**Quick Sort**

L**’idea di base** consiste in una funzione ricorsiva che dato un array ne prende in considerazione il valore di una cella casuale (chiamato pivot) e suddivide questo primo vettore in tre parti:

* la parte sinistra con tutti i valori più piccoli del pivot
* quella destra con tutti i valori più grandi
* quella centrale con tutti i valori uguali

Una volta suddiviso in “sotto-array” viene eseguito lo stesso procedimento per le parti di sinistra e di destra, sicché una volta terminati tutti i processi ricorsivi il vettore sia ordinato.

Per l’algoritmo ho scelto di utilizzare le liste di interi al posto dei vettori, per comodità

List<int> QuickSort ( List<int> vett)

{

//caso base

Se ( la lunghezza della lista passata come parametro è minore di 2)

{

restituisce la lista;

}

Altrimenti

{

pivot = vett[random]

List<int> sinistra

List<int> destra

List<int> centro

//Partiziona il vettore in 3:

Per tutta la lunghezza del vettore

(dove x è l’indice incrementale della cella corrente [0 → n])

{

se (vett[x] < pivot)

sinistra.Add(vett[x]);

invece se (vett[x] > pivot)

destra.Add(vett[x]);

altrimenti

centro.Add(vett[x]);

}

//riassembla il vettore/lista in una nuova, richiamando la funzione stessa

List<int> “Merge”

Merge.Aggiungi\_lista(QuickSort(sinistra))

Merge.Aggiungi\_lista(centro)

Merge.Aggiungi\_lista(Quicksort(destra))

Ritorna Merge

}

}