Universidad de Chile

Departamento de Ciencias de la Computación

Control 1 CC4102

Semestre: Otoño 2011

Solution MarkingScheme

Problema 1 (1.5 puntos)

a. **Cotas**: ¿Verdadero o Falso? (Justifica cada respuesta)

——————————begin solution——————————

Recuerde las definiciones:

$$O(g(n)) = \{f(n) : \exists c, n_0 > 0, \forall n \geq n_o, 0 \leq f(n) \leq cg(n)\}$$

$$\Omega(g(n)) = \{f(n) : \exists c, n_0 > 0, \forall n \geq n_o, 0 \leq cg(n) \leq f(n)\}$$

$$\Theta(g(n)) = \{f(n) : \exists c_1, c_2, n_0 > 0, \forall n \geq n_0, 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}$$

——————————end solution——————————

a) $n^2 + 10000n \in O(n^2)$

——————————begin solution——————————

Usando la definición, tenemos que $n^2 + 10000n \leq cn^2$ para una constante c y $n \geq n_0$. Luego

$$n^2 + 10000n \leq cn^2$$

$$1 + \frac{10000}{n} \leq c$$

Tomando cualquier c tal que $c \geq 10001$ para todo $n \geq 1$ se tiene el resultado.

——————————end solution——————————

b) $n^{3,00001} \in O(n^3)$

——————————begin solution——————————

$$n^{3,00001} \leq cn^3$$

$$n^{0,0001} \leq c$$

Como la función $f(n) = n^{0,00001}$ no es acotada superiormente, no puede existir tal constante c, luego la afirmación es falsa.

——————————end solution——————————

c) $n^2 - n \in \Omega(n^2)$

——————————begin solution——————————

$$cn^2 \leq n^2 - n$$

$$0 < c \leq 1 - \frac{1}{n}$$

Tomando $c = \frac{1}{2}$ se tiene el resultado $\forall n \geq 2$

——————————end solution——————————

d) $n^{2,99999} \in \Omega(n^3)$

————————begin solution————————

$$cn^3 \leq n^{2,99999}$$

$$0 < c \leq \frac{1}{n^{0,00001}}$$

Como $\frac{1}{n^{0,00001}}$ converge a 0 cuando n es suficientemente grande, no puede existir tal constante c y luego la afirmación es falsa.

————————end solution————————

e) $2^{n+1} \in O(2^n)$

————————begin solution————————

Como $2^{n+1} = 2 \cdot 2^n$, tomando $c = 2$ se tiene el resultado $\forall n > 0$

————————end solution————————

f) $(n+1)! \in \Omega(n!)$

————————begin solution————————

$(n+1)! = (n+1)n!$, luego $c \leq (n+1)$, tomando $c = 1$ se tiene el resultado $\forall n > 0$

————————end solution————————

g) $4^{\lg n} \in \Theta(n^2)$

————————begin solution————————

$4^{\lg n} = n^2$. Tomando $c_1 = c_2 = 1$ se tiene el resultado.

————————end solution————————

h) $\lg(n^2) + \lg n \in \Theta(\lg n^3)$

————————begin solution————————

$\lg(n^2) + \lg n = 3 \lg n$, tomando $c_1 = c_2 = 1$ se tiene el resultado.

————————end solution————————

i) $2^{2n} \in O(2^n)$

————————begin solution————————

$2^{2n} \leq c2^n$, luego $2^n \leq c$ pero como la función 2^n es no acotada superiormente, la afirmación es falsa.

————————end solution————————

b. Demostrar las siguientes series geométricas vistas en clase:

- $\displaystyle\sum_{i=1}^{n} \frac{1}{2^i} = 1$, para n grande.

——————begin solution——————

Sea $S = \sum_{i=1}^{n} \frac{1}{2^i}$ y considere

$$\begin{aligned} 2S &= 1 + \frac{1}{2} + \frac{1}{4} + \ldots + \frac{1}{2^{n-1}} \\ -S &= \quad - \frac{1}{2} - \frac{1}{4} - \ldots - \frac{1}{2^{n-1}} - \frac{1}{2^n} \\ 2S - S &= 1 - \frac{1}{2^n} \end{aligned}$$

Luego para n suficientemente grande, $S = 1$. *Otra forma de ver el resultado es sumar progresivamente la mitad del área restante de un cuadrado de lado 1. Al final se cubre toda el área, excepto por la mitad n-ésima, es decir,* $\frac{1}{2^n}$

——————end solution——————

- $\sum_{i=1}^{n} \frac{i}{2^i} = 2$, para n grande.

——————begin solution——————

Al igual que la anterior, considere $S = \sum_{i=1}^{n} \frac{i}{2^i}$. La suma $2S - S$ es de la forma $\sum_{i=1}^{n-1} \frac{1}{2^i} + 1 + \frac{n}{2^n} = 2 - \frac{n}{2^n} - \frac{1}{2^{n-1}} \to 2$ para n suficientemente grande.

——————end solution——————

Esquema de Puntajes:

a. 0,1 *puntos por cada asintótica*

b. 0,3 *puntos por cada suma*

Problema 2 (1.5 puntos)

El `Problema de las Torres de Hanoï` es un ejemplo clásico en recursividad, originalmente propuesto por Édouard Lucas. Un algoritmo recursivo es conocido desde 1892, moviendo los n discos de una torre en $2^n - 1$ movimientos. Este valor ha sido probado óptimo por una simple cota inferior.

Considere una variante para introducir más diversidad en las instancias, donde se permiten discos del mismo tamaño. Llamaremos a este problema el `Disk Pile Problem`. Esto obviamente introduce instancias mucho más fáciles, como la instancia en la cual la torre puede ser movida en tiempo lineal (Figura 1).

a. De un algoritmo optimal y recursivo para mover una `Disk Pile` de un pilar a otro, utilizando sólo un pilar adicional, sabiendo que $\forall i \in \{1, \ldots, s\}$, n_i es el número de discos de tamaño i.

————————begin solution————————

Es igual que el algoritmo para mover las Tower of Hanoï, excepto que siempre se moverán todos los discos de tamaño i en n_i movimientos consecutivos.

```
move(s,(n_1,..,n_s),A,B,C) = {
 if(s>0) {
  move(s-1,(n_1,..,n_{s-1}),A,C,B)
  move(n_s,A,C)
  move(s-1,(n_1,..,n_{s-1}),B,A,C)
 }
}
```

————————end solution————————

b. De y demuestre la complejidad exacta de su algoritmo en el peor caso sobre todas las instancias donde s y el vector $(n_1, \ldots, n_s)$ están fijos.

————————begin solution————————

Resolviendo la recurrencia directamente de la recursión del algoritmo, se obtiene que los n_s discos más grandes se mueven una vez, los n_{s-1} segundos más grandes se mueven dos veces, los n_{s-2} discos cuatro veces, y así sucesivamente hasta los n_1 discos más pequeños, los cuales se mueven 2^{s-1} veces. Sumando todos los movimientos se obtiene la cantidad de movimientos realizados por el algoritmo:

$$\sum_{i \in \{1,\ldots,s\}} n_i 2^{s-i}$$

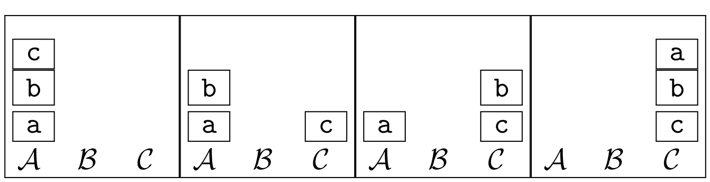


Figura 1: Moviendo una pila de discos de tamaño 3.

——————————end solution——————————

c. Pruebe que tal complejidad es óptima.

——————————begin solution——————————

Una cota inferior de $\sum_{i\in\{1,\ldots,s\}} n_i 2^{s-i}$ puede ser probada por inducción sobre el número de discos, o sobre el número de *tipos* de discos. Probaremos por inducción sobre la cantidad de tipos de discos s que cualquier pila de discos de tamaños $(n_1, \ldots, n_s)$ requiere $\sum_{i\in\{1,\ldots,s\}} n_i 2^{s-i}$ movimientos para ser movidos a otro pilar.

- *Caso base*: Para $s = 1$ la cota es n_1. Claramente, pues se necesitan n_1 movimientos para mover los n_1 discos.
- *Hipótesis de Inducción*: Suponga que hay $\sigma \geq 1$ tamaños, de forma que cualquier pila de discos de tamaños $(n_1, \ldots, n_\sigma)$ requiere $\sum_{i\in\{1,\ldots,\sigma\}} n_i 2^{\sigma-i}$ movimientos.
- *Paso inductivo*: Considere una pila de discos de tamaños $(n_1, \ldots, n_{\sigma+1})$; claramente todos los discos de tamaños menores que $\sigma + 1$ necesitan ser llevados a un único pilar antes de mover los discos más grandes, de forma de trasladar estos últimos en $n_{\sigma+1}$ movimientos, después de los cuales todos los discos de tamaño menor que $\sigma + 1$ sean puestos sobre los más grandes. Por la hipótesis inductiva, mover los discos más pequeños requiere $\mathbf{2} \sum_{i\in\{1,\ldots,\sigma\}} n_i 2^{\sigma-i}$ movimientos más los $n_{\sigma+1}$ restantes. Luego, cualquier pila de tamaños $(n_1, \ldots, n_{\sigma+1})$ requiere $\sum_{i\in\{1,\ldots,\sigma+1\}} n_i 2^{\sigma+1-i}$ movimientos para ser movida a otro pilar.
- *Conclusión*: La hipótesis de inducción es verificada para el caso $s = 1$, y se propaga para cualquier valor $s \geq 1$ gracias al paso inductivo. Concluimos que cualquier pila de discos de tamaños $(n_1, \ldots, n_s)$ para $s \geq 1$ requiere $\sum_{i\in\{1,\ldots,s\}} n_i 2^{s-i}$ movimientos para ser movida a otro pilar.

——————————end solution——————————

d. ¿Cuál es la complejidad exacta de su algoritmo en el peor caso sobre todas las instancias donde s y el total de discos n están fijos?

——————————begin solution——————————

El peor caso ocurre cuando $n_1 = n - s + 1$ y $n_2 = \ldots = n_s = 1$: los discos más pequeños se mueven más veces, así que maximizar su número maximiza la complejidad. Usando el resultado anterior, nos da una complejidad de $(n-s+1)2^{s-1} + \sum_{i\in[2..s]} 2^{s-i}$. Pero $\sum_{i\in[2..s]} 2^{s-i} = 2^0 + 2^1 + \cdots + 2^{s-2} = 2^{s-1} - 1$, por lo tanto, la complejidad es $(n-s+1)2^{s-1} + 2^{s-1} - 1 = (n-s+2)2^{s-1} - 1$.

——————————end solution——————————

Esquema de Puntajes:

a. 0,4 *puntos*

b. 0,4 *puntos*

c. 0,4 *puntos*

d. 0,3 *puntos*

Problema 3 (1.5 puntos)
Dado un arreglo ordenado A de n enteros y un entero x, ¿cuántas comparaciones con elementos del arreglo son necesarias para decidir si x pertenece a A (en el peor caso)? Calcule la cantidad exacta de comparaciones, de su clase asintótica más precisa, y demuestre sus resultados.

——————begin solution———————

Sea L_l el conjunto de todas las posibles posiciones de x en el arreglo A luego de la l-ésima comparación de algún algoritmo arbitrario (note que no sabemos el dominio de l *a prori*). Se tiene inicialmente que $L_0 = \{1, \ldots, n\}$. Utilizando la estrategia del adversario, al consultar por $i < j$ en la $(l+1)$-ésima comparación,

$$|L_{l+1}| \geq \lceil |L_l|/2 \rceil$$

A partir de esto, se deduce que cuando $k = \lceil \lg n \rceil$, entonces $|L_k| = 1$ (puede ser probado por inducción). Luego, dada la posición (el *rango de inserción*) de x en A, se necesita una comparación extra para determinar si se encuentra en el arreglo o no, por lo que se necesitan $1 + \lceil \lg n \rceil$ comparaciones en el peor caso, o bien $\Omega(\log n)$.

——————end solution———————

Esquema de Puntajes:

a. 0,5 *puntos para el orden asimptotico de la cota inferior*

b. 0,5 *puntos para el valor exacta*

c. 0,5 *puntos para la demostracion exacta*

Problema 4 (1.5 puntos)
Un centro médico ha contaminado a una persona dentro de N personas. La persona contaminada debe recibir un tratamiento dentro de los 15 próximos días.

Existe un test de sangre que, después de una incubación en una máquina por 15 días, indica si la sangre probada está contaminada o no. Los científicos del laboratorio quieren ejecutar las N pruebas en paralelo usando N máquinas, pero no poseen tantas.

Observe que la prueba puede detectar pequeñas cantidades del virus, como por ejemplo en una mezcla. Proponga un protocolo para ejecutar solamente $O(\log N)$ pruebas en paralelo, y explíquelo en un ejemplo para $N = 16$. En particular, su respuesta debería especificar

a. el conjunto de pruebas ejecutadas;

b. una corta prueba de que $O(\log N)$ pruebas son ejecutadas;

c. cómo analizar el resultado de las pruebas;

d. un ejemplo del protocolo para $N = 16$.

————————begin solution————————

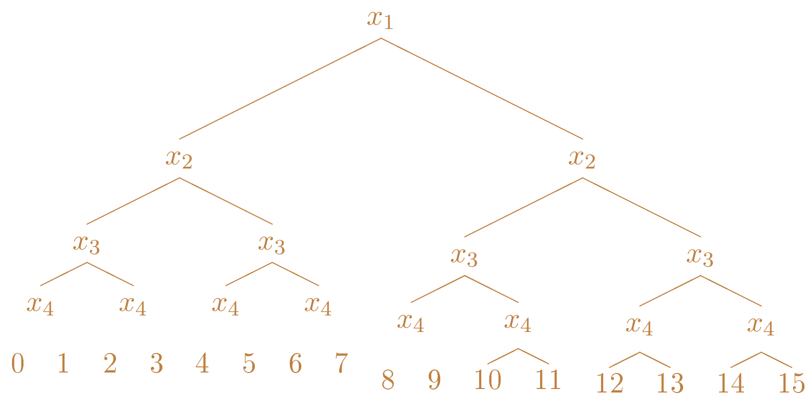
a. Preparar las pociones, cada una contienendo la sangre de un número de muestras como sigue: Enumere las muestras de 0 a $N-1$, escriba esos números en binario, y observe los bits de la codificación resultante de izquierda a derecha; la poción p contiene sangre de la muestra b si y solo si el p-ésimo bit de la codificación para la muestra b es 1. Este es el conjunto de pruebas a ser realizadas.

b. Note que con $\lceil \lg N \rceil$ bits se pueden codificar $2^{\lceil lgN \rceil} \geq N$ números diferentes en binario. El número de pociones necesarias es el número de bits necesarios, por lo que la cantidad de tests es $n = \lceil \lg N \rceil$.

c. Para determinar quién fue contaminado a partir de los resultados, defina $x_p = 1$ si la poción p salió positiva en el test. Los valores de $(x_i)_{i \leq n}$ forman un número x de n bits, y este número (entre 0 y $N-1$) es el número de la muestra contaminada.

d. Por ejemplo, en el caso en el cual una persona fue contaminada entre $N = 16$, el análisis para $n = 4$ pociones compuestas como se describió anteriormente permite encontrar a la persona, como se describe en el árbol a continuación.

Poción 1 es la mezcla de las muestras	8 a 15;
poción 2 es la mezcla de las muestras	4 a 7 y 12 a 15;
poción 3 es la mezcla de las muestras	$\{2, 3, 6, 7, 10, 11, 14, 15\}$;
poción 4 es la mezcla de las muestras	correspodientes a los números impares



——————end solution——————

Esquema de Puntajes:

a. 0,4 *puntos*

b. 0,4 *puntos*

c. 0,4 *puntos*

d. 0,3 *puntos*