Algoritmos 2016-17

Grado en Ingeniería Informática

Práctica 2

Ordenación por inserción y ordenación rápida

Fecha límite de entrega: viernes, 28 de octubre

El problema consiste en ordenar ascendentemente un vector de n números enteros. Como algoritmos de ordenación se utilizarán la *ordenación por inserción* y la *ordenación rápida*:

```
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

procedimiento Ordenación Rápida (v[1..n])
  Ordenación Rápida Auxiliar (v[1..n])
fin procedimiento

procedimiento Ordenación Rápida Auxiliar (v[iz..dr])
  si iz < dr entonces
    pivote := v[(iz+dr)/2] ;
    intercambiar(v[iz], v[(iz+dr)/2]) ;
    i := iz + 1 ;
    j := dr ;
    mientras i <= j hacer
      mientras i <= dr y v[i] < pivote hacer i := i + 1 fin mientras ;
      mientras v[j] > pivote hacer j := j - 1 fin mientras ;
      si i <= j entonces
        intercambiar (v[i], v[j]) ;
        i := i + 1 ;
        j := j - 1
      fin si
    fin mientras ;
    intercambiar (v[iz], v[j]) ;
    Ordenación Rápida Auxiliar (v[iz..j-1]) ;
    Ordenación Rápida Auxiliar (v[j+1..dr])
  fin si
fin procedimiento
```

1. Implemente los algoritmos de ordenación por inserción y rápida.

```
void ord_ins (int v [], int n) {
  /* ... */
}

void ord_rap_aux (int v [], int iz, int dr) {
  /* ... */
}

void ord_rap (int v [], int n) {
  ord_rap_aux(v, 0, n-1);
}
```

2. Valide el correcto funcionamiento de ambos métodos de ordenación (figura 1).

```
> ./test
Ordenacion por insercion con inicializacion aleatoria
3, -3, 0, 17, -5, 2, 11, 13, 6, 1, 7, 14, 1, -2, 5, -14, -2
ordenado? 0
ordenando...
-14, -5, -3, -2, -2, 0, 1, 1, 2, 3, 5, 6, 7, 11, 13, 14, 17
ordenado? 1

Ordenacion por insercion con inicializacion descendente
10, 9, 8, 7, 6, 5, 4, 3, 2, 1
ordenado? 0
ordenando...
1, 2, 3, 4, 5, 6, 7, 8, 9, 10
ordenado? 1
```

Figura 1: Posible test para la ordenación por inserción

3. Tanto para la ordenación por inserción como para la ordenación rápida calcule los tiempos de ejecución para distintos valores de n 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 (véase la figura 2).

4. Analice los tiempos obtenidos determinando empíricamente la complejidad de los algoritmos para cada una de las diferentes situaciones iniciales del vector (i.e., 6 tablas) (figura 3).

5. Deposite en `/PRACTICAS/GEI/Alg/P2/` (existe un directorio para cada estudiante) los ficheros `C` y el informe con el estudio empírico de la complejidad.

```c
#include <stdlib.h>
void inicializar_semilla() {
  srand(time(NULL));
}
void aleatorio(int v [], int n) {/* se generan números pseudoaleatorio entre -n y +n */
  int i, m=2*n+1;
  for (i=0; i < n; i++)
    v[i] = (rand() % m) - n;
}
void ascendente(int v [], int n) {
  int i;
  for (i=0; i < n; i++)
    v[i] = i;
}
```

Figura 2: Inicialización aleatoria y ascendente

```
Ordenación por inserción con inicialización descendente
           n           t(n)     t(n)/n^1.8       t(n)/n^2     t(n)/n^2.2
(*)      500         247.03       0.003425       0.000988       0.000285
        1000         953.00       0.003794       0.000953       0.000239
        2000        3818.00       0.004365       0.000955       0.000209
        4000       15471.00       0.005079       0.000967       0.000184
        8000       69474.00       0.006550       0.001086       0.000180
       16000      257089.00       0.006961       0.001004       0.000145
       32000     1023540.00       0.007959       0.001000       0.000126
```

Figura 3: Parte de la posible salida por pantalla de la ejecución del programa principal