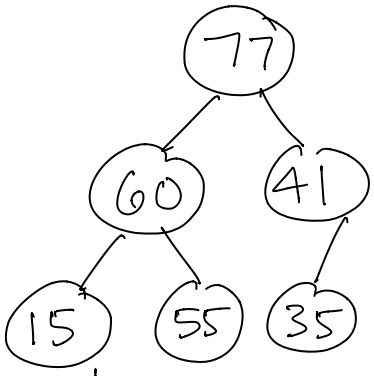
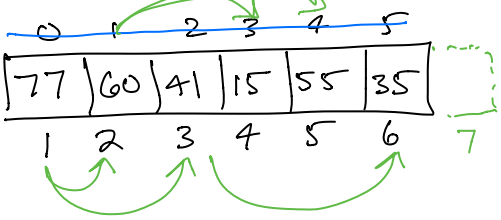
How to use a Heap to implement a P.Q.



parent
1 → 2, 3
2 → 4, 5
3 → 6, 7

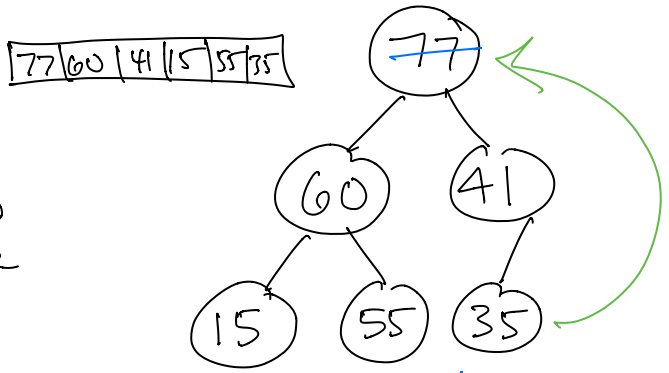
We are going to use an array to store a heap.

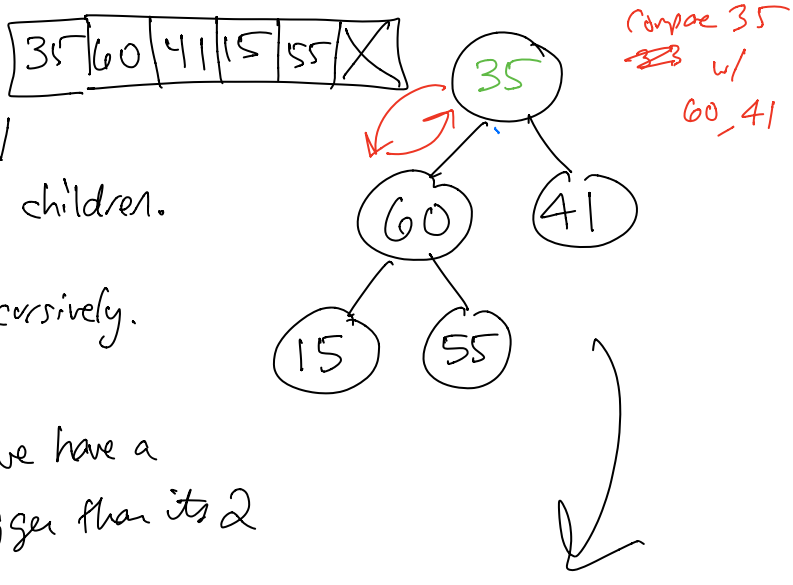


Use an array starting indices from 1.
Each node's children are stored at positions $2x$ & $2x+1$
where x is the position of the parent.

PQ

- deleteMax
  - Replace the item at the top of the tree (root node) w/ the last node in the tree.
  - We run an algorithm called heapdown
    - Exchange the 35 w/ the larger of its children.
      - Apply this recursively.
    - Keep going until we have a parent that is bigger than its 2 children.

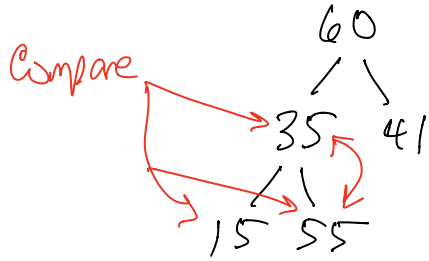


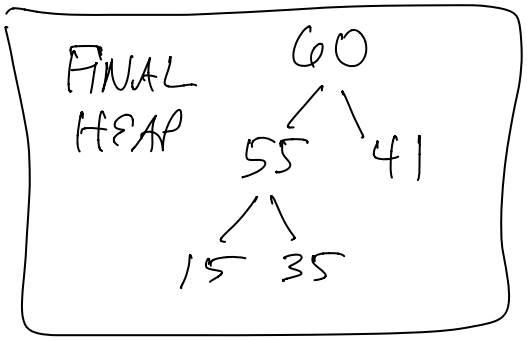


Big-oh?

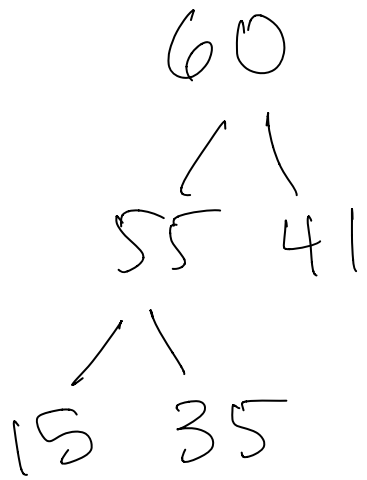
of deleteMax operation?

$O(\log n)$ → Avg & worst case





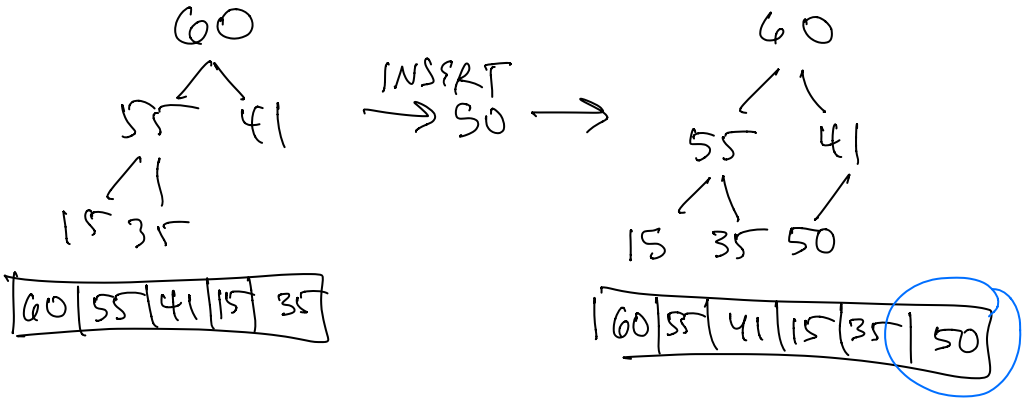
Insert?



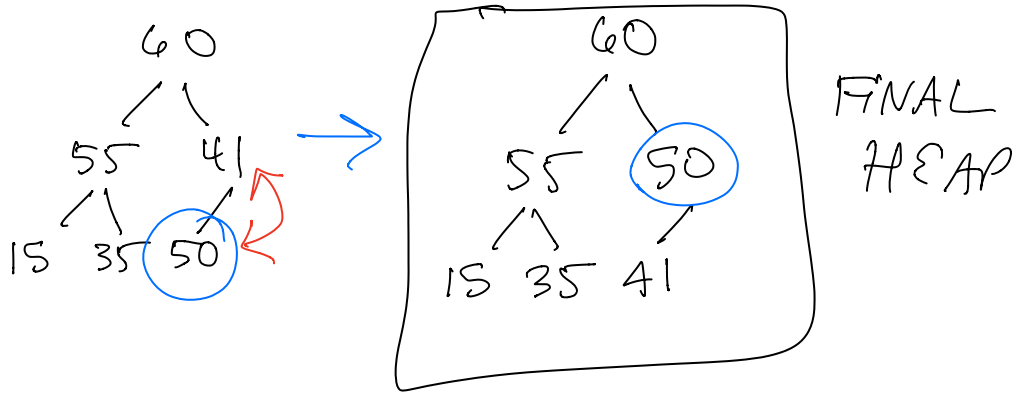
New

50

1. Add new item to the end of the array.



2. Run Heap-Up algorithm.
   - ↳ Swap the newitem w/ its parent as long as it is bigger than its parent.

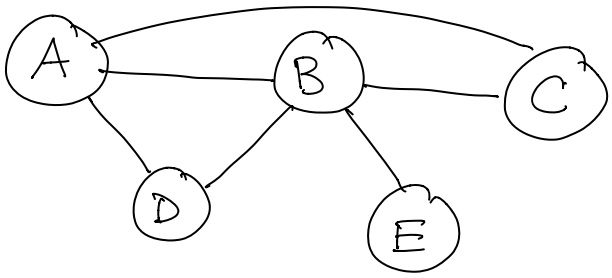


Big-oh?

$O(\log n)$ → worst/avg cases.

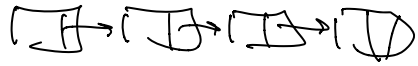
Graph

"Node"



List

- Each item always has one predecessor & one successor



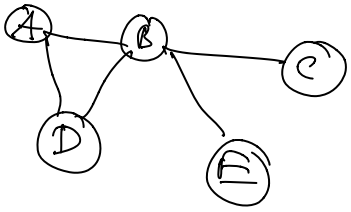
Tree

- Each item has one predecessor (parent) multiple successors

Graph - each node may have multiple predecessors/successors

Def'n of a graph

- Vertices (Nodes)
- Edges (an edge always connects 2 vertices)



What are the vertices?

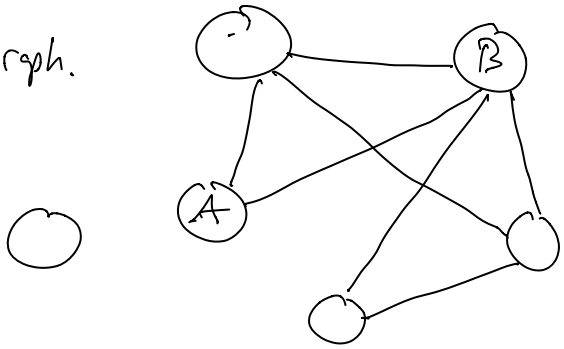
$\{A, B, C, D, E\}$

What are the edges?

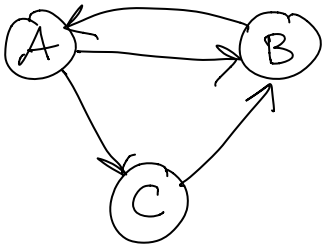
$\{(A, B), (A, D), (B, D), (B, E), (B, C)\}$

Uses of graphs

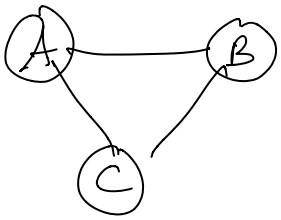
A social network → graph.



Directed

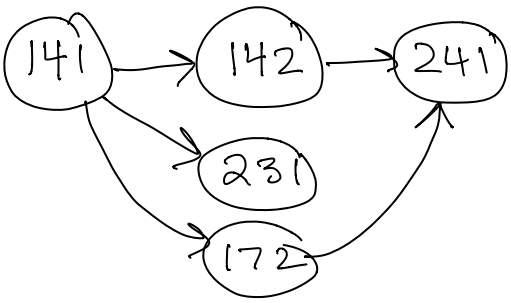


Undirected



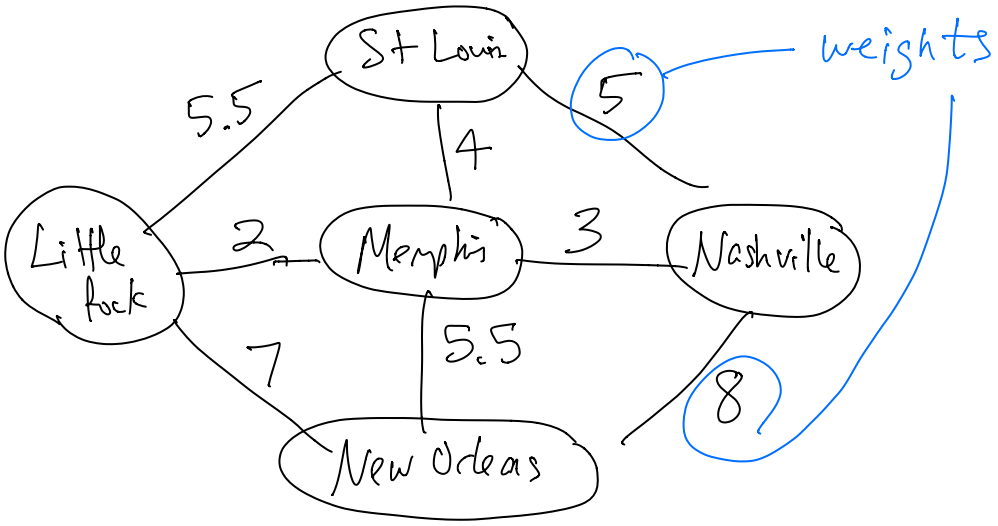
An arrow from X → Y
means "X is following Y on Instagram"

Graph of course pre-requisites



Weighted / Unweighted

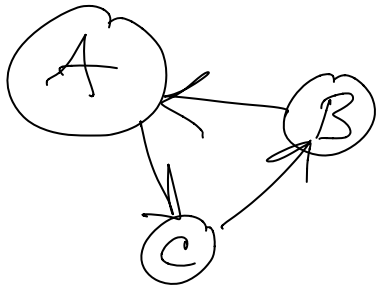
↳ Has a number associated w/ each edge.
(Usually that number is distance, price, time)



Adjacency: for undirected: A — B ⟹ A is adjacent to B
and
B is adjacent to A.

for directed A → B ⟹ B is adjacent to A.
A is not adjacent to B.

Path: A path is a sequence of vertices where each vertex is adjacent to the one that came before.



Cycle: Path where first vertex = last vertex