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

**int** N = a.length;

**for** (**int** k = N/2; k >= 1; k--) { // (1) create heap from array

*sink*(a, k, N);

}

// **TRUTH:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

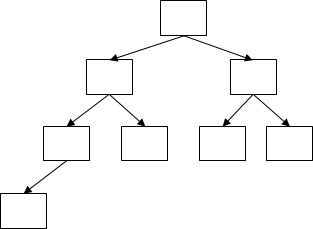
**while** (N > 1) { // (2) modify array in place, exchanging max

*exch*(a, 1, N--); // **TRUTH after:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*sink*(a, 1, N); // **TRUTH after:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

}

**static** **void** sink(Comparable[] a, **int** k, **int** N) {

**while** (2\*k <= N) {

**int** j = 2\*k;

**if** (j < N && *less*(a, j, j+1)) j++;

**if** (!*less*(a, k, j)) **break**;

*exch*(a, k, j);

k = j;

}

}

**static** **boolean** less(Comparable[] a, **int** i, **int** j) {

**return** a[i-1].compareTo(a[j-1]) < 0;

}

**static** **void** exch(Object[] a, **int** i, **int** j) {

Object swap = a[i-1];

a[i-1] = a[j-1];

a[j-1] = swap;

}

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| **S** | **O** | **R** | **T** | **E** | **X** | **A** | **M** |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |