Algoritmos y Estructuras de Datos II, 2ºdel Grado de Ingeniería Informática
Test de Análisis de Algoritmos, febrero 2024 - grupo 3.1

Responde y * RAZONA *** la respuesta para cada problema:**
Tiempo para hacer el examen: 1 hora y 15 minutos.

1) (3 puntos) Dado el código:

```
l=n;
while l>= 1 and par(T[1,l]);
   k=2; cell = T[k,l];
   while k<=n and cell<=T[k,l]
      for j=1..n^2
         if(T[k,l]>cell), cell++; endif
      endfor
      k=k+1;
   endwhile
   l--
endwhile
```

donde T:array$[1..n+1,\ 1..n+1]$ de enteros, estudiar los órdenes O, Ω y Θ de $t(n)$.

Solución. Para calcular la O y Ω del algoritmo, vamos a fijarnos el en peor caso y mejor caso respectivamente.

- El peor caso corresponde cuando los `while` se ejecutan si la posible interrupción causada por la segunda condición del `and` (es decir, `par(T[1,l])` y `cell<=T[k,l]`). Así pues, en este caso, tenemos 3 bucles anidados de aproximadamente n, n y n^2 pasos. Así pues, el tiempo en el peor caso es $\Theta(n^4)$.
- En el mejor caso, la condición `par(T[1,l])` es falsa y no se llega a entrar en los bucles. En ese caso, el tiempo de ejecución no depende de n siendo el mismo $\Theta(1)$.

En suma, $t(n) \in \Omega(1)$ y $O(n^4)$. Como Ω y O difieren, no existe el orden exacto del algoritmo.

2) (2 puntos) Resolver esta ecuación de recurrencia:

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

Solución.

- Ecuación recurrente: $t(n) - 4t(n-1) - 5t(n-2) = 3n^2 + n^3 5^{\frac{n}{4}} + 1$
- Ecuación característica (EC): $(x^2 - 4x - 5)(x-1)^3(x - 5^{1/4})^4 = 0$
- Soluciones EC: $5, -1, 1, 1, 1, 5^{1/4}, 5^{1/4}, 5^{1/4}, 5^{1/4}$
- Solución genérica:

$$t(n) = c_1 5^n + c_2(-1)^n + c_3 + c_4 n + c_5 n^2 + c_6 5^{n/4} + c_7 n 5^{n/4} + c_8 n^2 5^{n/4} + c_9 n^3 5^{n/4}$$

3) (1 punto) Ordenar los siguientes órdenes de ejecución, así como los Ω y Θ correspondientes, con la relación de inclusión:

$O\big(4^{\frac{n}{2}}\big)$, $O\Big(n^3 \ln^2 n + n^{\frac{5}{3}}\Big)$, $O\big(\log_2 n (\ln n)^2\big)$, $O(0.9^n)$, $O\Big(\log_5 n^{\frac{3}{2}}\Big)$

Solución. Primero notad que:

A. $O\big(4^{\frac{n}{2}}\big) = O(2^n)$

B. $O\Big(n^3 \ln^2 n + n^{\frac{5}{3}}\Big) = O\big(n^3 \ln^2 n\big)$

C. $O\big(\log_2 n (\ln n)^2\big) = O\big((\log n)^3\big)$

D. $O(0.9^n) \subset O(1)$ ya que $\lim_n 0.9^n = 0$

E. $O\left(\log_5 n^{\frac{3}{2}}\right) = O(\log n)$

De este modo, es fácil comprobar que:

$$O(D) \subset O(E) \subset O(C) \subset O(B) \subset O(A)$$

$$\Omega(D) \supset \Omega(E) \supset \Omega(C) \supset \Omega(B) \supset \Omega(A)$$

Por otro lado, los Θ no pueden ordenarse con la relación de inclusión por ser conjuntos disjuntos.

4) (1 punto) Indica si la siguiente afirmación es verdadera o falsa para cualquier algoritmo:

Si $t_p(n) \in \Omega(t_M(n))$ y $t(n) \in \Theta(t_p(n))$ entonces $t_m(n) \in \Omega(t_M(n))$

Solución. La proposición es cierta. Si $t_p(n) \in \Omega(t_M(n))$ entonces eso quiere decir que el tiempo promedio está acotado inferiomente por el tiempo en el peor caso (para un n grande). Pero, por definición, el tiempo promedio está actoado superiormente por el tiempo en el peor caso. Es decir, $t_p(n) \in O(t_M(n))$. Por tanto, $t_p(n) \in \Theta(t_M(n))$.

Ahora, usando que $t(n) \in \Theta(t_p(n))$, deducimos que $t(n) \in \Theta(t_M(n))$. En particular, se tiene que $t(n) \in \Omega(t_M(n))$. Como $t_m(n)$ es uno de los tiempos del algoritmo (uno de los $t(n)$) se puede concluir que $t_m(n) \in \Omega(t_M(n))$.

5) (3 puntos) Dada la función:

```
int g31(int Q[],int P[],int z):int
   hlp=1;
   k=1; while k<=z, k=k*2; endwhile
   if(z<=8)
      for i=1 to z, hlp++; endfor
   else
      if( mod(Q[z],4)<1)
         for i=1 to 8
            hlp+= g31(Q,P,z/2);
         endfor
      else
         for i=1 to 4
            hlp+= g31(P,Q,z/4);
         endfor
      endif
   endif
   return hlp;
```

donde P y Q son array$[1..n]$ de entero, y siendo la primera llamada con `g31(P,Q,n)`, estudiar los órdenes O, Ω, Θ y o-pequeña de su **tiempo promedio**. Para la o-pequeña, sobre su constante basta con indicar qué valores de n usar para calcularla. ¿Se podría eliminar la condición de que n sea potencia de 2?

Solución.

El $t_p(n)$ se puede expresar de la siguiente manera:

- Caso base, si $n \leq 8$: $t_p(n) = 5 + \log_2 n + n$
- En otro caso: $t_p(n) = 5 + \log_2 n + \frac{1}{4} 8 t_p(n/2) + \frac{3}{4} 4 t_p(n/4) = 5 + \log_2 n + 2t_p(n/2) + 3t_p(n/4)$

Para este problema, vamos a empezar asumiendo que n es potencia de dos, es decir, $n = 2^k$ para un $k \geq 0$. En tal caso, al sustituir, se nos queda

$$t_k - 2t_{k-1} - 3t_{k-2} = 5 + k.$$

Las raíces de la ecuación característica son: $-1, 3, 1, 1$. Así pues,

$$t_k = c_1 1^k + c_2 1^k k + c_3(-1)^k + c_4 3^k = c_1 + c_2 k + c_3(-1)^k + c_4 3^k \in \Theta(3^k)$$

Para calcular las constantes, podemos usar $k = 2$ $(n = 4)$, $k = 3$ $(n = 8)$, $k = 4$ $(n = 16)$, $k = 5$ $(n = 32)$.

Finalmente, deshaciendo el cambio $(k = \log_2 n)$, se tiene que

$$t(n) \in \Theta(3^{\log_2 n} | n = 2^k \text{ para } k \in \mathbb{N}) = \Theta(n^{\frac{1}{\log_3 2}} | n = 2^k \text{ para } k \in \mathbb{N}) \approx \Theta(n^{1.58} | n = 2^k \text{ para } k \in \mathbb{N})$$

Además, se puede tranquilamente eliminar la restricción de potencia de 2 ya que:

- $t_p(n)$ es eventualmente decreciente para un n grande
- $n^{1.58}$ es creciente
- $f(2n) = (2n)^{1.58} = 2^{1.58} n^{1.58} \in \Theta(n^{1.58})$