Prácticas de Algorítmica.
3º de Grado en Ingeniería Informática.
Curso 2019-2020.

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 varios algoritmos para distintos tamaños del ejemplar y posteriormente se estime, utilizando un enfoque híbrido, la complejidad computacional de esos algoritmos, aportando la función de tiempo en función del tamaño del ejemplar.

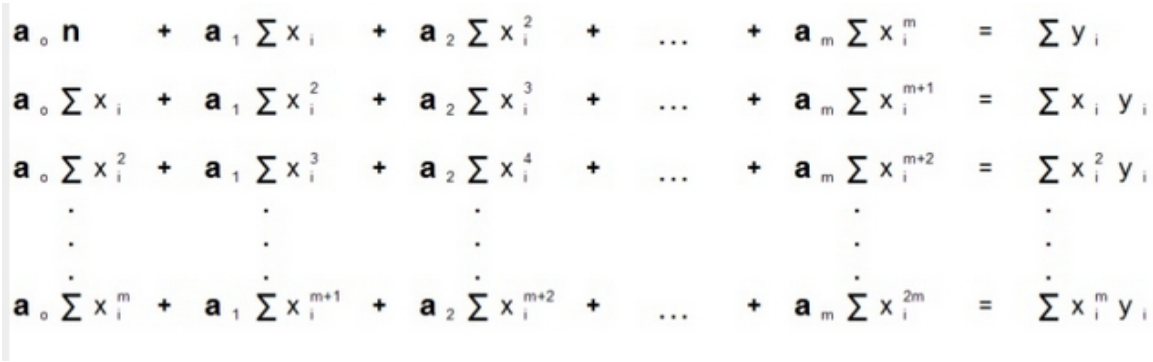
Enunciado de la parte obligatoria (Nota máxima: 8)

1. *Implementar el método de ordenación del quicksort y realizar con dicho método las siguientes pruebas:*
    1. *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).***
    2. ***I****mplementad una función que, utilizando asertos, compruebe que la ordenación se ha realizado correctamente* ***(bool estaOrdenado(const vector <int> &v);)***
    3. *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.*
    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. Los tiempos se almacenarán en un vector de la STL de tipo double.*
    5. *Almacenad en un fichero de texto los valores de n empleados (primera columna) y los tiempos correspondientes a esos valores de n (segunda columna) tiempos reales.*
    6. *Se probará ajustar una curva del tipo.* ***t(n) =a0 + a1*n*log(n).*** *Para ello se usará una función de prototipo* ***void ajusteNlogN(const vector <double> &n, const vector <double> &tiemposReales, double &a0, double &a1;).*** *Este ajuste se puede obtener usando el ajuste polinómico haciendo el cambio z = nlog(n)*
    7. *Ahora, teniendo en cuenta la función ajustada en el paso anterior, se obtendrán los tiempos estimados mediante el ajuste. Para ello usar la función de prototipo* ***void calcularTiemposEstimadosNlogN(const vector <double> &n, const vector <double> &tiemposReales, const double &a0, const double &a1, vector <double> &tiemposEstimados).*** *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.*

8. *Obtener el coeficiente de determinación del ajuste, sabiendo que es la varianza de los tiempos estimados dividida por la varianza de los tiempos reales. Para ello usar la función de prototipo* ***double calcularCoeficienteDeterminacion(const vector <double> &tiemposReales, const vector <double> &tiemposEstimados).***
9. *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.*
10. *Para finalizar, el programa ha de mostrar la ecuación de las curva ajustadas por mínimos cuadrados, su coeficiente 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 de un vector de 100000 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. Para ello usar la función* ***double calcularTiempoEstimadoNlogN(const double &n, const double &a0, const double &a1)***

2. ***Implementar el producto de matrices cuadradas de tipo float y realizad las siguientes pruebas:***
   1. *La matriz será de elementos de tipo double y se rellenerá aleatoriamente con valores entre 0.95 y 1.05. Para ello usad la función* ***void rellenarMatriz(vector<vector <double> &v).***
   2. *Realizar las mismas pruebas que en el caso del vector, teniendo en cuenta las siguientes diferencias:*
      1. *Para un valor de n solo se realiza una prueba.*
      2. *Se ajustará un polinomio de grado 3. La función del ajuste será* ***void ajustePolinomico(const vector <double> &n, const vector <double> &tiemposReales, vector <double> &a).*** *Donde a será un vector de coeficientes del polinomio de ajuste.*
      3. *La función para calcular los tiempos estimados será* ***oid calcularTiemposEstimadosPolinomico(const vector <double> &n, const vector <double> &tiemposReales, const vector <double> &a, vector <double> &tiemposEstimados).***
      4. ***La función para estimar el tiempo a partir de un valor de n será double calcularTiempoEstimadoPolinómico(const double &n, const vector <double> &a)***

   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 reales. 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 fuente 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: 16 de setiembre de 2019
Fecha de Entrega: 7 de Octubre de 2019.