

AyED

ANÁLISIS DE ALGORITMOS

Regla: Se ignoran las constantes.

$$3^* O(4) \Rightarrow O(1)$$

$$O(m/2) \Rightarrow O(m)$$

$$O(5m) \Rightarrow O(m)$$

$$T(n) = n + n^2 = O(n^2)$$

$$T(n) = n \lg n + n^2 = O(n^2)$$

Regla del máximo: Se toma la complejidad más grande.

Ejemplo: $T(n) = n + n^2 = O(n^2)$

Regla ignora términos de menor orden

$$O(3m^3 + 2m^2 + m) = O(m^3) \rightarrow \text{Se elimina la constante, y nos quedamos con la función más compleja}$$

Complejidad constante

$$x = x+1; \quad \left\{ \begin{array}{l} \text{no depende} \\ \text{de un tamaño de entrada} \end{array} \right\}$$

$$y = 200 * 3$$

$$O(1) + O(1) + O(1) = O(1)$$

Print(y)

Complejidad lineal

FOR (INT i=0; i < m; i++) $O(m)$

SOUT n $O(1)$

$$\left. \begin{array}{l} T(n) = O(m) + O(1) \\ = O(m) \end{array} \right\}$$

Complejidad cuadrática.

FOR (INT i=0; i < m; i++) {
 FOR (INT j=0; j < k; j++) $\rightarrow O(m)$
 SOUT (j, i) $\rightarrow O(m)$

$$} \quad T(n) = O(m) * O(m) = O(m^2)$$

húsares

Ejercicio 1)

$$\begin{aligned}
 & \text{1) } \left\{ \text{FOR(INT } h=0; h \leq z; h++ \right\} \boxed{O(m)} \\
 & \quad \text{SOUT}(h) \\
 & \text{2) } \left\{ \text{FOR(INT } i=0; i \leq n; i++ \right\} \boxed{O(m)} \\
 & \quad \left\{ \text{FOR(INT } j=0; j \leq i; j++ \right\} \boxed{O(m)} = O(m) * O(m) = \\
 & \quad \quad \text{SOUT}(j) \\
 & \quad \} \\
 & \text{3) } \left\{ \begin{aligned} T(m) &= O(m) + O(m^2) = \\ T(m) &= O(m^2) \end{aligned} \right.
 \end{aligned}$$

② multiplicamos porque hay un FOR dentro de otro

③ simplificamos el caso más complejo, en este caso es $O(m^2)$

Ejercicio 2)

$$\begin{aligned}
 & \text{IF(} x == \text{TRUE}) \\
 & \quad \# O(m) \\
 & \text{else} \\
 & \quad \# O(m \log m)
 \end{aligned}$$

nos quedamos con el peor caso.

$$\begin{aligned}
 T(m) &= O(m) + O(m \log m) = \\
 & \boxed{O(m \log m)}
 \end{aligned}$$

Ejercicio 3)

$$\begin{aligned}
 & \text{INT } x=0; \boxed{\text{CTE}} \\
 & \text{INT } i=1; \boxed{\text{CTE}} \\
 & \text{while(} i \leq m) \left\{ \begin{aligned} & \text{CTE}/2 * (m+1) \\
 & x = m+1 \\
 & i = i+2 \end{aligned} \right. \\
 & \quad \} \quad T(m) = O(m)
 \end{aligned}$$

Ejercicio 4)

$$\begin{aligned}
 & \text{INT } x=1; \boxed{\text{CTE}} \\
 & \text{while(} x \leq m) \left\{ \begin{aligned} & \text{CTE}_2 * \log m \\
 & x = 2 * x \end{aligned} \right. \\
 & \quad \} \quad T(m) = \text{CTE}_1 + \text{CTE}_2 * \log m \\
 & T(m) = O(\log m)
 \end{aligned}$$

Aclaración:

- Si m es potencia de 2: Realiza $\log(m)$ operaciones

- Si m no es potencia de 2: Realiza $\log(m) + 1$ iteraciones

Cual es la expresion correcta del siguiente organo.

Escravos.

$\text{For } \{1 \neq \emptyset; i < m; i++\} \rightarrow O(m)$

for ($i = 1$; $i \leq m$; $i++ = m/2$) $S(0)(m/2)$

$$x = x_{1,1} \quad T(n) = O(n) * O(m/2)$$

$$\text{a) } O(\sqrt{n})$$

b) $p(m)$

correct!

c) $O(m \log m)$ d) $O(m^2)$ e) $O(m^3)$

E-jenius occurs

a) Esonto $O(J_m)$

For (int i=0; i < m; i = (int) Math.sqrt(m))

{ } (.)

3

c) Ejeremto Olim logm)

ARRAYS. SORT (ARRAY); { SUPONIENDO QUE ARRAY
ES UN ARREGLO DE TAMAÑO n

d) Ejemplo $O(n^2)$

$$\text{For } \left(\text{int } i=0; i < n; i++ \right) \{ \quad \text{--- } \bullet \text{ (m)} \}$$

$$\sum_{k=0}^{\infty} \left(\frac{1}{2} \left(\frac{1}{2} + k \right) \Gamma \left(\frac{1}{2} + k \right) \right)^{-1} = \sqrt{\pi}$$

saw (v.)

$$T(m) = O(m)^* O(m) =$$

$$T(n) = O(n^2)$$

e) Ejem $O(n^3)$

$$\text{Factor}(\text{int } i=0) \ [cm, i+t)] \rightarrow O(m)$$

FOR(INT j=0; j<n; j++){ } $\rightarrow O(n)$

For(**int** k=0; k<n; k++) → O(n)
Solve it again

3000' SOUTHWEST

$$T(n) = O(n) * O(n) * O(n)$$

$$T(n) = O(n^3)$$

Ejercicio 6

```
int count = 0;  
int n = 1, length;  
  
for (int i=0; i<n; i++)  
    for (int j=0; j<n; j++)  
        a[i][j]++;  
  
y y a[2][2];
```

Sabemos que tarda 1 segundo cuando $n = 2500$

¿Cuanto tardaría aproximadamente para $n = 35000$. Justifique su respuesta.

Se podría plantear de dos formas:

① Sabemos que los dos for son bucles en líneas, y además iteran lo mismo.

Podemos clacularlo como si fuese un solo for , y después multiplicarlo por el mismo resultado. Porque en algún momento tenemos que llegar a $m^2 m$

Tendremos que: PARA $n = 2500$ Tarda 1 segundo

y PARA $n = 35000$ VA A TARDAR: $3500 \rightarrow 1 \text{ seg}$

$$((35000 * 1) / 3500) = 10 \quad 35000$$

10 segundos tarda UN FOR, pero tenemos DOS FOR ANDANDOS, por lo tanto tendriamos que hacer $m^2 m =$
Hacemos $10 * 10 = \boxed{100}$ [segundos] a la cantidad de segundos del cuadro

② Usamos la regla de 3, pero teniendo en cuenta que tenemos n^2 bucles. Lo que nos quedaria así:

$$\text{TIEMPO TOTAL: } 1 \text{ segundo} * ((35000 * 35000) / (3500 * 3500))$$

③ determinamos nuestro resultado en m^2 como tenemos m^2 el cuadro el resultado para tener el resultado correcto

$$3500 \rightarrow 1 \text{ seg} \quad m = ((35000 * 1) / (3500)) = 10$$

$$35000 \quad m = 10 = m^2 = 10^2 = \boxed{100} \text{ segundos}$$

Worst =

private void impares y pares (int m)

int x=0, int y=0;

FOR (int i=1; i<=m; i++)

IF (esPar(i))

FOR (int j=1; j<=m; j++) m

x++;

else

FOR (int j=1; j<=i; j++) m/2

y++

Public static EsPar (int n) {

if (numero. z == 0) return true } else,

use return false

$$T(m) = \sum_{i=1}^m c_1 e_1 + \sum_{i=1}^{m/2} \left(\sum_{j=1}^m c_2 e_2 + \sum_{j=1}^{2x_i} e_2 e_2 \right)$$

$$T(m) = c_1 e_1 * m + \sum_{i=1}^{m/2} c_2 e_2 * (m - 2 * i + 1 + 2 * (i - 1 + 1)) =$$

$$= c_1 e_1 * m + c_2 e_2 * (m + 2) * m/2$$

$$= c_1 e_1 + c_2 e_2 / 2 * m^2 + c_2 e_2 * m$$

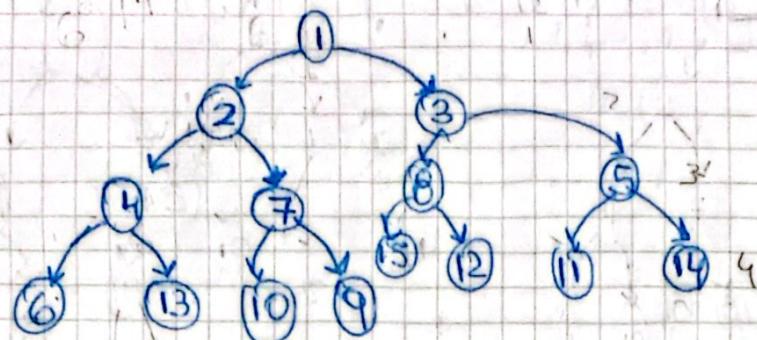
$$T(m) = O(m^2)$$

Ayer

① EXERCICIO DE TEORÍA DE HEAP

CONSTRUIR UNA HEAP A PARTIR DE LOS SIGUIENTES VALORES

6, 4, 15, 2, 10, 11, 8, 1, 13, 7, 9, 12, 5, 3, 14



②

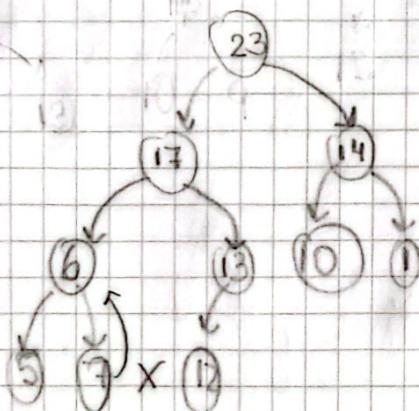
$$\text{h} = 1 \quad 2^0 = 2^{h-1}$$

$$\text{h} = 2 \quad 2^1 = 2^{h-1}$$

$$\text{h} = 3 \quad 2^2 = 2^{h-1} \text{ como máximo}$$

a) cuantos elementos hay al menos en una heap de altura h.

c) a siguiente arreglo es una maxHeap? [23, 17, 14, 6, 13, 10, 1, 5, 7, 12]?



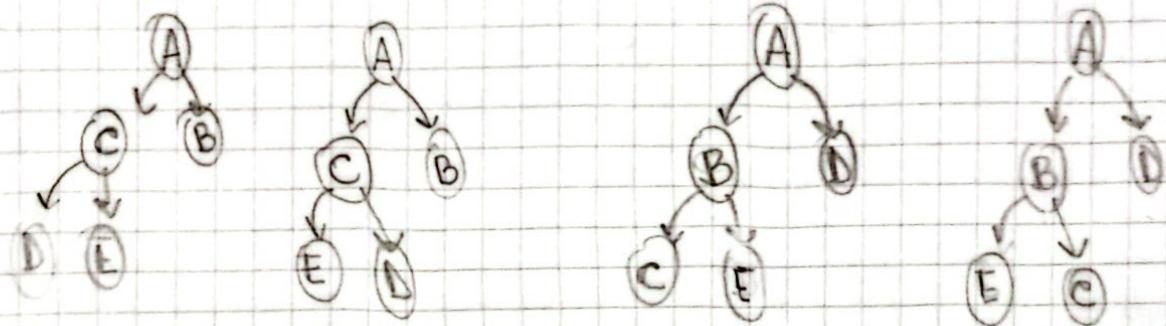
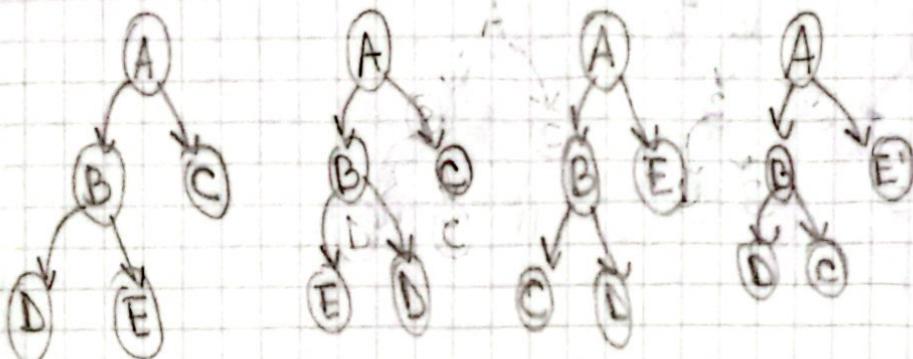
NO es MAX HEAP porque

6 es el padre de 7
DEBERIA SER AL REVERSO

Ejercicio 4:

Dibuja todos los minitrees posibles para este conjunto de datos

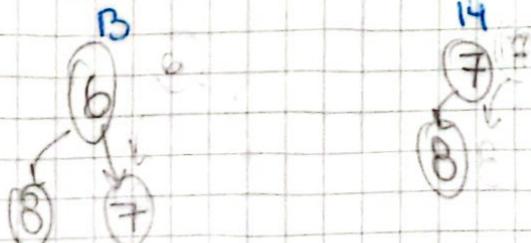
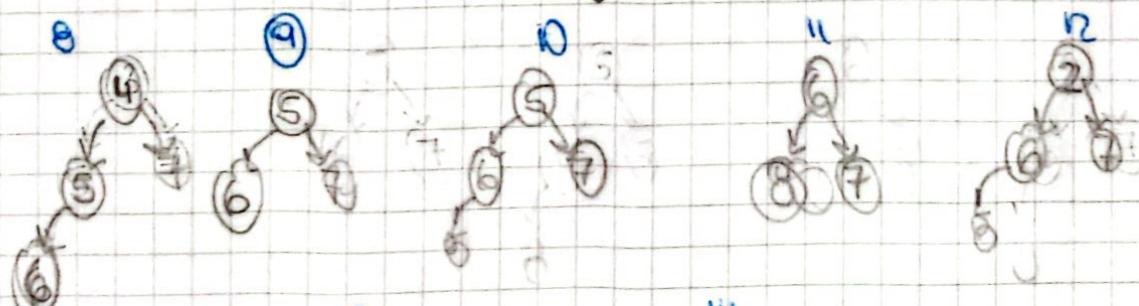
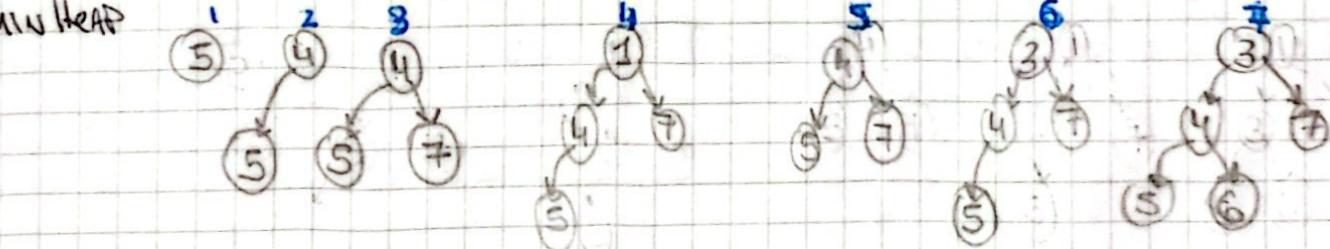
[A, B, C, D, E]



Ejercicio 5

insert(5), insert(4), insert(7), insert(1), deleteMin(), insert(3), insert(6)
deleteMin(), deleteMin(), insert(8), deleteMin(), insert(2), deleteMin(), deleteMin()

MIN HEAP



húsar

CONSULTAR ESTE Ayud

Aníque el algoritmo **HEAP-SELECT** PARA ORDENAR
desequilibradamente los elementos

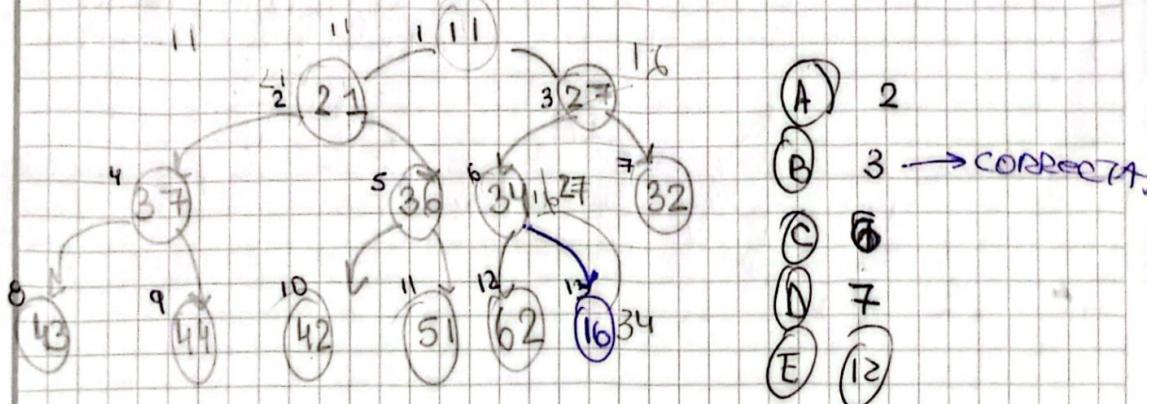
[15, 16, 40, 11, 7, 10, 23, 12, 140, 500, 11, 12, 13, 90]

Desequilibradamente → HABO MAX HEAP, y VOY CAMBIANDO EL
PRIMERO CON EL ÚLTIMO
DECREMENTO UN ELEMENTO EN HEAP?
FILTRO?

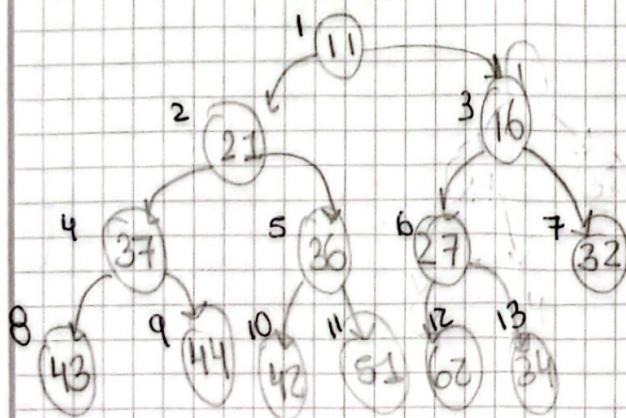
CÓMO AGREGO LA MAXHEAP? \rightarrow INSERCIÓN DE A UNO?
 \hookrightarrow BUILDHEAP

Ejercicio 9) Supongamos que una heap representa una caja de frascos de agua almacenada en el Almacén A. Si insercamos la clave 16 en que posición quedaría?

A [11 2 3 4 5 6 7 8 9 10 11 12]



FILTRANDO:



núsculos

① Problema: considerando que un ALGORITMO requiere $f(n)$ operaciones para resolver un problema y la COMPUTADORA procesa 100 operaciones por segundo.

Determinar el tiempo de ejecución en segundos requerido por el ALGORITMO para resolver un problema de tamaño = 10.000

a) $\log_{10} n = \log_{10}(10.000) \rightarrow 4$

b) $\sqrt{n} = \sqrt{10.000} = 100$

c) 100 OP 1 seg = $\frac{((4+1)/100)}{100} = 0,045$

b)

② Supón que un tiene un ALGORITMO ALGO-1 con UN TIEMPO DE EJECUCIÓN DE $10 \cdot n^2$, cuánto se hace más lento A_{60-1} cuando el tamaño de la ENTRADA n aumenta

a) EL DOBLE

b) EL TRIPLE

ANALISIS DE ALGORITMOS

AyED.

FOR (INT $i=1$; $i \leq m$; $i++$) /

 circle. writeLine(i)

} CTC

$i = 1$
 $2 = 2$
 $3 = 3$
 $4 = 4$

m VECES

$$T(m) = \sum_{i=1}^m \text{CTC} = \text{CTC} \sum_{i=1}^m$$

$$T(m) = m$$

FOR (INT $i=0$; $i \leq m$; $i++$)
 {
 circle. writeLine(i)
 }

① = 0
 ② = 1
 ③ = 2
 ④ = 3

Podemos llevar a sumatoria directa, si incrementamos o decrementamos de a uno

Si la suma de a 2,
 o multiplicando, dividiendo
 no la llevamos en forma directa.

Solo hace se multiplican o dividirán

$$T(m) = \text{CTC} \sum_{i=0}^m$$

$k = k+1$, esto se ejecuta $m-i$ veces

$$k = k - 1 = m - i$$

$$k = m - i + 1$$

$$k = m$$

$$T(m) = \text{CTC} + (m+1)$$

$$T(m) = m$$

$$\sum_{i=a}^b c = (b-a+1) \cdot c \rightarrow \sum_{i=5}^{m-3} c = ((m-3)-5+1) \cdot c = (m-7) \cdot c$$

FOR (INT $i=1$; $i \leq m$; $i++$)

{

 FOR (INT $j=1$; $j \leq m$; $j++$)

{

 circle. writeLine(i);

}

$T_2(m)$

$i = 1 \quad j = 1$

$i = 2 \quad j = 2$

$i = 3 \quad j = 3$

$k = k \quad K = K$

$\downarrow \quad \downarrow$

m VECES m VECES

Ejemplos de Algoritmo linea Atuesta.

$$T_2(m) = \sum_{i=1}^m \text{CTC}$$

$$T_2(m) = m \cdot \text{CTC}$$

$$T_1(m) = \sum_{i=1}^m T_2(m)$$

$$T_1(m) = \sum_{i=1}^m m \cdot \text{CTC}$$

$$T_1(m) = m \cdot (m \cdot \text{CTC})$$

$$T_1(m) = m \cdot m = m^2$$

$$T(m) = m^2$$

FOR (i=1; i<=m; i++)

{
P(1, 2, 3, 4, 5, 6)

FOR(INT i=1; i<=n; i++)

{ castle - warlike (i) $T_{3(m)}$

۷

$T_1(m)$

六 丁

1 1

2 1, 2

3 1,2,3

4 1,2,3,4

$$K = m \quad K = i$$

$$T_2(m) = \sum_{j=1}^i c_j e$$

$$T_i(m) = i \cdot CTC$$

$$T_1(m) = \sum_{i=1}^m T_2(n_i)$$

USAMOS LA PROPIEDAD:

$$T_1(m) = \sum_{i=1}^m i \cdot \text{cte} \left\{ \begin{array}{l} \text{tenemos que sacarlos} \\ \text{de } i, \text{ para que pue} \\ \text{se todo en turnos } \leq m \end{array} \right\} \sum_{j=1}^m i = \frac{m \cdot (m+1)}{2}$$

SOURCES IN CONFLUENCE

$$T_m = \text{cte} \sum_{i=1}^m i$$

RECAPITULAMOS LA SUMATORIA POR LA PROPIEDAD

$$T(m) = CTC + \frac{m(m+1)}{2}$$

$$T_1(m) = C \epsilon + \frac{m^2}{2} + m$$

$$T_1(m) = m^2$$

$$T(m) = m^2$$

Jerarquías while ; se calcula de misma manera

```
int i = 1;
while (i < m)
{ cout << writeLine(i);
  i++;
}
```

- ① $i = 1$
- ② $i = 2$

```
for(int i=1; i<m; i++)
{ cout << writeLine(i);
}
```

}

- ① $i = 1$
- ② $i = 2$

Para ambos casos:

$$T(m) = \sum_{i=1}^m \text{cte}$$

$$\rightarrow T(m) = m \cdot \text{cte}$$

$$T(m) = m$$

```
int i = 1;
while (i < m)
{ cout << writeLine(i);
  i = i * 2;
}
```

```
for(int i=1; i<m; i*=2)
{ cout << writeLine(i);
}
```

Ambos casos son iguales
se resuelven de la misma manera.

Iteración	Indice
①	$i = 1$
②	$i = 2$
③	$i = 4$
④	$i = 8$
⑤	$i = 16$
(k)	$i = 2^{k-1}$

Se ejecuta m veces

$$i = 2^k - m$$

$$2^{k-1} = m$$

Aplicamos logaritmo para lograr el exponente

$$k-1 = \log_2(m)$$

$$k = \log_2(m) + 1$$

$$T(m) = \sum_{k=1}^{\log_2(m)} \text{cte}$$

$$T(m) = \log_2(m)$$

① Hacemos tabla de iteraciones e índices, para sacar una $E(k)$

② Igualamos $i = 0$ m veces ejecuta \rightarrow iteraciones

③ Resolvemos

OTRO EJEMPLO DE ANÁLISIS DE ALGORITMOS

ITERACION INICP
 INT i, j, k;
 INT c3 A, B, C;

① $\begin{cases} i=1 \\ j=2 \\ k=1 \end{cases}$

$\begin{cases} i=1 \\ j=2 \\ k=1 \end{cases}$

$\begin{cases} i=1 \\ j=2 \\ k=2 \end{cases}$

$\begin{cases} i=1 \\ j=2 \\ k=3 \end{cases}$

② $\begin{cases} i=2 \\ j=3 \\ k=1 \end{cases}$

$\begin{cases} i=2 \\ j=3 \\ k=2 \end{cases}$

$\begin{cases} i=2 \\ j=3 \\ k=3 \end{cases}$

INT i, j, k;

FOR (i = 1; i <= m - 1; i++)

{ FOR (j = i + 1; j <= m; j++)

{ FOR (k = j - 1; k >= i; k--)

{ cout << i << endl;

CRE

T₂

T₃

$$T_3(n) = \sum_{k=1}^j \text{cte} = T_3(m) = J \cdot \text{cte}$$

$$T_2(n) = \sum_{j=i+1}^m T_3(n) = T_2(m) = \sum_{j=i+1}^m J \cdot \text{cte}$$

CUANDO UNA SUMATORIA NO ENCUENTRA EU 1

$$\text{Prop 1: } \sum_{i=1}^n \sum_{j=1}^k \sum_{l=i+1}^{k+1} \dots$$

CUANDO LA SUMATORIA NO ENCUENTRA EU 0 O 1

$$\text{Prop 2: } \sum_{i=1}^m i = \frac{m(m+1)}{2}$$

ANALIZAMOS PROBLEMAS:

- NO CONOCENOS EL VALOR INICIAL DE J (NECESITAMOS QUE VOS PUEDES J = 1) → PROPIEDAD 2
- TIENENOS MUCHAS ALIAS EN LA SUMATORIA POR J → PROPIEDAD 2.

$$T_2(n) = \sum_{j=i+1}^m j = \sum_{j=1}^m j - \sum_{j=1}^i j =$$

3)

$SUM = 0$

```

FOR( INT i=1 ; i<=m ; i++ ) -
    { FOR( INT j=0 ; j<i ; j++ )
        { cout << arr[i][j] }
    }
}

```

Iteration

1

2

3

(n)

Value

i=1

j=0

i=2

j=1

i=3

j=2

i=n

j=m-1

1st FOR: m times

2nd FOR: i-1 times

$$T_2(m) = \sum_{j=1}^i CTE$$

$$T_1(m) = \sum_{i=1}^m \sum_{j=1}^i CTE$$

$$T(m) = CTE + \sum_{i=1}^m \sum_{j=1}^i = CTE \sum_{i=1}^m$$

A NUEVA PARTE, NO IMPORTA CL
+1 o -1

ANÁLISIS DE ALGORITMOS

- ② MIRO EL PISO → como cambia la variable i
③ MIRO CUANTAS VECES entra al FOR

$T(n)$ = CANTIDAD de OPERACIONES

FOR(INT i=0; i*i < m; i++)
{
 SOUT ("Ayco")
}

↓

$i^2 = m-1$ HASTA DONDE ENTRA

DESPEJAMOS

$i^2 \geq m$ SALGO

$$\rightarrow i > \sqrt{m}$$

$$i = \sqrt{m}$$

$O(\sqrt{m})$

$$T(n) = \sqrt{m} \cdot \text{CTE}_1 + \text{CTE}_2$$

$$\sum_{i=0}^{\sqrt{m}} \text{CTE} = (m+1) \cdot \text{CTE}$$

$$\boxed{\text{LÍMITE SUPERIOR} - \text{LÍMITE INFERIOR} + 1 \\ m - 0 + 1 = m + 1}$$

CUANDO EXPRESAMOS EN SUMATORIA Y CUANDO NO?

→ SUMATORIA: CUANDO HAY DOS FOR, Y UNA VARIABLE DEPENDE DE LA OTRA

FOR(INT i=1; i<m; i = i * x)
{
 SOUT ("Ayco")
}

↓

- (1) $i=1$
(2) $i=x$
(3) $i=x^2$
(4) $i=x^3$
(5) $i=x^4$

(6)

$i = x^{(k-1)}$

↓ PROFE ELIGIÓ ESTE

- (6) $i=1$
(1) $i=x$
(2) $i=x^2$
(3) $i=x^3$
(4) $i=x^4$

(k)

$i=x^K$

$$x^K = m$$

APLICAMOS LOGARITMO

$$\log_x(x^K) = \log_x(m)$$

SIMPLIFICAMOS

$$K = \log_x(m)$$

hússares

CUANDO $x^K > m$ CREA EL FOR
DESPEJAMOS

$$x^K = m$$

$$\log_x(x^K) = \log_x(m)$$

$$K = \log_x(m)$$

FOR(
 INT $i = 3$; $i < m$; $i = i * \cancel{x}$)
 {
 $\cancel{\text{seor("yeso")}}$
 }

$$i = 1$$

$$i = x$$

$$i = x^2$$

$$i = x^3$$

$$i = x^4$$

$$i = x^5$$

$$k = x^k$$

CUANDO $x^k \geq m$ SALGO DEL WHILE

despejamos: $\log_x(m)$ PARA BAJAR EL EXPONENTE
en AMBOS LADOS

$$\log_x(x^k) \geq \log_x(m)$$

SIMPLIFICAMOS \log_x CON x

$$k \geq \log_x(m)$$

No es necesario hacerlo

$O(\log_x(m))$

$$x = 2 \quad m = 1024$$

$$\begin{array}{ll}
 i=1 & i=128 \\
 i=2 & i=256 \\
 i=4 & \\
 i=8 & i=512 \\
 i=16 & i=1024 \\
 i=32 & \\
 i=64 &
 \end{array}$$

$$\begin{aligned}
 \log_2(1024) &= \\
 2^{10} &= 1024
 \end{aligned}$$

CUANTOS RECORRIDOS POR ANIDADOS, SE MULTIPLICA EL COSTO
SI NO ESTAN EN ANIDADOS SE SUMAN.

$O(m)$

FOR (INT $i=0$; $i \leq m/2$; $i++$)

$O(m)$

$$T(m) = \frac{m}{2}$$

FOR (INT $y=1$; $y+m/2 \leq m$; $y++$)

FOR (INT $K=1$; $K \leq m$; $K = K * 2$) $\rightarrow O(\log_2(m))$

{sout("Ayes")
}

$O(m \cdot \log_2(m))$

$$T(m) = \frac{m}{2} \cdot \left(\frac{m}{2} + 1 \right) \cdot (\log_2(m) + 1) \cdot \text{cte}$$

+1 o -1 NO MODIFICA NADA.

IMPACTA AL COSTE

NO SE FIJAN SI AUMENTA

+1 o -1

EXPRESADO

$$\text{EN SUMATORIA} = \sum_{i=0}^{\frac{m}{2}-1} \left[\sum_{j=1}^{\frac{m}{2}+1} \left[\sum_{K=1}^{\log_2(m)+1} C_K \right] \right]$$

INT $i=1$
WHILE ($i \leq m$) $\rightarrow O(\log_2(m))$

{ INT $y=1$;
WHILE ($y \geq 0$) $\rightarrow O(\log_2(m))$

$$\downarrow y = j/2$$

}

$$O((\log_2(m))^2)$$

$$i = i * 2$$

}

FOR (INT $i=0$; $i \leq m/3$; $i++$)

$\rightarrow m$

{ FOR (INT $y=1$; $y \leq m$; $y = y + 4$)

{ sout("Ayes")
}

}

SIEMPRE ELEGIMOS
EL PUEDE
CONVIVIR ANTES DE

SI ENTREZCAN DESDE 1

- ① $y=1$
- ② $y=5$
- ③ $y=9$
- ④ $y=13$

- ① $y=1$
- ② $y=5$
- ③ $y=9$
- ④ $y=13$

$$K = 4K + 1 > m$$

$$K = \frac{m}{4} + 3$$

$$K = \frac{m}{4} + 1$$