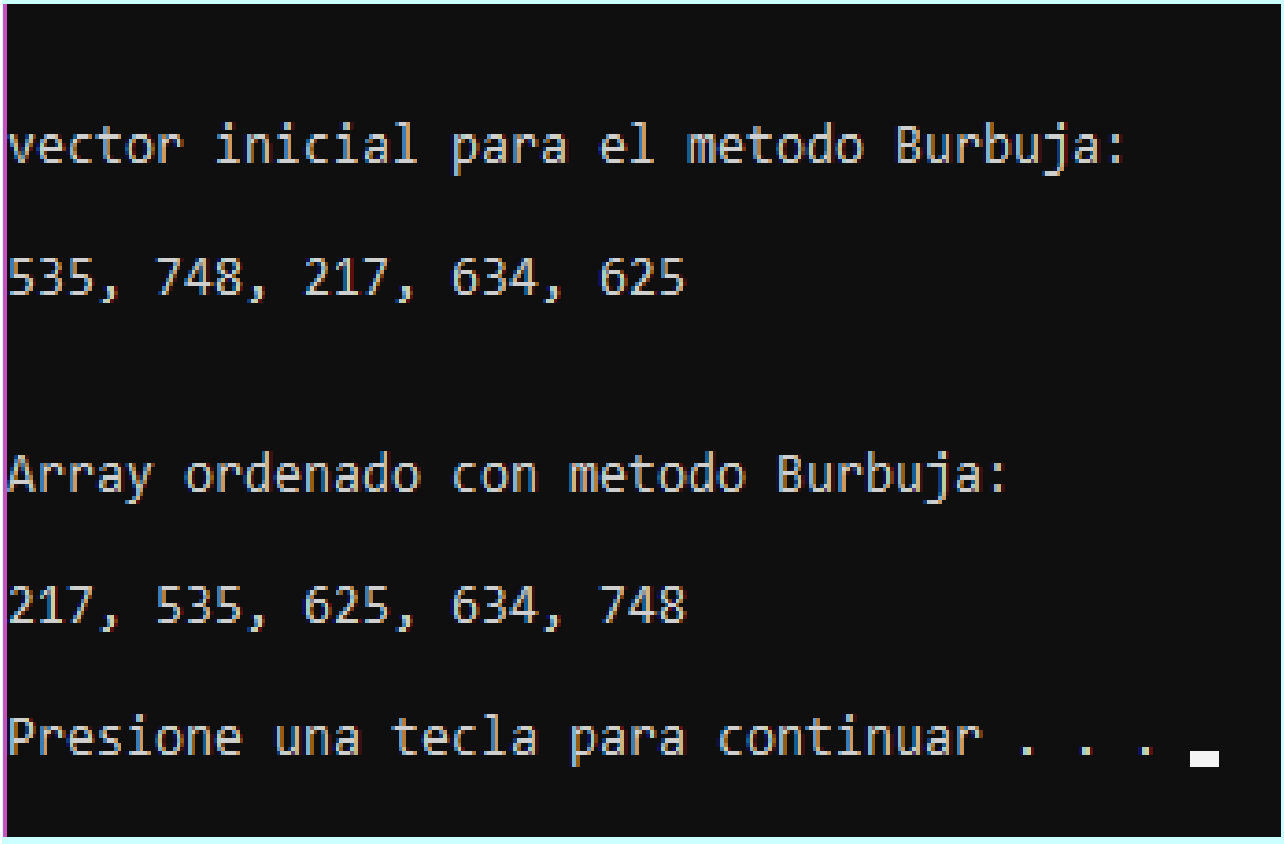
ANÁLISIS DE ALGORITMOS DE ORDENACIÓN



David Prieto Araujo david.prieto707@alu.uhu.es

Índice

Introducción

David Prieto Araujo david.prieto707@alu.uhu.es

El proceso de ordenar elementos (“sorting”) en base a alguna relación de orden es un problema muy frecuente. Nosotros nos centramos en el ordenamiento interno, es decir, cuando los elementos a ordenar están en la memoria interna.

Anteriormente habíamos estudiado el caso medio de los métodos de ordenación Inserción, Burbuja y Selección. Ahora nos vamos a centrar en el método de Quicksort u Ordenamiento rápido.

Este método es de los más utilizados debido a su eficacia y a que no consume memoria física. Este algoritmo se basa en la estrategia “divide y vencerás” donde a un valor del vector lo asigna como pivote y divide los valores menores al pivote a su izquierda y los mayores o iguales a él a su derecha.

Cálculo del tiempo teórico y experimental

Código del método de ordenación Quicksort(c++):

Se ha hecho uso de 3 funciones para implementación de su código:

```
void AlgoritmosOrdenacion::ordenaQuickSort(int v[],int size)
{
    QuickSort(v,0,size-1);
}

void AlgoritmosOrdenacion::QuickSort(int v[],int p,int size)
{
    int q;
    if(p<size)
    {
        q=Partition(v,p,size);
        QuickSort(v,p,q);
        QuickSort(v,q+1,size);
    }
}
```

Donde p seria el pivote asignado que en este caso sería el extremo izquierdo y Partition sería la función donde se va dividiendo el vector en vectores más pequeños hasta ordenar
el vector.

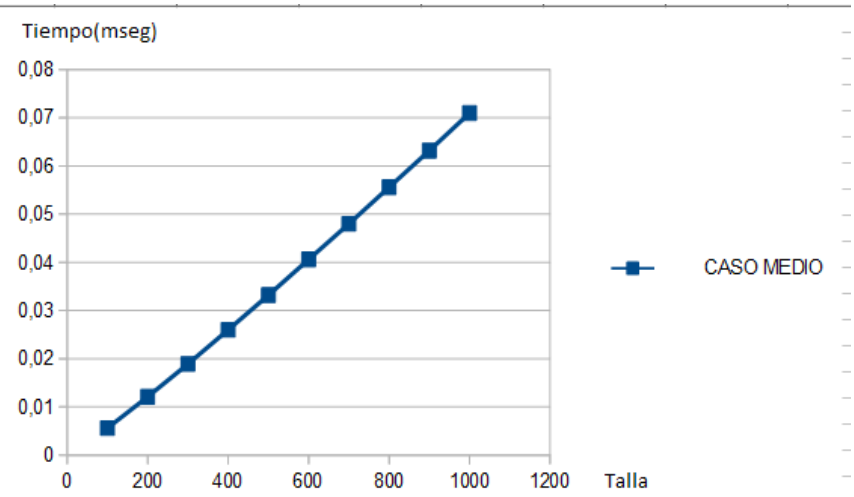
```
int AlgoritmosOrdenacion::Partition(int v[],int p, int r)
{
    int piv,i,j;
    piv=v[p];
    i=p-1;
    j=r+1;
    do
    {
      do
      {
          j--;
      }while(v[j]>piv);
      do
      {
          i++;
      }while(v[i]<piv);
      if(i<j)
      {
          int aux;
          aux=v[i];
          v[i]=v[j];
          v[j]=aux;
      }
    }while(i<j);
    return j;
}
```

Gráficas y tablas

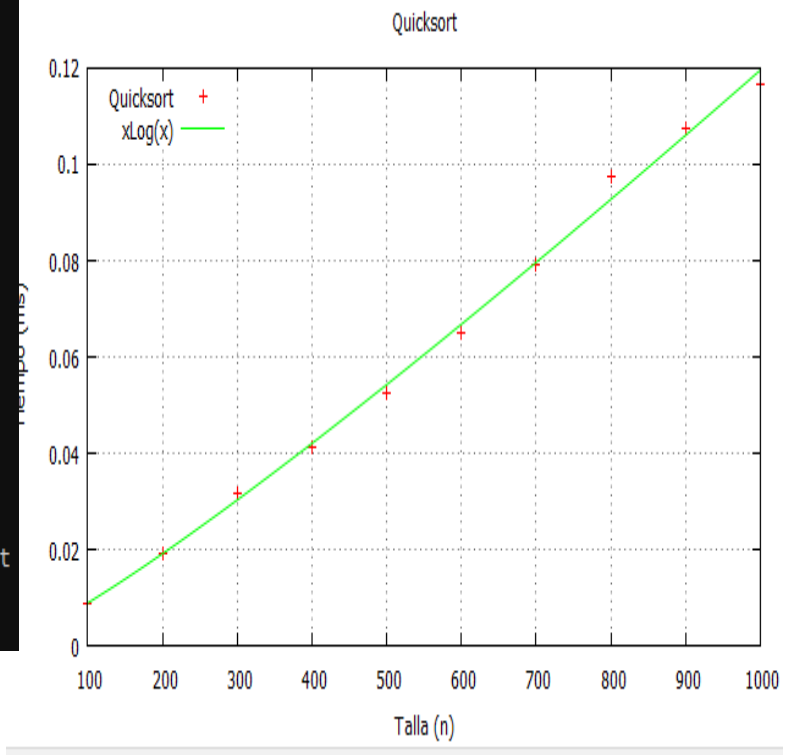
Aunque solo vamos a estudiar el caso medio, la diferencia entre el caso peor y los casos mejor y medio es que estos dos últimos son de orden logarítmico y el caso peor de orden cuadrático.

Como se puede observar en las gráficas teórica y experimental, ambas trayectorias tienden a ser una recta y el tiempo en ejecutar el algoritmo es similar.

	CASO MEDIO
	T(n)=(15*n*log(n)+26*n+1)/10^6
Talla	Tiempo(mseg)
100	0,0056
200	0,0121
300	0,0189
400	0,026
500	0,0332
600	0,0406
700	0,048
800	0,0556
900	0,0632
1000	0,071




```
***Ordenacion por Quicksort ***

Tiempos de ejecucion promedio

        Talla           Tiempo (mseg)

        100                0.0071
        200                 0.015
        300                 0.025
        400                 0.034
        500                 0.045
        600                 0.057
        700                 0.069
        800                 0.078
        900                 0.086
        1000                0.096

Datos guardados en el fichero Quicksort.dat

Generar grafica de resultados? (s/n): s
```



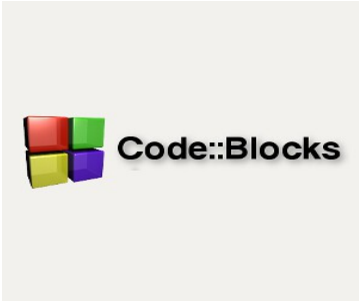
David Prieto Araujo david.prieto707@alu.uhu.es

Diseño de la aplicación

Para llevar a cabo la práctica, se ha hecho uso del programa codeblocks para ejecutar el código(del cual se ha implementado el archivo 'Principal.cpp', 'Graficas.cpp', 'AlgoritmosOrdenacion.cpp' y 'TestOrdenacion.cpp' y en el lenguaje de programación 'c++') y gnplot para crear las gráficas.

```
vector inicial para el metodo Quicksort:

709, 570, 87


Array ordenado con metodo Quicksort:

87, 570, 709

Presione una tecla para continuar . . .
```

```
*** Metodo a estudiar del caso medio ***

1.- Burbuja

2.- Insercion

3.- Seleccion

4.- Quicksort

0.- Volver al menu principal

-----------------------------------------

Elige una opcion: _
```



David Prieto Araujo david.prieto707@alu.uhu.es