Algoritmos http://www.lfcia.org/alg/ 2007-2008

Ingeniería Informática

Ingeniería Técnica de Informática de Gestión

Práctica 3

Fecha límite de entrega: viernes, 16 de noviembre[1]

Ordenación Rápida

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

```
procedimiento Qsort (V[izq..der])
  si izq+UMBRAL <= der entonces
    x := { número aleatorio en el rango [izq..der] };
    pivote := V[x];
    intercambiar (V[izq], V[x]);
    i := izq + 1;
    j := der;
    mientras i <= j hacer
      mientras i <= der 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[izq], V[j]);
    Qsort (V[izq..j-1]);
    Qsort (V[j+1..der])
    fin si
fin procedimiento

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

[1]Fecha límite para la entrega de las tres primeras prácticas. Deposite desde las máquinas `limia` o `xurxo` en `/PRACTICAS/EI/Alg` o `/PRACTICAS/ETIX/Alg`, según corresponda (existe un directorio para cada práctica y para cada estudiante) los ficheros *java* y los ficheros con los tiempos de ejecución y el estudio empírico de la complejidad

Se pide:

1. Copie de la segunda práctica las clases `Comparable`, `Entero`, `InicializacionDeVectores`, `Aleatoria`, `Ascendente`, `Descendente`, `Ordenacion` y cualquier otra que estime oportuna.

2. Implemente el algoritmo de ordenación rápida (figura 1).

```
public class OrdenacionRapida implements Ordenacion {
    private int umbral;
    private java.util.Random rand = new java.util.Random();
    public OrdenacionRapida(int u) {
        umbral = u;
    }
      .
      .
      .
    public void ordenar(Comparable [] v) {
        qsort(v, 0, v.length - 1);
        if (umbral > 1)
            (new OrdenacionPorInsercion()).ordenar(v);
    }
    public String toString() {
        return "Ordenacion Rapida con umbral igual a " + umbral;
    }
}
```

Figura 1: La clase `OrdenacionRapida`

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

4. 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.

5. 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?

6. 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).