Estructuras de Datos
Escuela de Informática-UNT
Tarea1: Algoritmos de Ordenación

Prof. José M. Saavedra R.
Prof. Teresa Bracamonte N.

Fecha de Entrega:17 de Febrero, 2010

1 Objetivo

El objetivo de esta tarea es realizar una pequeña investigación experimental, basada en seis algoritmos de ordenación.

2 Descripción

Esta tarea consiste en implementar y evaluar los seis siguientes algoritmos de ordenación:

1. Ordenación por Selección.
2. Ordenación Burbuja.
3. Ordenación por Inserción.
4. Shell Sort.
5. Ordenación por Mezcla.
6. Ordenación Rápida, usando pivote aleatorio.

Cinco de los seis algoritmos han sido estudiados en cátedra. El algoritmo *Shell Sort* se describe a continuación.

Shell Sort
Este algoritmo funciona de manera similar al algoritmo de inserción, pero los elementos se toman considerando un separación h que va disminuyendo en cada iteración. Lo que se obtiene con esto es que los elementos del arreglo ya están relativamente ordenados cuando se realiza la última iteración con $h = 1$, que es equivalente al algoritmos de inserción tradicional. Para esta tarea, utilice la siguiente serie de pasos:

$$h_i = \frac{3^i - 1}{2}, (h = \{1, 4, 13, 40, 121, \dots\})$$

en donde el paso mayor es el número de la serie que se encuentra dos posiciones antes del más cercano a n. Ejemplo: Si $n = 120$, entonces el más cercano en la serie es 121, y el paso mayor será 13, pues éste se ubica dos posiciones antes del 121.

2.1 Datos de Prueba

Para este experimento usted deberá crear 3 tipos de datos de prueba:

1. Un conjunto de elementos generados aleatoriamente (que por supuesto es un conjunto desordenado).
2. Un conjunto de elementos ordenados de menor a mayor (Genérelos aleatoriamente y luego ordénelos con alguno de los métodos).
3. Un conjunto de elementos ordenados de mayor a menor (Genérelos aleatoriamente y luego ordénelos con alguno de los métodos).

El tamaño de los conjuntos debe variar para cada tipo de datos. Considere conjuntos de tamaño $10^5, 2 \times 10^5, 3 \times 10^5, 4 \times 10^5, 5 \times 10^5, 6 \times 10^5, 7 \times 10^5, 8 \times 10^5, 9 \times 10^5$ y 10^6. Considere elementos enteros en el rango $[0, 10^6]$. Para generar datos aleatorios puede utilizar el siguiente código:

```
#include <stdlib.h>
#include <time.h>
#include <math.h>
#define MAX_LLAVE 1000000


int *generarAleatorios(int n)
{
        int *A=new int[n];
        srand ( time(NULL) );
        int i=0;
        for(i=0;i<n;i++)
        {
                A[i]=(int)((((double)rand())/RAND_MAX)*MAX_LLAVE);
        }
        return A;
}
```

2.2 Parámetros a evaluar

Se evaluarán el **número de comparaciones** que realiza el algoritmo así como el **tiempo** que necesita para ordenar. Para obtener el tiempo de un proceso puede usar el siguiente código:

```
#include<time.h>
 int getMilisegundos(clock_t c)
{
  int tiempo=0;
```

```
  tiempo = (int)((c/(double)CLOCKS_PER_SEC)*1000) ;
  return tiempo;
}

int main()
{
  clock_t t1, t2;
  int tiempo=0;
  t1= clock();
  procesoOrdenar(...);
  t2=clock();
  tiempo=getMilisegundos(t2-t1);
}
```

2.3 Informe

En el informe tiene que presentar tres tablas que midan comparciones, una para cada tipo de conjunto de prueba: Aleatorio, ordenado de menor a mayor y ordenado de mayor a menor. Cada tabla tiene 10 filas, una por cada tamaño del conjunto, y 6 columnas, una por cada método. Ejemplo:

	Selección	Burbuja	Inserción	Shell	Mezcla	Quick
10^5						
2×10^5						
3×10^5						
4×10^5						
5×10^5						
6×10^5						
7×10^5						
8×10^5						
9×10^5						
10^6						

Table 1: Número de Comparaciones por algoritmo para datos aletorios.

Similarmente, debe generar tres tablas bajo el mismo formato que mida tiempo de ejecución en milisegundos.

Para mayor claridad de los resultados, realice una gráfica por cada tabla (es decir deben existir 6 gráficos) que ilustre el comportamiento de los algoritmos bajo diferentes entradas.Es importante que describa cada uno de los gráficos y comente el comportamiento de los algoritmos.

Respecto al contenido del informe, éste debe contener:

- Introducción: Describe el trabajo desarrollado.
- Algoritmos de Ordenación:
  - Desripción del algoritmo indicando costo.
  - Descripción de la implementación.

- Resultados del Experimento, con su correspondiente análisis.
- Conclusiones.

3 Restricciones

- Asuma una ordenación ascendente.
- El trabajo se puede realizar de hasta 2 estudiantes por grupo.
- El lenguaje de programación debe ser C o C++.
- Todos los algoritmos de ordenación deben ser implementados por usted.
- Trabajos idénticos serán anulados.
- Los trabajos se entregan al inicio de la clase según fecha indicada.
- No se aceptan atrasos.