Algoritmos 2012/13

Grado en Ingeniería Informática

Práctica 3

Fecha límite de entrega: lunes, 19 de noviembre[1]

Ordenación por inserción y ordenación rápida

Ordenación por inserción

Algoritmo de 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
```

Se pide:

1. Implemente el algoritmo de ordenación por inserción.

   ```
   void ord_ins (int v [], int n);
   ```

2. Valide el correcto funcionamiento de la implementación

   ```
   > ./test
   Inicialización aleatoria
   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 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
   ```

3. Determine los tiempos de ejecución para distintos tamaños del vector 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 1).

4. Calcule empíricamente la complejidad del algoritmo para cada una de las diferentes situaciones iniciales del vector

[1] Deposite, desde alguna de las *máquinas de referencia* (consúltense en `wiki.fic.udc.es`) en `/PRACTICAS/GEI/Alg/P3/` (existe un directorio para cada estudiante) los ficheros `C` y los ficheros con los tiempos de ejecución y el estudio empírico de la complejidad.

```
#include <stdlib.h>
void inicializar_semilla() {
  srand(time(NULL));
}
void aleatorio(int v [], int n) {
  int i;
  for (i=0; i < n; i++)
    v[i] = rand();
}
void ascendente(int v [], int n) {
  int i;
  for (i=0; i < n; i++)
    v[i] = i;
}
```

Figura 1: Inicialización aleatoria y ascendente

Ordenación Rápida

Algoritmo de ordenación rápida (*quicksort*) con selección del pivote por mediana de tres y un umbral para detectar vectores pequeños:

```
procedimiento Mediana3 (V[i..j])
  k := (i + j) div 2 ;                  /* precondición: i < j */
  si V[k] > V[j] entonces intercambiar (V[k], V[j]) ;
  fin si
  si V[k] > V[i] entonces intercambiar (V[k], V[i]) ;
  fin si
  si V[i] > V[j] entonces intercambiar (V[i], V[j]) ;
  fin si
fin procedimiento

procedimiento OrdenarAux (V[izq..der])
  si izq+UMBRAL <= der entonces       /* UMBRAL >= 1 */
    Mediana3 (V[izq..der]) ;          /* el pivote está en 'izq' y en 'der' habrá */
                                      /* un valor mayor o igual que el pivote */
    pivote := V[izq] ;
    i := izq ;
    j := der ;
    repetir
      repetir i := i + 1 ; hasta V[i] >= pivote ;
      repetir j := j - 1 ; hasta V[j] <= pivote ;
      intercambiar (V[i], V[j]);
    hasta j <= i ;
    intercambiar (V[i], V[j]) ;       /* se deshace el último intercambio */
    intercambiar (V[izq], V[j]) ;
    OrdenarAux (V[izq..j-1]);
    OrdenarAux (V[j+1..der])
  fin si
fin procedimiento

procedimiento Ordenación Rápida (V[1..n])
  OrdenarAux(V[1..n]);
  si (UMBRAL > 1) entonces
    Ordenación por Inserción (V[1..n])
  fin si
fin procedimiento
```

Se pide:

1. Implemente el algoritmo de ordenación rápida.

```
void ord_rapida(int v [], int n) {
  rapida_aux(v, 0, n-1);
  if (UMBRAL > 1)
    ord_ins(v, n);
}
```

2. Valide el correcto funcionamiento de la implementación (con umbral $= 1$).

3. Ejecute el algoritmo con vectores de distinto tamaño y en distintas situaciones iniciales (vector ordenado ascendente o descendentemente, o desordenado), y con distintos valores de umbral: 1 (no se realiza la llamada al método de ordenación por inserción), 10 y 100.

4. Compare entre si los tiempos obtenidos para cada umbral usado. En función de la situación inicial del vector, ¿con qué umbral se obtienen los mejores tiempos? ¿por qué?

5. Calcule empíricamente la complejidad del algoritmo para cada una de las diferentes situaciones iniciales del vector y cada uno de los umbrales (i.e., 9 tablas).