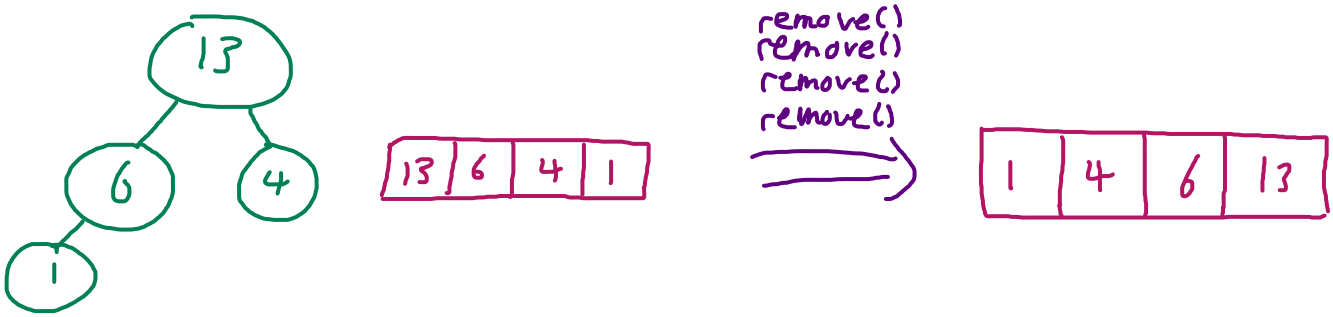
Heap Sort

1. Transform input array into a complete tree (later converted into a max-heap).
2. Starting from right→left, do "bottom-up heapification" (sink each node w/ max-heap priority); this organizes tree into max-heap.

3. Repeat N times: remove highest priority item and place item at end of heap array (sorting in place).



Memory? $\Theta(1)$ ← in place

Runtime?

Best case: $\Theta(N)$ ← if all elements same, sink and remove are constant operations.

Avg: $\Theta(N \log N)$

heapify = N "sinks" = $N \cdot O(\log N) = O(N \log N)$

N removes = $N \cdot O(\lg N) = O(N \log N)$

DO NOT USE THIS ALGORITHM!!

Heap sort is theoretically fast, but is in reality very slow on large inputs. It runs into caching problems (out of scope - 61C topic).