**public** **static** **void** sort(Comparable[] a) {

*shuffle(a);* *sort* (a, 0, a.length-1);

}

**static** **void** sort (Comparable[] a, **int** lo, **int** hi) {

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Q | T | R | I | C | S | O | U |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

**if** (hi <= lo) **return**;

**int** j = *partition*(a, lo, hi);

**// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*sort*(a, lo, j-1);

*sort*(a, j+1, hi);

**// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

}

**static** **int** partition(Comparable[] a, **int** lo, **int** hi) {

**int** i = lo;

**int** j = hi + 1;

Comparable v = a[lo];

**// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**while** (**true**) {

// find item on lo to swap

**while** (*less*(a[++i], v))

**if** (i == hi) **break**;

// find item on hi to swap

**while** (*less*(v, a[--j]))

**if** (j == lo) **break**;

**// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

// check if pointers cross

**if** (i >= j) **break**;

*exch*(a, i, j);

}

**// \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

// put partitioning item v at a[j]

*exch*(a, lo, j);

// now, a[lo .. j-1] <= a[j] <= a[j+1 .. hi]

**return** j;

}