Ejercicios en clase: División y conquista 2

Análisis y Diseño de Algoritmos

24 de abril de 2020

Ejercicio 1. Decimos que un vector $A[1..n]$ es *unimodal* si existe un índice p, llamado *pico* tal que $A[1..p]$ es una secuencia creciente, y $A[p+1..n]$ es una secuencia decreciente. Diseñe un algoritmo de división y conquista que recibe un vector unimodal y encuentra el pico de A. Su algoritmo debe tener complejidad $\Theta(\lg n)$ en el peor caso. Escriba la recurrencia y resuélvala usando los métodos visto en clase. Verifique que su recurrencia es correcta usando el teorema maestro.

Solución.

Primera parte: Diseñar un algoritmo de división y conquista.

Observamos que existen dos únicas posibilidades para el pico de un vector unimodal: o está en el lado ïzquierdo"del arreglo o está en lado "derecho"del arreglo.

Esto nos da la idea de un algoritmo de división y conquista. Sumado a esto, note que si A es un vector unimodal, entonces cualquier subarreglo en A es también unimodal, portanto se respetan las condiciones de los parámetros de entrada en las llamadas recursivas.

Recibe: Un vector unimodal $A[p..r]$ con elementos diferentes

Devuelve: el pico de A

PICO (A, p, r)	*cost*	*times*
1: **if** $p == r$	c_1	1
2: **return** $A[p]$	c_2	0
3: $q = \lfloor (r - p + 1)/2 \rfloor$	c_3	1
4: **if** $A[q] < A[q+1]$	c_4	1
5: **return** PICO$(A, q+1, r)$	$T(\lceil n/2 \rceil)$	1
6: **else**		
7: **return** PICO(A, p, q)	$T(\lfloor n/2 \rfloor)$	1

Segunda parte: Escribir la recurrencia y resolverla.

Tenemos la siguiente recurrencia para el tiempo de ejecución del algoritmo en el peor caso:

$$T(n) = \begin{cases} c_1 & n = 1 \\ T(\lceil \frac{n}{2} \rceil) + d & \text{caso contrário} \end{cases}$$

Resolveremos primero para $n = 2^k$ para algún k

$$\begin{aligned} T(n) &= T(2^k) \\ &= T(2^{k-1}) + d \\ &= T(2^{k-2}) + d + d \\ &= T(2^{k-j}) + dj \\ &= T(1) + dk \\ &= c_1 + dk \\ &= c_1 + d \lg n \end{aligned}$$

Ahora suponga que n no es potencia de 2. Luego $2^k \leq n < 2^{k+1}$ para algún entero k. Luego, como $T(n)$ es creciente (ver final),

$$T(n) < T(2^{k+1}) = c_1 + d(k+1) = c_1 + d + dk \leq c_1 + d + d \lg n$$

(la última desigualdad se obtiene porque $2^k \leq n$).

Vea también que

$$T(n) \geq T(2^k) = c_1 + dk = c_1 - d + d(k+1) > c_1 - d + d \lg n$$

(la última desigualdad se obtiene porque $2^{k+1} > n$).

Concluimos que

$$c - d + d \lg n < T(n) \leq c + d + d \lg n.$$

Portanto, $T(n) = \Theta(\lg n)$.

Faltaba demostrar que $T(n) = T(\lceil \frac{n}{2} \rceil) + d$ es una función creciente. Demostraremos por inducción en n que $T(n) \leq T(n+1)$. Si $n = 1$, tenemos que $T(1) = c_1 \leq c_1 + d = T(1) + d = T(2)$. Si $n > 1$, tenemos dos casos.

Si n es impar, $T(n) = T(\lceil \frac{n}{2} \rceil) + d = T(\lfloor \frac{n+1}{2} \rfloor) + d = T(n+1)$.

Si n es par, $T(n) = T(\lceil \frac{n}{2} \rceil) + d \leq T(\lceil \frac{n}{2} \rceil + 1) + d = T(\lceil \frac{n+1}{2} \rceil) + d = T(n+1)$.

Tercera parte: Comprobar usando teorema maestro.

Finalmente, el ejercicio nos pide comprobar nuestra solución mediante el teorema maestro. Para esto, note que, cuando n es potencia de 2, tenemos que $T(n) = T(n/2) + dn^0$. Luego $a = 1, b = 2, k = 0$ y $\lg 1 / \lg 2 = 0 = k$. Portanto estamos en el caso 2 del teorema maestro, y $T(n) = \Theta(n^0 \lg n) = \Theta(\lg n)$.

Ejercicio 2. Considere el siguiente problema. Entrada: Dos vectores $A[1..n]$, $B[1..n]$ con elementos distintos dos a dos, ordenados de manera creciente. Salida: La mediana del conjunto de elementos que están en A o en B. Es decir, el elemento v tal que existen exactamente $n-1$ elementos menores en A o B.

Por ejemplo, si $A = [10, 30, 50, 70]$, $B = [20, 40, 60, 80]$, la mediana correspondiente es 40, ya que $n = 4$ y existen 3 elementos menores que 40. Diseñe un algoritmo $\Theta(\lg n)$ para el problema de la mediana. En este ejercicio puede considerar que n es potencia de 2. Escriba el pseudocódigo, su recurrencia para el peor caso y concluya el tiempo de ejecución

Solución

Primera parte: Diseñar un algoritmo de división y conquista.

Note que existen 4 posibilidades en donde puede estar la mediana:$A[1..n/2], A[n/2 + 1..n], B[1..n/2], B[n/2 + 1..n]$.

Al comparar los puntos medios tenemos las siguientes posibilidaes

Caso 1 $A[n/2] < B[n/2]$. En ese caso, tenemos que todos los elementos en $A[1..n/2]$ son menores que todos los elementos en $A[n/2+1..n] \cup B[n/2..n]$ Como existen $n+1$ elementos en $A[n/2 + 1..n] \cup B[n/2..n]$, la mediana no puede estar en $A[1..n/2]$. Análogamente, observe que todos los elementos en $B[n/2+1..n]$ son mayores que todos los elementos en $B[1..n/2] \cup A[1..n/2]$. Luego la mediana no puede estar en $B[n/2 + 1..n]$.

Caso 2 $A[n/2+1] > B[n/2+1]$. De manera similar al caso anterior, deducimos que la mediana no puede estar ni en ni $B[1..n/2]$ ni en $A[n/2 + 1..n]$.

El análisis anterior nos permite diseñar el siguiente algoritmo.

Recibe: Dos vectores $A[p..q]$, $B[r..s]$ con elementos distintos dos a dos,ordenados de manera creciente.

Devuelve: La mediana del conjunto de elementos que están en A o en B

MEDIANA (A, p, q, B, r, s)	*cost*	*times*
if $p == q$	c_1	1
return mín$\{A[p], B[r]\}$		
$mid_A = (p + q - 1)/2$	c_2	1
$mid_B = (r + s - 1)/2$	c_3	1
if $A[mid_A] < B[mid_B]$	c_4	1
return MEDIANA$(A, mid_A + 1, q, B, r, mid_B)$	$T(n/2)$	1
else		
return MEDIANA$(A, p, mid_A, B, mid_B + 1, s)$	$T(n/2)$	1

Segunda parte: Escribir la recurrencia y resolverla.

El enunciado del problema nos permite asumir que n es potencia de 2, luego, el tiempo de ejecución viene dado por

$$T(n) = \begin{cases} c & n = 1 \\ T(\frac{n}{2}) + d & \text{caso contrário} \end{cases}$$

Como $n = 2^k$ para algún k natural, tenemos

$$\begin{aligned} T(n) &= T(2^k) \\ &= T(2^{k-1}) + d \\ &= T(2^{k-2}) + d + d \\ &= T(2^{k-j}) + dj \\ &= T(1) + dk \\ &= c + dk \\ &= c + d \lg n \end{aligned}$$

Luego $T(n) = \Theta(\lg n)$.

Tercera parte: Comprobar usando teorema maestro.

Finalmente, el ejercicio nos pide comprobar nuestra solución mediante el teorema maestro. Para esto, note que, cuando n es potencia de 2, tenemos que $T(n) = T(n/2) + dn^0$. Luego $a = 1, b = 2, k = 0$ y $\lg 1 / \lg 2 = 0 = k$. Portanto estamos en el caso 2 del teorema maestro, y $T(n) = \Theta(n^0 \lg n) = \Theta(\lg n)$.

Ejercicio 3. Entrada: un arreglo ordenado $A[1..n]$ de números enteros diferentes. Salida: El número de inversiones significativas, donde una *inversión significativa* es un par ordenado (i, j) tal que $i < j$ y $A[i] > 2A[j]$. El algoritmo debe tener complejidad $\Theta(n \lg n)$ Escriba el pseudocódigo del algoritmo anterior. Escriba una recurrencia para el peor caso de este algoritmo. Resuelva la recurrencia.

Ejercicio 4. Considere el siguiente problema de búsqueda. Entrada: un arreglo $A[1..n]$ de números enteros Salida: El número máximo de elementos con el mismo valor Por ejemplo, si $A = [2, 4, 2, 4, 2, 2, 1, 4, 3]$, el algoritmo debe devolver el valor 2. Su algoritmo debe consumir tiempo $\Theta(n \lg n)$ en el peor caso. Escriba el pseudocódigo del algoritmo anterior. Escriba una recurrencia para el peor caso de este algoritmo. Resuelva la recurrencia.