Reto 1

José Antonio Álvarez

22 de octubre de 2017

1. **Usando la notacion O, determinar la eficiencia de los siguientes segmentos de codigo:**

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

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

   a) En el primer algoritmo podemos observar que el bucle exterior es de orden $O(n)$ mientra que el interior de orden $O(log(n))$ ya que esta realizando tantas iteraciones como potencias de 2 menores que n. Esto es, $log_2(n)$. Por tanto la eficiencia del algortimo completo será de orden $O(n * log(n))$

   b) La única diferencia de este algoritmo (que representaremos por $f(n)$) respecto del anterior es el tope del bucle interior. Como siempre se cumple $i \leq n$, sabemos que $O(f(n)) \leq O(n * log(n))$. También sabemos que el bucle interior depende de n. Esto es, su orden de eficiencia es estrictamente mayor que $O(1)$. Por tanto $O(n) < O(f(n))$ (recordemos que el bucle exterior ya es de orden $O(n)$. Por tanto hemos obtenido que:

$$O(n) < O(f(n)) \leq O(n * log(n))$$

   Como no hay ningún orden de efiencia entre $O(n)$ y $O(n * log(n))$, hemos deducido que $O(f(n)) = O(n * log(n))$.

2. **Para cada funcion f(n) y cada tiempo t de la tabla siguiente, determinar el mayor tamano 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.)**

$f(n)$	t				
	1 segundo	1 hora	1 semana	1 año	1000 años
$log_2 n$	$\approx 10^{300000}$				
n				$\approx 3{,}15 \cdot 10^{15}$	
$nlog_2 n$		$1{,}33 \cdot 10^8$			
n^3				146645	
2^n	19				
$n!$		12			

La primera aproximación que tuve en cuenta era meramente algorítmica, pero en seguida me di cuenta de que no podía tomar este acercamiento para $f(n) \in \{log_2(n), n\}$. Por ello realicé los siguientes cálculos: el n que buscamos será el máximo de los i's que cumplen:

$$\frac{f(i)}{10^6} \leq t \leftrightarrow f(i) \leq t \cdot 10^6 \leftrightarrow i \leq f^{-1}(t \cdot 10^6)$$

Para $f(n) = log_2(n) \rightarrow f^{-1}(n) = 2^n$. Por tanto $i \leq 2^{t \cdot 10^6}$. Buscamos ahora expresarlo de la forma 10^x:

$$10^x = 2^{t \cdot 10^6} \leftrightarrow log_{10}(10^x) = log_{10}(2^{t \cdot 10^6}) \leftrightarrow x = t \cdot 10^6 \cdot log_{10}(2).$$

Obteniendo:

$$i \leq 10^{t \cdot 10^6 \cdot log_{10}(2).}$$

Para valores tan grandes podemos tomar la igualdad. Estudiémos ahora el segundo caso: $f(n) = n$).

$$\frac{f(i)}{10^6} = \frac{i}{10^6} \leq t \leftrightarrow i \leq t * 10^6$$

De nuevo tomaremos la igualdad. Para el resto de casos el estudio algorítmico es viable. Aplicando un ligero cambio en el incremento de j en el caso $f(n) = n \cdot log_2(n)$ para hacerlo aún más rápido conseguimos que su ejecución sea de apenas unos segundos.

Nota: El programa utilizado para los cálculos es el archivo adjunto **reto1.cpp**. Este es un ejemplo de ejecución:

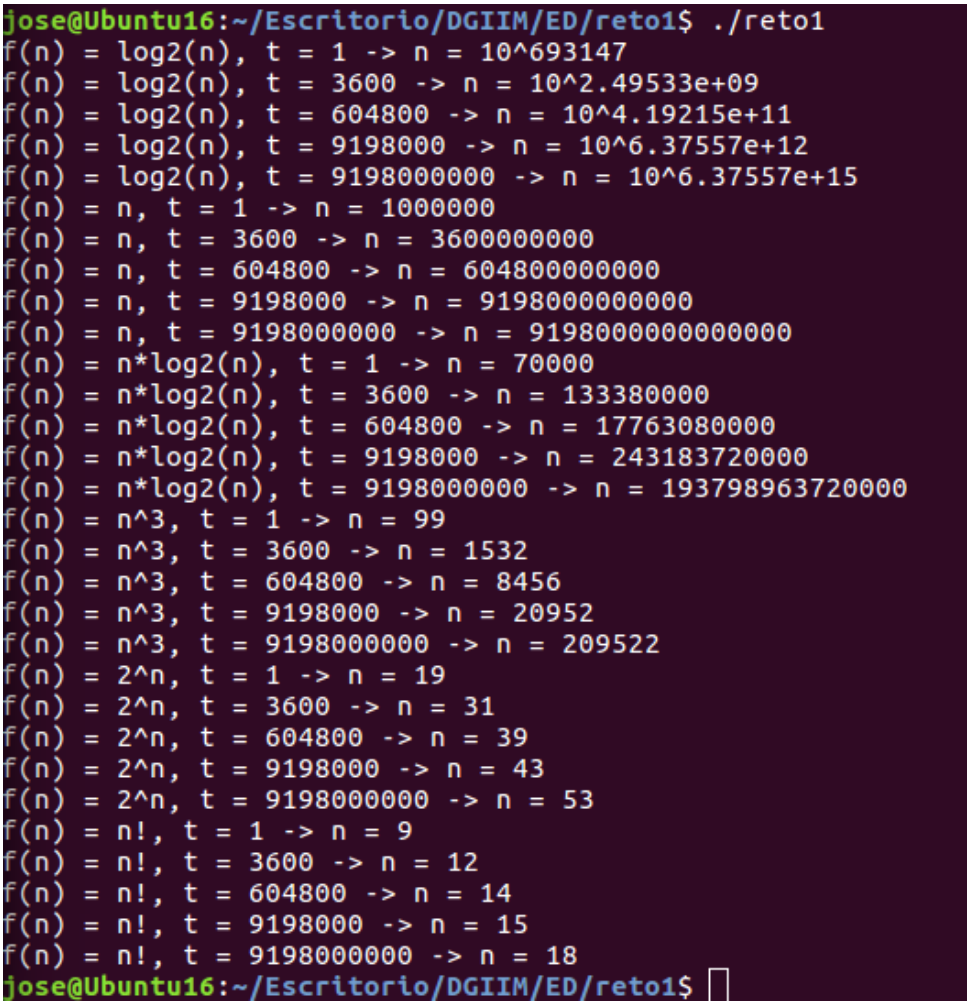

Figura 1: Cálculo final de valores de n

Por tanto la tabla queda de la siguiente forma:

$f(n)$	t				
	1 segundo	1 hora	1 semana	1 año	1000 años
$log_2 n$	10^{693147}	$10^{2,5 \cdot 10^9}$	$10^{4,19 \cdot 10^{11}}$	$10^{6,38 \cdot 10^{12}}$	$10^{6,38 \cdot 10^{15}}$
n	10^6	$3,6 \cdot 10^9$	$6,048 \cdot 10^{11}$	$9,198 \cdot 10^{12}$	$9,198 \cdot 10^{15}$
$nlog_2 n$	$7 \cdot 10^4$	$1,3338 \cdot 10^8$	$1,776308 \cdot 10^{10}$	$2,4318372 \cdot 10^{11}$	$1,9379896372 \cdot 10^{14}$
n^3	99	1532	8456	20952	209522
2^n	19	31	39	43	53
$n!$	9	12	14	15	18