Reto 1 - Estructura de Datos

Javier Sáez Maldonado
Luis Ortega Andrés

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

   Analizaremos ambos códigos escribiendo en comentarios la eficiencia de cada parte

   Código 1

```
int n,j; //O(2)
int i = 1; int x = 0; //O(4)
do{
        j = 1; //O(1)
        while( j <= n){ // O(1)
                j = j*2; // O(2)
                x++; //O(1)
        } // El bucle es de eficiencia O(l_2(n)) * O(1)
        i++; //O(1)

}while(i<=n); //O(n-1)
```

   Analicemos porque la eficiencia del bucle interior es $l_2(n)$:

   Llamemos $f(n)$ al número de veces que se ejecuta el bucle dependiendo del valor n, entonces tenemos (por j multiplicarse por 2) que:

$$f(x) = f\left(\frac{x}{2}\right) + 1 = f\left(\frac{x}{4}\right) + 1 + 1 = ... = \underbrace{1 + 1 + ... + 1}_{log_2(n)}$$

   Así, la eficiencia de nuestro código es:

$$O(2) + O(4) + O(n-1) * (O(l_2 n) * (O(1) + O(2) + O(1)) + O(1)) =$$
$$O(6) + O(n-1) * (O(4l_2 n + 1)) =$$
$$O(6) + O(4nl_2 n + n - 4l_2 n - 1) =$$
$$O((4n-4)l_2 n + n + 5)$$

   Que, como sabemos por la notación O, podemos reducir todo eso en el que tenga mayor relevancia, quedando así como resultado final:

$$O(nl_2 n)$$

Código 2

```
int n,j; int i = 2; int x = 0; //O(6)
do{
        j = 1; // O(1)
        while(j<=i){ // O(1)
                j = j*2; //O(2)
                x++; // O(1)
        } // El bucle tiene una eficiencia de  O(log_2(i)) * O(1)
        i++; // O(1)
}while(i<=n) //O(n-2)
```

Analicemos un poco cuantas iteraciones realizan nuestros bucles anidados con respecto de n: El bucle exterior se ejecuta $n-2$ veces, donde, en cada una de ellas, el bucle interior realiza $l_2(i)$ donde i va creciendo en cada iteración del bucle exterior desde 2 hasta n. Es decir, tenemos:

$$\sum_{i=2}^{n} log_2(i) = log_2(\prod_{i=2}^{n} i) = log_2(n!)$$

Omitiendo ya, aquellas iteraciones que sabemos que pasarian a ser nulas por la notación O grande.

Ejercicio 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 $f(n) * 10^{-6}$ sg.)

Para resolver este problema, lo hacemos de la siguiente forma:

$$10^{-6} f(n) = t \Rightarrow n = f^{-1}(t * 10^{6})$$

El resultado es el siguiente.

$f(n)$	t				
	1sg	1h	1semana	1 año	1000 años
$l_2 n$	10^{300000}	2^{10^6*3600}	$2^{10^6*3600*24*7}$	$2^{10^6*3600*24*7*52}$	$2^{10^6*3600*24*7*52*1000}$
n	100160256	360576923076,9	$6,057692308 * 10^{13}$	$3{,}15 * 10^{15}$	$3{,}15 * 10^{18}$
$nl_2 n$	62746,1	$1,33 * 10^{8}$	$1,77 * 10^{10}$	$7,9 * 10^{11}$	$6{,}39 * 10^{14}$
n^3	100	1532,6	8456,8	31565	315649
2^n	19,93	31,7	39,13	44,8	54,8
$n!$	9,44	12,78	14,6	16	18,2