Prácticas de Algorítmica.
3º de Grado en Ingeniería Informática.
Curso 2018-2019.

Práctica 1 (primera parte).

Objetivos.

Con esta práctica se pretende que el alumno se familiarice con el cálculo de tiempos de ejecución de un determinado algoritmo en función del tamaño del ejemplar y hacer una estimación empírica de esos tiempos en función de dicho tamaño. Para ello, el alumno deberá implementar un programa en C++ donde se calculen los tiempos de ejecución de dos algoritmos de ordenación y posteriormente se estime, utilizando un enfoque híbrido, la complejidad computacional de esos métodos.

Enunciado.

Implementad en C++ un programa que permita obtener los tiempos de ejecución de dos métodos de ordenación y a partir de esos tiempos estimar su complejidad computacional usando un enfoque híbrido. Para ello se implementará un método no sofisticado y otro sofisticado. Los elementos del conjunto a ordenar se almacenarán en un vector de la STL.

De entre los métodos no sofisticados se seleccionará uno de los siguientes, en función del último dígito del dni (suponemos que el dígito es d8):

1. *Método de inserción. (si d8 mod 5 == 0)*
2. *Método de inserción binaria. (si d8 mod 5 == 1)*
3. *Método burbuja. (si d8 mod 5 == 2)*
4. *Método de la sacudida. (si d8 mod 5 == 3)*
5. *Método de selección. (si d8 mod 5 == 4)*

De entre los métodos sofisticados se seleccionará uno de los siguientes, en función del penúltimo dígito del dni (suponemos que el dígito es d7):

6. *Método Shell. (si d7 mod 4 == 0)*
7. *Método de ordenación por montículo. (si d7 mod 4 == 1)*
8. *Método quicksort. (si d7 mod 4 == 2)*
9. *Método por contabilización de frecuencias. (si d7 mod 4 == 3)*

Para realizar las pruebas y calcular los tiempos, se procederá como sigue:

1. *El programa mostrará un menú en el que se seleccionará uno de los dos métodos.*
2. *Una vez seleccionado el método, el usuario introducirá, el valor mínimo del número de elementos del vector, el valor máximo, el incremento del valor del número de elementos y el número de veces que se repetirá la ordenación para cada valor del número de elementos. Por ejemplo, si el mínimo es 1000, el máximo es 5000, el incremento es 100 y el número de repeticiones es 50, se probará primero con 1000 elementos, repitiendo el experimento 50 veces, después con 1100 y se repite 50 veces y así hasta llegar a 5000.*
3. *El vector será de elementos de tipo entero y se rellenerá aleatoriamente con valores entre 0 y 9999. Para ello usad la función* ***void rellenarVector(vector<int> &v);***
4. *Si el número de repeticiones es por ejemplo 50, para cada valor de* ***n*** *entre el mínimo y el máximo se harán 50 pruebas de tal forma que si n=1200 se harán 50 pruebas para 1200 elementos. Posteriormente se calculará la media de tiempos de esas 50 pruebas y ese será el valor correspondiente de tiempo empleado para ese* ***n=1200****. Esto se hará para todos los posibles valores de* ***n****. En la documentación se adjunta un*

ejemplo para ver como se calculan los tiempos.

5. *Implementad una función que, utilizando asertos, compruebe que la ordenación se ha realizado correctamente (**bool estaOrdenado(const vector <int> v);**)*
6. *Almacenar en un fichero de texto los valores de n empleados (primera columna) y los tiempos correspondientes a esos valores de n (segunda columna)*

Fecha de comienzo: 13 de setiembre de 2018.
Fecha de Entrega: 4 de Octubre de 2018.