Examen 2

Análisis y Diseño de Algoritmos

3 de Febrero del 2022

Ejercicio 1 (3 ptos). ¿Cuantas hojas tiene un heap con n nodos? Demuestre.

Ejercicio 2 (3.5 ptos). Considere el arreglo $A_1 = [9, 13, 5, 12, 8, 7]$. Ilustre el algoritmo BUILD-MAX-HEAP en dicho arreglo. Sea A_2 el arreglo resultante. Ilustre el algoritmo HEAP-EXTRACT-MAX en el arreglo A_2. Luego de esta operación, ilustre el algoritmo HEAP-SORT en el arreglo A_2 (que acaba de ser modificado). Debe mostrar todos los pasos en su ilustración.

Ejercicio 3 (3 ptos). De un ejemplo de una permutación de los siete primeros números naturales (1...7) en donde el Quicksort hace la menor cantidad de comparaciones posibles. Justifique. ¿Cuántas comparaciones hace el Quicksort en su ejemplo? Observación: cuando hablamos de comparaciones, hablamos de comparaciones **entre elementos**.

Ejercicio 4 (4 ptos). Considere el siguiente algoritmo que determina el segundo mayor elemento de un vector $v[1 \ldots n]$ com $n \geq 2$ números positivos distintos.

```
ALGO(v, n)
 1: mayor = 0
 2: segundo_mayor = 0
 3: for i = 1 to n
 4:    if v[i] > mayor
 5:       segundo_mayor = mayor
 6:       mayor = v[i]
 7:    else
 8:       if v[i] > segundo_mayor
 9:          segundo_mayor = v[i]
10: return  segundo_mayor
```

Suponga que v es una permutación de 1 a n escogida de entre todas las permutaciones de 1 a n con distribución uniforme de probabilidad. Sea X la variable aleatória que guarda el número de veces que la variable *segundo_mayor* es alterada (osea, el número de ejecuciones de las líneas 5 y 9 del algoritmo) en una llamada a ALGO(v, n). Calcule el valor esperado de X.

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

$$B[k] = \begin{cases} A[i+k] & : 1 \leq i+k \leq n \\ A[(i+k)-n] & : \text{caso contrário} \end{cases}$$

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 de **división y conquista** (escriba el pseudocódigo del algoritmo) $\Theta(\lg n)$ en el peor caso. Escriba una recurrencia para el desempeño de este algoritmo. Compruebe el crecimiento Θ de esta recurrencia **usando teorema maestro**. *Además, use un ejemplo sencillo explicando brevemente cómo su algoritmo funciona para dicho caso en particular.*

Ejercicio 6 (3.5 ptos). Dado un arreglo $A[1..n]$ de números diferentes, decimos que está k-casi ordenado, si cada elemento está alejado como máximo k posiciones de su índice correspondiente en el arreglo ordenado de manera creciente. Por ejemplo, el arreglo $A = [2, 4, 1, 5, 3]$ está 2-casi ordenado, ya que el arreglo ordenado es $[1, 2, 3, 4, 5]$, y como podemos observar, cada elemento se ha desplazado como máximo dos posiciones de su posición original.

Diseñe un algoritmo que recibe un arreglo $A[1..n]$, y un número $k < n$, tal que A está k-ordenado, y devuelve un arreglo con los elementos de A ordenados de manera creciente. Su algoritmo deberá tener complejidad $O(n \lg k)$.

Deberá usar heap/fila de prioridades. Considere que tiene a la mano (no es necesario implementar) las funciones y operaciones vistas en clase (tanto para Min-Heap o Max-Heap). No es necesario escribir pseudocódigo (si ve que es necesario hágalo). **Basta dar una idea clara de su algoritmo y su complejidad.**