Angel Barrilao Bensrhir

Practica 1 – Eficiencia

Ejercicio 4

Características del computador usado:
- Procesador: Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz
- Ram: 8 GB
- SO: Ubuntu 16.04 (Linux version 4.4.0-96-generic)
- Compilador: gcc version 5.4.0 20160609 (Ubuntu 5.4.0-6ubuntu1~16.04.4)

NOTA: En la compilacion de las pruebas realizadas no se han utilizado flags de optimización ni de otro tipo.

Algoritmo

Mejor Caso

```
#include <iostream>
#include <ctime>    // Recursos para medir tiempos
#include <cstdlib>  // Para generación de números pseudoaleatorios

using namespace std;

void ordenar(int *v, int n) {
	for (int i=0; i<n-1; i++){
		for (int j=0; j<n-i-1; j++){
			if (v[j]>v[j+1]){
				int aux = v[j];
				v[j] = v[j+1];
				v[j+1] = aux;
			}
		}
	}
}


void sintaxis()
{
  cerr << "Sintaxis:" << endl;
  cerr << "  TAM: Tamaño del vector (>0)" << endl;
  cerr << "Se genera un vector de tamaño TAM con elementos aleatorios" << endl;
  exit(EXIT_FAILURE);
}

int main(int argc, char * argv[]){

	// Lectura de parámetros
	if (argc!=2){
		sintaxis();
	}
```

```
        int tam=atoi(argv[1]);     // Tamaño del vector

  // Generación del vector aleatorio
  int *v=new int[tam];       // Reserva de memoria
  srand(time(0));            // Inicialización del generador de números pseudoaleatorios

  int num=tam;

  for (int i=0; i<tam; i++)  // Recorrer vector
        v[i] = i;    // Generar aleatorio [0,vmax[

  clock_t tini;    // Anotamos el tiempo de inicio
  tini=clock();

        ordenar(v,tam); // Llamamos a la funcion

  clock_t tfin;    // Anotamos el tiempo de finalización
  tfin=clock();

  // Mostramos resultados
  cout << tam << "\t" << (tfin-tini)/(double)CLOCKS_PER_SEC << endl;

  delete [] v;     // Liberamos memoria dinámica
}
```

Peor Caso

```
#include <iostream>
#include <ctime>    // Recursos para medir tiempos
#include <cstdlib>  // Para generación de números pseudoaleatorios

using namespace std;

void ordenar(int *v, int n) {
        for (int i=0; i<n-1; i++){
                for (int j=0; j<n-i-1; j++){
                        if (v[j]>v[j+1]){
                                int aux = v[j];
                                v[j] = v[j+1];
                                v[j+1] = aux;
                        }
                }
        }
}


void sintaxis()
{
```

Angel Barrilao Bensrhir

```
  cerr << "Sintaxis:" << endl;
  cerr << "  TAM: Tamaño del vector (>0)" << endl;
  cerr << "Se genera un vector de tamaño TAM con elementos aleatorios" << endl;
  exit(EXIT_FAILURE);
}

int main(int argc, char * argv[]){

        // Lectura de parámetros
        if (argc!=2){
                sintaxis();
        }

        int tam=atoi(argv[1]);     // Tamaño del vector

  // Generación del vector aleatorio
  int *v=new int[tam];       // Reserva de memoria
  srand(time(0));            // Inicialización del generador de números pseudoaleatorios

  //Metemos valores totalmente ordenados a la inversa.
  int num=tam;
  for (int i=0; i<tam; i++)
        v[i] = num--;    // Generar aleatorio [0,vmax[

  clock_t tini;    // Anotamos el tiempo de inicio
  tini=clock();

        ordenar(v,tam); // Llamamos a la funcion

  clock_t tfin;    // Anotamos el tiempo de finalización
  tfin=clock();

  // Mostramos resultados
  cout << tam << "\t" << (tfin-tini)/(double)CLOCKS_PER_SEC << endl;

  delete [] v;     // Liberamos memoria dinámica
}
```

Script

```
#!/bin/csh
@ inicio = 100
@ fin = 30000
@ incremento = 100
set ejecutable1 = mejorcaso
set ejecutable2 = peorcaso
set salida1 = mejorcaso.dat
set salida2 = peorcaso.dat
```

Angel Barrilao Bensrhir

```
@ i = $inicio
echo > $salida1
while ( $i <= $fin )
  echo Ejecución tam = $i
  echo `./{$ejecutable1} $i` >> $salida1
  @ i += $incremento
end

@ i = $inicio

echo > $salida2
while ( $i <= $fin )
  echo Ejecución tam = $i
  echo `./{$ejecutable2} $i` >> $salida2
  @ i += $incremento
end
```

Comparativa de eficiencia empirica