Algoritmos y Estructuras de Datos II, 2ºdel Grado de Ingeniería Informática
Test de Análisis de Algoritmos, febrero 2024 - grupo 2.4

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

1) (3 puntos) Dado el código:

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

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 el `while` se ejecuta sin la interrupción causada por la segunda condición del `and` (es decir, `cell<=T[k,j]`). Así pues, en este caso, tenemos 3 bucles anidados de aproximadamente n^2, n y n pasos. Así pues, el tiempo en el peor caso es $\Theta(n^4)$.
- En el mejor caso, la condición `cell<=T[k,j]` es falsa y no se llega a entrar en el `while`. En ese caso, el tiempo de ejecución sería $\Theta(n^2)$ (correspondiente al primer bucle).

En suma, $t(n) \in \Omega(n^2)$ y $O(n^4)$. Como Ω y O difieren, no existe el orden exacto del algoritmo.

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

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

Solución.

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

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

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

$O(2.1^n)$, $O\big(\log_2 n (\ln n)^2\big)$, $O\big(16^{\frac{n}{2}}\big)$, $O\Big(n \ln^2 n + n^{\frac{5}{3}}\Big)$, $O\Big(\log_2 n^{\frac{5}{2}}\Big)$

Solución. Primero vamos a reducir las expresiones:

A. $O(2.1^n)$

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

C. $O\big(16^{\frac{n}{2}}\big) = O(4^n)$

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

E. $\mathrm{O}\left(\log_2 n^{\frac{5}{2}}\right) = \mathrm{O}(\log n)$

De este modo, es fácil comprobar que:

$$\mathrm{O}(E) \subset \mathrm{O}(B) \subset \mathrm{O}(D) \subset \mathrm{O}(A) \subset \mathrm{O}(C)$$

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

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 $t_p(n) \in \Theta(t_M(n))$ y $t(n) \in \Omega(t_p(n))$ entonces $t_M(n) \in \mathrm{O}(t_m(n))$

Solución. La proposición es cierta. Como $t_p(n) \in \Theta(t_M(n))$, entonces $\Theta(t_p(n)) = \Theta(t_M(n))$. Esto implica que $t(n) \in \Omega(t_p(n)) = \Omega(t_M(n))$. Por definición, $t(n) \in \mathrm{O}(t_M(n))$, de este modo, $t(n) \in \Theta(t_M(n))$. Como $t_m(n)$ es uno de los tiempos del algoritmo (uno de los $t(n)$), se sigue que $t_m(n) \in \Theta(t_M(n))$. Por tanto, $t_M(n) \in \Theta(t_m(n))$ y en particular $t_M(n) \in \mathrm{O}(t_m(n))$.

5) (3 puntos) Dada la función:

```
int g24(int Q[],int P[],int p):int
   crl=0;
   k=1; while k<=p, k=k*3; endwhile
   if(p<=32)
      for i=1 to p, crl++; endfor
   else
      if( mod(Q[p],4)<2)
         for i=1 to 2
            crl+= g24(Q,P,p/2);
         endfor
      else
         for i=1 to 4
            crl+= g24(P,Q,p/4);
         endfor
      endif
   endif
   return crl;
```

donde P y Q son array[1..n] de entero, y siendo la primera llamada con `g24(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 $n \leq 32$: $t_p(n) = 4 + \log_3 n + n$
- En otro caso: $t_p(n) = 4 + \log_3 n + \frac{1}{2}2t_p(n/2) + \frac{1}{2}4t_p(n/4) = 4 + \log_3 n + t_p(n/2) + 2t_p(n/4)$

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

$$t_k - t_{k-1} - 2t_{k-2} = 4 + k\log_3 2.$$

Las raíces de la ecuación característica son: $-1, 2, 1, 1$. Así pues,

$$t_k = c_1 1^k + c_2 1^k k + c_3(-1)^k + c_4 2^k = c_1 + c_2 k + c_3(-1)^k + c_4 2^k \in \Theta(2^k)$$

Para calcular las constantes, podemos usar $k = 4$ ($n = 16$), $k = 5$ ($n = 32$), $k = 6$ ($n = 64$), $k = 7$ ($n = 128$).

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

$$t(n) \in \Theta(2^{\log_2 n} | n = 2^k \text{ para } k \in \mathbb{N}) = \Theta(n | n = 2^k \text{ para } k \in \mathbb{N})$$

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

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