ED - Reto 1

Javier Gálvez Obispo

26 de octubre de 2017

1. Usando la notación O, determinar la eficiencia de los siguientes segmentos de código:

```
int n,j; int i=1; int x=0;          |     int n,j; int i=2; int x=0;
do{                                 |     do{
        j=1;                        |         j=1;
        while (j <= n){             |         while (j <= i){
                j=j*2;              |                 j=j*2;
                x++;                |                 x++;
        }                           |         }
        i++;                        |         i++;
}while (i<=n);                      |     }while (i<=n);
```

En el primer código tenemos:

- Varias operaciones del orden O(1) que podemos ignorar.
- Un bucle *while*, con eficiencia O($log_2(n)$) ya que la variable j se duplica en cada iteración.
- Un bucle *do-while* que se realiza n veces.

Por tanto el resultado es una eficiencia de O($nlog_2(n)$) ya que al estar anidados los bucles multiplicamos las eficiencias de ambos.

El segundo código se diferencia del primero en el bucle *while* que también depende de i para el número de itereaciones, por lo que se realizan $log_2(i)$ iteraciones.
En este caso no podemos multiplicar las eficiencias de ambos bucles y para calcular la eficiencia del código debemos tratar al bucle *do-while* como una sumatoria desde 1 hasta n:

$$\sum_{i=1}^{n} log_2(i) = log_2 \prod_{i=1}^{n} i = log_2(n!)$$

y utilizando las propiedades de los logaritmos llegamos a que la eficiencia del código es $log_2(n!)$

2. Para cada función $f(n)$ y cada tiempo t de la tabla siguiente, determinar el mayor tamaño de un problema que puede ser resuelto en un tiempo t (suponiendo que el algoritmo para resolver el problema tarda $f(n)$ microsegundos, es decir, $f(n) \cdot 10^{-6}$ sg).

Tenemos la siguiente expresión:

$$f(n) = t \cdot 10^6$$

y para cada $f(n)$ vamos a despejar n para poder calcularlo a partir del tiempo.

- $f(n) = log_2(n)$

$$log_2(n) = t \cdot 10^6 \implies n = 2^{t \cdot 10^6}$$

- $f(n) = n$

$$n = t \cdot 10^6$$

- $f(n) = nlog_2(n)$ (No hay forma analítica en este caso.)

$$nlog_2(n) = t \cdot 10^6$$

- $f(n) = n^3$

$$n^3 = t \cdot 10^6 \implies n = \sqrt[3]{t \cdot 10^6}$$

- $f(n) = 2^n$

$$2^n = t \cdot 10^6 \implies n = log_2(t \cdot 10^6)$$

- $f(n) = n!$ (No hay forma analítica en este caso.)

$$n! = t \cdot 10^6$$

Obtenemos la siguiente tabla al sustituir t por los valores que nos dicen:
(Para los cálculos se ha utilizado la página: Wolfram Alpha)

$f(n)$	t				
	1 sg.	1 h.	1 semana	1 año	1000 años
$log_2 n$	$9{,}9 \cdot 10^{301029}$	$2^{3600 \cdot 10^6}$	$2^{6 \cdot 10^{12}}$	$2^{3{,}15 \cdot 10^{13}}$	$2^{3{,}15 \cdot 10^{16}}$
n	10^6	$3600 \cdot 10^6$	$6{,}04 \cdot 10^{12}$	$3{,}15 \cdot 10^{13}$	$3{,}15 \cdot 10^{16}$
$nlog_2 n$	62746	$1{,}33 \cdot 10^8$	$1{,}77 \cdot 10^{10}$	$7{,}97 \cdot 10^{11}$	$6{,}41 \cdot 10^{14}$
n^3	100	1532	8456	31593	315938
2^n	19	31	39	44	54
$n!$	9	12	14	16	18