# CS61B Lecture #22

**Administrative:**

- Midterm graded:
  - Intended average: 20
  - Actual average: 23
  - Actual median: 22
- Midterms available at lecture today; after that, they'll be in 385 Soda.

**Today:** Project #1 retrospective, priority queues.



# Priority Queues, Heaps

- Priority queue: defined by operations "add," "find largest," "remove largest."
- Examples: scheduling long streams of actions to occur at various future times.
- Also useful for sorting (keep removing largest).
- Heap is common implementation.
- Enforces *heap property:* all labels in *both* children of node are less (or greater) than node's label.
- So node at top has largest (or smallest) label.
- Are free to add smaller value to less bushy subtree, thus maintaining bushiness (keeping tree balanced).
- Insertion and deletion always proportional to $\lg N$ in worst case.

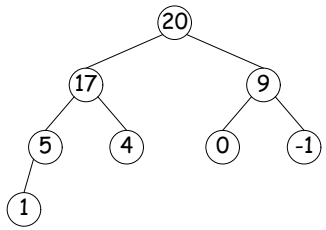


# Example: Inserting into a simple heap

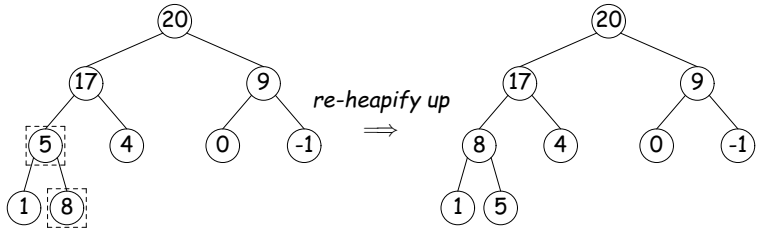
**Data:**

```
1 17 4 5 9 0 −1 20
```

**Initial Heap:**



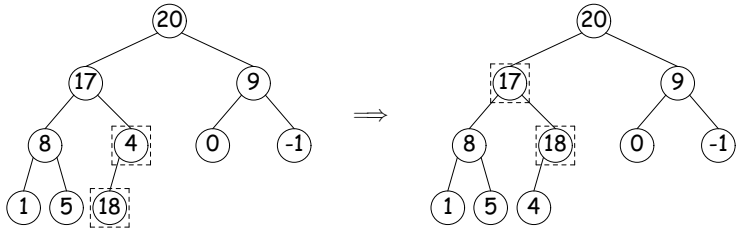

**Add 8:** Dashed boxes show where heap property violated

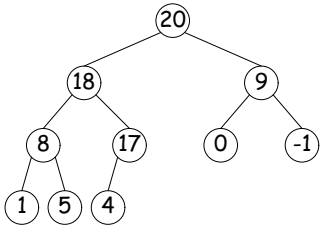





# Heap insertion continued

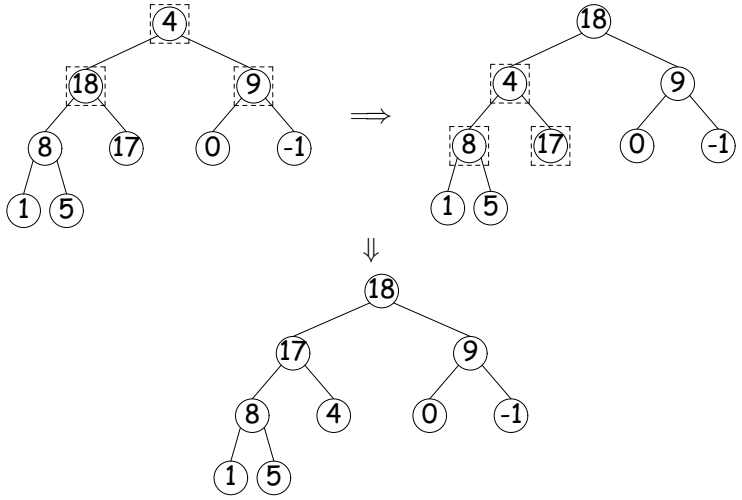
**Now insert 18:**



## Removing Largest from Heap

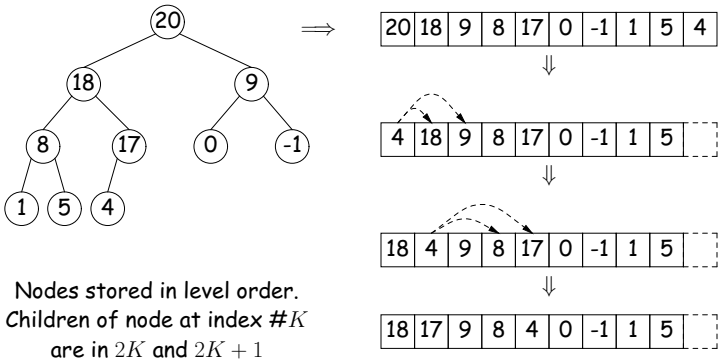
**To remove largest:** Move bottommost, rightmost node to top, then re-heapify down as needed (swap offending node with larger child) to re-establish heap property.






## Heaps in Arrays

- Since heaps are complete (missing items only at bottom level), can use arrays for compact representation.
- Example of removal from last slide (dashed arrows show children):




Nodes stored in level order. Children of node at index #$K$ are in $2K$ and $2K+1$