public abstract void join(int[], int, int, int, int);

public abstract int split(int[], int, int, int);

{
    {
        join(A, 0, 's', 'h');
        sort(A, 's', 'h');
        sort(A, 0, 's', 't');
        int s = split(A, 0, 't');
        if (to > 't')
        }
    }

public final void sort(int[], int, int, int);

Recall the abstract class Abstract in the handout.

Sorting by Divide and Conquer
- Because each `split()` divides the array into two (almost) equal-sized parts, each element is `join()`ed `log n` times.

- Merge Sort takes \( O(n \log n) \) steps.

- Merge Sort is a easy-split, hard-join method.

**Merge Sort**
Heap Sort

- Heap Sort is a hard-split, easy-join method.
- Think of Heap Sort as an improved (faster) version of Selection Sort.
- Specifically, \( \text{SPITT}() \), which finds the minimum (maximum) element in the subarray, is made to run in \( O(n) \) steps instead of \( O(n \log n) \) steps, where \( n \) is the subarray length.
- Since \( \text{SPITT}() \) is performed \( n \) times, where \( n \) is the (overall) array length, Heap Sort takes \( O(n \log n) \) steps.
The elements in the unsorted portion of the array are organized into a heap.

\[ \text{Unsorted: Heap Sorted} \]
children are heaps.

The root, if non-nil, is the largest key in the tree, and its left and right subtrees are themselves heaps.

- A heap is a binary tree that is almost balanced (we allow a variation of at most 1 in path lengths from the root to the leaves) and exhibits the heap property:

  - The largest key in the tree is in the root.

What is a Heap?
Implementing a Heap