Prácticas de Algorítmica.
3º de Grado en Ingeniería Informática.
Curso 2016-2017.
Práctica 3.

Objetivos.

Con esta práctica se pretende que el alumno implemente un algoritmo basado en la técnica divide y vencerás. Se proponen varios ejercicios de distinto nivel y el alumno ha de seleccionar uno de ellos. La puntuación máxima que se podrá obtener en cada uno de los ejercicios figura en su enunciado.

Enunciado 1:

Implementar el algoritmo recursivo para obtener el máximo y el mínimo elemento de un conjunto. El conjunto se implementará en una clase usando una plantilla para definir el tipo de elementos del conjunto y como mínimo contendrá métodos para insertar un elemento en el conjunto, borrar un elemento dada su posición y acceder a un elemento por su posición. En el programa de prueba se usará un conjunto de enteros que se rellenará aleatoriamente con valores comprendidos entre 1 y 10000. En la salida ha de aparecer el máximo y el mínimo elemento del vector así como el número de comparaciones realizadas. ***Puntuación máxima = 8.***

Enunciado 2:

Implementar el algoritmo, basado en el quicksort, para obtener los k-menores elementos de un conjunto. El conjunto se implementará en una clase usando una plantilla para definir el tipo de elementos del conjunto y como mínimo contendrá métodos para insertar un elemento en el conjunto, borrar un elemento dada su posición y acceder a un elemento por su posición. En el programa de prueba se usará un conjunto de enteros que se rellenará aleatoriamente con valores comprendidos entre 1 y 10000. En la salida del programa ha de aparecer el número de llamadas recursivas realizadas y se preguntará al usuario si se desea mostrar el vector resultante, resaltando los k-menores. ***Puntuación máxima = 8.***

Enunciado 3:

Implementar el algoritmo de ordenación quicksort para un conjunto de elementos. El conjunto se implementará en una clase usando una plantilla para definir el tipo de elementos del conjunto y como mínimo contendrá métodos para insertar un elemento en el conjunto, borrar un elemento dada su posición y acceder a un elemento por su posición. En el programa de prueba se usará un conjunto de enteros que se rellenará aleatoriamente con valores comprendidos entre 1 y 10000. El programa tendrá las siguientes opciones:

1. *Usando el método de ordenación por inserción binaria cuando el subconjunto a ordenar, resultante en las sucesivas iteraciones del quicksort, tenga un número de elementos menor o igual a* χ*, siendo* χ *un parámetro de la ordenación. En este caso el usuario introducirá el número de elementos y el valor de* χ
2. *Realizar pruebas siguiendo el enunciado de la opción 1, fijando un valor del número de elementos y para ese valor, analizar cual es el valor de* χ *que minimiza el tiempo de ordenación.*
3. *Usando el cálculo de la mediana de* χ *elementos seleccionados aletoriamente para determinar el pivote en el algoritmo de partición. Esta forma de seleccionar el pivote se hará siempre y cuando el número de elementos del subvector a tratar sea igual o superior a un límite l dado por el usuario. En este caso el usuario introducirá el número de elementos y el valor de* χ*.*
4. *Realizar pruebas siguiendo el enunciado de la opción 3, fijando un valor del número de elementos y del límite l, haciendo variar el valor de* χ*.*

Puntuación máxima = 9.

Enunciado 4:

Implementar el algoritmo de ordenación por fusión para un conjunto de n *elementos. Cuando el subconjunto a*

ordenar, resultante en las sucesivas iteraciones del método, tenga un número de elementos menor o igual a ***x****, siendo* ***x*** *un valor introducido por el usuario hay que usar el método de inserción directa. El conjunto se implementará en una clase usando una plantilla para definir el tipo de elementos del conjunto y como mínimo contendrá métodos para insertar un elemento en el conjunto, borrar un elemento dada su posición y acceder a un elemento por su posición. En el programa de prueba se usará un conjunto de enteros que se rellenará aleatoriamente con valores comprendidos entre 1 y 10000.*

El programa tendrá las siguientes opciones:

1. *Introducir el valor del número de elementos y de x y obtener el tiempo de la ordenación.*
2. *Realizar pruebas siguiendo el enunciado de la opción 1, fijando un valor del número de elementos y para ese valor, analizar cual es el valor de x que minimiza el tiempo de ordenación.*

Puntuación máxima = 9.

Enunciado 5:

Implementar el algoritmo para multiplicar enteros grandes. Para ello, el alumno ha de implementar una clase ***Entero*** *que ha de tener sobrecargados el operador *, el operador + (es necesario para el producto) y los operadores de entrada/salida (>> y <<). El número de dígitos máximo pueden tener dos números para multiplicarse de forma directa lo introducirá el usuario. El profesor suministrará una serie de algoritmos implementados en C y que deberán ser adaptados dentro de la clase* ***Entero****, que facilitarán la implementación de dicha clase.*

El programa tendrá un menú en el que se seleccione la operación que se desea realizar (suma o producto) y en cada operación se pedirán los números a utilizar. ***Puntuación máxima = 10.***

Enunciado 6:

Implementar el algoritmo de la exponenciación, en las tres variantes explicadas en clase para obtener la potencia de una matriz. El tipo matriz se implementará usando plantillas y en las pruebas se usará una matriz de enterors y estos se rellenarán con valores aleatorios enteros comprendidos entre -2 y 2. ***Puntuación máxima = 9.***

Fecha de comienzo: 27 de Octubre de 2016.
Fecha máxima de entrega: 10 de Noviembre de 2016.