conocer la eficiencia en casos típicos o mejor aún la distribución probabilística
- Para algoritmos aleatorizados, el análisis probabilístico es la herramienta adecuada

Introducción

El Análisis de Algoritmos tiene como objetivo describir de manera muy precisa y detallada la eficiencia de algoritmos, en particular, en situaciones "típicas".

- Los escenarios de caso peor pueden ser muy infrecuentes $\Longrightarrow$ el análisis de caso peor pierde relevancia y capacidad predictiva
- La notación asintótica oculta demasiados detalles $\Longrightarrow$ perdemos capacidad comparativa entre algoritmos asintóticamente equivalentes o muy próximos
- Queremos conocer la eficiencia en casos típicos o mejor aún la distribución probabilística
- Para algoritmos aleatorizados, el análisis probabilístico es la herramienta adecuada

Introducción

Unos cuantos ejemplos

- Quicksort
- Find (también conocido como Quickselect)
- Hashing
- Simplex
- Estructuras de datos aleatorizadas (skip lists, árboles de búsqueda aleatorizados)

Introducción

Unos cuantos ejemplos

- Quicksort
- Find (también conocido como Quickselect)
- Hashing
- Simplex
- Estructuras de datos aleatorizadas (skip lists, árboles de búsqueda aleatorizados)

Introducción

Unos cuantos ejemplos

- Quicksort
- Find (también conocido como Quickselect)
- Hashing
- Simplex
- Estructuras de datos aleatorizadas (skip lists, árboles de búsqueda aleatorizados)

Introducción

Unos cuantos ejemplos

- Quicksort
- Find (también conocido como Quickselect)
- Hashing
- Simplex
- Estructuras de datos aleatorizadas (skip lists, árboles de búsqueda aleatorizados)

Introducción

Unos cuantos ejemplos

- Quicksort
- Find (también conocido como Quickselect)
- Hashing
- Simplex
- Estructuras de datos aleatorizadas (skip lists, árboles de búsqueda aleatorizados)

1 Introducción

2 Ejemplos
- Ejemplo #1: Ordenación parcial
- Ejemplo #2: Fallos de predicción
- Ejemplo #3: El problema de la contratación

3 Técnicas

Ordenación parcial

- Ordenación parcial: dado un vector A de n elementos y un valor $1 \leq m \leq n$, reorganizar A de tal manera que sus m primeras posiciones contengan los m menores elementos de A en orden creciente
- Si $m = \Theta(n)$ entonces OK ordenar el vector; sino, demasiado trabajo...much work

Ordenación parcial

- Ordenación parcial: dado un vector A de n elementos y un valor $1 \leq m \leq n$, reorganizar A de tal manera que sus m primeras posiciones contengan los m menores elementos de A en orden creciente
- Si $m = \Theta(n)$ entonces OK ordenar el vector; sino, demasiado trabajo... much work

Soluciones usuales

- Idea #1: Heapsort parcial
  - Construir un heap con los n elementos y hacer m extracciones del mínimo del heap
  - El coste en caso peor es $\Theta(n + m \log n)$
  - Esta es la forma "tradicional" de implementar `partial_sort` en la STL de C++

Soluciones usuales

- Idea #1: Heapsort parcial
  - Construir un heap con los n elementos y hacer m extracciones del mínimo del heap
  - El coste en caso peor es $\Theta(n + m \log n)$
  - Esta es la forma "tradicional" de implementar `partial_sort` en la STL de C++

Soluciones usuales

- Idea #1: Heapsort parcial
  - Construir un heap con los n elementos y hacer m extracciones del mínimo del heap
  - El coste en caso peor es $\Theta(n + m \log n)$
  - Esta es la forma "tradicional" de implementar `partial_sort` en la STL de C++

Soluciones usuales

- Idea #2: Selección on-line
  - Construir un heap con los m primeros elementos, luego recorrer los restantes $n - m$ elementos actualizando el heap adecuadamente; finalmente extraer los m elementos del heap
  - El coste en caso peor es $\Theta(n \log m)$
  - No es una solución interesante a menos que m sea muy pequeña o que los n elementos no estén disponibles desde el principio y nos vayan llegando on-line

Soluciones usuales

- Idea #2: Selección on-line
  - Construir un heap con los m primeros elementos, luego recorrer los restantes $n - m$ elementos actualizando el heap adecuadamente; finalmente extraer los m elementos del heap
  - El coste en caso peor es $\Theta(n \log m)$
  - No es una solución interesante a menos que m sea muy pequeña o que los n elementos no estén disponibles desde el principio y nos vayan llegando on-line

Soluciones usuales

- Idea #2: Selección on-line
  - Construir un heap con los m primeros elementos, luego recorrer los restantes $n - m$ elementos actualizando el heap adecuadamente; finalmente extraer los m elementos del heap
  - El coste en caso peor es $\Theta(n \log m)$
  - No es una solución interesante a menos que m sea muy pequeña o que los n elementos no estén disponibles desde el principio y nos vayan llegando on-line

Soluciones usuales

- Idea #3: "Quickselsort"
  - Hallar el m-ésimo menor elemento usando quickselect, y luego hacer quicksort sobre los $m-1$ elementos que lo preceden
  - El coste promedio es $\Theta(n + m \log m)$
  - Usa dos algoritmos básicos ampliamente disponibles y bien optimizados en las bibliotecas estándar

Soluciones usuales

- Idea #3: "Quickselsort"
  - Hallar el m-ésimo menor elemento usando quickselect, y luego hacer quicksort sobre los $m-1$ elementos que lo preceden
  - El coste promedio es $\Theta(n + m \log m)$
  - Usa dos algoritmos básicos ampliamente disponibles y bien optimizados en las bibliotecas estándar

Soluciones usuales

- Idea #3: "Quickselsort"
  - Hallar el m-ésimo menor elemento usando quickselect, y luego hacer quicksort sobre los $m-1$ elementos que lo preceden
  - El coste promedio es $\Theta(n + m \log m)$
  - Usa dos algoritmos básicos ampliamente disponibles y bien optimizados en las bibliotecas estándar

Partial Quicksort

```
void partial_quicksort(vector<Elem>& A,
                       int i, int j, int m) {
    if (i < j) {
        int p = get_pivot(A, i, j);
        swap(A[p], A[l]);
        int k;
        partition(A, i, j, k);
        partial_quicksort(A, i, k - 1, m);
        if (k < m - 1)
            partial_quicksort(A, k + 1, j, m);
}  }
```

Análisis de Partial Quicksort

- Probabilidad de que el pivote elegido sea el k-ésimo de entre los n elementos: $\pi_{n,k}$
- Número medio de comparaciones $P_{n,m}$ para ordenar los m menores elementos de entre n:

$$P_{n,m} = n - 1 + \sum_{k=m+1}^{n} \pi_{n,k} \cdot P_{k-1,m} + \sum_{k=1}^{m} \pi_{n,k} \cdot (P_{k-1,k-1} + P_{n-k,m-k})$$

Análisis de Partial Quicksort

- Probabilidad de que el pivote elegido sea el k-ésimo de entre los n elementos: $\pi_{n,k}$
- Número medio de comparaciones $P_{n,m}$ para ordenar los m menores elementos de entre n:

$$P_{n,m} = n - 1 + \sum_{k=m+1}^{n} \pi_{n,k} \cdot P_{k-1,m} + \sum_{k=1}^{m} \pi_{n,k} \cdot (P_{k-1,k-1} + P_{n-k,m-k})$$

Análisis de Partial Quicksort

- Si $m = n$, partial quicksort $\equiv$ quicksort; sea q_n el número medio de comparaciones que hace quicksort
- La recurrencia queda así

$$P_{n,m} = n - 1 + \sum_{k=1}^{m} \pi_{n,k} \cdot q_{k-1} + \sum_{k=m+1}^{n} \pi_{n,k} \cdot P_{k-1,m} + \sum_{k=1}^{m} \pi_{n,k} \cdot P_{n-k,m-k} \qquad (*)$$

Análisis de Partial Quicksort

- Si $m = n$, partial quicksort $\equiv$ quicksort; sea q_n el número medio de comparaciones que hace quicksort
- La recurrencia queda así

$$P_{n,m} = n - 1 + \sum_{k=1}^{m} \pi_{n,k} \cdot q_{k-1} + \sum_{k=m+1}^{n} \pi_{n,k} \cdot P_{k-1,m} + \sum_{k=1}^{m} \pi_{n,k} \cdot P_{n-k,m-k} \qquad (*)$$

Qué es un fallo de predicción?

- En el hardware moderno se ejecutan varias instrucciones "simultáneamente" usando un arquitectura en pipeline: p.e., mientras se decodifica la instrucción i, se están obtienendo de la memoria los datos de la instrucción $i-1$, operando los de la $i-2$ y almacenando el resultado de la $i-3$
- El problema serio lo constituyen las instrucciones de salto, pues dependiendo del resultado el flujo de instrucciones seguirá por un lado o por otro.

Qué es un fallo de predicción?

- En el hardware moderno se ejecutan varias instrucciones "simultáneamente" usando un arquitectura en pipeline: p.e., mientras se decodifica la instrucción i, se están obtienendo de la memoria los datos de la instrucción $i-1$, operando los de la $i-2$ y almacenando el resultado de la $i-3$
- El problema serio lo constituyen las instrucciones de salto, pues dependiendo del resultado el flujo de instrucciones seguirá por un lado o por otro.

Qué es un fallo de predicción?

- Para aprovechar al máximo el "paralelismo", muchos procesadores modernos utilizan algún mecanismo para predecir cúal será el resultado de una instrucción de salto.
- Si hay un fallo en la predicción (Branch misprediction) hay que tomar medidas para desandar todo lo incorrectamente realizado ... Cada fallo de predicción tiene un coste alto y un alto impacto en la eficiencia del algoritmo ejecutado.

Qué es un fallo de predicción?

- Para aprovechar al máximo el "paralelismo", muchos procesadores modernos utilizan algún mecanismo para predecir cúal será el resultado de una instrucción de salto.
- Si hay un fallo en la predicción (Branch misprediction) hay que tomar medidas para desandar todo lo incorrectamente realizado ... Cada fallo de predicción tiene un coste alto y un alto impacto en la eficiencia del algoritmo ejecutado.

Fallos de predicción en Quicksort

La implementación típica de Quicksort particiona en cada etapa recursiva el array dado mediante dos bucles internos

- uno recorre el array de izquierda a derecha haciendo crecer la zona de elementos menores que el pivote
- el otro recorre el array de derecha a izquierda haciendo crecer la zona de elementos mayores que el pivote.

Aquí estudiaremos un predictor sencillo: se predice que cada rama condicional tendrá el mismo resultado que la vez anterior (1-bit predictor).

Fallos de predicción en Quicksort

Tenemos una permutación aleatoria σ de $[1..n]$ que recorremos de izquierda a derecha. Supongamos que $\sigma_1 = k$.

- Para $2 < i \leq k$, diremos que hay un fallo de predicción (FP) a la izquierda siempre que $\sigma_{i-1} < k$ y $\sigma_i > k$, o $\sigma_{i-1} > k$ y $\sigma_i < k$.
- Para $k \leq j < n$, hay un fallo de predicción a la derecha si $\sigma_{j+1} < k$ y $\sigma_j > k$, o $\sigma_{j+1} > k$ y $\sigma_j < k$.
- Además , hay un FP a la izquierda si $\sigma_2 > k$ y un FP a la derecha si $\sigma_n < k$.

Fallos de predicción en Quicksort

Tenemos una permutación aleatoria σ de $[1..n]$ que recorremos de izquierda a derecha. Supongamos que $\sigma_1 = k$.

- Para $2 < i \leq k$, diremos que hay un fallo de predicción (FP) a la izquierda siempre que $\sigma_{i-1} < k$ y $\sigma_i > k$, o $\sigma_{i-1} > k$ y $\sigma_i < k$.
- Para $k \leq j < n$, hay un fallo de predicción a la derecha si $\sigma_{j+1} < k$ y $\sigma_j > k$, o $\sigma_{j+1} > k$ y $\sigma_j < k$.
- Además , hay un FP a la izquierda si $\sigma_2 > k$ y un FP a la derecha si $\sigma_n < k$.

Fallos de predicción en Quicksort

Tenemos una permutación aleatoria σ de $[1..n]$ que recorremos de izquierda a derecha. Supongamos que $\sigma_1 = k$.

- Para $2 < i \leq k$, diremos que hay un fallo de predicción (FP) a la izquierda siempre que $\sigma_{i-1} < k$ y $\sigma_i > k$, o $\sigma_{i-1} > k$ y $\sigma_i < k$.
- Para $k \leq j < n$, hay un fallo de predicción a la derecha si $\sigma_{j+1} < k$ y $\sigma_j > k$, o $\sigma_{j+1} > k$ y $\sigma_j < k$.
- Además , hay un FP a la izquierda si $\sigma_2 > k$ y un FP a la derecha si $\sigma_n < k$.

Simplificando el modelo

- Para analizar el número medio de FPs, necesitaremos averiguar el número medio de FPs condicionado a que el pivote elegido sea el k-ésimo elemento del array.
- Transformamos el problema original en uno de conteo de strings de bits. Dado un string x de longitud k que comienza con un 0 y contiene en total $t+1$ 0's, el número de FPs izquierdos es el número de veces que hay un 0 seguido de 1 o un 1 seguido de un 0 en x.
- A una permutación σ le corresponde un string x con $x_i = 0$ si $\sigma_i \leq \sigma_1 = k$, y $x_i = 1$ en otro caso.

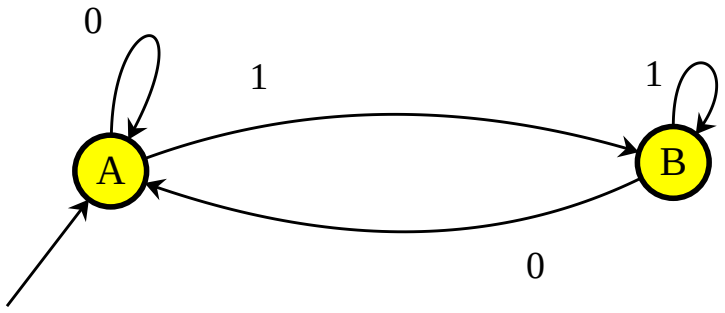
Simplificando el modelo

- Para analizar el número medio de FPs, necesitaremos averiguar el número medio de FPs condicionado a que el pivote elegido sea el k-ésimo elemento del array.
- Transformamos el problema original en uno de conteo de strings de bits. Dado un string x de longitud k que comienza con un 0 y contiene en total $t+1$ 0's, el número de FPs izquierdos es el número de veces que hay un 0 seguido de 1 o un 1 seguido de un 0 en x.
- A una permutación σ le corresponde un string x con $x_i = 0$ si $\sigma_i \leq \sigma_1 = k$, y $x_i = 1$ en otro caso.

Simplificando el modelo

- Para analizar el número medio de FPs, necesitaremos averiguar el número medio de FPs condicionado a que el pivote elegido sea el k-ésimo elemento del array.
- Transformamos el problema original en uno de conteo de strings de bits. Dado un string x de longitud k que comienza con un 0 y contiene en total $t+1$ 0's, el número de FPs izquierdos es el número de veces que hay un 0 seguido de 1 o un 1 seguido de un 0 en x.
- A una permutación σ le corresponde un string x con $x_i = 0$ si $\sigma_i \leq \sigma_1 = k$, y $x_i = 1$ en otro caso.

Simplificando el modelo



En términos de este autómata querremos saber cuántos strings x de longitud k con $t+1$ 0's saltan r veces de un estado a otro: las flechas entre los estados A y B corresponden a FPs!

El problema de la contratación

- Originalmente propuesto por Broder et al. (SODA 2008)
- Una secuencia (potencialmente infinita) de variables aleatorias i.i.d. Q_i, distribuídas uniformemente en $[0, 1]$
- En el paso i, se contrata o se descarta al candidato i basándose en su puntuación Q_i
- Las decisiones tomadas son irrevocables; no se conoce el futuro.
- Objetivo: contratar candidatos a un ritmo razonable para garantizar un crecimiento suficiente de la compañía, manteniendo un buen estándar de calidad (de hecho mejorándolo progresivamente) en la plantilla

El problema de la contratación

- Originalmente propuesto por Broder et al. (SODA 2008)
- Una secuencia (potencialmente infinita) de variables aleatorias i.i.d. Q_i, distribuídas uniformemente en $[0, 1]$
- En el paso i, se contrata o se descarta al candidato i basándose en su puntuación Q_i
- Las decisiones tomadas son irrevocables; no se conoce el futuro.
- Objetivo: contratar candidatos a un ritmo razonable para garantizar un crecimiento suficiente de la compañía, manteniendo un buen estándar de calidad (de hecho mejorándolo progresivamente) en la plantilla

El problema de la contratación

- Originalmente propuesto por Broder et al. (SODA 2008)
- Una secuencia (potencialmente infinita) de variables aleatorias i.i.d. Q_i, distribuídas uniformemente en $[0, 1]$
- En el paso i, se contrata o se descarta al candidato i basándose en su puntuación Q_i
- Las decisiones tomadas son irrevocables; no se conoce el futuro.
- Objetivo: contratar candidatos a un ritmo razonable para garantizar un crecimiento suficiente de la compañía, manteniendo un buen estándar de calidad (de hecho mejorándolo progresivamente) en la plantilla

El problema de la contratación

- Originalmente propuesto por Broder et al. (SODA 2008)
- Una secuencia (potencialmente infinita) de variables aleatorias i.i.d. Q_i, distribuídas uniformemente en $[0, 1]$
- En el paso i, se contrata o se descarta al candidato i basándose en su puntuación Q_i
- Las decisiones tomadas son irrevocables; no se conoce el futuro.
- Objetivo: contratar candidatos a un ritmo razonable para garantizar un crecimiento suficiente de la compañía, manteniendo un buen estándar de calidad (de hecho mejorándolo progresivamente) en la plantilla

El problema de la contratación

- Originalmente propuesto por Broder et al. (SODA 2008)
- Una secuencia (potencialmente infinita) de variables aleatorias i.i.d. Q_i, distribuídas uniformemente en $[0, 1]$
- En el paso i, se contrata o se descarta al candidato i basándose en su puntuación Q_i
- Las decisiones tomadas son irrevocables; no se conoce el futuro.
- Objetivo: contratar candidatos a un ritmo razonable para garantizar un crecimiento suficiente de la compañía, manteniendo un buen estándar de calidad (de hecho mejorándolo progresivamente) en la plantilla

El problema de la contratación

- Modelizamos el problema mediante una permutación σ de longitud n; el candidato i tiene puntuación $\pi(i)$, de manera que 1 es la peor puntuación y n la mejor
- Este modelo es equivalente al anterior tras hacer un "normalización" apropiada y el paso al límite, pero tiene la ventaja de que podemos aplicar técnicas de combinatoria analítica
- Nos interesa analizar parámetros tales como la puntuación media del último candidato contratado o el tamaño medio de la plantilla, en función del número de entrevistas n (= longitud de la permutación)

El problema de la contratación

- Modelizamos el problema mediante una permutación σ de longitud n; el candidato i tiene puntuación $\pi(i)$, de manera que 1 es la peor puntuación y n la mejor
- Este modelo es equivalente al anterior tras hacer un "normalización" apropiada y el paso al límite, pero tiene la ventaja de que podemos aplicar técnicas de combinatoria analítica
- Nos interesa analizar parámetros tales como la puntuación media del último candidato contratado o el tamaño medio de la plantilla, en función del número de entrevistas n (= longitud de la permutación)

El problema de la contratación

- Modelizamos el problema mediante una permutación σ de longitud n; el candidato i tiene puntuación $\pi(i)$, de manera que 1 es la peor puntuación y n la mejor
- Este modelo es equivalente al anterior tras hacer un "normalización" apropiada y el paso al límite, pero tiene la ventaja de que podemos aplicar técnicas de combinatoria analítica
- Nos interesa analizar parámetros tales como la puntuación media del último candidato contratado o el tamaño medio de la plantilla, en función del número de entrevistas n (= longitud de la permutación)

El problema de la contratación

Algunas de las estrategias analizadas:

- Por encima de un umbral
- Por encima del mejor
- Por encima del m-ésimo mejor (contratación por la élite)
- Por encima de la media
- Por encima de la mediana

Preliminares

- La eficiencia μ de un algoritmo con entradas del conjunto $\mathcal{I}$ ($\mu : \mathcal{I} \to \mathbb{N}$) depende de cada instancia particular
- Necesitamos una noción de tamaño de la entrada: $|\cdot| : \mathcal{I} \to \mathbb{N}$
- Podemos asumir que cada $\mathcal{I}_n = \{x \in \mathcal{I} \,|\, |x| = n\}$ es finito
- Caso peor

$$\mu^{[\mathrm{worst}]}(n) = \max\{\mu(x) \,|\, x \in \mathcal{I}_n\}$$

Preliminares

- La eficiencia μ de un algoritmo con entradas del conjunto $\mathcal{I}$ ($\mu : \mathcal{I} \to \mathbb{N}$) depende de cada instancia particular
- Necesitamos una noción de tamaño de la entrada: $|\cdot| : \mathcal{I} \to \mathbb{N}$
- Podemos asumir que cada $\mathcal{I}_n = \{x \in \mathcal{I} \,|\, |x| = n\}$ es finito
- Caso peor

$$\mu^{[\mathrm{worst}]}(n) = \max\{\mu(x) \,|\, x \in \mathcal{I}_n\}$$

Preliminares

- La eficiencia μ de un algoritmo con entradas del conjunto $\mathcal{I}$ ($\mu : \mathcal{I} \to \mathbb{N}$) depende de cada instancia particular
- Necesitamos una noción de tamaño de la entrada: $|\cdot| : \mathcal{I} \to \mathbb{N}$
- Podemos asumir que cada $\mathcal{I}_n = \{x \in \mathcal{I} \,|\, |x| = n\}$ es finito
- Caso peor

$$\mu^{[\text{worst}]}(n) = \max\{\mu(x) \,|\, x \in \mathcal{I}_n\}$$

Preliminares

- La eficiencia μ de un algoritmo con entradas del conjunto $\mathcal{I}$ ($\mu : \mathcal{I} \to \mathbb{N}$) depende de cada instancia particular
- Necesitamos una noción de tamaño de la entrada: $|\cdot| : \mathcal{I} \to \mathbb{N}$
- Podemos asumir que cada $\mathcal{I}_n = \{x \in \mathcal{I} \,|\, |x| = n\}$ es finito
- Caso peor

$$\mu^{[\text{worst}]}(n) = \max\{\mu(x) \,|\, x \in \mathcal{I}_n\}$$

Preliminares

- Para el análisis probabilístico necesitamos definir una distribución de probabilidad sobre las entradas y/o las decisiones aleatorias que toma el propio algoritmo
- La eficiencia vendrá entonces dada como una familia de variables aleatorias $\{\mu_n\}_{n\geq 0};\ \mu_n : \mathcal{I}_n \to \mathbb{N}$
- Caso promedio:

$$\mu^{[\text{avg}]}(n) = \mathbb{E}[\mu_n] = \sum_{k\geq 0} k\,\mathbb{P}[\mu_n = k]$$

Preliminares

- Para el análisis probabilístico necesitamos definir una distribución de probabilidad sobre las entradas y/o las decisiones aleatorias que toma el propio algoritmo
- La eficiencia vendrá entonces dada como una familia de variables aleatorias $\{\mu_n\}_{n\geq 0}$; $\mu_n : \mathcal{I}_n \to \mathbb{N}$
- Caso promedio:

$$\mu^{[\text{avg}]}(n) = \mathbb{E}[\mu_n] = \sum_{k\geq 0} k\,\mathbb{P}[\mu_n = k]$$

Preliminares

- Para el análisis probabilístico necesitamos definir una distribución de probabilidad sobre las entradas y/o las decisiones aleatorias que toma el propio algoritmo
- La eficiencia vendrá entonces dada como una familia de variables aleatorias $\{\mu_n\}_{n\geq 0};\ \mu_n : \mathcal{I}_n \to \mathbb{N}$
- Caso promedio:

$$\mu^{[\text{avg}]}(n) = \mathbb{E}[\mu_n] = \sum_{k\geq 0} k\,\mathbb{P}[\mu_n = k]$$

Preliminares

Si suponemos que todas las entradas de tamaño n posibles tienen la misma probabilidad (distribución uniforme) entonces

$$\mathbb{P}[x] = \frac{1}{\#\mathcal{I}_n}, \text{para todo } x \in \mathcal{I}_n$$

y nuestro problema es en definitiva de conteo:

$$\mathbb{E}[\mu_n] = \frac{\sum_{x \in \mathcal{I}_n} \mu(x)}{\#\mathcal{I}_n}$$

Funciones generatrices

Entre las herramientas fundamentales del análisis de algoritmos están las funciones generatrices.
Para una secuencia dada $\{a_n\}_{n\geq 0}$ su función generatriz es:

$$A(z) = \sum_{n\geq 0} a_n z^n$$

Esto es puramente formal. Por ejemplo, si $a_n = 2^n$ entonces $A(z) = 1 + 2z + 4z^2 + 8z^3 + 16z^4 + \cdots = \frac{1}{1-2z}$.
Un caso importante es una función generatriz de valores esperados, esto es $a_n = \mathbb{E}[\mu_n]$.

Funciones generatrices

Las funciones generatrices son muy útiles para resolver recurrencias. Por ejemplo, tomemos los números de Fibonacci.

- $F_0 = F_1 = 1$
- $F_n = F_{n-1} + F_{n-2}$ si $n > 1$

Funciones generatrices

$$F(z) = \sum_{n \geq 0} F_n z^n = F_0 + F_1 z + \sum_{n \geq 2} F_n z^n$$

$$= 1 + z + \sum_{n \geq 2} (F_{n-1} + F_{n-2}) z^n$$

$$= 1 + z + z \sum_{n \geq 2} F_{n-1} z^{n-1} + z^2 \sum_{n \geq 2} F_{n-2} z^{n-2}$$

$$= 1 + z + z(F(z) - 1) + z^2 F(z) = 1 + zF(z) + z^2 F(z) = \frac{1}{1 - z - z^2}$$

Funciones generatrices

$$F(z) = \frac{-1}{(z-\phi)(z-\phi')} = \frac{1}{\sqrt{5}}\left(\frac{1}{z-\phi'} - \frac{1}{z-\phi}\right)$$
$$= \frac{1}{\sqrt{5}}\left(\frac{\phi}{1-z/\phi'} - \frac{\phi'}{1-z/\phi}\right)$$

donde $\phi = (\sqrt{5}-1)/2 \approx 0.618$,
$\phi' = -1/\phi = -(\sqrt{5}+1)/2 = -1.618$.
Nos interesa $F_n = [z^n]F(z)$.

Funciones generatrices

$$[z^n]F(z) = \frac{1}{\sqrt{5}}\left(\phi \cdot [z^n]\frac{1}{1-z/\phi'} - \phi' \cdot [z^n]\frac{1}{1-z/\phi}\right)$$

$$\frac{1}{\sqrt{5}}\left(\phi \cdot (\phi')^{-n} - \phi' \cdot (\phi)^{-n}\right)$$

$$\frac{1}{\sqrt{5}}\left(\phi \cdot (-\phi)^{n} - \phi' \cdot (-\phi')^{n}\right)$$

$$\sim \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^{n+1}$$

Esto es, $F_n = \Theta(1.618\ldots^n)$.

Funciones generatrices

Otro tipo importante de FGs son las funciones generatrices exponenciales:

$$A(z) = \sum_{n \geq 0} a_n \frac{z^n}{n!}$$

Se usan cuando los a_n's crecen superexponencialmente. También se usan a menudo funciones generatrices bivariadas: para $\{a_{n,m}\}_{n,m \geq 0}$ definimos

$$A(z,u) = \sum_{n \geq 0} \sum_{m \geq 0} a_{n,m} z^n u^m$$

El método simbólico

El método simbólico traduce ciertas construcciones combinatorias usuales a las correspondientes ecuaciones funcionales sobre FGs.

Ejemplo: Consideremos la FG de conteo de una clase combinatoria $\mathcal{A}$:

$$A(z) = \sum_{n \geq 0} a_n z^n = \sum_{\alpha \in \mathcal{A}} z^{|\alpha|}$$

donde $a_n = \#\mathcal{A}_n = \#\{x \in \mathcal{A} \,|\, |x| = n\}$.

El método simbólico

Si $\mathcal{A} = \mathcal{B} \times \mathcal{C}$ entonces

$$A(z) = \sum_{\alpha \in \mathcal{A}} z^{|\alpha|} = \sum_{(\beta,\gamma) \in \mathcal{B} \times \mathcal{C}} z^{|\beta|+|\gamma|} = \left(\sum_{\beta \in \mathcal{B}} z^{|\beta|} \right) \left(\sum_{\gamma \in \mathcal{C}} z^{|\gamma|} \right)$$
$$= B(z) \cdot C(z)$$

El método simbólico

Si $\mathcal{C} = \mathrm{Seq}(\mathcal{A})$ entonces

$$\mathcal{C} = \lambda + \mathcal{A} + \mathcal{A} \times \mathcal{A} + \cdots + \underbrace{\mathcal{A} \times \cdots \times \mathcal{A}}_{k \text{ veces}} + \cdots$$

Por tanto

$$C(z) = 1 + A + A^2 + \cdots + A^k + \cdots = \frac{1}{1 - A(z)}$$

El método simbólico

Un diccionario de construcciones combinatorias y las FGs correspondientes

$\{\epsilon\}$	1
$\{Z\}$	z
$\mathcal{A} + \mathcal{B}$	$A + B$
$\mathcal{A} \times \mathcal{B}$	$A \cdot B$
$\mathrm{Seq}(\mathcal{A})$	$\frac{1}{1-A}$
$\mathrm{Set}(\mathcal{A})$	$\exp(A)$
$\mathrm{Cycle}(\mathcal{A})$	$\log \frac{1}{1-A}$

El método simbólico

Un ejemplo sencillo: un árebol binario es o bien vacío (una hoja) o bien consiste en una raíz y dos subárboles binarios:

$$\mathcal{B} = \{\epsilon\} + \{Z\} \times \mathcal{B} \times \mathcal{B}$$

La FG de conteo de los árboles binarios es:

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

El método simbólico

Si resolvemos la ecuación cuadrática para $B(z)$ y teniendo en cuenta que $B(0) = b_0 = 1$,

$$B(z) = \begin{cases} \frac{1-\sqrt{1-4z}}{2z} & z \neq 0, \\ 1 & z = 0. \end{cases}$$

Si se extrae el coeficiente de z^n en $B(z)$ usando el teorema del binomio de Newton obtenemos

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

los famosos números de Catalan.

El método simbólico

Otro ejemplo son los derangements, permutaciones sin puntos fijos. El problema suele aparecer de la siguiente manera: en un teatro n personas dejan sus sombreros y a la salida se devuelven al azar; cuál es la probabilidad de que al menos uno persona recupere su propio sombrero?

Empecemos con las permutaciones. Una permutación cualquiera consiste en una secuencia de átomos etiquetados. Sea $\mathcal{P}$ el conjunto de todas las permutaciones. Entonces

$$P(z) = \sum_{\sigma \in \mathcal{P}} \frac{z^{|\sigma|}}{|\sigma|!} = \sum_{n \geq 0} n! \frac{z^n}{n!} = \frac{1}{1-z},$$

lo que corresponde a que $\mathcal{P} = \mathsf{Seq}(Z)$.

El método simbólico

Pero una permutación también se puede caracterizar como un conjunto de ciclos.

$$\mathcal{P} = \mathsf{Set}(\mathcal{C})$$
$$\mathcal{C} = \mathsf{Cycle}(\mathcal{Z})$$

La función generatriz C de los ciclos etiquetados es $\ln \frac{1}{1-z}$ y por tanto la de las permutaciones es

$$P = \exp(C) = \exp\left(\ln \frac{1}{1-z}\right) = \frac{1}{1-z}$$

El método simbólico

Una permutación consiste en un derangement y un conjunto de ciclos de un sólo elemento cada uno. Podemos escribir

$$\mathcal{P} = \mathcal{D} \times \mathcal{F}$$

donde $\mathcal{D}$ es la clase de los derangements y $\mathcal{F}$ es la clase de los conjuntos de ciclos unitarios. Un ciclo unitario es lo mismo que un "átomo", así $F(z) = \mathrm{exp}(z)$. Por tanto

$$P(z) = \frac{1}{1-z} = D(z) \cdot \mathrm{exp}(z) \implies D(z) = \frac{e^{-z}}{1-z}$$

El método simbólico y funciones bivariadas

Supongamos que queremos analizar un algoritmo donde el coste que nos interesa es la variable aleatoria μ_n. Definimos

$$B(z,u) = \sum_{n\geq 0}\sum_{k\geq 0} \mathbb{P}[\mu_n = k]\, z^n u^k$$

Para la distribución uniforme tendremos

$$[z^n u^k]B(z,u) = \frac{[z^n u^k]\sum_{n\geq 0}\sum_{k\geq 0} a_{n,k} z^n u^k}{[z^n]\sum_{n\geq 0} a_n z^n} = \frac{[z^n u^k]A(z,u)}{[z^n]A(z,1)}$$

donde $a_{n,k} = \#\{x \in \mathcal{I} \,|\, |x| = n \wedge \mu(x) = k\}$, y $a_n = \#\mathcal{I}_n$

El método simbólico y funciones bivariadas

Simbólicamente podemos escribir

$$A(z,u) = \sum_{x \in \mathcal{I}} z^{|x|} u^{\mu(x)}$$

El coeciente entre el coeficiente $[z^n]A(z,u)$ y $[z^n]A(z,1)$ es la función generatriz de probabilidad de μ_n

$$p_n(u) = \mathbb{E}[u^{\mu_n}] = \sum_{k \geq 0} \mathbb{P}[\mu_n = k]\, u^k = \frac{[z^n]A(z,u)}{[z^n]A(z,1)}$$

El método simbólico y funciones bivariadas

Si tomamos derivadas respecto a u y hacemos $u = 1$ podemos extraer los sucesivos momentos: valor esperado, segundo momento factorial, etc.

$$A^{(r)}(z) = \left.\frac{\partial^r A(z,u)}{\partial u^r}\right|_{u=1} = \sum_{n\geq 0} \mathbb{E}[\underline{\mu_n{}^r}]\, z^n$$

Por ejemplo,

$$\mathbb{V}[\mu_n] = \mathbb{E}\left[\underline{\mu_n{}^2}\right] - \mathbb{E}[\mu_n]^2 = [z^n]A^{(2)}(z) - ([z^n]A(z))^2$$

Ejemplo: Partial Quicksort

- Definimos estas FGs

$$P(z,u) = \sum_{n\geq 0} \sum_{1\leq m\leq n} P_{n,m} z^n u^m$$

$$T(z,u) = \sum_{n\geq 0} \sum_{1\leq m\leq n} t_{n,m} z^n u^m$$

  donde $t_{n,m}$ es el coste no recursivo.

- La recurrencia delos $P_{n,m}$'s se traduce en

$$\frac{\partial P}{\partial z} = \frac{P(z,u)}{1-z} + \frac{u\,P(z,u)}{1-uz} + \frac{\partial T}{\partial z} \qquad (*)$$

Ejemplo: Partial Quicksort

- Definimos estas FGs

$$P(z,u) = \sum_{n\geq 0} \sum_{1\leq m\leq n} P_{n,m} z^n u^m$$

$$T(z,u) = \sum_{n\geq 0} \sum_{1\leq m\leq n} t_{n,m} z^n u^m$$

  donde $t_{n,m}$ es el coste no recursivo.

- La recurrencia delos $P_{n,m}$'s se traduce en

$$\frac{\partial P}{\partial z} = \frac{P(z,u)}{1-z} + \frac{u\,P(z,u)}{1-uz} + \frac{\partial T}{\partial z} \qquad (*)$$

Ejemplo: Partial Quicksort

Se resuelve la ecuación diferencial (*) y se extraen coeficientes y obtenemos finalmente

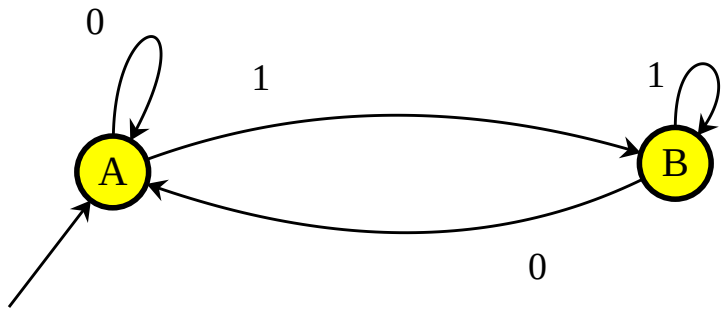
$$P_{n,m} = 2n + 2(n+1)H_n - 2(n+3-m)H_{n+1-m} - 6m + 6$$

Si comparamos partial quicksort con "quickselect+quicksort" se hacen

$$2m - 4H_m + 2$$

comparaciones en promedio menos. Se puede calcular de manera similar el ahorro en intercambios, etc.

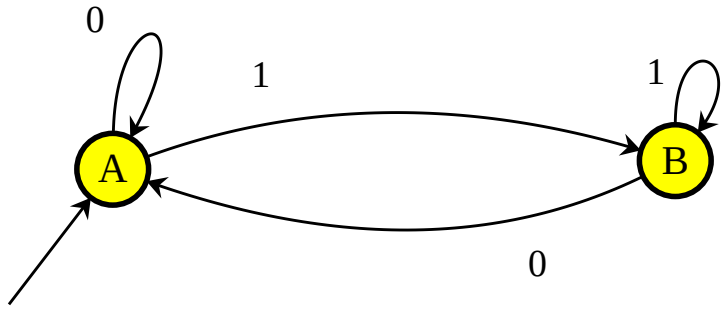
Ejemplo: Fallos de predicción



$$A(w,u,z) = \sum_{x \in A} u^{\mathrm{FP}(x)} w^{\mathrm{zeros}(x)} z^{|x|},$$

$$B(w,u,z) = \sum_{x \in B} u^{\mathrm{FP}(x)} w^{\mathrm{zeros}(x)} z^{|x|}$$

Ejemplo: Fallos de predicción



$$A = 1 + wzA + wuzB,$$

$$B = zB + zuA,$$

$$C = A + B = \frac{1 - z + uz}{(1 - z)\left(1 - uy - \frac{wu^2z^2}{1-z}\right)}$$

Ejemplo: El problema de la contratación

- A una estrategia de contratación la llamamos Basada en el rango si y sólo si las decisiones se toman en función del rango relativo del candidato en curso y no en función de su puntuación en términos absolutos.
- $\mathcal{H}(\sigma)$ = el conjunto de los candidatos contratados en la permutación σ; $h(\sigma) = \#\mathcal{H}(\sigma)$

Ejemplo: El problema de la contratación

- A una estrategia de contratación la llamamos Basada en el rango si y sólo si las decisiones se toman en función del rango relativo del candidato en curso y no en función de su puntuación en términos absolutos.
- $\mathcal{H}(\sigma)$ = el conjunto de los candidatos contratados en la permutación σ; $h(\sigma) = \#\mathcal{H}(\sigma)$

Ejemplo: El problema de la contratación

Sea $\sigma \circ j$ la permutación que obtenemos tras reetiquetar $j, j+1, \ldots, n = |\sigma|$ como $j+1, j+2, \ldots, n+1$ y añadir j por el final. Ejemplo: $32451 \circ 3 = 425613$ Sea

$X_j(\sigma) = 1$ si un candidato con puntuación j es contratado tras σ y $X_j(\sigma) = 0$ sino.

$$h(\sigma \circ j) = h(\sigma) + X_j(\sigma)$$

Ejemplo: El problema de la contratación

Sea $X(\sigma)$ el número de j's tales que $X_j(\sigma) = 1$.

Sea

$$H(z,u) = \sum_{\sigma \in \mathcal{P}} \frac{z^{|\sigma|}}{|\sigma|!} u^{h(\sigma)}.$$

Entonces

$$(1-z)\frac{\partial}{\partial z} H(z,u) - H(z,u) = (u-1) \sum_{\sigma \in \mathcal{P}} X(\sigma) \frac{z^{|\sigma|}}{|\sigma|!} u^{h(\sigma)}.$$

Ejemplo: Contratación por encima del mejor

El candidato i es contratado si y sólo si su puntuación es superior a la del mejor candidato contratado hasta el momento.

- $X(\sigma) = 1$
- $\mathcal{H}(\sigma) = \{i :$
  i es un máximo de izquierda a derecha en $\sigma\}$
- $\mathbb{E}[h_n] = [z^n] \left.\frac{\partial H}{\partial u}\right|_{u=1} = \ln n + O(1)$
- La varianza también es $\ln n + O(1)$ y tras la normalización apropiada $h_n^* = (h_n - \mathbb{E}[h_n])/\sqrt{\mathbb{V}[h_n]}$ converge a $\mathcal{N}(0, 1)$

Ejemplo: Contratación por encima del mejor

El candidato i es contratado si y sólo si su puntuación es superior a la del mejor candidato contratado hasta el momento.

- $X(\sigma) = 1$
- $\mathcal{H}(\sigma) = \{i :$
  i es un máximo de izquierda a derecha en $\sigma\}$
- $\mathbb{E}[h_n] = [z^n] \left.\frac{\partial H}{\partial u}\right|_{u=1} = \ln n + O(1)$
- La varianza también es $\ln n + O(1)$ y tras la normalización apropiada $h_n^* = (h_n - \mathbb{E}[h_n])/\sqrt{\mathbb{V}[h_n]}$ converge a $\mathcal{N}(0,1)$

Ejemplo: Contratación por encima del mejor

El candidato i es contratado si y sólo si su puntuación es superior a la del mejor candidato contratado hasta el momento.

- $X(\sigma) = 1$
- $\mathcal{H}(\sigma) = \{i :$
  i es un máximo de izquierda a derecha en $\sigma\}$
- $\mathbb{E}[h_n] = [z^n] \left.\frac{\partial H}{\partial u}\right|_{u=1} = \ln n + O(1)$
- La varianza también es $\ln n + O(1)$ y tras la normalización apropiada $h_n^* = (h_n - \mathbb{E}[h_n])/\sqrt{\mathbb{V}[h_n]}$ converge a $\mathcal{N}(0,1)$

Ejemplo: Contratación por encima del mejor

El candidato i es contratado si y sólo si su puntuación es superior a la del mejor candidato contratado hasta el momento.

- $X(\sigma) = 1$
- $\mathcal{H}(\sigma) = \{i :$
  i es un máximo de izquierda a derecha en $\sigma\}$
- $\mathbb{E}[h_n] = [z^n] \left.\frac{\partial H}{\partial u}\right|_{u=1} = \ln n + O(1)$
- La varianza también es $\ln n + O(1)$ y tras la normalización apropiada $h_n^* = (h_n - \mathbb{E}[h_n])/\sqrt{\mathbb{V}[h_n]}$ converge a $\mathcal{N}(0,1)$

Ejemplo: Contratación por encima de la mediana

El candidato i es contratado si y sólo si su puntuación está por encima de la mediana de las puntuaciones de los candidatos contratados.

- $X(\sigma) = \lceil (h(\sigma) + 1)/2 \rceil$
- $\sqrt{\frac{n}{\pi}}(1 + O(n^{-1})) \leq \mathbb{E}[h_n] \leq 3\sqrt{\frac{n}{\pi}}(1 + O(n^{-1}))$
- El resultado se obtiene al resolver las ecuaciones diferenciales que salen del Teorema anterior, usando $X_L(\sigma) = (h(\sigma) + 1)/2$ and $X_U(\sigma) = (h(\sigma) + 3)/2$ como cota inferior y superior, respectivamente.

Ejemplo: Contratación por encima de la mediana

El candidato i es contratado si y sólo si su puntuación está por encima de la mediana de las puntuaciones de los candidatos contratados.

- $X(\sigma) = \lceil (h(\sigma) + 1)/2 \rceil$
- $\sqrt{\frac{n}{\pi}}(1 + O(n^{-1})) \leq \mathbb{E}[h_n] \leq 3\sqrt{\frac{n}{\pi}}(1 + O(n^{-1}))$
- El resultado se obtiene al resolver las ecuaciones diferenciales que salen del Teorema anterior, usando $X_L(\sigma) = (h(\sigma) + 1)/2$ and $X_U(\sigma) = (h(\sigma) + 3)/2$ como cota inferior y superior, respectivamente.

Ejemplo: Contratación por encima de la mediana

El candidato i es contratado si y sólo si su puntuación está por encima de la mediana de las puntuaciones de los candidatos contratados.

- $X(\sigma) = \lceil (h(\sigma) + 1)/2 \rceil$
- $\sqrt{\frac{n}{\pi}}(1 + O(n^{-1})) \leq \mathbb{E}[h_n] \leq 3\sqrt{\frac{n}{\pi}}(1 + O(n^{-1}))$
- El resultado se obtiene al resolver las ecuaciones diferenciales que salen del Teorema anterior, usando $X_L(\sigma) = (h(\sigma) + 1)/2$ and $X_U(\sigma) = (h(\sigma) + 3)/2$ como cota inferior y superior, respectivamente.

Combinatoria analítica: una breve incursión

Los últimos resultados y otros muchos se puden obtener aplicando las técnicas de la combinatoria analítica; son técnicas basadas en la teoría de variable compleja.
En esencia, lo que hacemos es ver el objeto formal $F(z)$ como una función de variable compleja.

Combinatoria analítica: una breve incursión

Una función generatriz es analítica en una cierta región del plano complejo; es decir, la serie de potencias converge para todo z dentro del llamado disco de convergencia de radio R.
Un resultado importante es que si $R < \infty$ entonces $[z^n]F(z) < (\frac{1}{R} + \epsilon)^n$ y $(\frac{1}{R} - \epsilon)^n < f_n$ para todo $\epsilon > 0$; es decir, el radio de convergencia nos da inmediatamente el ritmo de crecimiento exponencial (si $R \neq \infty$).

Combinatoria analítica: una breve incursión

En muchos casos una función cesa de ser analítica porque no está definida en un cierto punto o alguna de sus derivadas no lo está. Muchas funciones importantes con radio de convergencia finito, tienen solo un conjunto finito de puntos donde la función no es analítica, se denominan singularidades.
La distancia al origen de la singularidad más cercana al origen (llamada dominante) no da el radio de convergencia $\Longrightarrow$ el ritmo de crecimiento exponencial.

Combinatoria analítica: una breve incursión

El resultado esencial a partir del cual se han obtenido otros muchos es el teorema de Cauchy: si $F(z)$ es analítica en una región R que contiene a $z = 0$ y Γ es una curva cerrada simple dentro de R con $z = 0$ en su interior

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

Se usa un contorno Γ fácil de integrar en la zona "alejada" de las singularidades y cerca de la singularidad se usa una expansión local de $F(z)$

Combinatoria analítica: una breve incursión

La naturaleza de la singularidad nos da factores polinómicos, logarítmicos, etc..
En condiciones bastante generales se pueden aplicar resultados del tipo

$$F(z) \sim_{z\to\rho} G(z) \implies f_n \sim g_n$$

donde $z=\rho$ es la singularidad dominante de $F(z)$, y la función $G(z)$ es una de la que sabemos extraer fácilmente los coeficientes, p.e.,

$$[z^n]\left(1-\frac{z}{\rho}\right)^{-\alpha} = \rho^{-n}\frac{n^\alpha}{\Gamma(\alpha)}(1+O(1/n)),$$

siendo $\alpha \neq -1,-2,-3,\ldots$

Combinatoria analítica: una breve incursión

Por ejemplo, la función generatriz de conteo de los árboles binarios es

$$B(z) = \frac{1-\sqrt{1-4z}}{2z} \sim_{z\to\frac{1}{4}} -2\left(1-\frac{z}{1/4}\right)^{1/2},$$

por lo tanto

$$b_n \sim -2\cdot 4^n \frac{n^{-3/2}}{\Gamma(-3/2)}(1+O(1/n)) = 4^n \frac{1}{n\sqrt{\pi n}}(1+O(1/n)).$$

Combinatoria analítica: una breve incursión

Otro ejemplo son los derangements.

$$D(z) = \frac{e^{-z}}{1-z} \sim_{z\to 1} \frac{e^{-1}}{1-z},$$

por lo tanto

$$d_n \sim e^{-1} \cdot n!(1 + O(1/n))$$

Y la respuesta al problema original es que la probabilidad de que nadie recupere su sombrero es

$$\frac{d_n}{n!} \sim e^{-1} \approx 0.386\ldots$$

Combinatoria analítica: una breve incursión

A menudo un FG satisface una ecuación funcional más o menos compleja, pero podemos utilizar técnicas del análisis complejo (ej: teorema de la función implícita) para saber su radio de convergencia y computar una expasión local cerca de la singularidad dominante, y de ahí sacar los f_n sin necesidad de resolver explícitamente la ecuación satisfecha por $F(z)$

Existen otros muchos métodos útiles como los de punto de silla (saddle point methods) y otros que no recurrenc al análisis de variable compleja (p.e. los teoremas Tauberianos).

Y además tenemos otras técnicas como los teoremas de resolución de recurrencias divide-y-vencerás.

Esto se acaba ...

y no he hecho más que arañar la superfície; serían necesarias dos horas más para empezar a ver con un poco más de detalle el territorio que sólo hemos sobrevolado ...

GRACIAS!!

Reconocimientos

- Kanela Kaligosi, Peter Sanders: Branch mispredictions in Quicksort
- Margaret Archibald: The hiring problem