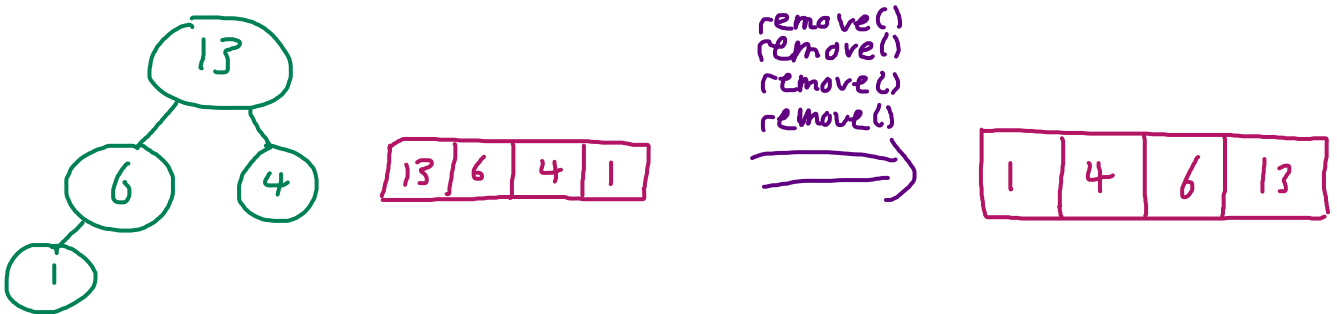


Heap Sort

- ① Transform input array into a complete tree (later converted into a max-heap).
- ② Starting from right $\rightarrow$ 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? $\Theta(1)$ $\leftarrow$ in place

Runtime?

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

Avg: $\Theta(N \log N)$ $\leftarrow$ heapify = N "sinks" = $N \cdot O(\log N) = O(N \log N)$
 N removes = $N \cdot O(\log N) = O(N \log N)$

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).

