

Soluciones del tutorial 11

Para los códigos se ha utilizado Java™ software.

Problemas del tutorial

Problema 1

- Falso. En una ordenación por inserción, los elementos se seleccionan de uno en uno de un conjunto sin ordenar y se insertan en una lista ordenada.
- Verdadero
- Verdadero
- Falso. Algunas implementaciones de la *collection* está desordenadas y otras ordenadas.
- Verdadero

Problema 2

Parte A

Dada la siguiente lista

10 20 30 40 25 35

considere el algoritmo de ordenación por inserción (insertion sort) tratado en clase

- ¿Cuántas comparaciones requeriría este array por parte de *insertion sort* para realizar una ordenación completa?

8

- Muestre el estado del array después de cada comparación

10	20	30	40	25	35	(3-1)
10	20	30	40	25	35	(3-2)
10	20	30	40	25	35	(3-3)
10	20	30	25	40	35	
10	20	25	30	40	35	
10	20	25	30	40	35	(3-4)
10	15	20	25	35	40	
10	15	20	25	35	40	(3-5)

- En este método de inserción por ordenación tenemos dos bucles. En su respuesta al apartado (2), marque el estado del array al final de cada iteración del bucle más remoto **(3-1) => (3-5)**

Parte B

Simule la ejecución de *quicksort* en el siguiente array, utilizando el algoritmo *Quicksort* más sencillo tratado en clase para obtener así los dos primeros subarrays:

6 12 18 9 15 3

Pivote = 6

6 12 18 9 15 3

L H

3 12 18 9 15 6

L H

3 12 18 9 15 6

H L

|

Dos subarrays son: {3} y {12, 18, 9, 15, 6}

Problema 3

{Alice=2, Andrew=1, Ben=1, Jennifer=1, Julia=1, Melissa=1, Steve=3}

Ejercicios de diseño

Parte 1.

A.

Respuesta: b.

B.

Si se utiliza el número de identificación de los estudiantes como clave, sólo es necesario realizar $O(\log n)$ número de comparaciones para hallar la calificación de un estudiante en particular.

Si se utiliza la calificación de los estudiantes como clave, es necesario realizar $O(n)$ número de comparaciones para hallar la calificación de un estudiante en particular, ya que debemos buscar por todo el árbol para encontrar un estudiante con un número de identificación que se corresponda.

Parte 2.

Utilizaremos los números de identificación de los estudiantes como claves para las tablas hash y los árboles de búsqueda binaria. Las tablas hash nos proporcionarán el mejor resultado de consulta (tiempo de consulta constante), suponiendo que los números de identificación de los estudiantes estén distribuidos de manera uniforme.

Parte 3.

Respuesta: b.

Parte 4.

Parte 5.

A.

```
public class Student {
    String name;
    int id;
    int grade;
    public int hashCode() {
        return id;
    }
    public boolean equals (Object s) {
        //inserte su código abajo
        Student stu = (Student) s;
        if (stu.id == id) {
            return true;
        } else {
            return false;
        }
    }
}
```

B.

Respuesta: 100, 200, 300, 400, 500