Algoritmos http://www.madsgroup.org/docencia/alg/ Octubre de 2008

Ingeniería Informática

Ingeniería Técnica de Informática de Gestión

Práctica 2

Ordenación por inserción

El problema consiste en ordenar ascendentemente un vector de n números enteros. Como algoritmo de ordenación se utilizará la *ordenación por inserción*:

```
procedimiento Ordenación por inserción (var v[1..n])
  para i := 2 hasta n hacer
    x := v[i] ;
    j := i-1 ;
    mientras j > 0 y v[j] > x hacer
      v[j+1] := v[j] ;
      j := j-1
    fin mientras ;
    v[j+1] := x
  fin para
fin procedimiento
```

1. Implemente el algoritmo de ordenación por inserción (figura 1).

```
void ordenacion_por_insercion (int v [], int n) {
  int i, j, x;
  for (i=1; i<n; i++) {
/*
 *
 */
  }
}
```

Figura 1: Ordenación por inserción

2. Valide el correcto funcionamiento de este método de ordenación (figura 2).
3. Determine los tiempos de ejecución para distintos valores de n $(500, 1000, 2000, 4000, \ldots)$ y para tres diferentes situaciones iniciales: (a) el vector ya está ordenado en orden ascendente, (b) el vector ya está ordenado en orden descendente, y (c) el vector está inicialmente desordenado (figura 3).
4. Calcule empíricamente la complejidad del algoritmo para cada una de las diferentes situaciones iniciales del vector (figura 4).

```
$ ./test
Inicialización aleatoria módulo 100
22, 78, 70, 88, 23, 55, 53, 62, 83, 38
¿ordenado? 0
Ordenación por Inserción
22, 23, 38, 53, 55, 62, 70, 78, 83, 88
¿ordenado? 1

Inicialización ascendente
0, 1, 2, 3, 4, 5, 6, 7, 8, 9
¿ordenado? 1
Ordenación por Inserción
0, 1, 2, 3, 4, 5, 6, 7, 8, 9
¿ordenado? 1

Inicialización descendente
10, 9, 8, 7, 6, 5, 4, 3, 2, 1
¿ordenado? 0
Ordenación por Inserción
1, 2, 3, 4, 5, 6, 7, 8, 9, 10
¿ordenado? 1
```

Figura 2: Test

```
#include <stdlib.h>

void aleatorio(int v [], int n) {
  int i;
  srand(time(NULL));
  for (i=0; i < n; i++)
    v[i] = rand();
}
```

Figura 3: Inicialización aleatoria

```
Ordenación por inserción con inicialización descendente
           n           t(n)     t(n)/n^1.8       t(n)/n^2     t(n)/n^2.2
(*)      500         686.601       0.009518       0.002746       0.000792
(*)     1000        2659.378       0.010587       0.002659       0.000668
        2000        9695.000       0.011084       0.002424       0.000530
        4000       42872.000       0.014076       0.002680       0.000510
        8000      168412.000       0.015879       0.002631       0.000436
       16000      645125.000       0.017467       0.002520       0.000364
       32000     2713383.000       0.021098       0.002650       0.000333
```

Figura 4: Parte de la posible salida por pantalla de la ejecución del programa principal