

## Intraduction to algorithms

1. Insertion-Sort:  $T_a(n) = n^2 - 4n - 8, n \geq 0$  [8 steps]

Merge-Sort:  $T_b(n) = 8n \lg(n) + 4n + 2, n \geq 0$  [steps]

When is  $T_a$  faster than  $T_b$ ?

i.e.  $T_b > T_a$  for  $n \geq 0$

$$8n \lg(n) + 4n + 2 > n^2 - 4n - 8$$

$$8n \lg(n) + 8n + 10 - n^2 > 0$$

First of all,

$$\inf \{n \in \mathbb{R}^+ : 8n \lg(n) + 8n + 10 - n^2 > 0\} = 0$$

Because we can see from  $T_a$  and  $T_b$  domain that at least  $n=0$  (a good place to start).

Second,

$$\sup \{n \in \mathbb{R}^+ : 8n \lg(n) + 8n + 10 - n^2 > 0\} \approx 54.2838$$

Since Raphson's method we solve

$$8n \lg(n) + 8n + 10 - n^2 = 0$$

$$n \approx 54.2838$$

As  $n$  is unique we guess that is supremum.

Hence,

That  $n \in (0, 54.243819618056]$  our domain such that  $T_b > T_a$ .

But we made a mistake by  $n$  and  $f(n)$  are  $\mathbb{Z}$ :  $f: \mathbb{Z} \rightarrow \mathbb{Z}$ , ( $f = T_b - T_a$ )

In computational world

$$n \in (0, 54]$$

As

$$\frac{f(n)}{\text{Operations}} = 10^9 \text{ ns}$$

$f(n)$  : instructions       $O$  : instructions by ns

Then

$$f(n) = (10^9)(O)$$

But  $O = 1$  instruction by 1 ns

So,

$$f(n) = 10^9 \text{ ns}$$

$$n = f^*(10^9) \quad [\text{Max instructions}]$$

Where parameter of  $f^*$  is ns

$$1 \text{ s} : n = f^*(10^9)$$

$$1 \text{ h} : n = f^*(10^9 \cdot 60 \cdot 60)$$

$$50 \text{ days} : n = f^*(10^9 \cdot 60 \cdot 60 \cdot 24 \cdot 30)$$

$$1 \text{ century} : n = f^*(10^9 \cdot 60 \cdot 60 \cdot 24 \cdot 30 \cdot 12 \cdot 100)$$

Note - NEN, which  $f(n)$  is monotonically increasing. Then we can say  $f(n) \leq c$ ,  $n \geq n_0$

$\lg n$	1 seg	1 hora	30 dias	1 año
$\lg n$	$2(10^9)$	$(10^9 \cdot 60 \cdot 60)$ $2$	$(10^9 \cdot 60 \cdot 60 \cdot 24 \cdot 30)$ $2$	$(10^9 \cdot 60 \cdot 60 \cdot 24 \cdot 30 \cdot 365)$ $2$
$n$	$10^9$	$3.6 \times 10^{12}$	$2.592_{\text{e+15}}$	$3.1104_{\text{e+18}}$
$n^2$	$31623$	$1897367$	$50911689$	$8.64_{\text{e+8}}$
$2^n$	30	92	52	6193

Note.  $2^{10} \approx 10^3$ , thus we have  $(2^{10})(2^{10}) \approx (10^3)(10^3)$

But the error is greater than 100%.

### 3 Problema de búsqueda.

Entrada. Se encuen dan números  $A = \langle a_1, a_2, \dots, a_n \rangle$  y un

Salida. Un índice  $i$  tal que  $v == A[i]$  o  
un valor especial NIL si  $v$  no se encuentra en  $A$ .

buscar( $A, v$ )

```
for i = 1 to A.length
    if A[i] == v
        return i
return NIL
```

Nota:  $t_0 = i + 1$

Invariante. En el rango de cada iteración del  
for de las líneas 1-3,  $v$  no está en  $A[1 \dots i-1]$   
porque  $v == A[i]$  (alternativa 1) o hasta que  $i+1 > n$

Mantenimiento. En la primera iteración del  
for de las líneas 1-3,  $v$  no está en el  
subarreglo  $A[1 \dots i-1] \subseteq [ ]$  (corregido cuando)  
el cual por definición, no tiene ningún  
elemento, cumpliéndose a priori la invariante de  
lazo.

Mantenimiento. Antes de la iteración,  
esto es,  $A[1 \dots i-1]$  no contiene a  $v$   
por invariante de lazo.

Línea #2 Verifica que  $A[i] == v$ , si es  
cierta, se cumple la alternativa 1 de la  
invariante, es decir,  $v$  se ha visto y se resuelve  
el problema (#3 línea 7)

En caso de no ser cierta, en la iteración  
siguiente  $v$  no está en  $A[1 \dots i-1]$   
( $i$  sumó una unidad). Cumpliendo la invariante

Finalización. La primera alternativa de  
la iteración se obtuvo en el mantenimiento  
cuando  $A[i] == v$ .

La segunda alternativa ocurre cuando  
 $i+1 > n$  de acuerdo a la invariante  
 $v$  no está en  $A[1 \dots i-1] = A[1 \dots n]$ ,  
siguiendo con un NIL.

Se ha resuelto el problema.

Mejor caso:  $E_3$  da menor respuesta.

A  $\{i=1\}$  se le asigna el valor de la concentración en la otra población.

$$T_{C(i)} = C_1 + C_2 + T_3$$

$$T_{C(i)} = C \quad (C \text{ es una constante})$$

(\*) ¿entonces  $C(i) = \Theta(A)$ ?

$$T_{C(i)} = \Theta(A)$$

$$C = \Theta(A)$$

Entonces  $C_1 + C_2 > 0 \Rightarrow$  no habrá que

$$0 \leq C_1 q(n) + f(n) \leq C_2 q(n)$$

$$0 \leq C_1 q(n) \leq C_2 q(n)$$

$$0 \leq \frac{C_1}{C} \leq 1 \leq \frac{C_2}{C}$$

Para  $n_0 = 1,000,000$ ,

$$1 \in (\Theta, \Theta)$$

Por lo tanto,

$$0 \leq \frac{C_1}{C} \leq 1$$

$$\frac{C_2}{C} \geq 1$$

Es cuando que  $C = \Theta(A)$

$$O(1) \leq O(n)^2$$

Entonces  $C_1$  y  $n_0$  se obtienen.

$$O(n^2) \leq C_1 n$$

$$O(n^2) \leq C_1 n^2$$

Para  $n_0 = 1$  som 6.6 0.21

$$1 \in [1-0]$$

Para lo anterior

$$\frac{C_1}{n} \geq \frac{1}{n}$$

Se cumple que  $C = O(\gamma)$ .

$$\text{II: } C \leq O(1)^2$$

Entonces  $C_1$  y  $n_0$  se obtienen.

$$O(n^2) \leq C_1 n^2$$

$$O(n^2) \leq C_1 n^2$$

Entonces que  $n_0 = 1$ ,  $1 \in [1-0]$

$$1 \geq \frac{C_1}{n}$$

Se cumple que  $C = O(1)$

Por otro lado, Ocurre el caso de que el vector  $v$  no esta en el array.

$$T(n) = C_1(n+1) + C_2 n + C_3$$

$$T(n) = an + b \quad a, b \in \mathbb{R}$$

$$\Theta: \epsilon_{an+b} = \Theta(n)$$

$$\exists C_1, C_2 > 0 \text{ y } \forall n \geq n_0$$

$$T(n) \leq C_1 n \leq an + b \leq C_2 n$$

$$0 \leq C_1 \leq a + \left(\frac{b}{n}\right) \leq C_2$$

$$\text{Como } n_0 = 1$$

$$a + \frac{b}{n} \in (a, a+b]$$

Teniendo

$$0 < C_1 \leq a$$

$$C_2 > a + b$$

$$\text{Se cumple que } an + b = \Theta(n^2)$$

$$\Theta: \delta an + b = \Theta(n^2)$$

$$\exists C > 0 \text{ y } \forall n \geq n_0$$

$$0 \leq \frac{a}{n} + \frac{b}{n^2} \leq C$$

$$\text{Para } n_0 = 1$$

$$\frac{a}{n} + \frac{b}{n^2} \in (0, a+b]$$

$$\text{Asi } C > a + b$$

$$\text{Se cumple } an + b = \Theta(n^2)$$

$$n \geq a_n + b = \Omega(n^{\frac{1}{2}}) ?$$

$$\exists c > 0 \text{ et } n_0 \in \mathbb{N} \quad 0 \leq c \leq a n^{\frac{1}{2}} + \frac{b}{n^{\frac{1}{2}}}$$

Pour  $n_0 = 1$   
 $a n^{\frac{1}{2}} + \frac{b}{n^{\frac{1}{2}}} \in [a+b - \epsilon, a+b]$

Asr

$$0 < c \leq a + b$$

Obtenons

$$a n + b = \Omega(n^{\frac{1}{2}})$$

Caso particular

$$T_{\text{cas}} = E[X] = \sum_{i=1}^m x_i f(x_i) = \sum_{i=1}^m i \left(\frac{1}{m+1}\right) = \frac{1}{m+1} (m+1)$$

$X =$  Cantidad de pares que son 2 al azar  
(CUPA)

$f(x)$ : Todos tienen la misma probabilidad  $f(x) = \frac{1}{m+1}$   
 $x_i = 1, 2, 3, 4, \dots, m+1$  (longitudes de 2)

Para calcular que

$$T_{\text{cas}} = an + b$$

esta dividido por 1 (como se demuestra arriba):

$$\Theta(n) = an + b$$

$$\Theta(n^2) = an + b$$

$$\sqrt{n} = an + b$$

Ast

$$a = \frac{1}{2} \quad \text{y} \quad b = 1$$

Son casos particulares del caso Universal

24

I.4

insertion-sort (seq, n)

If  $n > 1$

insertion-sort (seq, n-1)

merge (seq, n-2) //El mismo que el merge

return seq

Dividir. Calcular  $n-1$ , toma un tiempo constante.

Vencer. Recursivamente el resolver el subproblema, cada vez de tamaño  $n-1$ , lo que suministra  $T(n-1)$  en el tiempo de ejecución.

Combinar. El procedimiento merge toma  $\Theta(n)$  pasos en promedio de  $n$  elementos.

(Por que merge es el mismo que el insertion sort + iteraciones).

I.4

Hacer con Combinatoria o recursión.

$$T(n) = \begin{cases} \Theta(1) & \text{os nsi} \\ T(n-1) + \Theta(1) & n > 1 \end{cases}$$

$\Theta(1)$  para la combinación porque solo se ejecuta una vez.

Para  $n > 1$

$$T(n) = T(n-1) + \Theta(1)$$

$$T(n) = T(n-1) + C$$

Por abajo de recursividad

$$\begin{array}{c} C \quad 0 \\ | \\ C \quad 1 \\ | \\ C \quad 2 \\ | \\ C \quad 3 \\ | \\ C \quad \dots \end{array}$$

$$C \quad n-2 \quad (n-1=2)$$

Ast  $T(n) = \sum_{i=0}^{n-2} C = C(n-2) = Cn - 2C$

$$T(n) = an + b$$

Entonces  $a = \Theta(n)$ ,  $b = \Theta(n)$ ,  $an+b = \Theta(n)$ ,  $an+b = O(n^2)$

Para car. cuando el enemigo está muerto

$$T(n) = \begin{cases} \Theta(1) & 0 \leq n \leq 1 \\ T(n-1) + \Theta(n) & n > 1 \end{cases}$$

Le combinará la  $\Theta(n)$  porque invertir los datos  
para  $n > 1$

$$T(n) = T(n-1) + \Theta(n)$$

$$T(n) = T(n-1) + Cn$$

A rbd de recurrencias

$$Cn \quad \underline{-}^{\Theta}$$

$$C(n-1) \quad \underline{+}$$

$$C(n-2) \quad \underline{-}^2$$

$$C2 \quad \underline{n-i=2} \Rightarrow i=n-2$$

Entonces

$$T(n) = \sum_{i=0}^{n-2} (Cn - i) = an^2 + bn + c \text{ para } a, b, c \in \mathbb{R}^+$$

(4)  $i.e. \exists C_1, C_2 \in \mathbb{R}^+ \text{ y } \forall n \geq n_0 :$

$$0 \leq C_1 n^2 \leq an^2 + bn + c \leq C_2 n^2$$

$$0 \leq C_1 \Leftrightarrow a + \frac{b}{n} + \frac{c}{n^2} \leq C_2$$

Para  $n_0 = 1$

$$a + \frac{b}{n} + \frac{c}{n^2} \in [a, a+b+c]$$

Obteniendo

$$0 \leq C_1 \leq a \quad \text{y} \quad C_2 \geq a+b+c$$

Se cumple que  $\exists C_1, C_2 \in \mathbb{R}^+ \text{ y } \forall n \geq n_0 :$

$0 \leq an^2 + bn + c = O(n^2)$ ?

$\exists C_1, C_2 \in \mathbb{R}^+ \text{ y } \forall n \geq n_0 :$

$$an^2 + bn + c \leq C_1 n^2$$

$$a + \frac{bn}{n} + \frac{c}{n^2} \leq C_1 n$$

Para  $n_0 = 1$

$$a + \frac{b}{n} + \frac{c}{n^2} \in [0, a+b+c]$$

Obteniendo

$$C_1 \geq a+b+c$$

Se cumple que  $an^2 + bn + c = O(n^2)$

$\exists C_1, C_2 \in \mathbb{R}^+ \text{ y } \forall n \geq n_0 :$

$$0 \leq C_1 n^{1/2} \leq an^2 + bn + c n^{1/2}$$

$$C_1 \leq a n^{1/2} + b n^{1/2} + c n^{1/2}$$

Para  $n=1 \Rightarrow a n^{1/2} + b n^{1/2} + c n^{1/2} = [a+b+c] n^{1/2}$

Obteniendo  $O(C_1) \leq O(n^{1/2}) \Rightarrow an^2 + bn + c = \Omega(n^{1/2})$

I. S  $a, b = T(n) = \Theta(n^4), n \geq 2$

(1) Asymptotic upper and lower bound for  $T(n)$

$$T(n) = 2T\left(\frac{n}{2}\right) + n^4, n > 2$$

We have  $a=2$ ,  $b=2$  and  $f(n)=n^4$

Since

$$f(n) = \sqrt{n}^{1024(1)+\epsilon}$$

$$n^4 = \sqrt{n}^{1+\epsilon}, \text{ where } 1+\epsilon > 1, \epsilon = 3$$

$$n^4 = \sqrt{n}(n^4)$$

and for sufficiently large  $n$

we have that

$$2f\left(\frac{n}{2}\right) \leq K n^4$$

$$2 \cdot \frac{n^4}{2^4} \leq K n^4$$

$$\frac{n^4}{2^4} \leq K n^4, \text{ choosing } K = \frac{1}{2^4}$$

So it follows from the third case of the master theorem:

$$T(n) = \Theta(n^4)$$

ear Images

Show

$$b T(n) = \lceil \frac{7n}{10} \rceil + n, \quad n > 2$$

We have  $a=1$ ,  $b=\frac{7}{10}$  and  $f(n)=n$ .

Since

$$n = \lceil n^{1+\epsilon} \rceil, \text{ where } \alpha+\epsilon>0, \epsilon=1$$

$$n = \lceil n^{\alpha} \rceil$$

and for sufficiently large  $n$

we have  $\lceil \cdot \rceil$  hat.

$$f\left(\frac{7}{10}n\right) \leq K f(n)$$

$$\frac{7}{10}n \leq Kn, \text{ we choose } K=\frac{7}{10}$$

So

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

Clear Images

seconds

Clear Images

$T(n) = T(n-1) + \frac{1}{n}$

$T(n) - \frac{1}{n} \rightarrow c - \frac{1}{n}$

$T(n-1) - \frac{1}{n-1} \rightarrow c - \frac{1}{n-1}$

$T(n-2) - \frac{1}{n-2} \rightarrow c - \frac{1}{n-2}$

$\vdots$

$T(1) \rightarrow \Theta(1)$

$$T(n) = \sum_{i=0}^{n-1} c \frac{1}{n-i} + \Theta(1)$$

$$= \frac{c}{n} \sum_{i=0}^{n-1} i^{-1} + \Theta(1)$$

$$= \frac{c}{n} \sum_{i=0}^{n-1} i^{-1} + \Theta(1)$$

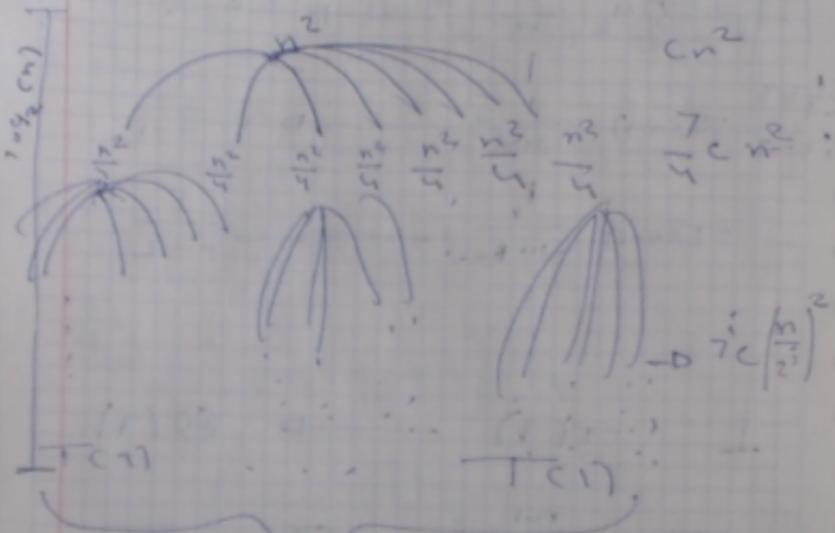
$$= \frac{1}{2} (-n^2 n + 2) + \Theta(n)$$

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

Clear Images

Clear Image

$$d. T(n) = 7T\left(\frac{n}{2}\right) + n^2$$



$$C7^{\log_2(n)} = Cn^{\log_2(7)} = \Theta(n^{\log_2(7)})$$

$$\text{Thus } T(n) = \sum_{i=0}^{\log_2(n)-1} \left(\frac{7}{4}\right)^i cn^2 + \Theta(n^{\log_2(7)})$$

$$cn^2 \left( \frac{4}{3} \left[ \left(\frac{7}{4}\right)^{\log_2(n)} - 1 \right] \right) + \Theta(n^{\log_2(7)})$$

$$< \sum_{i=0}^{\infty} \left(\frac{7}{4}\right)^i cn^2 + \Theta(n^{\log_2(7)}) \rightarrow \infty$$

Clear Image

Lemma 4.3

$$g(n) = \sum_{i=0}^{\log_2(n)-1} \binom{2^i}{n} c n^2$$

If  $n^2 = O(n^{t_0(n)-\epsilon})$ ,  $\epsilon \approx 0.8$

$$n^2 = O(n^{2-\epsilon})$$

But

$$f\left(\frac{n}{2^i}\right) = O\left(\left(\frac{n}{2^i}\right)^{1+\log_2(7)-1}\right)$$
$$g(n) = O\left(\sum_{i=0}^{\log_2(n)-1} a_i \left(\frac{n}{2^i}\right)^{1+\log_2(7)-\epsilon}\right)$$

Geometric sum

$$g(n) = O\left(n^{1+\log_2(7)-\epsilon} \left[\frac{n^\epsilon - 1}{b^\epsilon - 1}\right]\right)$$

Since  $b$  and  $\epsilon$  are constants

$$n^{1+\log_2(7)-\epsilon} O(n^\epsilon) = O(n^{1+\log_2(7)})$$

$$g(n) = O(n^{1+\log_2(7)})$$

Lemma 4.4

$$\begin{aligned} T(n) &= \textcircled{H}(n^{1+\log_2(7)}) + \textcircled{H}(n^{1+\log_2(7)}) \\ &= \textcircled{H}(n^{1+\log_2(7)}) \end{aligned}$$

seconds

L6



Clear Image

Sea  $X_{ij}$  una variable aleatoria  
tal que

$$X_{ij} = \mathbb{I}\{\sum_{k=1}^n A_k > nC_j\} \text{ para } 1 \leq j \leq n$$

Sabemos que  $\sum_{k=1}^n K$

$$P(X_{ij}) = \frac{1}{(n-1)(n)} = \frac{1}{2}$$

Por Lema 2.1

$$\mathbb{E}[X_{ij}] = \frac{1}{2}$$

Sea  $X$  la variable aleatoria que denota  
el número de éxitos en  $n$  tiradas

$$X = \sum_{i=1}^{n-1} \sum_{j=1}^n X_{ij}$$

Queremos el valor esperado de esta variable:

$$\mathbb{E}[X] = \mathbb{E}\left[\sum_{i=1}^{n-1} \sum_{j=1}^n X_{ij}\right]$$

$$= \sum_{i=1}^{n-1} \sum_{j=1}^n \mathbb{E}[X_{ij}]$$

$$= \sum_{i=1}^{n-1} \sum_{j=1}^n \frac{1}{2}$$

$$= \binom{n}{2} \frac{1}{2}$$

$$= \frac{n(n-1)}{2}$$

JPG Change

Clear Images

Show

17

Análisis amortizado

Sea  $C_i$  el costo de la operación  $i$ .

$$C_i = \begin{cases} i & \text{si } \lg(i) \text{ es entero} \\ 1 & \text{de otra manera} \end{cases}$$

Entonces

Figura.

$$\sum_{i=1}^n C_i \leq \sum_{i=1}^n 2^i + \sum 1$$

$$\leq \sum_{i=1}^n 2^i + n$$

$$\leq \sum_{i=0}^{3n} 2^i + n$$

$$= n + (2^n - 1)$$

$$= 3n - 1$$

$$\leq 3n < O(n)$$

$$\Rightarrow \frac{O(n)}{n} = O(1), \text{ el costo amortizado por operación}$$

### 1.7 Método de contralíneas

Sea  $C_i$  el costo por la operación  $i$ .

$$C'_i = \begin{cases} 1 & \text{si } i \text{ pertenece a la tira} \\ 0 & \text{de otra manera} \end{cases}$$

$i$	$C'_i$	$C_i$	$b_i$
1	3	7	2
2	3	2	3
3	3	7	5
4	3	4	4
5	3	1	6
...	...	...	...

Se propone  $C'_i = 3$  porque es el  $C'_i$  menor tal que  $b_i + C'_i - C_i > 0$ ,  $b_i = 0$

Así, el costo anotado es  $C'_i = 3$   
obteniendo que  $\sum_{i=1}^n C'_i = 3n$

Así

$$\hat{C}'_i > C_i$$

$$\sum_{i=1}^n \hat{C}'_i \geq \sum_{i=1}^n C_i, \text{ porque } b_i > 0$$

Por el costo anotado  $\rightarrow O(n)$ ,  
entonces el costo total de  $n$  operaciones  
 $\rightarrow O(n)$ .

Snap it!!

Clear Images

9

$\langle 2, 3, 8, 6, 1 \rangle$

Intervales

(1, 5)

(2, 5)

(3, 5)

(4, 5)

(3, 4)

- b) Partiendo obtener que los números más grandes son cercanos al inicio o a suavemente, así el arreglo que tiene más proximidad es:

$\langle n, n-1, \dots, 2, 1 \rangle$

Sea  $X_{i,j}$  una variable aleatoria

dada por  $X_{i,j} = \begin{cases} 1 & \text{si } A_i > A_{i+1} \\ 0 & \text{en caso contrario} \end{cases}$  para  $i, j \in \{1, 2, \dots, n\}$

Sabemos que  $P(X_{i,j}) = 1$  ( $A_i > A_{i+1}$  siempre).

Aplicamos el teorema mencionado en I.6

$$E[X] = \sum_{i=1}^n \sum_{j=i+1}^n 1$$

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

$$= \frac{1}{2}(n-1)n \quad (\text{total de comparaciones})$$

C. Encuentra un algoritmo y ordena en concepto por inserción sort, es una  
algoritmo muy parecido al de t.  
en que el mayor elemento no ordena.  
Así,  $i < j$  y  $A[i:j] > [j]$  en la  
misma situación.

También sabemos que en el  
peor caso Insertion-sort: se

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

que es el mismo que la cantidad  
de inserciones se dan

$$\frac{n(n-1)}{4} = \Theta(n^2)$$

Otra manera de pensarlo es que  
el tiempo de ejecución de Insertion sort  
es una constante por el tiempo de  
ejecución las veces el número de  
inserciones.