Ejercicios en clase: División y conquista 2

Análisis y Diseño de Algoritmos

25 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.

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

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 índice i tal que existen más de $n/2$ números iguales que $A[i]$. Si no existe tal índice, devolver -1. Por ejemplo, si $A = [2, 4, 2, 4, 2, 2, 1, 4, 2]$, 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.

Ejercicio 5. Dado un arreglo $A[1..n]$, una k-rotación de A es un arreglo $B[1..n]$ tal que

$$B(k) = \begin{cases} A[i+k] & i+k \leq n \\ A[(i+k) \mod n] & \text{caso contrário} \end{cases}$$

Por ejemplo, si $A = [3, 6, 9, 10]$, una 2-rotación de A es $B = [9, 10, 3, 6]$.

Considere el siguiente problema. Entrada: Una k-rotación B de un arreglo ordenado de elementos diferentes. Salida: El número k. Por ejemplo, si $B = [9, 10, 3, 6]$, el algoritmo debe devolver el valor 2.

Diseñe un algoritmo $\Theta(\lg n)$ para el peor caso del problema. Escriba el pseudocódigo del algoritmo. Escriba una recurrencia para el peor caso de este algoritmo. Resuelva la recurrencia. Verifique con teorema maestro. Verifique usando inducción.

Ejercicio 6. Considere el siguiente problema. Entrada: Un arreglo $A[1..n]$ de números enteros positivos ordenado de manera creciente. Salida: El mínimo número entero positivo que no está en A. Por ejemplo, si $B = [1, 2, 3, 4, 7, 8, 10, 12]$, el algoritmo debe devolver el valor 5.

Diseñe un algoritmo $\Theta(\lg n)$ en el peor caso. Escriba el pseudocódigo del algoritmo anterior. Escriba una recurrencia para el peor caso de este algoritmo. Resuelva la recurrencia. Verifique con teorema maestro. Verifique usando inducción.

Ejercicio 7. Considere el siguiente problema.

Entrada: Un arreglo $A[1..n]$ de números enteros positivos diferentes (no necesariamente ordenado).

Salida: Un índice k tal que $A[1], \ldots, A[k-1] < A[k]$ y $A[k] > A[k+1], \ldots, A[n]$. Si no existe tal índice k, devolver -1. Por ejemplo, si $A = [1, 3, 2, 4, 8, 7, 5, 6]$, el algoritmo debe devolver el índice 5.

Diseñe un algoritmo de división y conquista que consuma tiempo $\Theta(n)$ en el peor caso. Escriba el pseudocódigo del algoritmo anterior. Escriba una recurrencia para el peor caso de este algoritmo. Resuelva la recurrencia. Verifique con teorema maestro. Verifique usando inducción.

Ejercicio 8. Considere el siguiente problema.

Entrada: Un arreglo $A[1..n]$ de números enteros positivos diferentes (no necesariamente ordenado).

Salida: Un par de índices (i, j) tales que $A[i] = A[i+1] = \cdots = A[j]$ y $j - i$ tiene valor máximo Por ejemplo, si $A = [1, 3, 2, 2, 8, 7, 7, 7, 3, 4, 4, 3]$, el algoritmo debe devolver el par $(6, 8)$.

Diseñe un algoritmo de división y conquista que consuma 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. Verifique con teorema maestro. Verifique usando inducción.

Ejercicio 9. Considere el siguiente problema.

Entrada: Un conjunto de k arreglos $A_1, A_2, \ldots A_k$ que en conjunto tienen tamaño n.

Salida: Un arreglo $B[1..n]$ con todos los elementos de la entrada ordenados Por ejemplo, si $A_1 = [1, 3], A_2 = [2, 8, 7], A_3 = [7, 3]$, el algoritmo debe devolver $B = [1, 2, 3, 3, 7, 7, 8]$.

Diseñe un algoritmo de división y conquista que consuma tiempo $\Theta(n \lg k)$ en el peor caso. Escriba el pseudocódigo del algoritmo anterior. Escriba una recurrencia para el peor caso de este algoritmo. Resuelva la recurrencia. Verifique con teorema maestro. Verifique usando inducción.