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, aportando la función de tiempo en función del tamaño del vector a ordenar.

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****.*

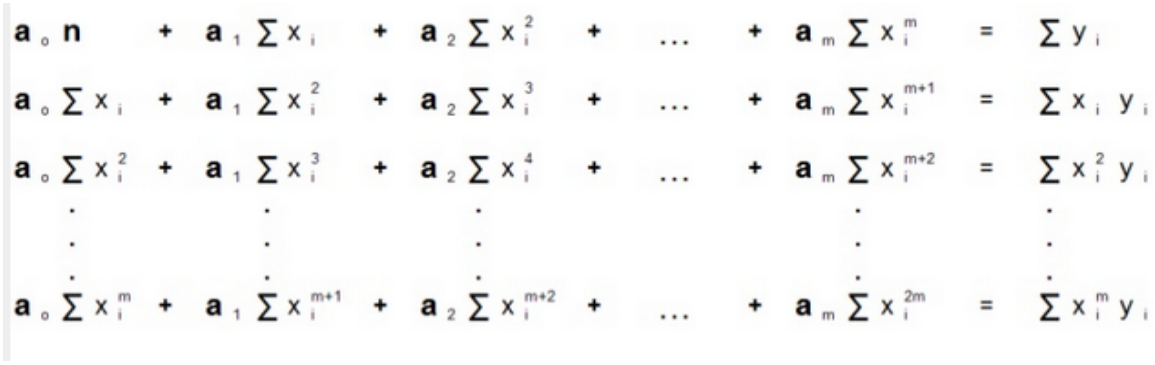
*Esto se hará para todos los posibles valores de **n**. En la documentación se adjunta un ejemplo para ver como se calculan los tiempos.*

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)*
7. *Una vez obtenidos y almacenados los valores de n y los valores de tiempo, se realizará un ajuste a una curva por mínimos cuadrados teniendo en cuenta que la variable independiente es el tamaño del ejemplar (n) y que la variable dependiente es el tiempo de ejecución. Al final del enunciado se explica como realizar el ajuste.*
8. *Para los métodos no sofisticados se probará ajustar un polinomio de grado 2. **t(n) =a0 + a1*n + a2*n$^{2.}$***
9. *Para los métodos sofisticados se probará ajustar una curva del tipo. **t(n) =a0 + a1*n*log(n),** excepto para el método de contabilización de frecuencias que se probará un ajuste del tipo **t(n) =a0 + a1*n.** El primero de estos ajustes se puede convertir en lineal haciendo el cambio z = nlog(n).*
10. *Para todos los casos la ecuación de la curva obtenida nos proporciona la complejidad computacioal de ese algoritmo usando un enfoque híbrido.*
11. *Ahora, teniendo en cuenta las curvas de ajuste obtenidas en el paso 7, se calcularán los tiempos estimados a partir de dichas curvas para cada uno de los tamaños de ejemplar probados para obtenerlas. De esta forma, al final de este paso, tendremos los tiempos reales de los algoritmos, que son los tiempos obtenidos en el apartado 4, y los tiempos estimados mediante los ajustes por mínimos cuadrados.*
12. *Una vez obtenidos los tiempos estimados se guardarán en un fichero de texto para poder representarlos posteriormente usando el programa **gnuplot** (se suministra un ejemplo de uso)**.** La información se guardará por columnas en el siguiente orden: tamaño del ejemplar, tiempo real y tiempo estimado.*
13. *Para finalizar, el programa ha de mostrar las ecuaciones de las curvas ajustadas por mínimos cuadrados y sus coeficientes de determinación y dar la posibilidad al usuario si quiere hacer una estimación de tiempos para un determinado valor del tamaño del ejemplar, en cuyo caso mostrará el tiempo de ese estimación en días. Esta opción ha de poder repetirse hasta que el usuario introduzca un tamaño de ejemplar igual a 0. Esto es útil cuando el tiempo es muy elevado para un tamaño de ejemplar relativamente grande. Por ejemplo la ordenación por el método burbuja de un vector de 100 millones de elementos tardaría en calcularse varios días,, y mediante la curva ajustada podemos obtener de una forma muy fiable el tiempo que tardaría en calcularse.*

Notas estadísticas y matemáticas:

El coeficiente de determinación se calcula dividiendo la varianza de los tiempos estimados entre la varianza de los tiempos reales. Este valor está comprendido entre 0 y 1, y para que el ajuste sea bueno ha de ser superior a 0,9.

Para estimar los parámetros de un ajuste polinómico de orden m se puede usar el siguiente sistema de acuaciones (http://es.slideshare.net/diegoegas/regresion-polinomial-2512264):



- *En este sistema de ecuaciones las incógnitas son los valores de los a_i. Los valores x_i se corresponden con el tamaño del ejemplar y los valores y_i se corresponden con los tiempos observados. La n que aparece en el primer término de la primera ecuación del sistema es el tamaño de la muestra de tiempos con los que estamos trabajando.*
  Cada sumatorio tendrá tantos sumandos como valores de n se hayan usado para el tamaño del ejemplar. Por ejemplo, si al ordenar se usan valores de n desde 1000 hasta 10000, de 1000 en 1000 (10 valores de n), cada sumatorio tendrá 10 elementos.
- *Se suministra el código objeto en versiones para linux local y ubuntu de 64 bits, de una función para resolver un sistema lineal de ecuaciones, cuyo prototipo es:*

void resolverSistemaEcuaciones(vector < vector < double > > A, vector < vector < double > > B, int n, vector < vector < double > > &X);

donde:
A es la matriz de coeficientes de nxn
B es la matriz de terminos independientes de nx1
n es el orden de las matrices
X es el valor de las variables que se obtienen resolviendo el sistema de orden nx1

Declaracion y reserva de matrices usando el tipo vector de la STL
vector < vector < double > > matrizDatos;
matrizDatos = vector< vector< double > >(filas, vector< double >(columnas)); //Matriz de filas x columnas

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