Tarea 2
Complejidad

Montaño Pérez Joshua Said

1. Sea Π un problema. El desempeño computacional en el peor de los casos para Π es $O(n^2)$ y también es $\Omega(n \cdot log_2\ n)$. Sea **A** un algoritmo que soluciona Π ¿Cuáles de las siguientes afirmaciones resultan consistentes con la información sobre Π?

   Respuesta

   El algoritmo **A** tiene en el peor caso complejidad

   a) $\Theta(n^3)$

   Si es pues $n^3 > n^2$ desde $N = 2$.

   b) $\Omega(n)$

   Es similar al anterior, la constante tendría que ser mayor a n.

   c) $\Theta(n \cdot log\ n)$

   No es. De acuerdo a la propiedad 2 habría que ver que $n \cdot log\ n \in \Theta(n^2)$, lo cual no es cierto pues n^2 no acota por abajo.

   d) $\Theta(n^3)$

   El enunciado nos dice que el peor de los casos es $O(n^2)$, entonces toda ejecución esta acotada por n^2. Nos faltaría ver que sea acotada también por abajo. Pero no podemos garantizar que sea así, el enunciado nos dice que es $\Omega(n \cdot log_2\ n)$, pero no sabemos sí eso se cumple.

2. Supongamos que un algoritmo **A** se ejecuta en el peor de los casos con tiempo f(n) y el algoritmo B toma tiempo g(n), en el peor caso.

   Responda las siguientes preguntas con **sí**, **no** o **tal vez** y **justifica formalmente tu respuesta**.

   ¿Es **A** más rápido que **B**, para toda n mayor que alguna n_0 ...

   a) ... si $g(n) \in \Omega(f(n)log$ n)? **b)** ... si $g(n) \in \Theta(f(n)log$ n)?

   Respuesta

   No para el inciso a) tenemos que la definicón de Ω, ∃c∈ R^+ y $\forall n > n_0$, g(n)geq c f(n) log(n) y como log(n) es mayor a **1** y con cualquier valor de **N**, entonces g(n) leq c f(n) log(n) leq c f(n). Entonces podemos decir que g(n) es $\Omega f(n)$, lo cual nos dice que **B** no puede ser más rápido que A.

   Por otro lado en el inciso b) sabemos que $f(n) \in \Omega(f(n)log(n))$ y luego que $g(n) \in \Theta(f(n)log$ n) y como $g(n) \in \Omega(f(n)log$ n) podemos decir que $\Omega(f(n)logn) \subseteq \Omega f(n)$, entonces también $g(n) \in \Omega(f(n))$. Por lo tanto f(n) es uno de los mejores caso de **B**, pero de los peores casos para **A**, entonces **B** no es más rapido que A y podemos concluir que **A** es más rápido que **B**.

3. Determina el desempeño computacional T(n) de los siguientes ciclos anidados. Para facilitar las operaciones aritméticas, suponer que n es potencia de 2: n = 2^k para algún entero positivo k.

```
...
    i <-- n;
        while i > 0 do
            j <-- i;
                while not (j > n)
                    <cuerpo del repeat> // requiere O(1)
```

```
            j <-- j * 2
        end_w
    i <-- i div 2
end_w
...
```

Respuesta
La complejidad que tiene este pseudocódigo es de O(log^2n), debido a que tenemos dos whiles anidados, debido a que en el primer while se verifica si $i > 0$ y este va dividiendo el valor de i en 2 con valores enteros(con *log* n iteraciones), mientras que el otro while se basa en la iteraciones del primero teniendo como resultado otro (*log* n), además como los *whiles* estan anidados, ahora tenemos que multiplicarlos obteniendo como resultado log^2n.

4. Si en el código anterior cambiamos la asignación

   $i \leftarrow i$ div 2 por $i \leftarrow i/2$

   ¿Cuál sería la complejidad del algotimo?

   Respuesta
   Este no tendría complejidad ya que se vuelve un bucle infinito, esto debido a que la división es sobre los reales, entonces cada vez que si divide los números se van haciendo cada vez mas pequeños hasta nunca acabar.

5. Considera las siguientes funciones de complejidad.

   - n^2log log n
   - n^9
   - $n^{3/2}$
   - $n!$
   - 3^n
   - $2n^3 - 2n^2 + n + 7$
   - $7n^4 - 2n^3 + 3^{n+2}$
   - $log(n!)$
   - n^2 log n
   - 3^{2n}
   - n^{2n}
   - n^{logn}

   Usando la definicón de O, Ω ,Θ ,o y w así como las relaciones estar contenida, $\subset$, y ser igual, $=$, **ordenar** las siguientes funciones de complejidad en términos de O, Ω y Θ
   Respuesta

O	Ω	Θ
O(log(n!))	$\Omega(n^{2n})$	$\Theta(7n^4)$-$2n^3$+3^{n+2}
O($n^{3/2}$)	$\Omega(n^{logn})$	$\Theta(3^n)$
O(n^2log(log n))	$\Omega(n!)$	
O($2n^3$-$2n^2$+n+7)	$\Omega(3^{2n})$	
O($2n^9$)	$\Omega(7n^4$-$2n^3$+$3^{n+2})$	
O($7n^4$-$2n^3$+3^{n+2})	$\Omega(n^4)$	
O(3^{2n})	$\Omega(2n^3$-$2n^2 + n + 7)$	
O(n!)	$\Omega(n^2logn)$	
O(n^{logn})	$\Omega(n^2log(logn))$	
O(n^{2n})	$\Omega(n^{3/2})$	
	$\Omega(log(n!))$	

   Esta tabla muestra las funciones que $\subseteq$ en su respectiva columna de O, Ω y Θ y la unica igualdad que tendriamos seria la de $\Theta(7n^4)$-$2n^3$+$3^{n+2} = \Theta(3^n)$.

6. **(Opcional)** Proporcionar un algoritmo (código) cuyo desempeño computacional sea $\Theta(n^2logn)$. Debe usar operaciones básicas, **no** debes usar procesos.

   Justifica formalmente que su algoritmo alcanza el tiempo pedido

   Respuesta