Pontificia Universidad Católica de Chile
Departamento de Ciencia de la Computación
IIC1253 - Matemáticas Discretas

Tarea 3

15 de noviembre de 2015

2º semestre 2015 - Profesores G. Diéguez - F. Suárez

Marcel Emile Behar Fernández - 12638609

Respuestas

Pregunta 2

Pregunta 2.a

Para calcular la complejidad es necesario contar la cantidad de veces que se repiten las operaciones de comparación dentro de las iteraciones del algoritmo.
En este caso, la comparación se realiza en la linea 5 del algoritmo donde se evalúa si $p1 \neq p2$.
Como esta instrucción se encuentra de una doble iteración la instrucción se repite n veces para el loop interno y otras n veces para el loop externo.
Esto da como resultado que la complejidad del algoritmo sea $\mathcal{O}(n^2 + c)$ o bien simplemente $\mathcal{O}(n^2)$.

Pregunta 2.b

Al igual que en el caso anterior en necesario contar las veces que se ejecutan las lineas de comparación o, en este caso, de recursión.
Las líneas donde se produce la recursión son las líneas 21 y 22. Estas recursiones se ejecutan $\left\lfloor \frac{n}{2} \right\rfloor$ y $\left\lceil \frac{n}{2} \right\rceil$ veces respectivamente.
Además, existen lineas que se ejecutan n veces durante el algoritmo, por lo que la función de recurrencia sería:

$$\mathcal{T}(n) = \begin{cases} c & \leq 3 \\ \mathcal{T}\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + \mathcal{T}\left(\left\lceil \frac{n}{2} \right\rceil\right) + n & > 3 \end{cases}$$

Probaremos por inducción que esta nueva función $\mathcal{T}(n)$ es no decreciente.
PD: $(\exists n_0 \in \mathbb{N})(\forall n \geq n_0)(\mathcal{T}(n) \leq \mathcal{T}(n+1))$

- Caso base: Para $n = 3$ se tiene que $\mathcal{T}(n) = \mathcal{T}(3) = c \leq 2c + 4 = \mathcal{T}(4) = \mathcal{T}\left(\left\lfloor \frac{4}{2} \right\rfloor\right) + \mathcal{T}\left(\left\lceil \frac{4}{2} \right\rceil\right) + 4 = \mathcal{T}(n+1)$
- Hipótesis de inducción: $\mathcal{T}(n) \leq \mathcal{T}(n+1)$
- Tesis de inducción:

$$\mathcal{T}\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + \mathcal{T}\left(\left\lceil \frac{n}{2} \right\rceil\right) + n < \mathcal{T}\left(\left\lfloor \frac{n+1}{2} \right\rfloor\right) + \mathcal{T}\left(\left\lceil \frac{n+1}{2} \right\rceil\right) + n + 1 \tag{1}$$

$$0 < \underbrace{\mathcal{T}\left(\left\lfloor \frac{n+1}{2} \right\rfloor\right) - \mathcal{T}\left(\left\lfloor \frac{n}{2} \right\rfloor\right)}_{>0} + \underbrace{\mathcal{T}\left(\left\lceil \frac{n+1}{2} \right\rceil\right) - \mathcal{T}\left(\left\lceil \frac{n}{2} \right\rceil\right)}_{>0} + 1 \tag{2}$$

Por lo que se demuestra que la función es asintóticamente no decreciente desde $n_0 = 3$.

Ahora queremos dejar la función de recurrencia solamente en función de n y no de $\mathcal{T}(n)$. Para esto consideremos n de la forma $n = 2^i$ para eliminar los pisos y techos.

$$\begin{aligned}
\mathcal{T}(n) &= 2\mathcal{T}\left(\frac{n}{2}\right) + n && (3)\\
&= 2\left(2\mathcal{T}\left(\frac{n}{4}\right) + \frac{n}{2}\right) + n \\
&= 4\mathcal{T}\left(\frac{n}{4}\right) + 2n && (4)\\
&= 4\left(2\mathcal{T}\left(\frac{n}{8}\right) + \frac{n}{4}\right) + n \\
&= 8\mathcal{T}\left(\frac{n}{8}\right) + 3n && (5)\\
&\vdots \\
\mathcal{T}(n) &= 2^i\mathcal{T}\left(\frac{n}{2^i}\right) + in && (6)
\end{aligned}$$

Como esto es solo una conjetura hay que demostrarla por inducción.

- Caso base: Para $i = 1$ se tiene que $\mathcal{T}(n) = 2^1\mathcal{T}\left(\frac{n}{2^1}\right) + 1n = 2\mathcal{T}\left(\frac{n}{2}\right) + n = \mathcal{T}(n)$
- Hipótesis de inducción: $\mathcal{T}(n) = 2^i\mathcal{T}\left(\frac{n}{2^i}\right) + in$

- Tesis de inducción:

$$\mathcal{T}\left(\frac{n}{2^i}\right) = 2\mathcal{T}\left(\frac{n}{(2^i)2}\right) + \frac{n}{2^i} \tag{7}$$

$$\mathcal{T}(n) = 2^i\left(2\mathcal{T}\left(\frac{n}{(2^i)2}\right) + \frac{n}{2^i}\right) + in$$

$$\mathcal{T}(n) = 2^{i+1}\mathcal{T}\left(\frac{n}{2^{i+1}}\right) + n(i+1) \tag{8}$$

Que es lo que se quería demostrar.

Luego usando la sustitución $i = \log_2 n$ se tiene que:

$$\begin{aligned}\mathcal{T}(n) &= 2^i\mathcal{T}\left(\frac{n}{2^i}\right) + in &(6)\\ &= n\mathcal{T}(1) + n\log_2 n\\ &= nc + n\log_2 n &(9)\end{aligned}$$

Por lo tanto se tiene que $\mathcal{T}(n) \in \mathcal{O}(n\log n|POTENCIA_2)$

Por último, como las funciones $\mathcal{T}(n)$ y $nlogn$ son asisntóticamente no decrecientes, $nlogn$ es b-armónica y $\mathcal{T}(n) \in \mathcal{O}(n\log n|POTENCIA_2)$ se tiene que $\mathcal{T}(n) = \mathcal{O}(n\log n)$.

Pregunta 2.c

Como se observó en la pregunta anterior, la función de recurrencia para el algoritmo sería:

$$\mathcal{T}(n) = \begin{cases} c & \leq 3 \\ \mathcal{T}\left(\left\lfloor\frac{n}{2}\right\rfloor\right) + \mathcal{T}\left(\left\lceil\frac{n}{2}\right\rceil\right) + n & > 3 \end{cases}$$

Utilizando el Teorema Maestro para funciones recursivas del tipo "Dividir y Conquistar":

$$\mathcal{T}(n) = \begin{cases} \mathcal{O}(n^d) & a_1 + a_2 < b^d \\ \mathcal{O}(n^d\log n) & a_1 + a_2 = b^d \\ \mathcal{O}(n^{\log_b(a_1+b_2)}) & a_1 + a_2 > b^d \end{cases}$$

Es fácil notar que $a_1 = a_2 = 1, b = 2, c = 1$ y $d = 1$.
Luego, en este caso se tiene que $a_1 + a_2 = b^n$ por lo que por el Teorema Maestro se llega a que $\mathcal{T}(n) = \mathcal{O}(n^d\log n) = \mathcal{O}(n\log n)$ corroborando lo obtenido en (b).

Pregunta 2.d

Es claro notar que el mejor algoritmo es el de Raúl Dibbs puesto que su complejidad es $\mathcal{O}(n\log n)$ mientras que la complejidad del algoritmo de Claudio Beauchef es $\mathcal{O}(n^2)$, la cual es mayor.