Juan Menduiña

Ejercitación Teórica N° 3:
Análisis de Algoritmos.

Ejercicio 1.

Ordenar las siguientes funciones: $\sqrt{n}$, *n*, 3^n, n^2, *cte*, 2^n, log_2^2 *n*, log_3 *n*, log_2 *n*, *según su velocidad de crecimiento.*

cte $< log_3$ n $< log_2$ n $< log_2^2$ n $< \sqrt{n} <$ n $< n^2 < 2^n < 3^n$.

Ejercicio 2.

Expresar de qué orden es el siguiente fragmento de código:

```
for (int j = 4; j < n; j=j+2) {
      val = 0;
      for (int i = 0; i < j; ++i) {
            val = val + i * j;
            for (int k = 0; k < n; ++k){
                  val++;
            }
      }
}
```

(a) *O(n log n).* **(b)** *O*(n^2). **(c)** *O*(n^2 *log n*). **(d)** *O*(n^3).

Iteraciones del primer *for*:

j= 4.
j= 4 + 2.
j= 4 + 2 * 2.
…
j= 4 + 2 (k - 1).

4 + 2 (k - 1)= n - 1
2 (k - 1)= n - 1 - 4
2 (k - 1)= n - 5
k - 1= $\frac{n-5}{2}$
k= $\frac{n-5}{2}$ + 1
k= $\frac{n-3}{2}$.

Iteraciones del segundo *for*:

i= 0.
i= 1.
i= 2.
…
i= k - 1.

k - 1= j - 1
k= j - 1 + 1
k= j.

Iteraciones del tercer *for*:

k= 0.

k= 1.
k= 2.
…
k= k´ - 1.

k - 1= n - 1
k= n - 1 + 1
k= n.

Entonces:

T(n)= $\sum_{j=1}^{\frac{n-3}{2}} cte_1\ (\sum_{i=1}^{j} cte_2 + \sum_{k=1}^{n} cte_3)$
T(n)= $\sum_{j=1}^{\frac{n-3}{2}} cte_1\ (\sum_{i=1}^{j} cte_2 + n * cte_3)$
T(n)= $\sum_{j=1}^{\frac{n-3}{2}} cte_1 * j * (cte_2 + n * cte_3)$
T(n)= $cte_1\ (cte_2$ + n $cte_3)\ \sum_{j=1}^{\frac{n-3}{2}} j$
T(n)= $cte_1\ (cte_2$ + n $cte_3)\ \frac{\frac{n-3}{2}(\frac{n-3}{2}+1)}{2}$
T(n)= $cte_1\ (cte_2$ + n $cte_3)\ \frac{\frac{n-3}{2}\frac{n-1}{2}}{2}$
T(n)= $cte_1\ (cte_2$ + n $cte_3)\ \frac{\frac{n^2-n-3n+3}{4}}{2}$
T(n)= $cte_1\ (cte_2$ + n $cte_3)\ \frac{\frac{n^2-4n+3}{4}}{2}$
T(n)= $cte_1\ (cte_2$ + n $cte_3)\ \frac{n^2-4n+3}{8}$
T(n)= $cte_1\ (cte_2$ + n $cte_3)$ ($\frac{1}{8} n^2$ - $\frac{1}{2}$ n + $\frac{3}{8}$)
T(n)= ($\frac{1}{8} n^2$ - $\frac{1}{2}$ n + $\frac{3}{8}$) $cte_1\ cte_2$ + ($\frac{1}{8} n^3$ - $\frac{1}{2} n^2$ + $\frac{3}{8}$ n) $cte_1\ cte_3$.

Por lo tanto, el fragmento de código es de orden O(n^3).

Juan Menduiña

Ejercicio 3.

Suponer que se dispone de un algoritmo A, que resuelve un problema de tamaño n, y su función de tiempo de ejecución es T(n)= n log(n). Este algoritmo se ejecuta en una computadora que procesa 10.000 operaciones por segundo. Determinar el tiempo que requerirá el algoritmo para resolver un problema de tamaño n= 1.024.

T(n)= n log_2 n.

T(1024)= 1024 log_2 1024
T(1024)= 1024 * 10
T(1024)= 10240.

10.000 operaciones por segundo.

Tiempo (en segundos)= $\frac{10240}{10000}$
Tiempo (en segundos)= 1,024.

Por lo tanto, el tiempo en segundos que requerirá el algoritmo para resolver un problema de tamaño n= 1.024 es 1,024.

Ejercicio 4.

¿Cuál es el resultado de la siguiente sumatoria? $\sum_{i=3}^{8} ni$.

(a) *(8-3+1) n.* **(b)** *(8-3+1) in.* **(c)** *33n.*
(d) *5n.* **(e)** *8i.* **(f)** *Ninguna de las otras opciones.*

$\sum_{i=3}^{8} ni$= n $\sum_{i=3}^{8} i$
$\sum_{i=3}^{8} ni$= n ($\sum_{i=1}^{8} i$ - $\sum_{i=1}^{2} i$)
$\sum_{i=3}^{8} ni$= n ($\sum_{i=1}^{8} i$ - $\sum_{i=1}^{2} i$)
$\sum_{i=3}^{8} ni$= n ($\frac{8*9}{2}$ - $\frac{2*3}{2}$)
$\sum_{i=3}^{8} ni$= n ($\frac{72}{2}$ - $\frac{6}{2}$)
$\sum_{i=3}^{8} ni$= n (36 - 3)
$\sum_{i=3}^{8} ni$= 33n.

Ejercicio 5.

¿Cuál de las siguientes sentencias es correcta, según la definición vista en clase?

(a) n^2 *es* $O(n^2)$.
(b) n^2 *es* $O(n^3)$.
(c) n^2 *es* $O(n^2 \log n)$.
(d) *Opciones a y b.*
(e) *Opciones a, b y c.*
(f) *Ninguna de las otras opciones.*

Ejercicio 6.

Dado el siguiente algoritmo:

```
void ejercicio5 (int n) {
   if (n ≥2) {
      2 * ejercicio5 (n/2);
      n = n/2;
      ejercicio5 (n/2);
   }
}
```

Indicar el T(n) para n ≥ 2:

(a) $T(n)= d + 3\ T(\frac{n}{2})$.
(b) $T(n)= d + 2\ T(\frac{n}{2}) + T(\frac{n}{4})$.
(c) $T(n)= d + T(\frac{n}{2}) + T(\frac{n}{4})$.
(d) $T(n)= d + T(\frac{n}{2}) + T(\frac{n}{2})$.
(e) $T(n)= d + T(\frac{n}{2}) + T(\frac{n}{2}) + T(\frac{n}{4})$.

Ejercicio 7.

Dada la recurrencia:

$$T(n)= \begin{cases} 1, para\ n \leq 1 \\ T(\frac{n}{3}) + c, para\ n > 1 \end{cases}.$$

(a) *¿Cómo se reemplaza $T(\frac{n}{3})$, considerando $\frac{n}{3} > 1$?*

(i) *$T(\frac{n}{3})$ + c.*
(ii) *Ninguna de las otras opciones.*
(iii) *$T(\frac{n}{3})$ + 1.*
(iv) *$T(\frac{\frac{n}{3}}{3})$ + c.*
(v) *$T(\frac{\frac{n}{3}}{3})$ + 1.*

(b) *Desarrollar la función T(n).*

Paso 1:

T(n)= T($\frac{n}{3}$) + c, si n $>$ 1.

Paso 2:

T(n)= T($\frac{\frac{n}{3}}{3}$) + c + c
T(n)= T($\frac{n}{9}$) + 2c, si n $>$ 2.

Paso 3:

T(n)= T($\frac{\frac{n}{9}}{3}$) + 2c + c
T(n)= T($\frac{n}{27}$) + 3c, si n $>$ 3.

Paso i (Paso general):

T(n)= T($\frac{n}{3^i}$) + ic, si n $>$ i.

$\frac{n}{3^i}$= 1
1 * 3^i= n
3^i= n
i log_3 3= log_3 n
i * 1= log_3 n
i= log_3 n.

Entonces:

T(n)= T($\frac{n}{3^{log_3 \text{ n}}}$) + log_3 n * c
T(n)= T($\frac{n}{n}$) + log_3 n * c
T(n)= T(1) + log_3 n * c
T(n)= 1 + log_3 n * c ≤ O(log_3 n).

Ejercicio 8.

Considerar el siguiente fragmento de código:

```
int count = 0; int n = a.length;
  for (int i = 0; i < n; i+=n/2)   {
              for (int j = 0; j < n; j++)  {
       a[j]++;
              }
  }
```

Este algoritmo se ejecuta en una computadora que procesa 100.000 operaciones por cada segundo. Determinar el tiempo aproximado que requerirá el algoritmo para resolver un problema de tamaño n= 1.000.

(a) *0,01 seg.* **(b)** *0,1 seg.* **(c)** *1 seg.* **(d)** *Ninguna de las opciones anteriores.*

T(n)= cte_1 + $\sum_{i=1}^{2} \sum_{j=1}^{n} cte_2$
T(n)= cte_1 + $\sum_{i=1}^{2} n * cte_2$
T(n)= cte_1 + 2n cte_2.

T(1000)≅ 1000.

100.000 operaciones por segundo.

Tiempo (en segundos)≅ $\frac{1000}{100000}$
Tiempo (en segundos)≅ 0,01.

Por lo tanto, el tiempo aproximado que requerirá el algoritmo para resolver un problema de tamaño n= 1.000 es 0,01 segundos.

Ejercicio 9.

Considerar la siguiente recurrencia:

T(1)= 4.
$T(n)= 2\, T(\frac{n}{2}) + 5n + 1 \; (n \geq 2)$.

¿Cuál es el valor de T(n) para n= 4?

(a) *51.* **(b)** *38.* **(c)** *59.* **(d)** *79.* **(e)** *Ninguna de las opciones anteriores.*

$T(4)= 2\, T(\frac{4}{2}) + 5 * 4 + 1$
T(4)= 2 T(2) + 20 + 1
T(4)= 2 T(2) + 21
$T(4)= 2\, [2\, T(\frac{2}{2}) + 5 * 2] + 21$
T(4)= 2 [2 T(1) + 10 + 1] + 21
T(4)= 2 [2 T(1) + 11] + 21
T(4)= 2 (2 * 4 + 11) + 21
T(4)= 2 (8 + 11) + 21
T(4)= 2 * 19 + 21
T(4)= 38 + 21
T(4)= 59.

Ejercicio 10.

Expresar la función T(n) del siguiente segmento de código:

```
public static void ejercicio (int n) {
    int x = 0;
    int j = 1;
    while ( j <= n ) {
        for ( int i = n*n ; i >=1 ; i = i - 3 )
            x = x + 1 ;
        j = j * 2 ;
    }
}
```

(a) *T(n)*= $\frac{1}{3} n^2 + log_2$ *n.*
(b) *T(n)*= $n^2 + \frac{1}{3} log_2$ *n.*
(c) *T(n)*= $\frac{1}{3} log_2$ *n.*
(d) *T(n)*= $\frac{1}{3} n^2$ $log_2 n + log_2$ *n.*

Iteraciones del *while*:

j= 1.
j= 2.
j= 4.
…
j= 2^{k-1}.

2^{k-1}= n
log_2 2^{k-1}= log_2 n
(k - 1) log_2 2= log_2 n
(k - 1) * 1= log_2 n
k - 1= log_2 n
k= log_2 n + 1.

Iteraciones del *for*:

i= n^2.
i= n^2 - 3.
i= n^2 - 3 * 2.
…
i= n^2 - 3 (k - 1).

n^2 - 3 (k - 1)= 1
3 (k - 1)= n^2 - 1

k - 1= $\frac{n^2-1}{3}$
k= $\frac{n^2-1}{3}$ + 1
k= $\frac{n^2+2}{3}$.

Entonces:

T(n)= $\sum_{j=1}^{log_2 n+1} \sum_{i=1}^{\frac{n^2+2}{3}} cte$
T(n)= $\sum_{j=1}^{log_2 n+1} \frac{n^2+2}{3}$
T(n)= (log_2 n + 1) $\frac{n^2+2}{3}$
T(n)= (log_2 n + 1) ($\frac{1}{3} n^2$ + $\frac{2}{3}$)
T(n)= $\frac{1}{3} n^2$ log_2 n + $\frac{2}{3}$ log_2 n + $\frac{1}{3} n^2$ + $\frac{2}{3}$.

Ejercicio 11.

¿Cuál es el tiempo de ejecución del siguiente método?

```
void fun(int n,  int arr[])
{    int i = 0, j = 0;
    for (; i < n; ++i)
        while (j < n && arr[i] < arr[j])
            j++;
}
```

Iteraciones del *for*:

i= 0.
i= 1.
i= 2.
…
i= k - 1.

k - 1= n - 1
k= n - 1 + 1
k= n.

Iteraciones del *while*:

j= 0.
j= 1.
j= 2.
…
j= k - 1.

k - 1= n - 1
k= n - 1 + 1
k= n.

Entonces:

T(n)= cte_1 + $\sum_{i=1}^{n} \sum_{j=1}^{n} cte_2$
T(n)= cte_1 + $\sum_{i=1}^{n} n * cte_2$
T(n)= cte_1 + nn cte_2
T(n)= cte_1 + n^2 cte_2.

Por lo tanto, el tiempo de ejecución del método es T(n)= cte_1 + n^2 cte_2.

Ejercicio 12.

¿Cuál es el valor que retorna el método fun1?

```
int fun1 (int n) {
   int i, j, k, p, q = 0;
   for (i = 1; i < n; ++i)    {
      p = 0;
      for (j = n; j > 1; j = j/2)
         ++p;
      for (k = 1; k < p; k = k*2)
         ++q;
   }
   return q;
}
```

Iteraciones del primer *for*:

i= 1.
i= 2.
i= 3.
…
i= k.

k= n - 1.

Iteraciones del segundo *for*:

j= n.
j= $\frac{n}{2}$.
j= $\frac{n}{4}$.
…
j= $\frac{n}{2^{k-1}}$.

$\frac{n}{2^{k-1}}$= 1 + 1
$\frac{n}{2^{k-1}}$= 2
$2 * 2^{k-1}$= n
2^k= n
$log_2\ 2^k$= log_2 n
k log_2 2= log_2 n
k * 1= log_2 n
k= log_2 n.

Iteraciones del tercer *for*:

k= 1.
k= 2.
k= 4.
…
k= $2^{k'-1}$.

2^{k-1}= p - 1
log_2 2^{k-1}= log_2 (p - 1)
(k - 1) log_2 2= log_2 (p - 1)
(k - 1) * 1= log_2 (p - 1)
k - 1= log_2 (p - 1)
k= log_2 (p - 1) + 1, si p $>$ 1.

Entonces:

q= (n - 1) [log_2 (log_2 n - 1) + 1], si n $>$ 2.

Por lo tanto, el valor que retorna *fun1* es q= (n - 1) [log_2 (log_2 n - 1) + 1], si n $>$ 2.

Ejercicio 13.

¿Cuál es el tiempo de ejecución del siguiente código?

```
void fun(int n)
{
    for (int i = 0; i < n / 2; i++)
        for (int j = 1; j + n / 2 <= n; j++)
            for (int k = 1; k <= n; k = k * 2)
                System.out.print("AyED");
}

int main()
{
  int n=8;
  fun(3);
}
```

Iteraciones del primer *for*:

i= 0.
i= 1.
i= 2.
…
i= k - 1.

k - 1= $\frac{n}{2}$ - 1
k= $\frac{n}{2}$ - 1 + 1
k= $\frac{n}{2}$.

Iteraciones del segundo *for*:

j= 1.
j= 2.
j= 3.
…
j= k.

k + $\frac{n}{2}$= n
k= n - $\frac{n}{2}$
k= $\frac{n}{2}$.

Iteraciones del tercer *for*:

k= 1.
k= 2.
k= 4.
…
k= $2^{k'-1}$.

2^{k-1}= n
log_2 2^{k-1}= log_2 n
(k - 1) log_2 2= log_2 n
(k - 1) * 1= log_2 n
k - 1= log_2 n
k= log_2 n + 1.

Entonces:

T(n)= $\sum_{i=1}^{\frac{n}{2}} \sum_{j=1}^{\frac{n}{2}} \sum_{k=1}^{log_2 n+1} cte$
T(n)= $\sum_{i=1}^{\frac{n}{2}} \sum_{j=1}^{\frac{n}{2}} (log_2 n + 1)\ cte$
T(n)= $\sum_{i=1}^{\frac{n}{2}} \frac{n}{2} (log_2 n + 1)\ cte$
T(n)= $\frac{n}{2} \frac{n}{2}$ (log_2 n +1) cte
T(n)= $\frac{1}{4}$ n^2 (log_2 n + 1) cte.

Por lo tanto, el tiempo de ejecución del código es T(n)= $\frac{1}{4}$ n^2 (log_2 n + 1) cte.

Ejercicio 14.

¿Cuál es el tiempo de ejecución del siguiente código?

```
void fun(int a, int b)
{
    // Consider a and b both are positive integers
    while (a != b) {
        if (a > b)
            a = a - b;
        else
            b = b - a;
    }
}
```

Este algoritmo implementa el cálculo del máximo común divisor (MCD) utilizando el método de resta sucesiva. La complejidad temporal de este enfoque es O(max(a, b)), ya que, en cada iteración del bucle *while*, la variable más grande se reduce por la cantidad de la menor. En el peor de los casos, si *a* y *b* son muy desiguales, el número de iteraciones podría ser proporcional a la variable con mayor valor.

Ejercicio 15.

¿Cuál es el tiempo de ejecución del siguiente código?

```
void fun(int n)
{
  for(int i=0;i*i<n;i++)
    System.out.print("AyED");
}
```

Iteraciones del *for*:

i= 0.
i= 1.
i= 2.
…
i= k - 1.

(k - 1) (k - 1)= n - 1
$(k-1)^2$= n - 1
k - 1= $\sqrt{n-1}$
k= $\sqrt{n-1}$ + 1.

Entonces:

T(n)= $\sum_{i=1}^{\sqrt{n-1}+1} cte$
T(n)= ($\sqrt{n-1}$ + 1) cte.

Por lo tanto, el tiempo de ejecución del código es T(n)= ($\sqrt{n-1}$ + 1) cte.

Ejercicio 16.

¿Cuál es el tiempo de ejecución del siguiente código?

```
int fun(int n)
{
    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j < n; j += i)
        {
            // Some O(1) task
        }
    }
}
```

Nota: Tener en cuenta que $(\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{n})$ *se puede acotar con O(log n).*

Iteraciones del primer *for*:

i= 1.
i= 2.
i= 3.
…
i= k.

k= n.

Iteraciones del segundo *for*:

j= 1.
j= 1 + i
j= 1 + i * 2
…
j= 1 + i (k - 1).

1 + i (k - 1)= n - 1
i (k - 1)= n - 1 - 1
i (k - 1)= n - 2
k - 1= $\frac{n-2}{i}$
k= $\frac{n-2}{i}$ + 1.

Entonces:

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

T(n)= $\sum_{i=1}^{n}(\frac{n-2}{i} + 1)\ cte$
T(n)= $\sum_{i=1}^{n}\frac{n-2}{i}cte + cte$
T(n)= $\sum_{i=1}^{n}\frac{n-2}{i}cte$ + $\sum_{i=1}^{n}cte$
T(n)= (n - 2) cte $\sum_{i=1}^{n}\frac{1}{i}$ + n cte
T(n)≅ cte (n - 2) log_2 n + n cte
T(n)≅ [(n - 2) log_2 n + n] cte.

Por lo tanto, el tiempo de ejecución del código es T(n)≅ [(n - 2) log_2 n + n] cte.