

# HEAPS

---

,

# Announcements

- PA03 checkpoint due tomorrow (05/31) at midnight. Submit to pa03-checkpoint assignment on gradescope
- PA03 due next Friday 06/08 at midnight. Submit to the pa03 assignment on gradescope
- Final exam on Wed 06/13 (8a -11a)
- Review session on Tuesday (06/05): first session 2:00p to 3:00p and the other from 3:00p to 4:00p

# How is PA03 going?

- A. Done
- B. On track to finish
- C. Having trouble with the checkpoint (design)
- D. Just started
- E. Haven't started

# Heaps: Supported Operations

Min-Heaps

Max-Heap

BST

- Insert :
- Min:
- Delete Min:
- Max
- Delete Max

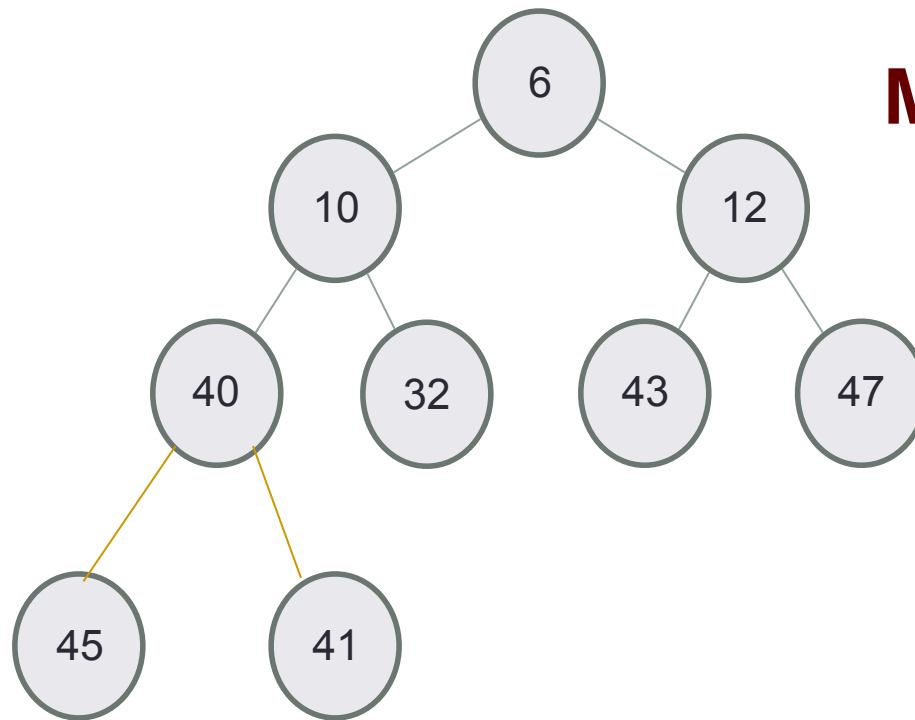
**Choose heap if you are doing repeated insert/delete/(min OR max) operations**

## **Applications:**

- Efficient sort
- Finding the median of a sequence of numbers
- Compression codes

# Heaps as binary trees

- Rooted binary tree that is as complete as possible
- In a **min-Heap**, each node satisfies the following **heap property**:  
 $\text{key}(x) \leq \text{key}(\text{children of } x)$

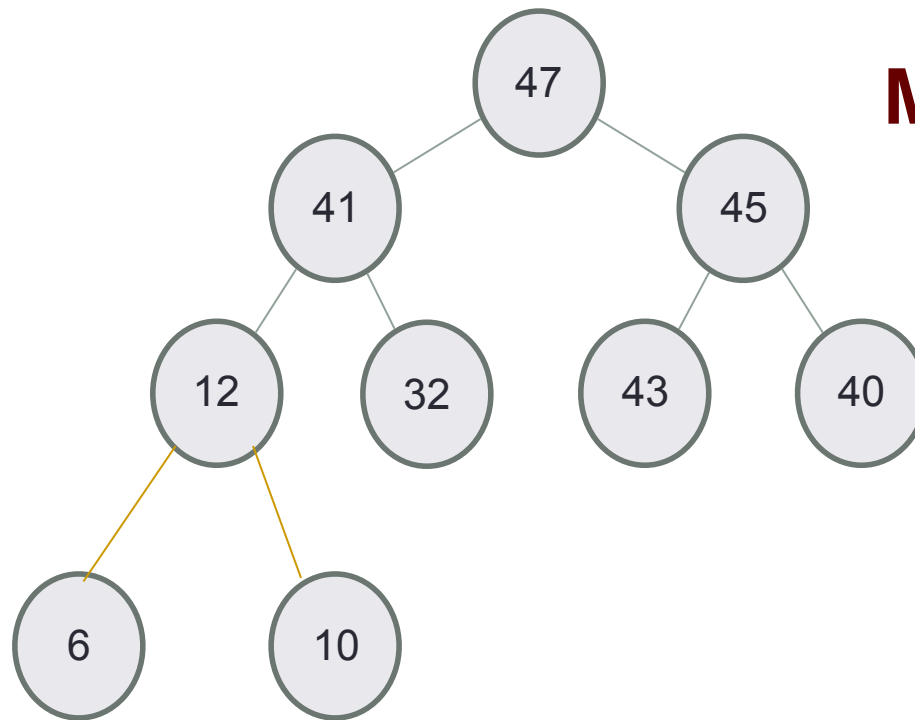


**Min Heap with 9 nodes**

Where is the minimum element?

# Heaps as binary trees

- Rooted binary tree that is as complete as possible
- In a max-Heap, each node satisfies the following **heap property**:  
 $\text{key}(x) \geq \text{key}(\text{children of } x)$



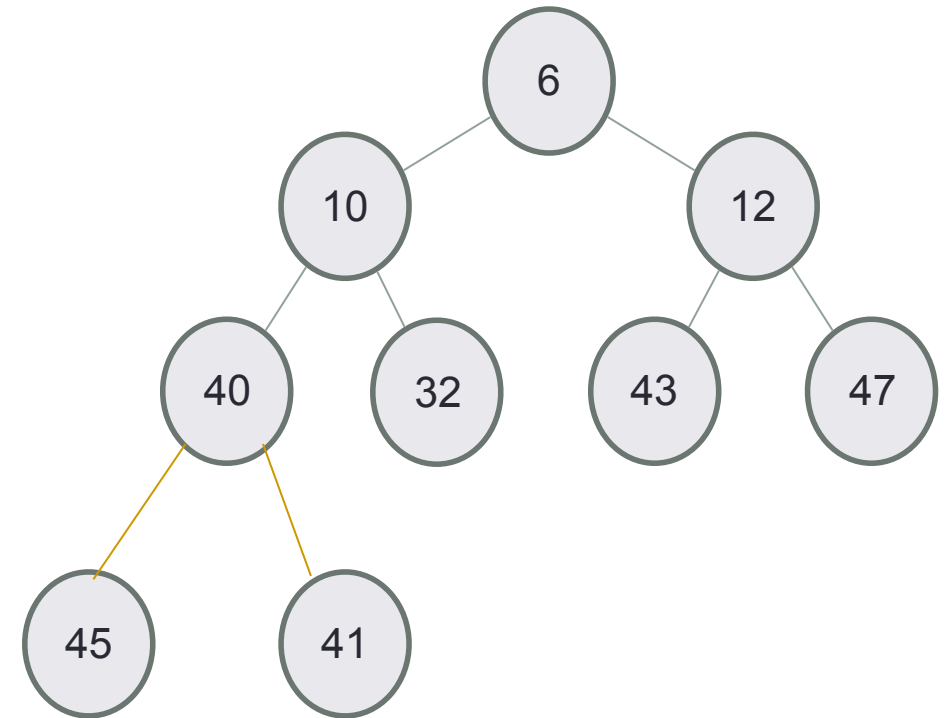
**Max Heap with 9 nodes**

Where is the maximum element?

# Identifying heaps

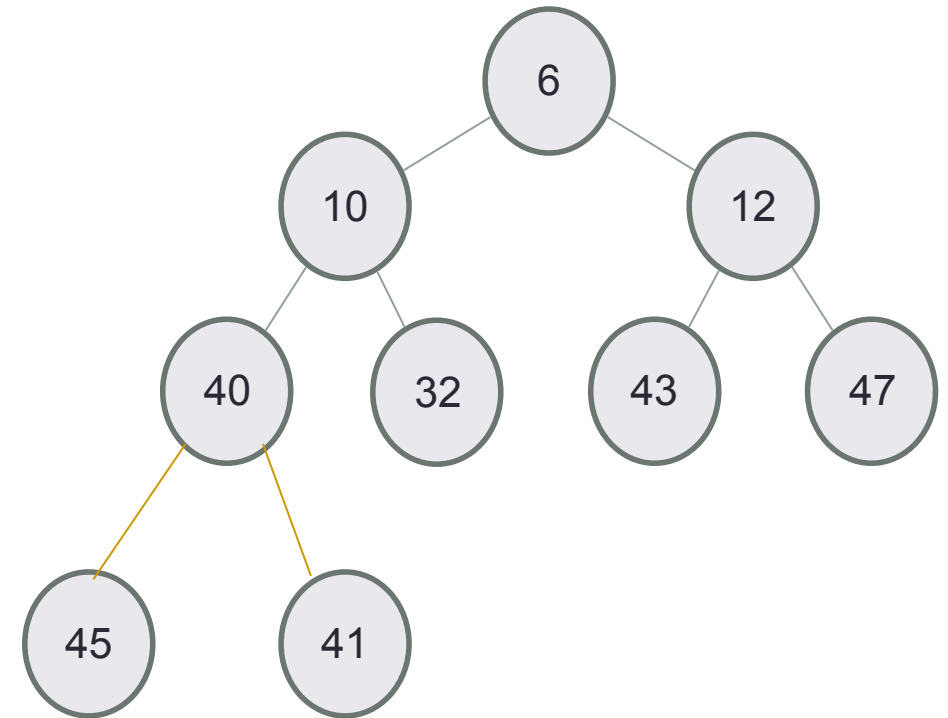
Starting with the following min Heap which of the following operations will result in something that is NOT a min Heap

- A. Swap the nodes 40 and 32
- B. Swap the nodes 32 and 43
- C. Swap the nodes 43 and 40
- D. Insert 50 as the left child of 45
- E. C&D



# Structure: Complete binary tree

**A heap is a complete binary tree: Each level is as full as possible.  
Nodes on the bottom level are as far left as possible**





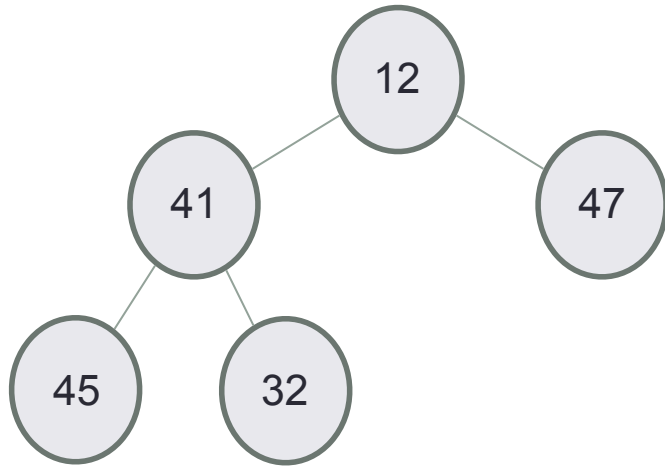
# Insert into a heap

- Insert key(x) in the first open slot at the last level of tree (going from left to right)
- If the heap property is not violated - Done

Insert the elements {12, 41, 47, 45, 32} in a min-Heap

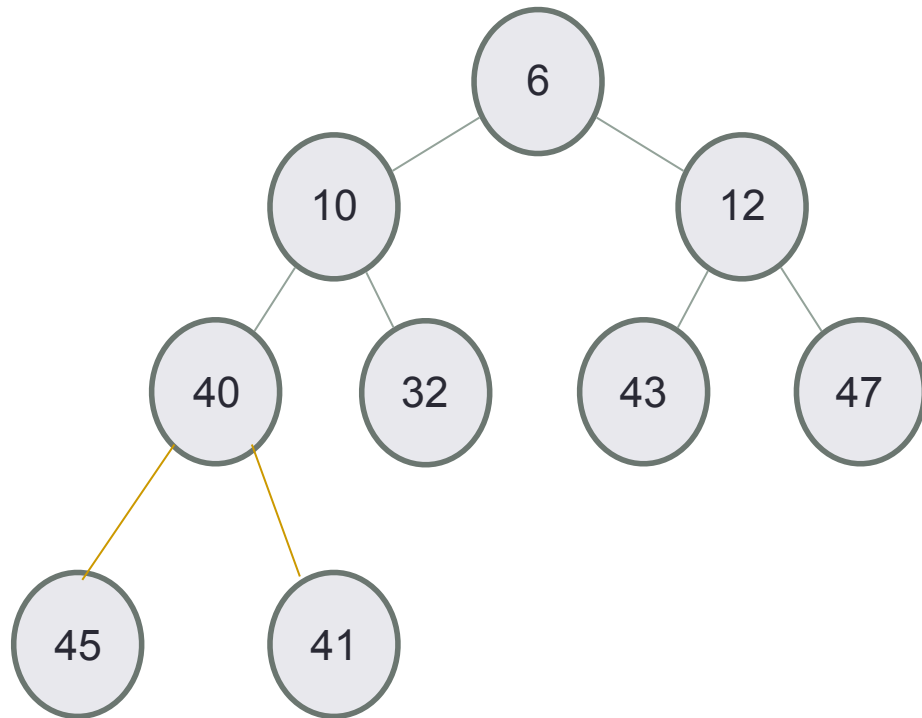
# Insert 32 into a heap

- Insert  $\text{key}(x)$  in the first open slot at the last level of tree (going from left to right)
- If the heap property is not violated - Done
- Else: while( $\text{key}(\text{parent}(x)) > \text{key}(x)$ ) swap the  $\text{key}(x)$  with  $\text{key}(\text{parent}(x))$



# Implementing heaps as arrays

Value										
Index	0	1	2	3	4	5	6	7	8	

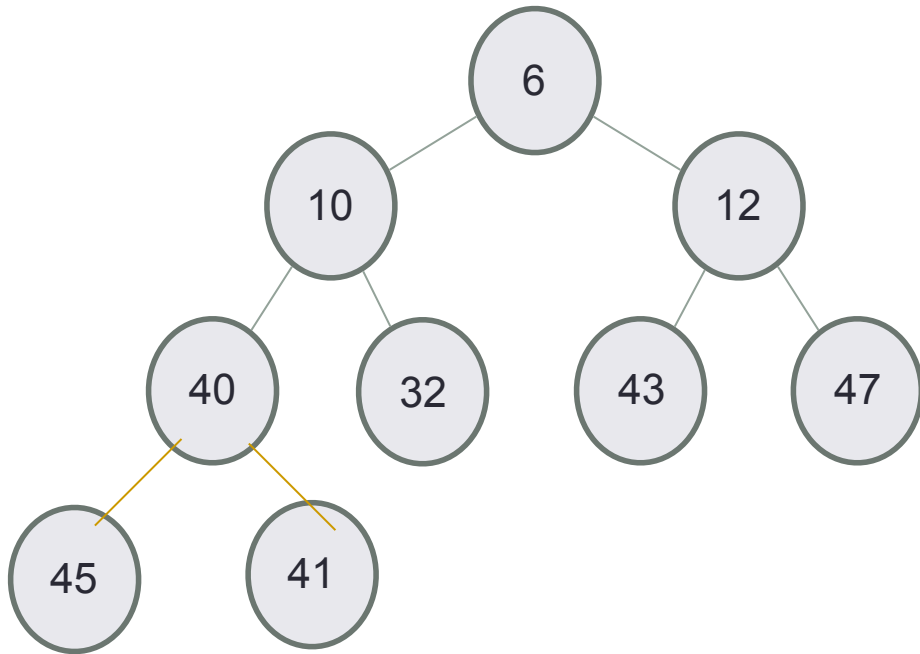


How is the array implementation of the heap useful?

- More space efficient
- Accessing parent and children of a node is  $O(1)$
- Easier to insert elements in the heap

## Conceptualize heaps as trees, implement as arrays

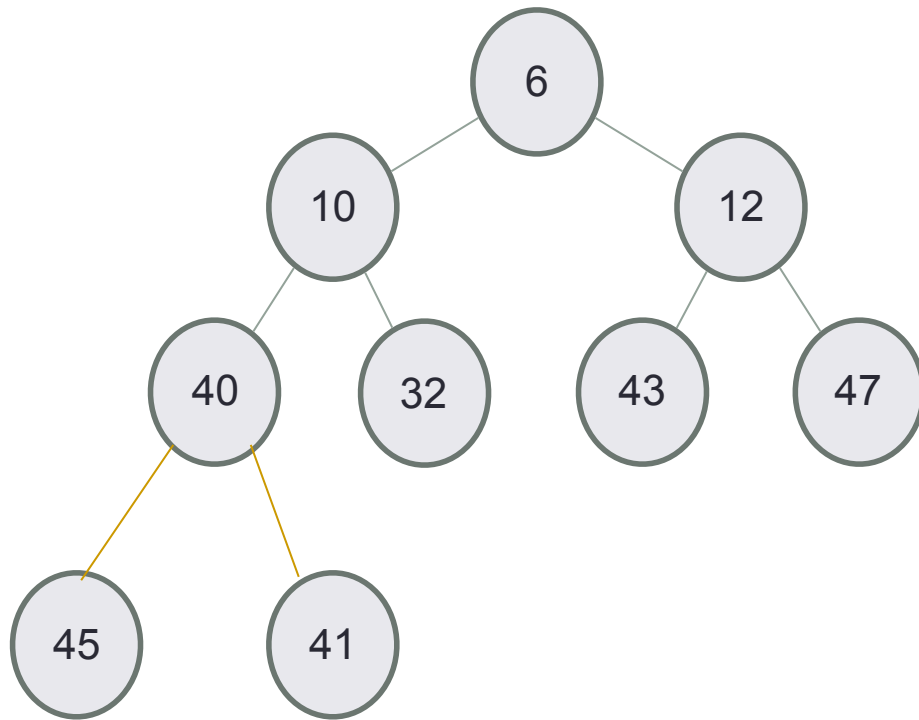
Value	6	10	12	40	32	43	47	45	41	
Index	0	1	2	3	4	5	6	7	8	



For a node at index  $i$ , what is the index of the left and right children?

- A.  $(2*i, 2*i+1)$
- B.  $(2*i+1, 2*i+2)$
- C.  $(\log(i), \log(i)+1)$
- D. None of the above

# Conceptualize heaps as trees, implement as arrays



**For a node at index  $i$ , index of the parent is:**

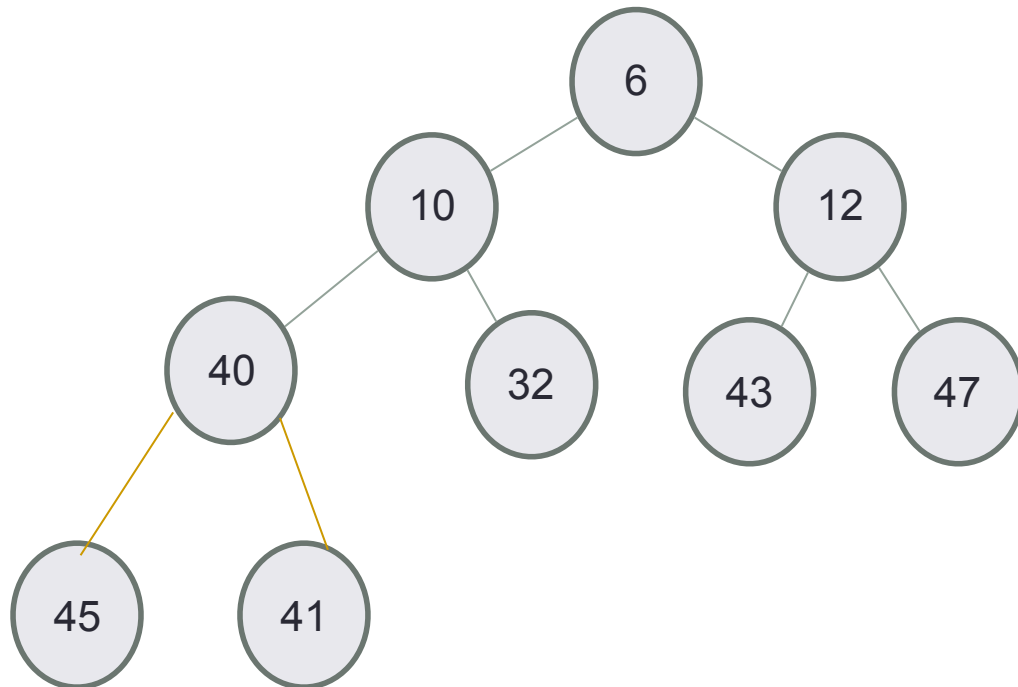
**$i/2 - 1$ ,      if  $i$  is even**

**$(i-1)/2$ ,      if  $i$  is odd**

Value	Index	Index of parent	Index of children
6	0	-	1, 2
10	1	0	3, 4
12	2	0	5, 6
40	3	1	7, 8
32	4	1	
43	5	2	
47	6	2	
45	7	3	
41	8	3	

# Insert 50, then 35

Value	6	10	12	40	32	43	47	45	41	
Index	0	1	2	3	4	5	6	7	8	



For a node at index  $i$ ,

\* index of the parent is:

$i/2 - 1$ , if  $i$  is even

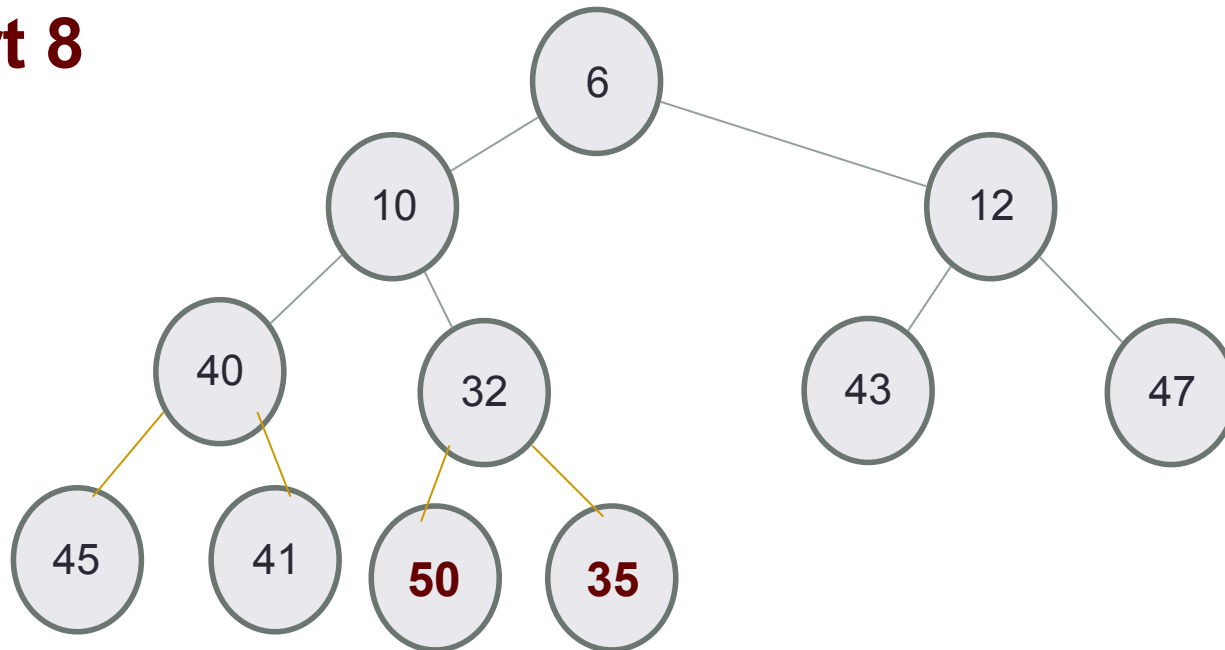
$(i-1)/2$ , if  $i$  is odd

\* index of the children are  $(2i+1, 2i+2)$

# Insert 8 into a heap

Value	6	10	12	40	32	43	47	45	41	50	35
Index	0	1	2	3	4	5	6	7	8	9	10

## Insert 8



Which node is the parent of 8 after the insertion has been completed

- A. Node 6
- B. Node 12
- C. None 43
- D. None - Node 8 will be the root

**For a node at index  $i$ ,**

**\* index of the parent is:**

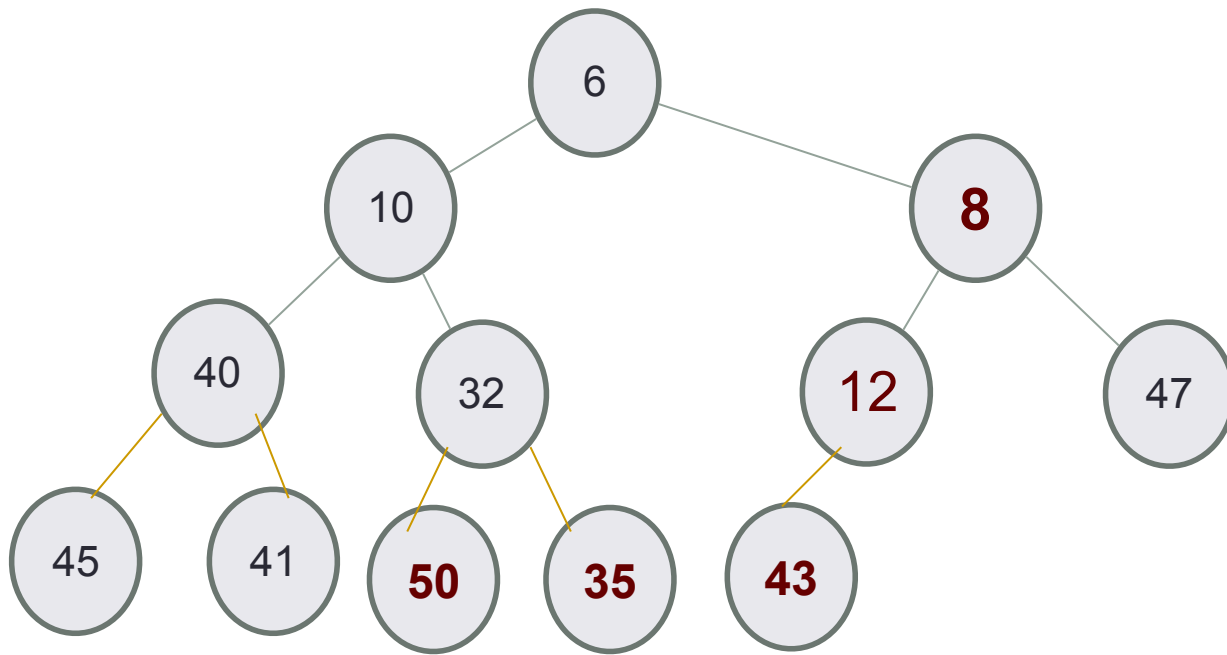
**$i/2 - 1$ , if  $i$  is even**

**$(i-1)/2$ , if  $i$  is odd**

**\* index of the children are  $(2i+1, 2i+2)$**

# Delete min

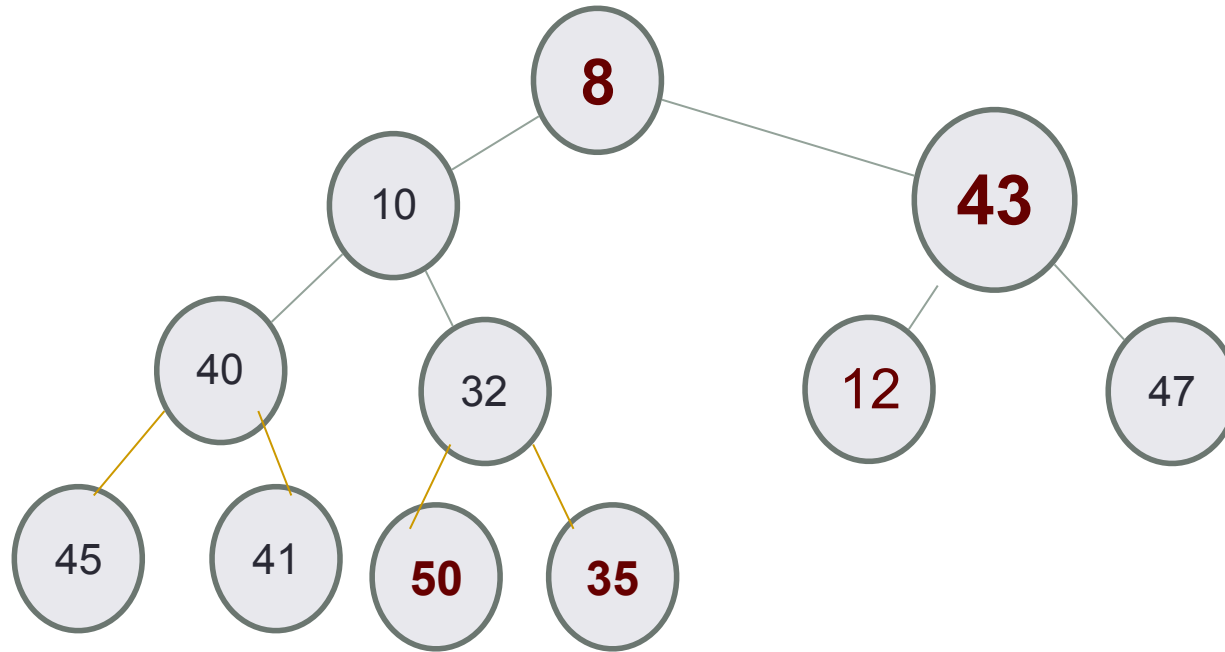
- Replace the root with the rightmost node at the last level
- “Bubble down”- swap node with one of the children until the heap property is restored





# Delete min

- Delete the root:
  - Replace the root with the last node in the array
  - If heap property is violated - swap with the child that has the **LOWEST** key value, repeat until heap property is restored



To fix the heap property on a delete **BUBBLE DOWN!**  
Worst case:  $O(\log N)$

# Applications

- Efficient sort
- Finding the median of a sequence of numbers
- Compression codes