Estructuras de Datos y Algoritmos – iic2133

Control 1

20 de marzo, 2019

Nombre: ____

1) Escribe el algoritmo *quicksort3*, que, en lugar de particionar el arreglo A en dos, como lo hace *quicksort*, lo particiona en tres: datos menores que el pivote, datos iguales al pivote, y datos mayores que el pivote. Puedes suponer que las particiones van a parar a listas diferentes o bien al mismo arreglo —especifica. Usa una notación similar a la usada en las diapositivas.

Solución: Pueden haber muchos algoritmos correctos. Las distribución de puntaje se hizo de la siguiente forma:

- [1 pto.] Que se haya hecho en pseudocódigo y sin funciones específicas de ningún lenguaje, es decir, usando la notación que se usa en las diapositivas.
- [3 ptos.] Haya separación correcta entre los menores, iguales y mayores y que se vaya reordenando la lista según corresponda.
- [1 pto.] La nueva implementación de partition retorne dos pivotes.
- [1 pto.] Que el algoritmo termine de forma correcta.

2) Considera el siguiente algoritmo de ordenación, para ordenar el arreglo A de largo n :

$\boldsymbol{sort}(A, n)$:

  $\boldsymbol{for}\ \ g \in \{5,\ 3,\ 1\}$:

    $\boldsymbol{for}\ i \in [g, n[$:

      $j \leftarrow i$

      $\boldsymbol{while}\ (j \geq g\ \wedge\ A[j] < A[j-g])$:

        $A[j] \rightleftarrows A[j-g]$

        $j \leftarrow j - g$

Demuestra que este algoritmo es correcto según los **dos** criterios vistos en clases.

Solución: El algoritmo es finito y es correcto.

- Es finito:
  [0.5 pto.] El primer “for” termina ya que recorre un conjunto finito

  [0.5 pto.] El segundo “for” termina ya que recorre un conjunto finito

  [2 pto.] La segunda condición no podemos predecirla. Por otro lado, como J es monótonamente decreciente y G es finita, tenemos que el “while” siempre va a terminar ya que la primera condición se va a romper (J >= G).

- Es correcto:
  - ❑ Opción 1:

  [3 pto.] Con G = 1, el algoritmo es igual a Insertion Sort. Como sabemos que Insertion Sort es correcto y siempre se va a ejecutar el algoritmo con G = 1, el algoritmo Sort() es correcto.

- [ ] Opción 2:

Solo considerando con G = 1. Los otros valores de G no aportan a la demostración.

Invariante: Luego de la iteración i, el arreglo está ordenado hasta el índice i.

Lo demostramos por Inducción.

[1 pto.] Caso Base. i = 1. El primer elemento del arreglo.

Un arreglo de largo uno está siempre ordenado.

[0.5 pto.] Hipótesis Inductiva. Tras la iteración i, A está ordenado hasta el índice i.

[1.5 pto.] En la iteración i+1 existen dos casos:

- a_i <= a_{i+1} -> A está ordenado hasta el índice i+1
- a_i > a_{i+1} ->
  Llamemos a_j= a_{i+1}

  Como ya estaba ordenado hasta a_i, se tiene

  a_1 <= a_2 <= … <= a_{i-1} <= a_i > a_j

  en cada paso el elemento a_j se cambia de posición con el anterior, dejando ordenado a ambos lados:

  a_1 <= a_2 <= … > a_j <= … <= a_{i-1} <= a_i -> while continúa.

  a_1 <= a_2 <= … <= a_j <= … <= a_{i-1} <= a_i -> while termina, y los elementos están ordenados hasta el índice i+1.

Por inducción, después de la iteración n el arreglo está ordenado hasta el índice n, por lo tanto, está completamente ordenado.