- Eficiència temporal i espacial
- Notació asimptòtica
- Propietats de la notació asimptòtica

- Formes de creixement freqüents
- Anàlisi asimptòtica de l'eficiència temporal
- Anàlisi asimptòtica de l'eficiència espacial

- Equacions de recurrència
- Problemes

Els problemes marcats amb una estrelleta ($\star$) són més difícils. Si no es diu el contrari i us cal, considereu que tots els logaritmes són en base 2.

- 1.2
- 1.3
- 1.4
- 1.6
- $\star$ 1.7
- 1.12
- 1.13
- 1.14
- 1.15
- 1.16
- 1.17
- 1.19
- 1.20 (un o dos)
- 1.21 (un o dos)
- 1.22 (un o dos)
- 1.23 (un o dos)
- 1.24 (un o dos)
- 1.25 (un o dos)
- 1.27
- 1.28
- 1.31 (en teoria)
- 1.32 (en teoria)
- 1.34
- 1.35
- 1.36
- 1.37

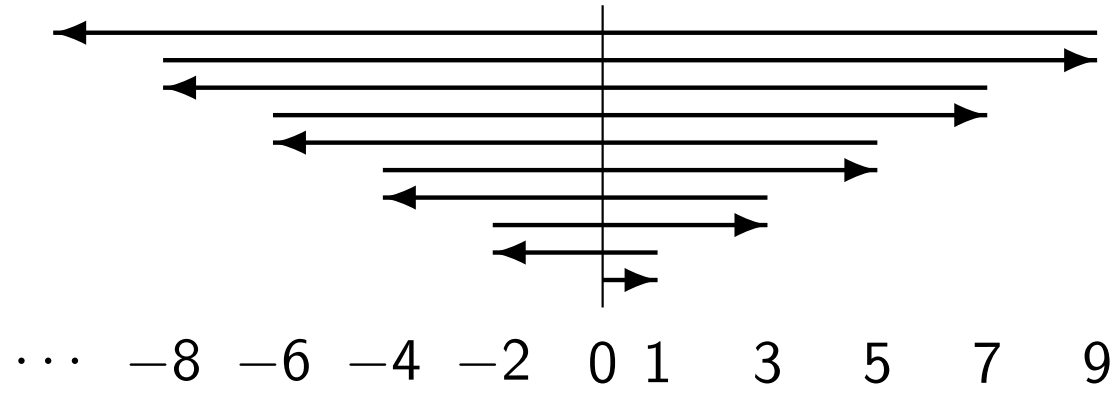
- El concepte d'eficiència és un concepte relatiu, en el sentit que es compara l'eficiència d'un algorisme amb la d'un altre que resol el mateix problema.
- Es diu que un algorisme és ineficient si n'hi ha un altre de més eficient per resoldre el mateix problema.
- Mesurar l'eficiència d'un algorisme equival a mesurar la quantitat de recursos (temps i espai) necessaris per a la seva execució, com a funció de les dades d'entrada.

- Ens trobem davant d'un mur que s'allarga indefinidament en totes dues direccions. Hi ha una porta que travessa el mur, però no sabem a quina distància o en quina direcció. És una mica fosc, però portem un fanalet que ens permet de veure la porta quan ja hi som aprop.

- Ens trobem davant d'un mur que s'allarga indefinidament en totes dues direccions. Hi ha una porta que travessa el mur, però no sabem a quina distància o en quina direcció. És una mica fosc, però portem un fanalet que ens permet de veure la porta quan ja hi som aprop.

- Mostreu que hi ha un algorisme que ens permet trobar la porta caminant $\Theta(n)$ passes, on n és el nombre de passes que haguérem fet si haguéssim sabut on era la porta i hi haguéssim caminat directament.
- Quina és la constant multiplicativa en l'anàlisi asimptòtic de l'eficiència temporal d'aquest algorisme?

- Primer intent de resolució.



– Tenim que

$$T(n) = 2\sum_{i=1}^{n-1} i + n = 2\frac{(n-1)n}{2} + n = n^2\,,$$

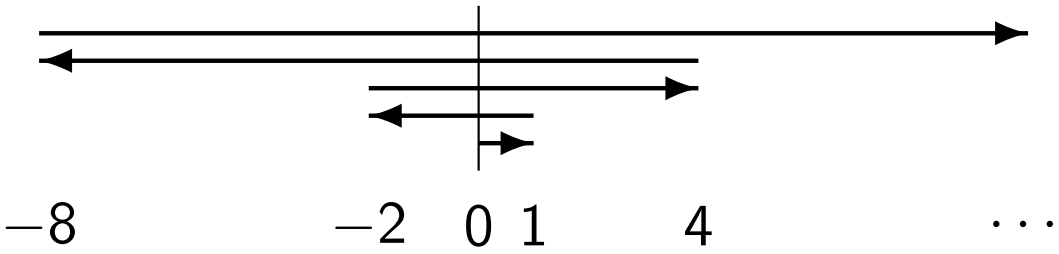
atès que

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} \qquad \text{per a tot } n \geqslant 0\,.$$

– Aleshores

$$T(n) = \Theta(n^2)\,.$$

- Segon intent de resolució.



– Suposem que $n = 2^k$. Aleshores

$$T(n) = 2\sum_{i=0}^{k-1} 2^i + 2^k = 2(2^k - 1) + 2^k = 3n - 2\,,$$

atès que

$$\sum_{i=0}^{n} 2^i = 2^{n+1} - 1 \qquad \text{per a tot } n \geqslant 0\,.$$

– Aleshores

$$T(n) = \Theta(n)\,.$$

- La notació asimptòtica serveix per classificar funcions en base a la seva velocitat relativa de creixement.
- La idea és la d'establir un ordre relatiu entre funcions. Donades dues funcions $f = f(x)$ i $g = g(x)$, hi pot haver valors de x on $f(x) < g(x)$ i valors de x on $f(x) > g(x)$, per la qual cosa no té sentit de dir que, per exemple, $f(x) < g(x)$.
- Per exemple, $1000\,n > n^2$ per a valors petits de n, però n^2 creix més ràpidament que $1000\,n$ i eventualment $n^2 > 1000\,n$. És a dir, només a partir de $n = 1000$ té sentit dir que $1000\,n \leqslant n^2$.

- Les constants poden ser extremadament grans
  - $x \uparrow n = xx \cdots x$
  - $x \uparrow\uparrow n = x \uparrow (x \uparrow (\cdots \uparrow x) \cdots)$
  - Per exemple,

$$\begin{aligned} 10 \uparrow\uparrow 10 &= 10^{10^{10^{10^{10^{10^{10^{10^{10^{10}}}}}}}}} \\ &= 1 \text{ seguit de devers} \\ &\quad 40000 \text{ zeros} \end{aligned}$$

  - La regla general és

$$x \overbrace{\uparrow\uparrow \cdots \uparrow}^{k \text{ fletxes}} n =$$

$$= \underbrace{x \overbrace{\uparrow \cdots \uparrow}^{k-1} (x \overbrace{\uparrow \cdots \uparrow}^{k-1} (\cdots \overbrace{\uparrow \cdots \uparrow}^{k-1} x) \cdots)}_{n \text{ ocurrències de } x}$$

Siguin $f, g : \mathbb{N} \to \mathbb{N}$.

- O gran $O(g)$ denota el conjunt de les funcions f que com a molt creixen tan ràpidament com g.

$$O(g) = \{ f \mid \exists c \in \mathbb{R}^+, n_0 \in \mathbb{N} : \forall n \geqslant n_0 : f(n) \leqslant c\, g(n) \}$$

- Omega gran $\Omega(g)$ denota el conjunt de les funcions f que creixen tan ràpidament com o més ràpidament que g.

$$\Omega(g) = \{ f \mid \exists c \in \mathbb{R}^+, n_0 \in \mathbb{N} : \forall n \geqslant n_0 : f(n) \geqslant c\, g(n) \}$$

- Theta gran $\Theta(g)$ denota el conjunt de les funcions f que creixen exactament al mateix ritme que g.

$$\Theta(g) = \{ f \mid f \in O(g), f \in \Omega(g) \}$$

- $O(g)$ denota el conjunt de les funcions f que com a molt creixen tan ràpidament com g.

$$O(g) = \{ f \mid \exists c \in \mathbb{R}^+, n_0 \in \mathbb{N} : \forall n \geqslant n_0 : f(n) \leqslant c\, g(n) \}$$

- La constant n_0 indica a partir de quin punt la funció g és una fita superior per a la funció f.
- La constant c formalitza l'expressió "mòdul una constant multiplicativa".
- Per exemple,

$$2n^2 \in O(n^4) \qquad n^4 \notin O(n^2)$$
$$2n^2 \in O(n^3) \qquad n^3 \notin O(n^2)$$
$$2n^2 \in O(n^2)$$

- $\Omega(g)$ denota el conjunt de les funcions f que creixen tan ràpidament com o més ràpidament que g.

$$\Omega(g) = \{ f \mid \exists c \in \mathbb{R}^+, n_0 \in \mathbb{N} : \forall n \geqslant n_0 : c\, g(n) \leqslant f(n) \}$$

- La constant n_0 indica a partir de quin punt la funció g és una fita inferior per a la funció f.
- La constant c formalitza l'expressió "mòdul una constant multiplicativa".
- Per exemple,

$$n^4 \in \Omega(n^2) \qquad n^2 \notin \Omega(n^4)$$
$$n^3 \in \Omega(n^2) \qquad n^2 \notin \Omega(n^3)$$
$$n^2 \in \Omega(n^2)$$

- $\Theta(g)$ denota el conjunt de les funcions f que creixen exactament al mateix ritme que g.

$$\begin{aligned}\Theta(g) &= \{ \quad f \mid f \in O(g), f \in \Omega(g) \quad \} \\ &= \{ \quad f \mid \exists c_1, c_2 \in \mathbb{R}^+, n_0 \in \mathbb{N} : \\ &\qquad \forall n \geqslant n_0 : \\ &\qquad c_1\, g(n) \leqslant f(n) \leqslant c_2\, g(n) \quad \}\end{aligned}$$

- La constant n_0 indica a partir de quin punt la funció g és una fita inferior per a la funció f.
- Per exemple,

$$n \notin \Theta(n^2) \qquad n^2 \notin \Theta(n)$$
$$100\, n^2 \in \Theta(n^2)$$

- Reflexivitat
  $f \in \Theta(f)$.
- Transitivat
  Si $f \in \Theta(g)$ i $g \in \Theta(h)$ aleshores $f \in \Theta(h)$.
- Regla de la suma
  Si $f_1 \in \Theta(g_1)$ i $f_2 \in \Theta(g_2)$ aleshores $f_1 + f_2 \in \Theta(\max(g_1, g_2))$.
- Regla del producte
  Si $f_1 \in \Theta(g_1)$ i $f_2 \in \Theta(g_2)$ aleshores $f_1\, f_2 \in \Theta(g_1\, g_2)$.
- Invariança additiva
  $\forall c \in \mathbb{R}^+ : f \in O(g)$ si i només si $c + f \in O(g)$.
- Invariança multiplicativa
  $\forall c \in \mathbb{R}^+ : f \in \Theta(g)$ si i només si $c\, f \in \Theta(g)$.
- Simetria
  $f \in O(g)$ si i només si $g \in \Omega(f)$.

Teorema (Reflexivitat)

$f \in \Theta(f)$.

Demostració.

Amb $c_1, c_2 = 1$ resulta, per a qualsevol $n_0 \in \mathbb{N}$, $c_1\, f(n) \leqslant f(n) \leqslant c_2\, f(n)$ per a tot $n \geqslant n_0$. □

Teorema (Transitivat)

Si $f \in \Theta(g)$ *i* $g \in \Theta(h)$ *aleshores* $f \in \Theta(h)$.

Demostració.

Sigui $f \in \Theta(g)$ i sigui $g \in \Theta(h)$. Per definició, existeixen $c_1, c_2, c_3, c_4 \in \mathbb{R}^+$ i $n_0, n_1 \in \mathbb{N}$ tals que, per a tot $n \geqslant n_0$, $c_1\, g(n) \leqslant f(n) \leqslant c_2\, g(n)$ i per a tot $n \geqslant n_1$, $c_3\, h(n) \leqslant g(n) \leqslant c_4\, h(n)$.

Això vol dir que per a tot $n \geqslant \max(n_0, n_1)$, $c_1\, c_3\, h(n) \leqslant f(n) \leqslant c_2\, c_4\, h(n)$.

Aleshores $f \in \Theta(h)$. □

Teorema (Regla de la suma)

Si $f_1 \in \Theta(g_1)$ *i* $f_2 \in \Theta(g_2)$ *aleshores* $f_1 + f_2 \in \Theta(\max(g_1, g_2))$.
És a dir, $\Theta(g_1) + \Theta(g_2) = \Theta(\max(g_1, g_2))$.

Demostració.

Siguin $f_1 \in \Theta(g_1)$ i $f_2 \in \Theta(g_2)$. Per definició, existeixen $c_1, c_2, c_3, c_4 \in \mathbb{R}^+$ i $n_1, n_2 \in \mathbb{N}$ tals que, per a tot $n \geqslant n_1$, $c_1\, g_1(n) \leqslant f_1(n) \leqslant c_2\, g_1(n)$ i per a tot $n \geqslant n_2$, $c_3\, g_2(n) \leqslant f_2(n) \leqslant c_4\, g_2(n)$.

Això vol dir que per a tot $n \geqslant \max(n_1, n_2)$,
$c_1\, g_1(n) + c_3\, g_2(n) \leqslant f_1(n) + f_2(n) \leqslant c_2\, g_1(n) + c_4\, g_2(n)$, per la qual cosa
$\min(c_1, c_3)\,(g_1(n) + g_2(n)) \leqslant f_1(n) + f_2(n) \leqslant \max(c_2, c_4)\,(g_1(n) + g_2(n))$
per a tot $n \geqslant \max(n_1, n_2)$ i, per tant,
$\min(c_1, c_3)\ \max(g_1(n), g_2(n)) \leqslant f_1(n) + f_2(n) \leqslant 2\ \max(c_2, c_4)\ \max(g_1(n), g_2(n))$ per a tot $n \geqslant \max(n_1, n_2)$.

Aleshores $f_1 + f_2 \in \Theta(\max(g_1, g_2))$. □

Teorema (Regla del producte)

Si $f_1 \in \Theta(g_1)$ *i* $f_2 \in \Theta(g_2)$ *aleshores* $f_1\, f_2 \in \Theta(g_1\, g_2)$. *És a dir,* $\Theta(g_1)\,\Theta(g_2) = \Theta(g_1\, g_2)$.

Demostració.

Siguin $f_1 \in \Theta(g_1)$ i $f_2 \in \Theta(g_2)$. Per definició, existeixen $c_1, c_2, c_3, c_4 \in \mathbb{R}^+$ i $n_1, n_2 \in \mathbb{N}$ tals que, per a tot $n \geqslant n_1$, $c_1\, g_1(n) \leqslant f_1(n) \leqslant c_2\, g_1(n)$ i per a tot $n \geqslant n_2$, $c_3\, g_2(n) \leqslant f_2(n) \leqslant c_4\, g_2(n)$.

Això vol dir que per a tot $n \geqslant \max(n_1, n_2)$,
$c_1\, c_3\, g_1(n)\, g_2(n) \leqslant f_1(n)\, f_2(n) \leqslant c_2\, c_4\, g_1(n)\, g_2(n)$.

Aleshores $f_1\, f_2 \in \Theta(g_1\, g_2)$. □

Teorema (Invariança additiva)

$\forall c \in \mathbb{R}^+ : f \in O(g)$ si i només si $c + f \in O(g)$.

Demostració.

Regla de la suma amb $f_2(n) = c$ (per a la demostració directa) i amb $f_2(n) = -c$ (per a la demostració inversa) per a tot $n \in \mathbb{N}$. □

Teorema (Invariança multiplicativa)

$\forall c \in \mathbb{R}^+ : f \in \Theta(g)$ si i només si $c\,f \in \Theta(g)$.

Demostració.

Regla del producte amb $f_2(n) = c$ (per a la demostració directa) i amb $f_2(n) = 1/c$ (per a la demostració inversa) per a tot $n \in \mathbb{N}$. □

Teorema (Simetria)

$f \in O(g)$ *si i només si* $g \in \Omega(f)$.

Demostració.

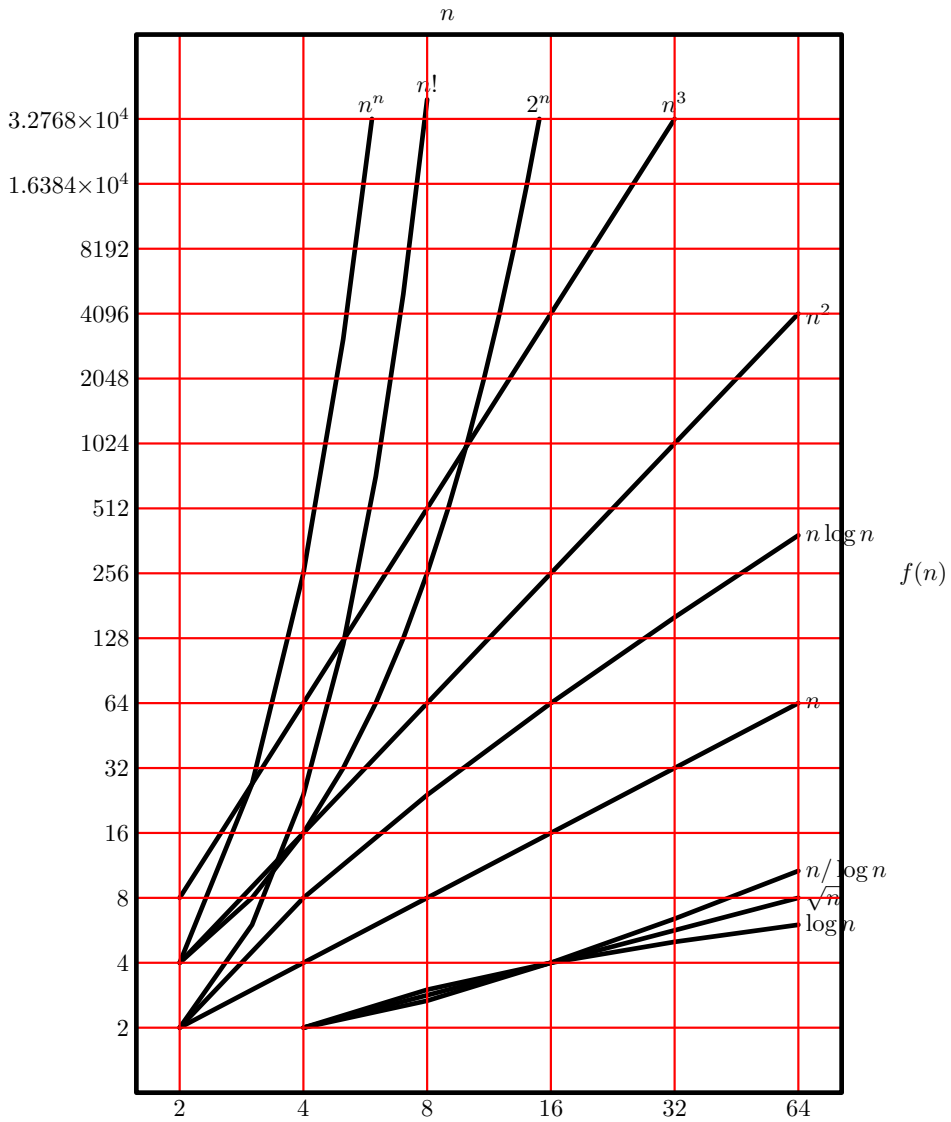
Sigui $f \in O(g)$. Per definició, existeixen $c \in \mathbb{R}^+$ i $n_0 \in \mathbb{N}$ tals que, per a tot $n \geqslant n_0$, $f(n) \leqslant c\, g(n)$.

Donat que $c \neq 0$, $(1/c)\, f(n) \leqslant g(n)$ per a tot $n \geqslant n_0$. Aleshores $g \in \Omega(f)$.

La demostració inversa és similar. □

- Cost constant $\Theta(1)$. No depèn de la mida de les dades.
- Cost logarísmic $\Theta(\log n)$. Per exemple, la cerca dicotòmica en un vector ordenat de n elements.
- Cost lineal $\Theta(n)$. Per exemple, la cerca seqüencial d'un element en un vector desordenat de n elements.
- Cost quasi-lineal $\Theta(n \log n)$. Per exemple, l'ordenació d'un vector de n elements.
- Cost polinòmic $\Theta(n^k)$ amb k constant.
  - Cost quadràtic $\Theta(n^2)$. Per exemple, el recorregut de una matriu de n files i n columnes.
  - Cost cúbic $\Theta(n^3)$. Per exemple, el producte de dues matrius de n files i n columnes.
- Cost exponencial $\Theta(k^n)$ amb k constant. Per exemple, la cerca (heurística) en un espai d'estats de amplada k i profunditat n.

n	$\log n$	$\sqrt{n}$	n	$n \log n$	n^2	n^3	2^n	$n!$
1	0	1	1	0	1	1	2	1
10	2.3	3	10	23	100	10^3	1024	10^6
100	4.6	10	100	461	10^4	10^6	10^{30}	10^{158}
10^3	6.9	32	10^3	6 908	10^6	10^9	10^{301}	$10^{2\,567}$
10^4	9.2	100	10^4	92 103	10^8	10^{12}	10^{3010}	$10^{35\,659}$
10^5	11.5	316	10^5	1 151 290	10^{10}	10^{15}	$10^{30\,103}$	$10^{456\,573}$


n
3.2768×10^4
1.6384×10^4
8192
4096
2048
1024
512
256
128
64
32
16
8
4
2
2
4
8
16
32
64
n^n
n!
2^n
n^3
n^2
n log n
n
n/log n
√n
log n
f(n)

Problema (1.2)

Contesteu les preguntes següents sobre logaritmes:

- *Quants bits calen per representar un nombre natural entre 0 i* $n-1$*?*
- *Començant per* $x = 1$*, quants cops cal doblar* x *fins que sigui més gran o igual que* n*?*
- *Començant per* $x = n$*, quants cops cal dividir* x *per dos fins que sigui menor o igual que* 1*? I si es divideix per tres?*

Problema (1.3)

Demostreu les igualtats següents per inducció:

- $\sum_{i=1}^{n} i = n(n+1)/2$.
- $\sum_{i=1}^{n} i^2 = n(n+1)(2n+1)/6$.

Problema (1.4)

Demostreu que $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$.

Problema (1.6)

Demostreu les propietats asimptòtiques següents:

- *Tot polinomi* $p(n) = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_0 n^0$ *amb* $a_k > 0$ *pertany a* $\Theta(n^k)$.
- *Per a qualssevol bases* $a, b > 1$, *es té que* $\log_a n = \Theta(\log_b n)$.

Problema (1.7)

Demostreu que $\log(n!) = \Theta(n \log n)$.

Problema (1.12)

Per a cada funció $f(n)$ i temps t de la taula següent, determineu la talla més gran que es pot resoldre en temps t d'un problema que necessita un temps de $f(n)$ μs per executar-se sobre una entrada de talla n.

	1 segon	*1 minut*	*1 hora*	*1 dia*	*1 mes*	*1 any*	*1 segle*
$\log n$							
$\sqrt{n}$							
n							
n^2							
n^3							
2^n							

Problema (1.13)

La tecnologia actual permet resoldre, en un cert temps T, un problema de talla fins a $n = 10^8$ amb un algorisme de complexitat $\Theta(n)$. Suposeu que la constant amagada en la notació asimptòtica és la mateixa per a tots els algorismes. Quina és la talla n de la instància més gran que es pot resoldre en temps T amb un algorisme de complexitat $\Theta(n\sqrt{n})$ i per què?

Suposeu que amb la tecnologia d'aquí a deu anys, els processadors siguin 100 cops més ràpids. Quina serà llavors la talla n de la instància més gran que es podrà resoldre en temps T amb un algorisme de complexitat $\Theta(n\sqrt{n})$?

Problema (1.14)

Considereu les funcions següents: $5n \ln n$, $4n\sqrt{n}$, $\log_\pi(n^n)$, $n/\log_2 n$, $n/\ln n$, $3\ln(7^n)$. *Ordeneu-les, de menor a major, segons el seu creixement asimptòtic. Si dues o més funcions tenen el mateix grau de creixement, indiqueu-ho.*

Problema (1.15)

Agrupeu les funcions següents de manera que $f(n)$ i $g(n)$ pertanyin al mateix grup si i només si $f(n) \in \Theta(g(n))$, i enumereu els grups segons el seu ordre de creixement:

n, $\sqrt{n}$, $n^{1,5}$, n^2, $n \log n$, $n \log \log n$, $(\log \log n)^2$, $n \log^2 n$, $n^3/(n-1)$, $n^{\log n}$, $n \log(n^2)$, $21n$, 2^n, $2^{\sqrt{n}}$, $2^{\sqrt{\log n}}$, $2^{n/2}$, 37, 2^{2^n}, $2/n$ *i* $n^2 \log n$.

- Referència a un objecte (de tipus predefinit o bé escalar) i la seva manipulació mitjançant operacions aritmètiques i lògiques, incloent-hi indexacions de vectors i seleccions de tuples. Constant.
- Avaluació d'una expressió. Suma dels costos de les subexpressions que la componen i dels operadors que les combinen.
- Invocació a una funció. Suma del cost d'avaluar els paràmetres reals més el cost d'executar el cos de la funció.
  - Es considera que el pas de paràmetres d'entrada no exigeix fer-ne una còpia si la funció no els modifica.
  - No es considera la formulació d'equacions de recurrència, típiques del cas de les funcions recursives.
- Composició seqüencial d'instruccions. Suma dels costos de les instruccions que componen la seqüència.
- Assignació. Suma del cost d'avaluar l'expressió que identifica la part esquerra i del cost d'avaluar l'expressió de la part dreta.

- Composició alternativa d'instruccions. Suma dels costos d'avaluar les diferents expressions condicionals i les instruccions associades.
  - S'examinen totes les branques possibles i predomina el cost de la condició o de l'expressió més costosa.
- Composició iterativa d'instruccions. Suma del cost d'avaluar la condició de la iteració i de les instruccions que en formen el cos multiplicades pel nombre de vegades que s'executa la iteració.
- Invocació a una acció. Suma del cost d'avaluar els paràmetres reals més el cost d'executar el cos de l'acció.
  - Es considera que el pas de paràmetres d'entrada no exigeix fer-ne una còpia si l'acció no els modifica.
- Retorn d'un valor. Cost d'avaluar l'expressió.
- Tractament d'errors. Constant.

Exemple

Ordenació d'un vector pel mètode de la bombolla.

procedure BubbleSort $(A, 1, n)$
  for $i := 1$ **to** $n - 1$ **do**
    for $j := n$ **downto** $i + 1$ **do**
      if $A[j-1] > A[j]$ **then**
        $A[j-1] \leftrightarrow A[j]$
      end if
    end for
  end for
end procedure

- $n - 1$ iteracions exteriors, $n - i$ iteracions interiors

$$\sum_{i=1}^{n-1}(n-i) = n(n-1) - \frac{n(n-1)}{2} = \frac{n(n-1)}{2} = \Theta(n^2)$$

- Espai que ocupa un objecte (de tipus predefinit o bé escalar). Constant.
- Espai que ocupa un vector. Producte de l'espai que ocupa cada component per la dimensió del vector.
- Espai que ocupa una estructura. Suma de l'espai que ocupen els components de l'estructura.

Problema (1.16)

De cadascun dels fragments de codi següents, analitzeu el seu cost.

1. ```
   int s = 0;
   for (int i=0; i<n; ++i) {
     ++s;
   }
   ```
2. ```
   int s = 0;
   for (int i=0; i<n; i+=2) {
     ++s;
   }
   ```

Problema (1.16)

De cadascun dels fragments de codi següents, analitzeu el seu cost.

3.
```
int s = 0;
for (int i=0; i<n; ++i) {
  ++s;
}
for (int j=0; j<n; ++j) {
  ++s;
}
```

4.
```
int s = 0;
for (int i=0; i<n; ++i) {
  for (int j=0; j<n; ++j) {
    ++s;
} }
```

Problema (1.16)

De cadascun dels fragments de codi següents, analitzeu el seu cost.

5.
```
int s = 0;
for (int i=0; i<n; ++i) {
  for (int j=0; j<i; ++j) {
    ++s;
} }
```

6.
```
int s = 0;
for (int i=0; i<n; ++i) {
  for (int j=i; j<n; ++j) {
    ++s;
} }
```

Problema (1.16)

De cadascun dels fragments de codi següents, analitzeu el seu cost.

7.
```
int s = 0;
for (int i=0; i<n; ++i) {
  for (int j=0; j<n; ++j) {
    for (int k=0; k<n; ++k) {
      ++s;
} } }
```

8.
```
int s = 0;
for (int i=0; i<n; ++i) {
  for (int j=0; j<i; ++j) {
    for (int k=0; k<j; ++k) {
      ++s;
} } }
```

Problema (1.16)

De cadascun dels fragments de codi següents, analitzeu el seu cost.

9.
```
int s = 0;
for (int i=1; i<=n; i*=2) {
  ++s;
}
```

10.
```
int s = 0;
for (int i=0; i<n; ++i) {
  for (int j=0; j<i*i; ++j) {
    for (int k=0; k<n; ++k) {
      ++s;
} } }
```

Problema (1.16)

De cadascun dels fragments de codi següents, analitzeu el seu cost.

11.
```
int s = 0;
for (int i=0; i<n; ++i) {
  for (int j=0; j<i*i; ++j) {
    if (j%i==0) {
      for (int k=0; k<n; ++k) {
        ++s;
} } } }
```

Problema (1.17)

Digueu quin és el cost de les instruccions (1), (2) i (3) del programa següent.

```
int funcio1 (vector<int>& v) { return v[0]; }
int funcio2 (vector<int>  v) { return v[0]; }

int f (int n) {
  vector<int> v(n,0); // (1)
  int a = funcio1(v); // (2)
  int b = funcio2(v); // (3)
  return a+b;
}
```

Problema (1.19)

Utilitzeu la notació asimptòtica més indicada (O, Ω, Θ) per indicar el cost en temps de la cerca seqüencial.

- *en el cas pitjor,*
- *en el cas millor,*
- *en el cas mitjà.*

Problema (1.20)

Considereu una taula T amb n elements.

- *Escriviu un algorisme que usi exactament $n-1$ comparacions per trobar l'element mínim de T.*
- *Escriviu un algorisme que usi exactament $n-1$ comparacions per trobar l'element màxim de T.*
- $\star$ *Escriviu un algorisme que trobi els elements mínim i màxim de T amb només unes $3n/2$ comparacions.*

Problema (1.21)

Quin és el cost en temps i en espai dels algorismes "escolars" per

- *sumar dos naturals de n dígits,*
- *multiplicar dos naturals de n dígits.*

Problema (1.22)

Escriviu l'algorisme clàssic per sumar dues matrius $n \times n$. Expliqueu perquè aquest algorisme de cost $O(n^2)$ és lineal.

Demostreu que qualsevol algorisme per sumar dues matrius $n \times n$ requereix $\Omega(n^2)$ passos.

Problema (1.23)

Escriviu l'algorisme clàssic per multiplicar dues matrius $n \times n$. Quin és el seu cost en funció de n? Quin és el seu cost en funció de la talla de les entrades?

Demostreu que qualsevol algorisme per multiplicar dues matrius $n \times n$ requereix $\Omega(n^2)$ passos.

Problema (1.24)

Escriviu un algorisme per determinar si un número natural n és primer. Quin és el seu cost en funció de n?

Problema (1.25)

Escriviu i analitzeu l'algorisme corresponent al garbell d'Eratóstenes per determinar tots els nombres primers entre 2 i n.

- Una equació de recurrència és una expressió que dóna el valor de $f(n)$ en termes de $f(n')$ per a un valor $n' \leqslant n$ més petit de la variable

$$\begin{aligned} C(n,1) &= 1 \\ C(n,n) &= 1 \\ C(n,k) &= C(n-1,k) + C(n-1,k-1) \end{aligned}$$

- Les equacions de recurrència es poden reduir al que s'anomena una solució tancada, una expressió que pot ser avaluada en una quantitat fixa d'operacions aritmètiques elementals (sumes i diferències, productes i quocients, exponenciació i radicació)
- També s'admet com a expressió tancada el factorial d'un nombre (encara que, de fet, correspon a una quantitat variable de multiplicacions) i sovint també expressions amb sumatoris

$$C(n,k) = \binom{n}{k} = \frac{n!}{k!\,(n-k)!}$$

Exemple

Factorial.

$$T(n) = \begin{cases} k_1 & \text{si } n = 0 \\ T(n-1) + k_2 & \text{si } n \geqslant 1 \end{cases}$$

Per a $n > 0$,

$$\begin{aligned} T(n) &= T(n-1) + k_2 \\ &= T(n-2) + k_2 + k_2 \\ &= \cdots \\ &= T(n-n) + \underbrace{k_2 + \cdots + k_2}_{n} \\ &= T(0) + n\,k_2 \\ &= n\,k_2 + k_1 \\ &= \Theta(n) \end{aligned}$$

- La talla del problema decreix aritmèticament. Una crida recursiva sobre un problema de talla n genera a crides recursives sobre subproblemes de talla $n - c$, amb c constant.

$$T(n) = a \cdot T(n - c) + g(n)$$

- La talla del problema decreix geomètricament. Una crida recursiva sobre un problema de talla n genera a crides recursives sobre subproblemes de talla n/b, amb b constant i $b > 1$.

$$T(n) = a \cdot T(n/b) + g(n)$$

Teorema (Màster I)

Sigui $T(n)$ el cost d'un algorisme recursiu descrit per la recurrència

$$T(n) = \begin{cases} f(n) & \text{si } 0 \leqslant n < n_0 \\ a \cdot T(n-c) + g(n) & \text{si } n \geqslant n_0 \end{cases}$$

on n_0 és una constant, $c \geqslant 1$, $f(n)$ és una funció arbitrària i $g(n) = \Theta(n^k)$ amb k constant. Aleshores,

$$T(n) = \begin{cases} \Theta(n^k) & \text{si } a < 1 \\ \Theta(n^{k+1}) & \text{si } a = 1 \\ \Theta(a^{n/c}) & \text{si } a > 1 \end{cases}$$

Exemple

La solució de l'equació de recurrència

$$\begin{aligned} T(1) &= 1 \\ T(n) &= T(n-1)+n \qquad \text{per a } n \geqslant 2 \end{aligned}$$

és $T(n) = \Theta(n^2)$, atès que $a = 1$ i $n^{k+1} = n^2$.

Teorema (Màster II)

Sigui $T(n)$ el cost d'un algorisme recursiu descrit per la recurrència

$$T(n) = \begin{cases} f(n) & \text{si } 0 \leqslant n < n_0 \\ a \cdot T(n/b) + g(n) & \text{si } n \geqslant n_0 \end{cases}$$

on n_0 és una constant, $b > 1$, $f(n)$ és una funció arbitrària i $g(n) = \Theta(n^k)$ amb $k \geqslant 0$ constant. Aleshores,

$$T(n) = \begin{cases} \Theta(n^k) & \text{si } a < b^k \\ \Theta(n^k \log n) & \text{si } a = b^k \\ \Theta(n^{\log_b a}) & \text{si } a > b^k \end{cases}$$

Exemple

La solució de l'equació de recurrència

$$\begin{aligned} T(1) &= 0 \\ T(n) &= 2\,T(n/2) + n \qquad \text{per a } n \geqslant 2 \end{aligned}$$

és $T(n) = \Theta(n \log n)$, atès que $2 = a = b^k = 2^1$.

Demostració.

Considerem $T(n) = \Theta(n^k) + a \cdot T(n/b)$ per a $n \geqslant n_0$. Sigui $\alpha = \log_b a$, i sigui $C(n) = T(n)/n^\alpha$. Aleshores,

$$\begin{aligned} C(n) &= \frac{T(n)}{n^\alpha} = \Theta(n^{k-\alpha}) + \frac{a}{n^\alpha}\, T(n/b) \\ &= \Theta(n^{k-\alpha}) + \frac{a}{n^\alpha}\left(\frac{n}{b}\right)^\alpha C(n/b) \\ &= \Theta(n^{k-\alpha}) + \frac{a}{b^\alpha}\, C(n/b) \\ &= \Theta(n^{k-\alpha}) + C(n/b) \end{aligned}$$

Demostrarem que

$$C(n) = \begin{cases} \Theta(n^{k-\alpha}) & \text{si } k-\alpha > 0 \\ \Theta(\log n) & \text{si } k-\alpha = 0 \\ \Theta(1) & \text{si } k-\alpha < 0 \end{cases}$$

Demostració.

amb la qual cosa

$$T(n) = \begin{cases} \Theta(n^k) & \text{si } k - \alpha > 0 \equiv a < b^k \\ \Theta(n^k \log n) & \text{si } k - \alpha = 0 \equiv a = b^k \\ \Theta(n^\alpha) & \text{si } k - \alpha < 0 \equiv a > b^k \end{cases}$$

De fet, sigui $C(n) = \Theta(n^c) + C(n/b)$ per a $n \geqslant n_0$. Aleshores,

$$\begin{aligned} C(n) &= \Theta(n^c) + \Theta((n/b)^c) + C(n/b^2) = \cdots \\ &= \Theta(n^c + (n/b)^c + (n/b^2)^c + \cdots + (n/b^{\log_b n})^c \\ &= \Theta(n^c)\left(1 + 1/b^c + 1/b^{2c} + \cdots + 1/b^{(\log_b n)c}\right) \end{aligned}$$

Demostració.

Cal distingir tres casos:

$(c > 0)$

$$C(n) = \Theta(n^c)\Theta(1) = \Theta(n^c)$$

$(c = 0)$

$$C(n) = \Theta(1)(1 + \underbrace{1/1 + \cdots + 1/1}_{\Theta(\log n)}) = \Theta(\log n)$$

$(c < 0)$

$$\begin{aligned} C(n) &= \Theta(n^c)\left(1 + b^{-c} + b^{-2c} + \cdots + b^{-(\log_b n)c}\right) \\ &= \Theta(n^c)\Theta(n^{-c}) = \Theta(1) \end{aligned}$$

Problema (1.27)

Resoleu les recurrències subtractores següents:

- $T(n) = T(n-1) + \Theta(1)$,
- $T(n) = T(n-2) + \Theta(1)$,
- $T(n) = T(n-1) + \Theta(n)$,
- $T(n) = 2T(n-1) + \Theta(1)$.

Problema (1.28)

Resoleu les recurrències divisores següents:

- $T(n) = 2T(n/2) + \Theta(1)$,
- $T(n) = 2T(n/2) + \Theta(n)$,
- $T(n) = 2T(n/2) + \Theta(n^2)$,
- $T(n) = 2T(n/2) + O(1)$,
- $T(n) = 2T(n/2) + O(n)$,
- $T(n) = 2T(n/2) + O(n^2)$,
- $T(n) = 4T(n/2) + n$,
- $T(n) = 4T(n/2) + n^2$,
- $T(n) = 4T(n/2) + n^3$,
- $T(n) = 9T(n/3) + 3n + 2$,
- $T(n) = T(9n/10) + \Theta(n)$,
- $T(n) = 2T(n/4) + \sqrt{n}$.

Problema (1.31)

Digueu quins algorismes d'ordenació elementals són estables. Per als que ho són, justifiqueu per què ho són; per als que no ho són, doneu un contraexemple.

Problema (1.32)

Digueu quan triguen els algorismes d'ordenació elementals quan l'entrada es troba...

- *permutada a l'atzar,*
- *ordenada,*
- *ordenada del revés,*
- *amb tots els elements iguals.*

Problema (1.34)

Tenim una taula t amb n elements i desitgem saber si k_n altres elements són dins de la taula t o no. Hi ha, com a mínim, dues maneres possibles de procedir:

Opció 1: *Per a cadascun dels k_n elements, fem una cerca lineal a la taula t.*

Opció 2: *Ordenem primer la taula t amb un algorisme $\Theta(n \log n)$ i, després, per a cadascun dels k_n elements, fem una cerca dicotòmica a la taula t.*

Digueu quin ha de ser el valor mínim de k_n (utilitzant notació asimptòtica) perquè la segona opció sigui més ràpida o igual que la primera.

Problema (1.35)

Analitzeu el cost de les funcions recursives següents per calcular x^n per a $n \geqslant 0$.

1. ```
   double potencia_1 (double x, int n) {
     if (n==0) {
       return 1;
     } else {
       return x*potencia_1(x,n-1);
   } }
   ```
   ```

Problema (1.35)

Analitzeu el cost de les funcions recursives següents per calcular x^n per a $n \geqslant 0$.

2.
```
double potencia_2 (double x, int n) {
  if (n==0) {
    return 1;
  } else if (n%2==0) {
    int y = potencia_2(x,n/2);
    return y*y;
  } else {
    int y = potencia_2(x,n/2);
    return y*y*x;
} }
```

Problema (1.35)

Analitzeu el cost de les funcions recursives següents per calcular x^n per a $n \geqslant 0$.

3.
```
double potencia_3 (double x, int n) {
  if (n==0) {
    return 1;
  } else if (n%2==0) {
    return potencia_3(x,n/2) * potencia_3(x,n/2);
  } else {
    return potencia_3(x,n/2) * potencia_3(x,n/2) * x;
} }
```

Problema (1.36)

Considereu les dues funcions següents per trobar el màxim d'una taula de n reals (amb $n \geqslant 1$). Calculeu el cost dels dos algorismes en funció de la talla de la taula n. Quin d'ells escolliríeu? Per què?

1. ```
   // Crida inicial: maxS(t,n-1)
   double maxS (vector<double>& t, int i) {
     if (i==0) {
       return t[0];
     } else {
       double m = maxS(t,i-1);
       return t[i]<m ? m : t[i];
   } }
   ```
   ```

Problema (1.36)

Considereu les dues funcions següents per trobar el màxim d'una taula de n reals (amb $n \geqslant 1$). Calculeu el cost dels dos algorismes en funció de la talla de la taula n. Quin d'ells escolliríeu? Per què?

2.
```
// Crida inicial: maxD(t,0,n-1)
double maxD (vector<double>& t, int i, int j) {
  if (i==j) {
    return t[i];
  } else {
    int k  = (i+j)/2;
    double m1 = maxD(t,i,k);
    double m2 = maxD(t,k+1,j);
    return m1>m2 ? m1 : m2;
} }
```

Problema (1.37)

Un algorisme A té un cost donat per la recurrència $T_A(n) = 7T_A(n/2) + n^2$. Un algorisme competidor B té cost $T_B(n) = xT_B(n/4) + n^2$. Quin és l'enter x més gran per al qual B és asimptòticament millor que A?