

## Build Heap

1. Let  $\text{index} = \text{length}/2 - 1$ . This is the parent of the last node in the tree, i.e.  $\text{list}[\text{index} + 1] \dots \text{list}[\text{length} - 1]$  are leaves
2. Convert the subtree with root of  $\text{list}[\text{index}]$  into a heap.
  - a. Given  $\text{list}[a]$  is root of tree,  $\text{list}[b]$  is left child ( $\text{root} * 2 + 1$ ),  $\text{list}[c]$  is right child ( $\text{root} * 2 + 2$ ), if exists
  - b. Compare  $\text{list}[b]$  with  $\text{list}[c]$  to determine larger child,  $\text{list}[\text{largerIndex}]$
  - c. Compare  $\text{list}[a]$  with  $\text{list}[\text{largerIndex}]$ . If  $\text{list}[a] < \text{list}[\text{largerIndex}]$ , then swap, else already a heap
  - d. If swap, repeat step 2 for the subtree of  $\text{list}[\text{largerIndex}]$
3. Convert the subtree with the root of  $\text{list}[\text{index} - 1]$  into a heap, repeat until  $\text{list}[0]$

## Heap Sort

1. Swap the root with the end of the list.
2. Heapify the list up to but not including the root
3. Repeat until there is only one node in the list

Simulate the heapsort algorithm manually to sort the array:

Show all steps

1. Make into a heap
2. Sort

# Max-Heap

[illegible]

[illegible]

[illegible]