Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2133 — Estructuras de Datos y Algoritmos
2022 - 1

Pauta Interrogación 1

Pregunta 1

Existen varios algoritmos de ordenación conocidos. aquí algunos ejemplos:

1. QuickSort:
   a) complejidad O(n^2)
   b) mejor caso, O($nlog(n)$) (mejor caso es cuando se eligen los elementos medios a cada iteración del algoritmo). peor caso O(n^2), cuando se elijen los elementos de los extremos a cada iteración del algoritmo.
2. MergeSort:
   a) complejidad O($nlog(n)$)
   b) mejor caso O($nlog(n)$), no hay mejor ni peor caso. todos tienen complejidad similar peor caso O($nlog(n)$)

   Ojo: Mergsort debe poseer modificacones considerables para servir en el problema

Nota: Ejecutar 2 insertionsort consecutivos no soluciona el problema ya que desordenaria la lista original
Distribución puntaje

a. definir el algoritmo (0,6)
b. calcular la complejidad (0,7)
c. ver mejor y peor caso (0,7)

OBS: la complejidad calculada en b debe ser la del peor caso, no la promedio

Pregunta 2

a)

El algoritmo *Selection Sort* propone que para una secuencia inicial de datos, A:

1. Definir una secuencia ordenada, B, inicialmente vacía
2. Buscar el menor dato x en A
3. Sacar x de A e insertarlo al final de B
4. Si quedan elementos en A, volver a 2

Por lo tanto, aplicado al arreglo de nombres y edades, la lista B se construiría como se muestra a continuación:
I_1 = [(él, 2)]
I_2 = [(él, 2), (vosotras, 5)]
I_3 = [(él, 2), (vosotras, 5), (ella, 10)]
I_4 = [(él, 2), (vosotras, 5), (ella, 10), (ustedes, 10)]
I_5 = [(él, 2), (vosotras, 5), (ella, 10), (ustedes, 10), (todos, 12)]
I_6 = [(él, 2), (vosotras, 5), (ella, 10), (ustedes, 10), (todos, 12), (vosotros, 15)]
I_7 = [(él, 2), (vosotras, 5), (ella, 10), (ustedes, 10), (todos, 12), (vosotros, 15), (tú, 20)]
I_8 = [(él, 2), (vosotras, 5), (ella, 10), (ustedes, 10), (todos, 12), (vosotros, 15), (tú, 20), (yo, 23)]
I_9 = [(él, 2), (vosotras, 5), (ella, 10), (ustedes, 10), (todos, 12), (vosotros, 15), (tú, 20), (yo, 23), (nosotros, 25)]
I_{10} = [(él, 2), (vosotras, 5), (ella, 10), (ustedes, 10), (todos, 12), (vosotros, 15), (tú, 20), (yo, 23), (nosotros, 25), (ellos, 25)]
I_{11} = [(él, 2), (vosotras, 5), (ella, 10), (ustedes, 10), (todos, 12), (vosotros, 15), (tú, 20), (yo, 23), (nosotros, 25), (ellos, 25), (ellas, 30)]
Nota: El algoritmo que utiliza swap no deja el mismo orden, por lo que no se considera correcto.

Distribución puntaje
Puntaje total parte a: **1 pts.**

1 Ejecutar el algoritmos correctamente (con el ejemplo o mostrandos los pasos a seguir) sin swap (**0,75 pts.**)

2 Mostrar que al ejecutar el algorimo es estable en el ejemplo (**0,25 pts.**)

b)

Distribución puntaje
Puntaje total parte b: **1 pts.**

1 Crear un algoritmos que ordene que sea derivado del algoritmo selection sort (que ordene correctamente) (**0,3 pts.**)

2 El algoritmo creado es estable y lo demuestra (**0,7 pts.**)

Pregunta 3

(A) Debe describir un algoritmo de dividir y vencer para realizar los calculos de las sumas. (1,1pto)

(B) Cualquier estructura que almacene la suma y ambos datos es suficiente (0,2pts)

(C) Cualquier algoritmo que tenga en consideración la estructura anterior es valida (0,7pts)

Pregunta 4

si interpretación de la A mas como un genérico inductivo, que termina (demostración que funciona)
BI arreglos de largo 2, al ordenar por max y luego min se tienen los elementos del medio si se tiene elem1, elem2, elem3, elem4; resultando (elem2 + elem3)/2 que es la mediana.

HI para 2 arreglos de largo $k > 2$ ordenados el algoritmo funciona

TI para 2 arreglos de largo k+1 ordenados el algoritmo funciona por demostrar lo de arriba, se tienen 2 casos $m_1 = m_2$ y que sean distintos, si $m_1 = m_2$ se encontró la mediana, en caso de ser distintos se aplica el algoritmo donde se reduce el espacio a la mitad, por HI funciona.

Explicación de porque retorna bien la mediana si m_1 y m_2 son distintos, como un elemento es mayor que el otro y ambos subarreglos se encuentran ordenados se deduce que el resto de elementos para la derecha de una mediana son mayores y los de su izquierda son menores, entonces las llamadas recursivas abarca espacios con sentido para llegar a elementos del medio (sigan como mayores o menores respecto a mediana)

si solo se ve que la cosa termine

a. 2 casos para el término:

   a) **0.25pts** $m_1 = m_2$, explicación directa da el mismo término

   b) **0.75pts** Los arreglos resultantes ambos son de largo 2
   una breve explicación de como llegar a este caso, se tiene n iteraciones donde las medianas m_1 y m_2 no fueron iguales, en cada llamado recursivo disminuye el espacio de búsqueda (0 a mediana, y medina término del subarreglo), "la mitad de dicho espacio", así hasta llegar a un largo 2.

b. **1pts** Del pseudo

```
Median(AR1, AR2):
    m1 = AR1[len(AR1)/2]
    m2 = AR1[len(AR2)/2]

    if (m1 > m2)
    SubArr1 = AR1[0:m1]
    SubArr2 = AR2[m2: ]
    Median(SubArr1, SubArr2)
[...]
```

Se puede apreciar que el espacio de búsqueda tiene un comportamiento $n, \frac{n}{2}, \frac{n}{4}, ..,1$ de esto se puede concluir que se comporta como $log_2 n$, esto sale directo como los datos vienen bien y no hay que realizarles otro tipo de operación.