

Ejercicio 1-3

Comparación básica de tiempos de ejecución

Algoritmos y Estructuras de Datos

Tema 1: análisis de algoritmos no recursivos

1º Grado en Ingeniería en Desarrollo de Contenidos Digitales
Profesor Dr. Carlos Grima Izquierdo (www.carlosgrima.com)
U-tad (www.u-tad.com), curso 2013/14

Vamos a comparar el tiempo que se tarda en ordenar un vector con un mal algoritmo y con un buen algoritmo, independientemente de lo bueno que sea el ordenador.

Para ello:

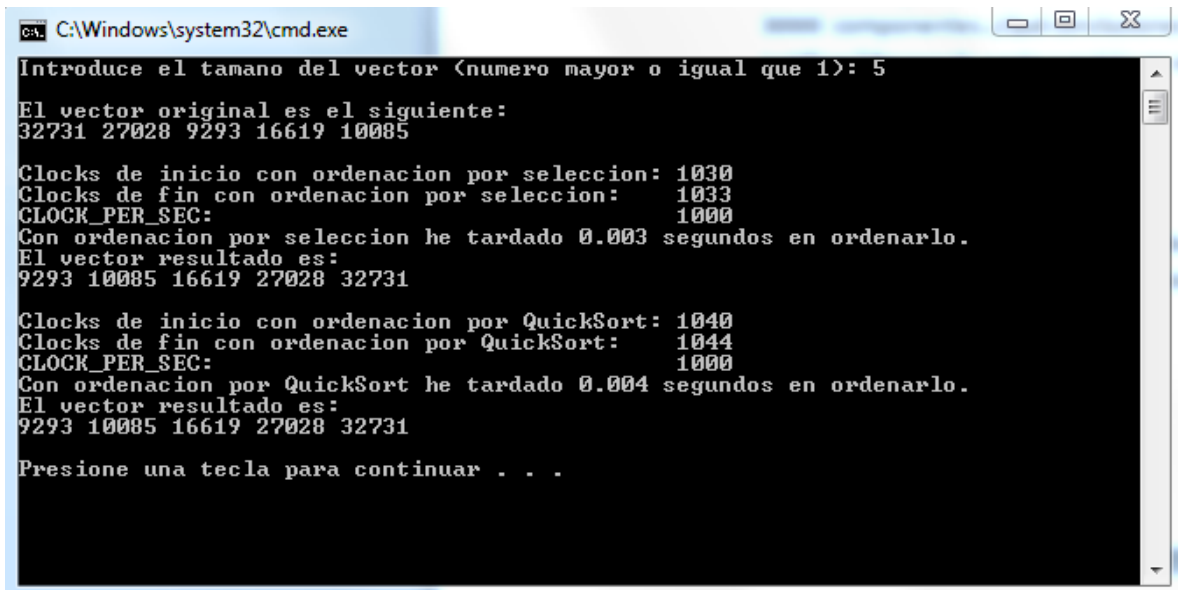
1. Idear un algoritmo para ordenar un array de número enteros. Se realizará una función en la cual el array de enteros se pasa como argumento (se pasa el puntero a su primer elemento), y la función lo ordena. O bien se creará una clase "VectorEnteros" que contenga como atributo un array de ints y que tenga métodos para ordenarse.
2. Programar un main() que haga lo siguiente:
 - a. Se pide por teclado el número de componentes del array que se ordenará (al menos 1). Se imprime el array por pantalla. Se imprimen todos los números seguidos, separados por un espacio (no por un retorno de carro)
 - b. Se ordena el array utilizando el algoritmo ideado. Se imprime por pantalla el array resultado y el tiempo empleado en ordenarlo.
 - c. Se ordena el array utilizando el método "qsort()", que implementa el algoritmo "Quick Sort", el cual es tremendamente eficiente cuando el tamaño del array es muy grande. Se imprime por pantalla el array resultado y el tiempo empleado en ordenarlo.
3. Se harán dos pruebas: una con un array de 5 componentes, y otra con un array de 30000 componentes. ¿Qué conclusiones extraes de los resultados y cómo los justificas?

Tener en cuenta lo siguiente:

- En ningún momento de este curso podrás usar la clase predefinida de C++ llamada "vector".
- Para medir el tiempo entre dos secciones de código, utiliza el método "clock()" de la biblioteca "ctime" (búscalo en www.cplusplus.com).

- Imprime el resultado en segundos, con suficientes decimales. Si es necesario, utiliza "printf()" y establece el número de decimales a imprimir. Busca los detalles de cómo usar printf() en www.cplusplus.com
- El método "qsort()" se encuentra en la biblioteca "cstdlib". Busca y comprende los detalles de cómo usarlo en www.cplusplus.com
- Para generar números aleatorios usar "srand()" (para establecer la semilla) y "rand()" (para generar el número aleatorio). Busca y comprende los detalles de cómo usarlos en www.cplusplus.com
- Ten en cuenta que el operador "/" en C/C++ hace la división entera si ambos operandos son de tipo "int". En caso de que uno de ellos sea "float", hace la división real. Tenlo en cuenta a la hora de calcular los segundos a partir de los clocks transcurridos.
- Como el tamaño del array no es constante (se define en tiempo de ejecución, pues lo introduce el usuario por teclado), es necesario usar malloc() para crear el array. Igualmente, habrá que liberar la memoria con free(). En su lugar, también se pueden utilizar "new" y "delete", pero se recomienda utilizar malloc() y free() para entrenar con ellos.

La prueba con 5 componentes daría el siguiente resultado (este resultado puede variar debido a que el array es aleatorio):



```

C:\Windows\system32\cmd.exe
Introduce el tamaño del vector (número mayor o igual que 1): 5
El vector original es el siguiente:
32731 27028 9293 16619 10085

Clocks de inicio con ordenación por selección: 1030
Clocks de fin con ordenación por selección: 1033
CLOCK_PER_SEC: 1000
Con ordenación por selección he tardado 0.003 segundos en ordenarlo.
El vector resultado es:
9293 10085 16619 27028 32731

Clocks de inicio con ordenación por QuickSort: 1040
Clocks de fin con ordenación por QuickSort: 1044
CLOCK_PER_SEC: 1000
Con ordenación por QuickSort he tardado 0.004 segundos en ordenarlo.
El vector resultado es:
9293 10085 16619 27028 32731

Presione una tecla para continuar . . .

```

Para la prueba con 30000 componentes, primero comentamos en el main las secciones en las cuales se imprimen por pantalla los vectores. El resultado sería:

ca. C:\Windows\system32\cmd.exe

Introduce el tamaño del vector (número mayor o igual que 1): 30000

El vector original es el siguiente:

Clocks de inicio con ordenación por selección: 6317
Clocks de fin con ordenación por selección: 8635
CLOCK_PER_SEC: 1000
Con ordenación por selección he tardado 2.318 segundos en ordenarlo.
El vector resultado es:

Clocks de inicio con ordenación por QuickSort: 8642
Clocks de fin con ordenación por QuickSort: 8670
CLOCK_PER_SEC: 1000
Con ordenación por QuickSort he tardado 0.028 segundos en ordenarlo.
El vector resultado es:

Presione una tecla para continuar . . .