Algoritmos 2019/20

Grado en Ingeniería Informática

Práctica 2

Fecha límite de entrega: jueves, 31 de octubre

Ordenación por inserción y ordenación *shell*: 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 shell*:

```
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 shell (v[1..n])
  incremento := n;
  repetir
    incremento := incremento div 2;
    para i := incremento+1 hasta n hacer
      tmp := v[i];
      j := i;
      seguir := cierto;
      mientras j-incremento > 0 y seguir hacer
        si tmp < v[j-incremento] entonces
          v[j] := v[j-incremento];
          j := j-incremento
        sino  seguir := falso
        fin si
      fin mientras;
      v[j] := tmp
    fin para
  hasta incremento = 1
fin procedimiento
```

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

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

2. Valide el correcto funcionamiento de la implementación.

   ```
   > ./test
   Inicializacion aleatoria
   3, -3, 0, 17, -5, 2, 11, 13, 6, 1, 7, 14, 1, -2, 5, -14, -2
   ordenado? 0
   Ordenacion por Insercion
   -14, -5, -3, -2, -2, 0, 1, 1, 2, 3, 5, 6, 7, 11, 13, 14, 17
   ordenado? 1
   ```

```
Inicializacion descendente
10, 9, 8, 7, 6, 5, 4, 3, 2, 1
ordenado? 0
Ordenacion por Insercion
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).

```
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 1: Inicialización aleatoria y ascendente

4. Calcule empíricamente las complejidades de los algoritmos para cada una de las diferentes situaciones iniciales del vector (i.e., 6 tablas) (figura 2).

5. Entregue los ficheros con el código C y el informe en una carpeta **P2** mediante SVN en el repositorio **https://svn.fic.udc.es/grao2/alg/19-20/*pepe.perez*** (sustituya ***pepe.perez*** por su login).

```
Ordenacion por insercion con inicializacion 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 2: Parte de la posible salida por pantalla de la ejecución del programa principal