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

IIC2133 — Estructuras de Datos y Algoritmos 1'2023

Interrogación 1

3 de abril de 2023

Condiciones de entrega. Debe entregar solo 3 de las siguientes 4 preguntas.

Nota. Cada pregunta tiene 6 puntos (+1 punto base). La nota es el promedio de las 3 preguntas entregadas.

Uso de algoritmos. En sus diseños puede utilizar llamados a cualquiera de los algoritmos vistos en clase. No debe demostrar la correctitud o complejidad de estos llamados, salvo que se especifique lo contrario.

1. Análisis de algoritmos

Para ordenar una secuencia de datos implementada como arreglo A se propone el algoritmo `GnomeSort`, apodado *stupid sort* debido a que para ciertos inputs puede realizar una cantidad de iteraciones mayor a n.

input : Arreglo $A[0, \ldots, n-1]$, largo $n \geq 2$

```
GnomeSort(A, n):
1    p ← 0
2    while p < n :
3        if p = 0 ∨ A[p] ≥ A[p − 1] :
4            p ← p + 1
5        else:
6            A[p] ⇌ A[p − 1]
7            p ← p − 1
```

(a) [3 ptos.] Demuestre que luego de la k-ésima iteración del **while** de `GnomeSort`, $A[0 \ldots p-1]$ está ordenada. Note que p no necesariamente coincide con el número de iteraciones que se han ejecutado hasta el momento. *Pista:* use inducción sobre k.

(b) [1 pto.] Demuestre que `GnomeSort` termina, es decir, que se logra $p = n$ al término de alguna iteración del **while**.

(c) [1 pto.] Determine la complejidad de tiempo y espacio de `GnomeSort` en el peor caso.

(d) [1 pto.] ¿Tiene un mejor caso? Justifique y en caso afirmativo, determine su complejidad.

2. Diseño de algoritmos

Su compañía empleadora ha sido contratada para actualizar el sistema de control de un aeropuerto. Le corresponde modificar el sistema que selecciona el próximo avión que debe aterrizar, que actualmente utiliza una cola FIFO implementada en un arreglo $A[0..n-1]$ en que cada avión P_i que llega a la cola trae asociado un número de secuencia S_i que lo identifica. El valor n es adecuado para el tráfico del aeropuerto.

Además de S_i, cada avión en la cola Q cuenta con su autonomía de vuelo F_i (una medida de cuánto tiempo puede permanecer en el aire sin caer) y el número de pasajeros que transporta T_i. La autonomía F_i se actualiza para todos los aviones P_i de la cola Q cada un minuto y esta autonomía cambia en tasas diferentes para cada avión P_i. Los aviones nuevos que llegan a la cola Q ingresan con su secuencia S_i, autonomía F_i y número de pasajeros T_i definidas.

Para los siguientes escenarios, proponga una algoritmo que resuelva lo solicitado.

(a) [3 ptos.] Se busca disminuir el riesgo de caída, por lo que se requiere ordenar la cola Q primero por la autonomía F_i de cada avión P_i en la cola Q. Para autonomías iguales se debe ordenar por número de pasajeros T_i y para los casos con igual autonomía y número de pasajeros se debe ordenar por la secuencia S_i del avión. El siguiente avión a aterrizar debe quedar en la cabeza de la cola Q, i.e. en la posición $Q[0]$. Escriba en pseudo código el algoritmo `sortTrafico`(Q, i, f) para ordenar la cola Q según lo indicado.

(b) [3 ptos.] Para permitir una forma eficiente de desviar tráfico aéreo a otros terminales se requiere encontrar en la cola Q el avión P_i con la menor autonomía mayor que un valor D para desviarlo a otro aeropuerto. Escriba en pseudo código el algoritmo `desviaTrafico`(Q, i, f, D) que retorne la posición en la cola Q del avión P_i que cumple lo solicitado.

3. Estrategias algorítmicas

Considere dos arreglos $A[0 \dots n-1]$ y $B[0 \dots n-1]$ ordenados y del mismo tamaño. Asuma que n es impar.

(a) [3 ptos.] Proponga el pseudocódigo de un algoritmo que utilice la estrategia dividir para conquistar que retorne la mediana del arreglo $C[0 \dots 2n-1]$ que se obtendría al combinar los arreglos A y B. Su algoritmo debe tener una complejidad asintótica mejor que lineal en el peor caso.

(b) [2 ptos.] Determine justificadamente la complejidad de tiempo en el peor caso para su algoritmo.

(c) [1 pto.] ¿Su algoritmo tiene un mejor caso? Justifique, y en caso afirmativo, entregue la complejidad de tiempo en el mejor caso.

4. Modificación de algoritmos

Se tiene un arreglo $A[0 \dots n-1]$ originalmente ordenado, tal que varios de sus elementos fueron desordenados. Se sabe que en este minuto, al menos un 80 % de sus elementos están en su posición correcta ordenada. En su empresa se piensa usar `Quicksort` para ordenar nuevamente A.

(a) [2 ptos.] Un(a) ingeniero(a) de software (SI) propone que la elección del pivote sea la mediana entre los valores en los índices 0, $\lfloor n/2 \rfloor$ y $n-1$ del arreglo A. Indique en qué líneas o zona debe hacer la modificación en la versión de `Quicksort` vista en clases y especifique el fragmento de pseudocódigo nuevo para incorporar el cambio propuesto.

(b) [2 ptos.] Un segundo SI propone que cuando el subarreglo a ordenar sea de tamaño 20 o menos se utilice `InsertionSort`, ya que si está ordenado, este es su mejor caso. Indique en qué líneas o zona debe hacerse la modificación en la versión de `Quicksort` vista en clases y especifique el fragmento de pseudocódigo nuevo para incorporar el cambio propuesto.

(c) [2 ptos.] Un tercer SI propone reemplazar `Quicksort` por `InsertionSort` para todo el proceso, afirmando que si el arreglo A está 80 % ordenado, el desempeño de `InsertionSort` se debe parecer más a su mejor caso, que es mejor que `Quicksort`. ¿Es correcto lo propuesto? Justifique.