Università di Torino – Facoltà di Scienze MFN
Corso di Studi in Informatica
Curriculum SR (Sistemi e Reti)

Laboratorio di Algoritmi a.a. 2009-10

Compito 7

Quicksort alla Hoare generico

Si definisca in Java una classe *Sorting* contenente una realizzazione del quicksort per array di *int* e una per array *generici*, con la partizione alla Hoare illustrata nella lezione 15, e con ottimizzazione con insertion-sort sotto una certa soglia (da determinarsi sperimentalmente, vedi slide successiva).

```
public class Sorting {
  static final int SOGLIA = …;
  private static Random generatore;

   public static <T extends Comparable<T>> void qsort(T[] a) {
   generatore = new Random();
   qs(a, 0, a.length - 1);
   …
 }
```

Ovviamente si devono definire tutti i metodi ausiliari necessari (con tipo generico *T extends Comparable<T>*).

Nella classe *Sorting* si definiscano inoltre i metodi
public static boolean isSorted(int[] a) oppure èOrdinato
p…static <T extends Comparable<T>> boolean isSorted(T[] a)
che controllano se un array è ordinato.

Classe RandomArrays

Si definisca una classe *RandomArrays* contenente i metodi:

public static int[] randomIntArray(int lungh, int max)
che costruisce e restituisce un array di lunghezza lungh di int casuali compresi fra 0 e max (escluso)

public static Integer[] randomIntegerArray(int lungh, int max)
che costruisce e restituisce un array di lunghezza lungh di Integer casuali compresi fra 0 e max (escluso)

p… static Double[] randomDoubleArray(int lungh, double max)
che costruisce e restituisce un array di lunghezza lungh di Double casuali compresi fra 0 e max;

facoltativo:
p… static String[] randomStringArray(int lungh, int lunghString)
che costruisce e restituisce un array di lunghezza lungh di String casuali di lunghezza lunghString;

Main di prova

Si definisca una classe *ProvaSorting* contenente un main il quale, utilizzando i metodi delle classi *RandomArrays* e *Sorting*, provi il quicksort su array casuali di int di lunghezze diverse e il quicksort generico su array di tipi diversi (Integer, Double, String) e lunghezze diverse, e misuri i tempi di esecuzione.

Si aggiusti per tentativi il valore di soglia.

Nota: si devono usare i metodi isSorted per controllare il funzionamento dell'algoritmo su array di grandi dimensioni:

```
…
out.println("l'array" + (isSorted(ar) ? "" : " NON") + " e' ordinato");
out.println("sto ordinando con il qsort generico");
t0 = nanoTime();
qsort(ar);
…
```