

QUEUES AND HEAPS

Problem Solving with Computers-II

C++

```
#include <iostream>
using namespace std;

int main(){
    cout<<"Hola Facebook\n";
    return 0;
}
```

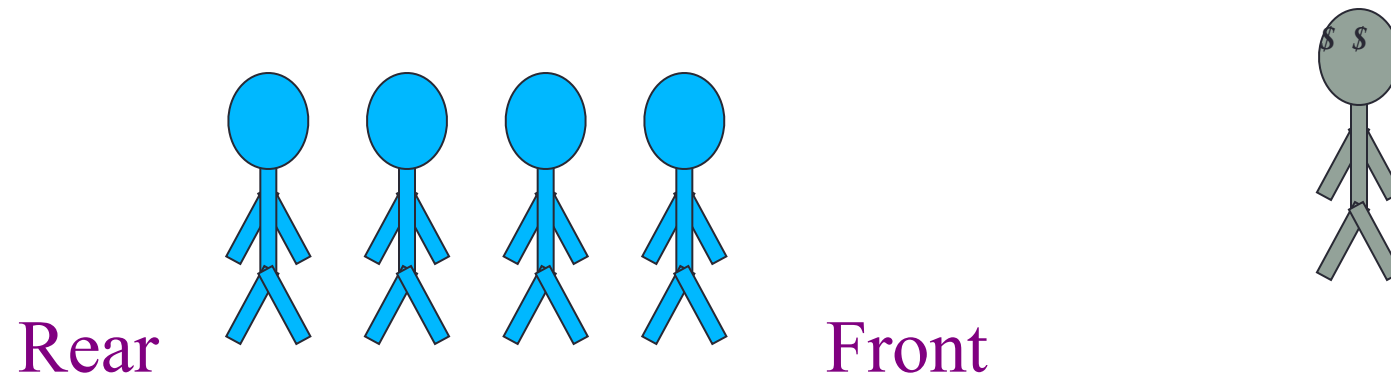


How is PA02 going?

- A. Done
- B. On track to finish
- C. Having some difficulties
- D. Just started
- E. Haven't started

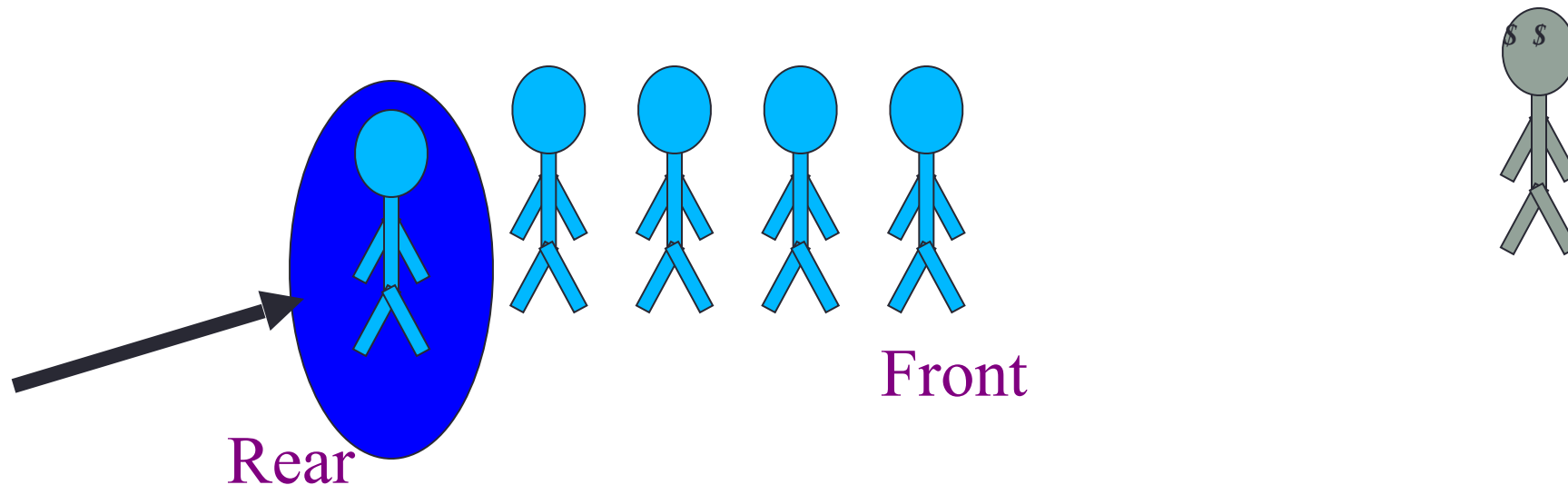
The Queue Operations

- A queue is like a line of people waiting for a bank teller.
- The queue has a front and a rear.



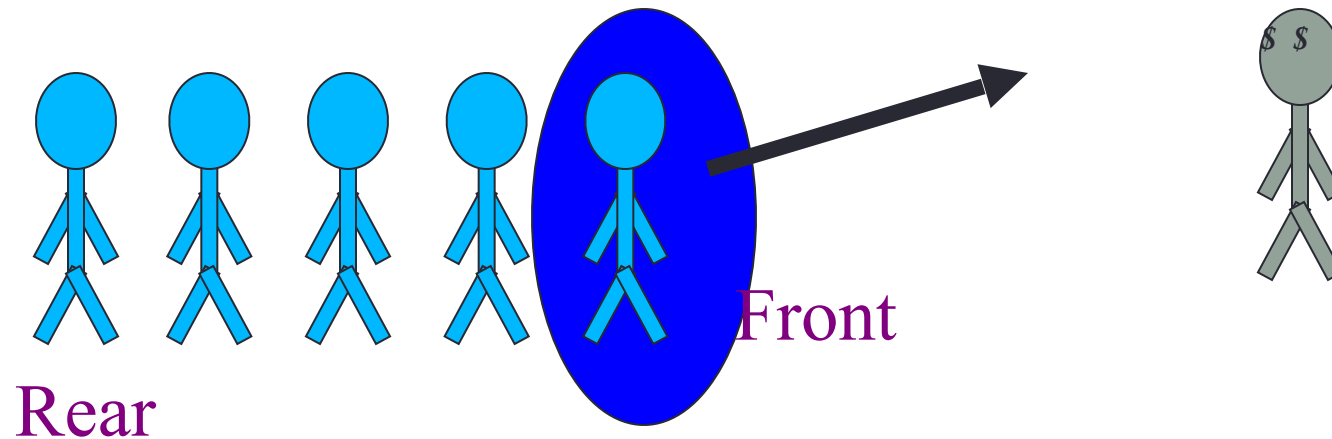
The Queue Operations

- New people must enter the queue at the rear. The C++ queue class calls this a push, although it is usually called an enqueue operation.



The Queue Operations

- When an item is taken from the queue, it always comes from the front. The C++ queue calls this a pop, although it is usually called a dequeue operation.



The Queue Class

- The C++ standard template library has a queue template class.
- The template parameter is the type of the items that can be put in the queue.

```
template <class Item>  
class queue<Item>  
{  
public:  
    queue( );  
    void push(const Item& entry);  
    void pop( );  
    bool empty( ) const;  
    Item front( ) const;  
    ...
```

Heaps

- Clarification
 - *heap*, the data structure is not related to *heap*, the region of memory
- What are the operations supported?
- What are the running times?

Heaps

Min-Heaps

Max-Heap

BST

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

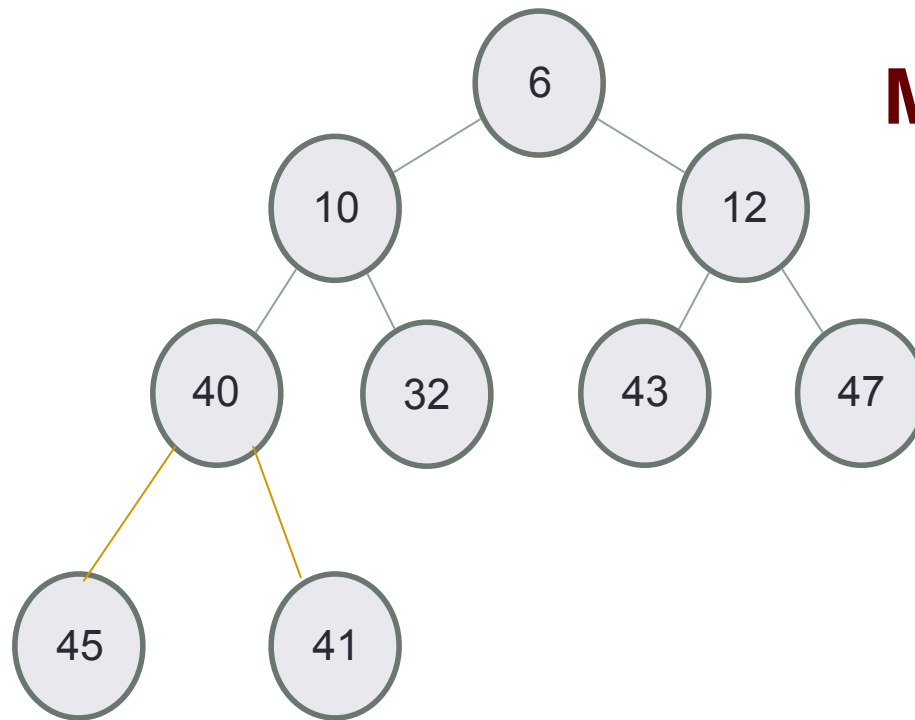
Choose heap if you are doing repeated insert/delete/(min OR max) operations

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**:
 $\text{key}(x) \leq \text{key}(\text{children of } x)$

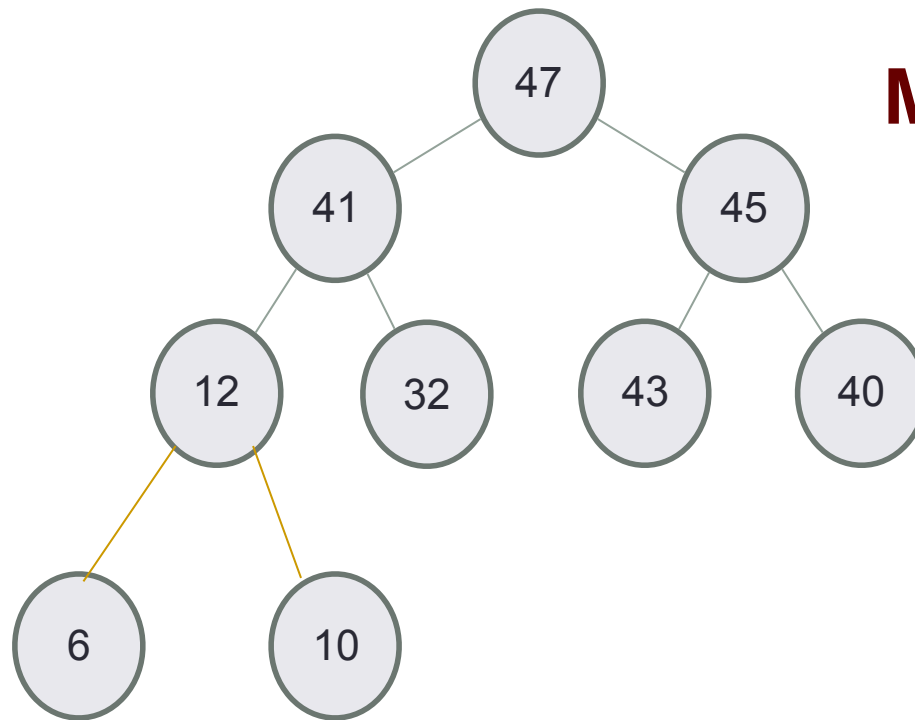


Min Heap with 9 nodes

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**:
 $\text{key}(x) \geq \text{key}(\text{children of } x)$



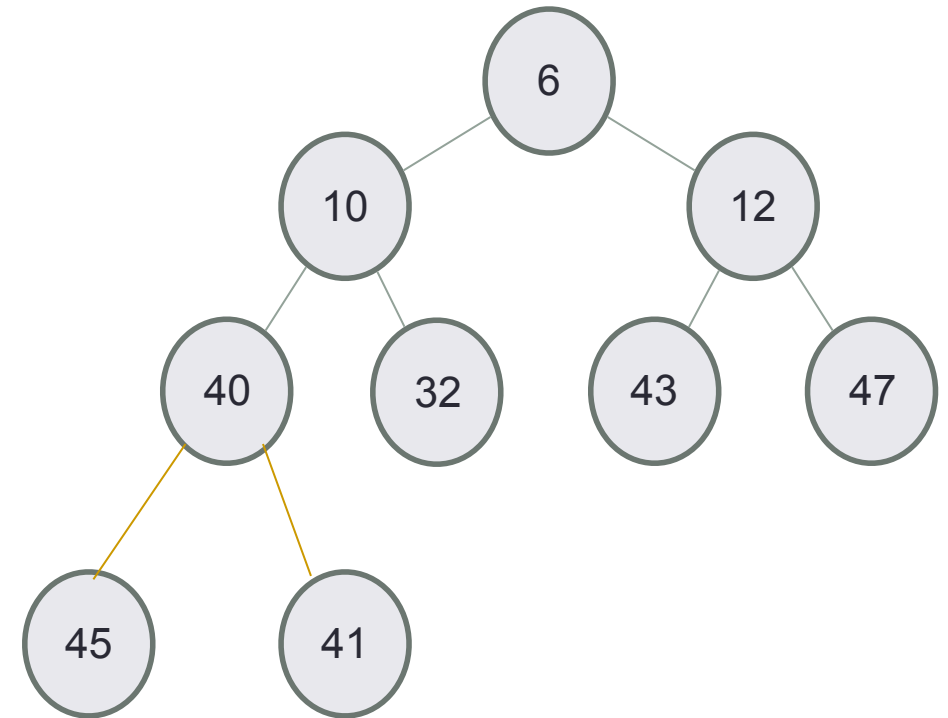
Max Heap with 9 nodes

Where is the maximum element?

Identifying heaps

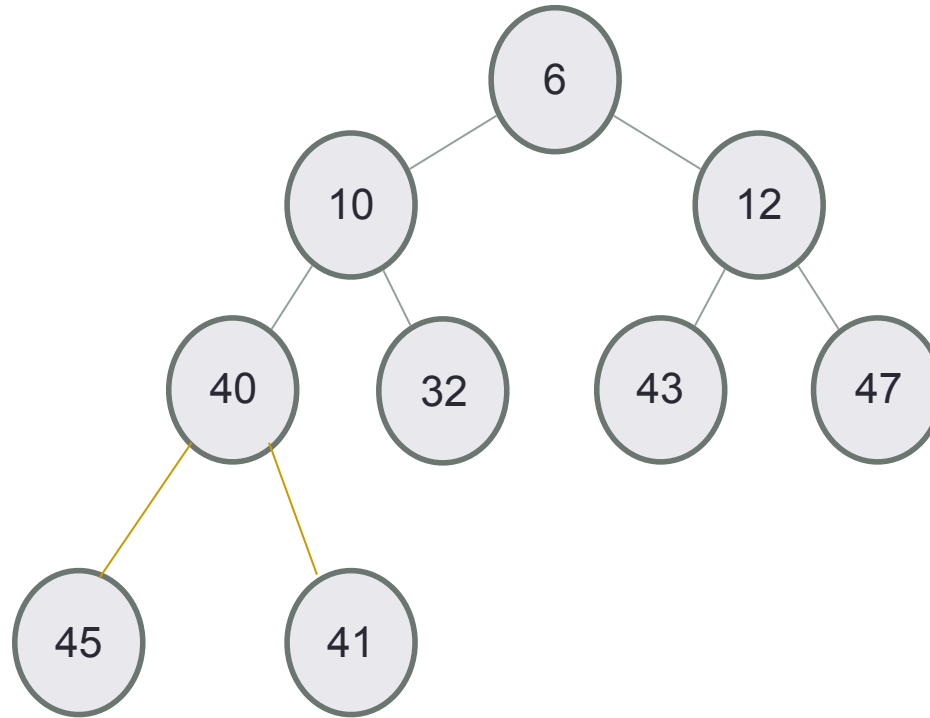
Starting with the following min Heap which of the following operations will result in something that is NOT a min Heap

- 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



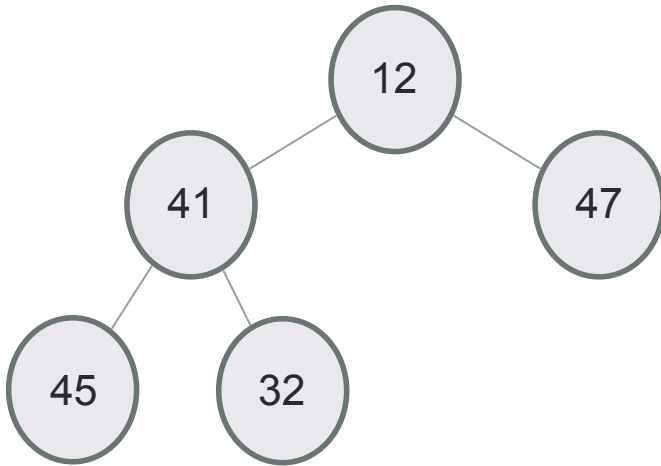
Structure: Complete binary tree

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

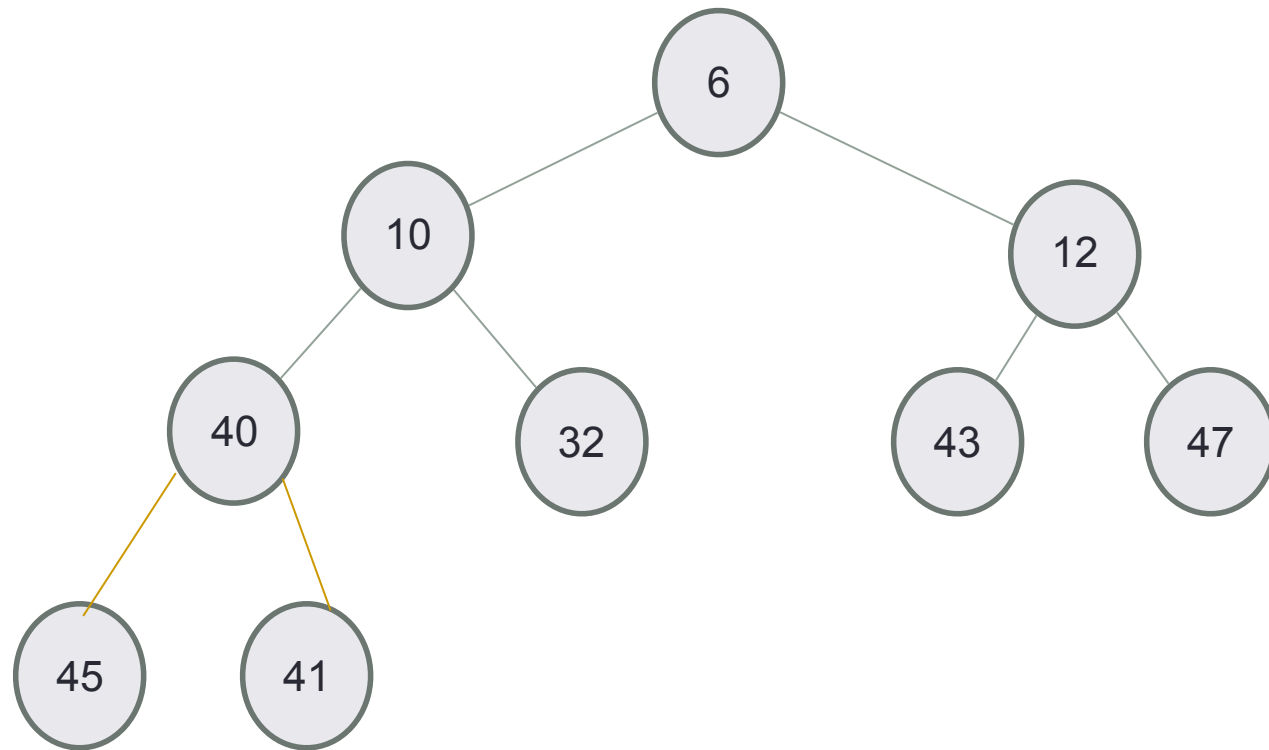


Insert 32 into a heap

- 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: $\text{while}(\text{key}(\text{parent}(x)) > \text{key}(x))$ swap the $\text{key}(x)$ with $\text{key}(\text{parent}(x))$

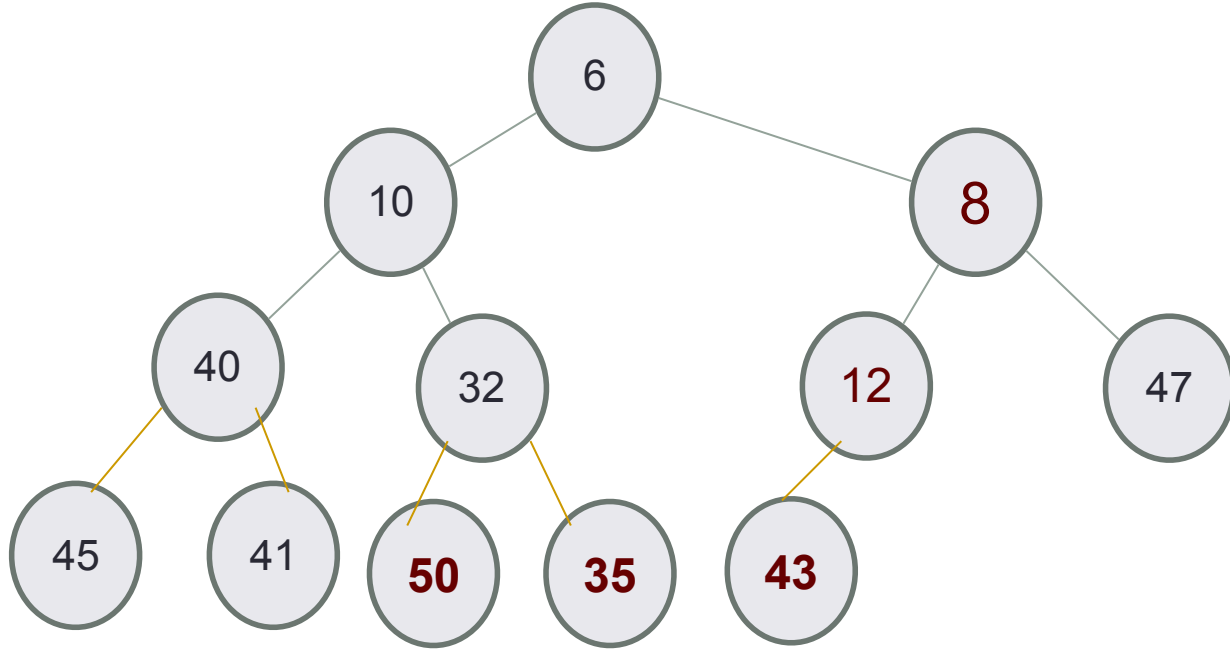


Insert 50, then 35, then 8



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

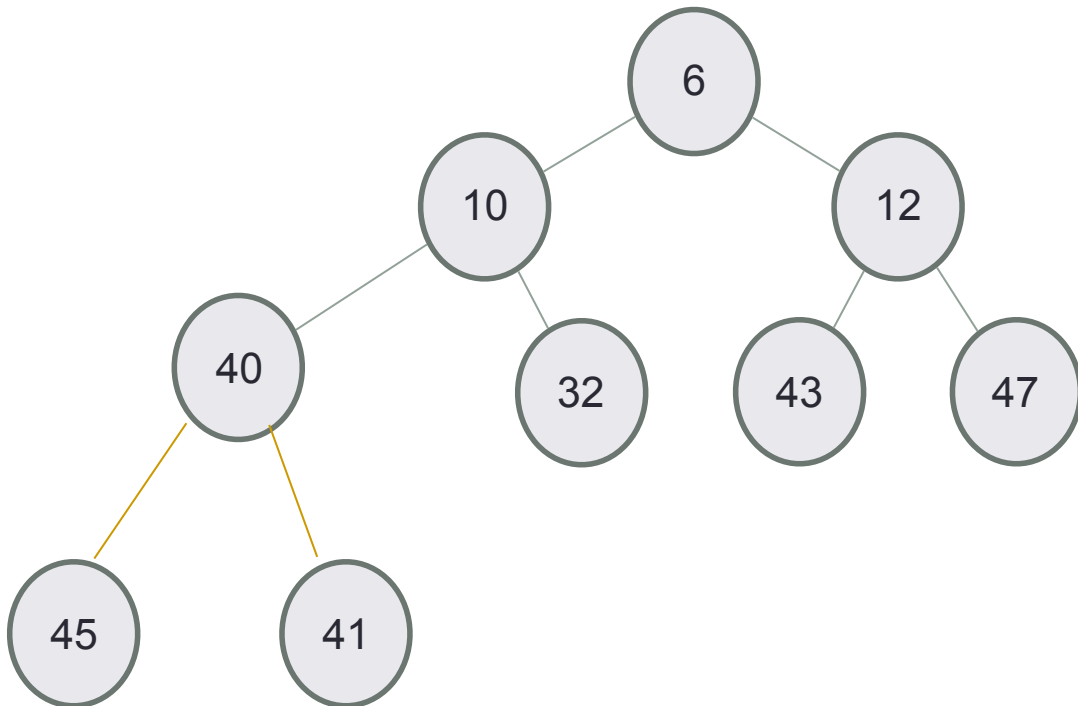


Under the hood of heaps

- An efficient way of implementing heaps is using vectors
- Although we think of heaps as trees, the entire tree can be efficiently represented as a vector!!

Implementing heaps using an array or vector

Value										
Index	0	1	2	3	4	5	6	7	8	



Using vector as the internal data structure of the heap has some advantages:

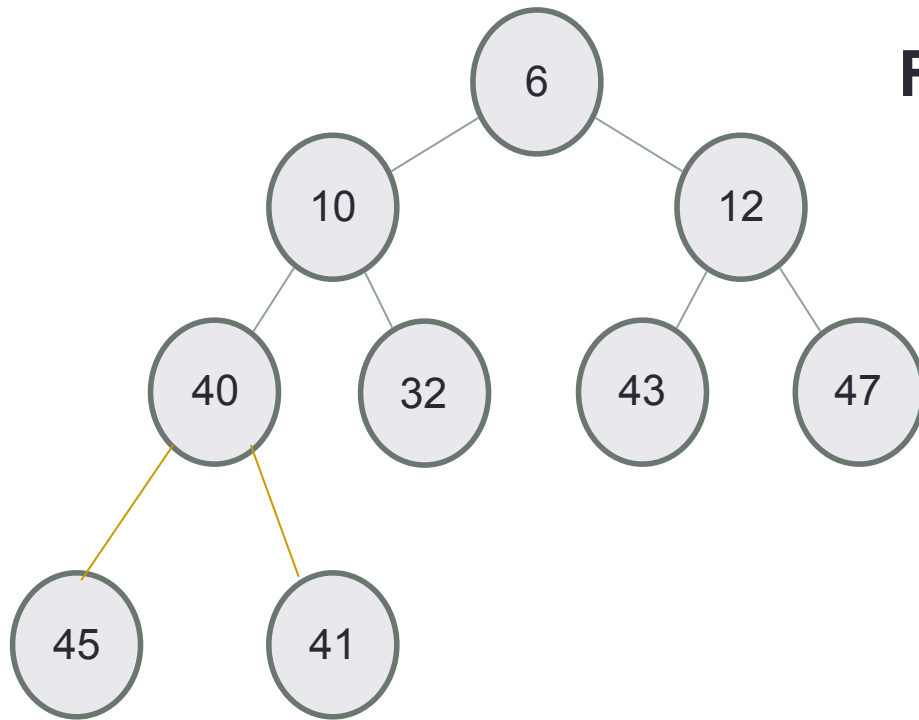
- **More space efficient than trees**
- **Easier to insert nodes into the heap**

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
- Else....

Insert the elements {12, 41, 47, 45, 32} in a min-Heap using the vector representation of the heap

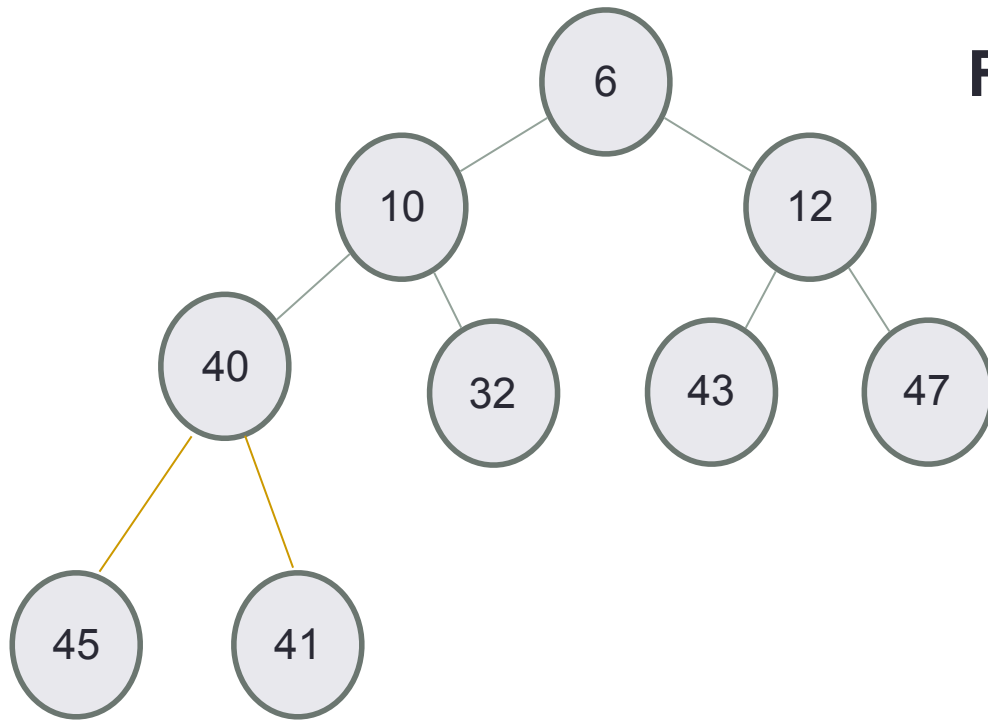
Traversing up the “tree”



For a node at index i , index of the parent is $\text{int}(i/2)$

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

Insert 50, then 35



For a node at index i , index of the parent is $\text{int}(i/2)$

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

Insert 8 into a heap

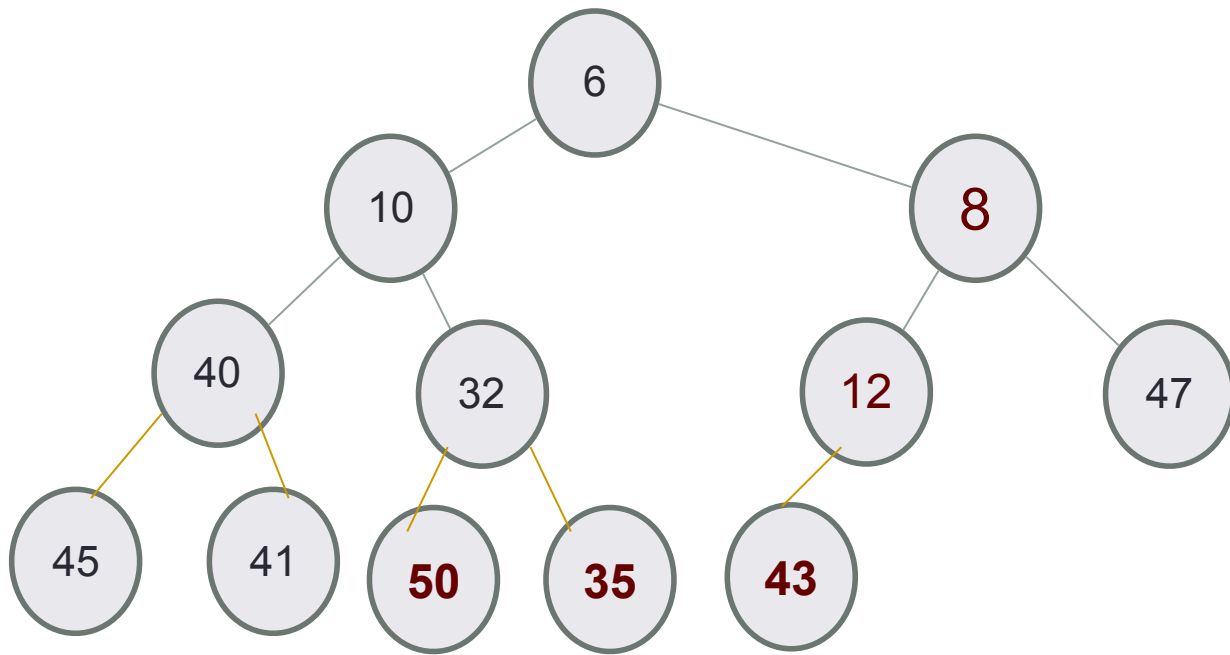
Value	6	10	12	40	32	43	47	45	41	50	35
Index	0	1	2	3	4	5	6	7	8	9	10

After inserting 8, which node is the parent of 8 ?

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

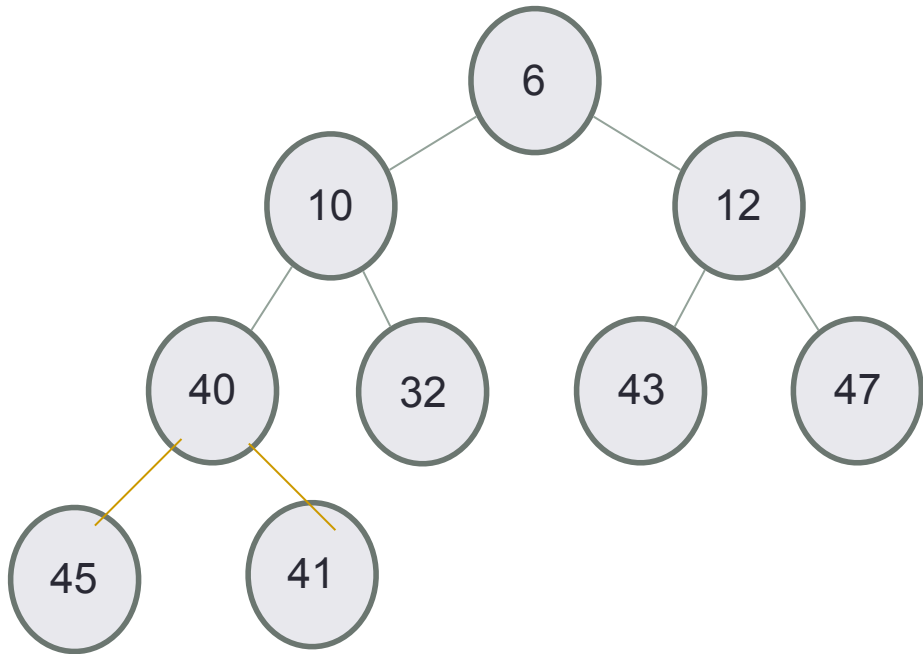
Delete min

- 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



Traversing down the tree

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

Next lecture

- **STL implementation of heaps (priority queues)**