C=4.5

Examen I

Análisis y Diseño de Algoritmos I

Universidad del Valle

Mayo de 2023

Nombre: Brandon Calderón Prieto Código: 2125974

(El examen consta de 7 preguntas. Cada pregunta desde la pregunta 1 a la 5 vale 1 punto, las preguntas 6 y 7 valen 2 puntos cada una . Donde el máximo puntaje en el examen es 9)

1) Indique cuál de las siguientes secuencias de funciones está ordenada ascendentemente según el orden de crecimiento.

Seleccione una:

a. (log n)*(log n), (log (3*$n^{0.5}$)), (log n*n), (log (log n)), log (n^n), $n^{0,5}$*log n , n^3 , n!

b. (log (log n)), (log (3*$n^{0,5}$)), (log n*n), (log n)*(log n), $n^{0,5}$*log n, n^3, log (n^n), n!

c. (log (3 * $n^{0,5}$)), (log (log n)), (log n*n), (log n)*(log n), log (n^n), $n^{0,5}$*log n, n^3 , n!

d. (log (log n)), (log (3*$n^{0,5}$)), (log n*n), (log n)*(log n), $n^{0,5}$*log n, log (n^n), n^3 , n!

2) Una y sólo una de las siguientes afirmaciones es verdadera, indique cuál es:

Seleccione una:

a. n^2 + (log n) es Θ(n^2 (log n))

b. Si f(n) es Θ (n^2) y g(n) es Θ (n^2), entonces f(n) * g(n) es O(n^3)

c. Si f(n) es O(n^2) y g(n) es O(n), entonces f(g(n)) es O(n^3)

d. Si f(g(n)) es O(n^3)), entonces f(n) es O(n^2) y g(n) es O(n).

3) Considere un arreglo con n elementos [a1, a2, a3, . . . , an] tal que los valores del arreglo cuya posición (índice) es impar vale 2 y los valores del arreglo cuya posición (índice) es par vale 3 (es decir el arreglo se vería así: [2 3 2 3 2 3 ....]). Seleccione el algoritmo de ordenamiento que tendrá el mayor orden de complejidad para dicho tipo de entrada:

Seleccione una opción:

a. InsertionSort b. Quicksort c. Mergesort

4) Empareje cada orden de crecimiento de la izquierda con la correspondiente ecuación de recurrencia en la derecha

a) O($n^{1/2}$)
b) Ω(n^4)
c) Θ (n^3)

a T(n) = T(n/3) + Θ (1)
c T(n) = $log\ n^n$ + 8T(n/2) 8T(
b T(n) = n^4 + 16T(n/2)

$n^{\log_2 16} = n^4 = n^4$ $n^{\log_2 8}$

5) Analice que ocurriría si el Merge-Sort visto en clase se modificara y en lugar de dividirse en dos mitades, se dividiera en 3 partes del mismo tamaño (las 3 partes unidas corresponden a todo el arreglo). En particular indique cuál sería la ecuación de recurrencia del Merge-Sort en el peor de los casos, haga la respectiva explicación de la ecuación (Asuma que la cantidad de elementos del arreglo es potencia de 3).

6) Considere el siguiente algoritmo:

(Asuma que A es un arreglo de números *positivos con* al menos dos elementos.)

```
1.    Another-sort(A,p,r)   p: inicion  r: fin
2.       if A[p] > A[r]
3.          exchange A[p] with A[r]
4.       if p + 1 < r   hayan dos elementos
5.          k = ⌊(r-p+1)/3⌋
6.          Another-sort(A,p,r-k)
7.          Another-sort(A,p+k,r)
8.          Another-sort(A,p,r-k)
```

Responda las siguientes preguntas:

a) Defina la ecuación de recurrencia que modele el peor caso (puede considerar la línea 2 como operación básica); resuelva la ecuación y precise el orden de complejidad de dicha solución en términos de θ

b) Compare el orden de complejidad en el peor caso de este algoritmo con respecto a los de los algoritmos de ordenamiento vistos en clase; indique qué tan eficiente es este algoritmo (en el peor caso) con respecto a cada uno de los vistos en clase.

6

7) Considere el siguiente seudocódigo:

(Asuma que A es un arreglo de números *positivos con* al menos dos elementos.)

```
1.Sort(A[1..n])
2.     for(j = 1; j <= n; j = j+1)
3.        for(i = 1; i <= n-1; i = i+1)
4.           if A[i] > A[i+1]
5.              intercambiar(A[i], A[i+1])
```

[2,4,3,1,5]

Responda las siguientes preguntas (Asuma que el arreglo de entrada puede tener elementos repetidos):

a) Considerando la línea 5 como básica, determine la complejidad tanto en el mejor como en el peor de los casos. ¿Cómo serían dichas entradas? Explique sus respuestas.

b) ¿En caso de que el algoritmo sea correcto, es estable? Explique su respuesta. (Si considera que el algoritmo no es correcto, no necesita determinar su estabilidad).