Chapter 7
Priority Queue

By: Dr. Aryaf A. Al-adwan
Faculty of Engineering Technology
Computer and Networks Engineering Dept.
Data Structures Course

The ADT Priority Queue

- A ***priority queue*** is an ADT in which items are ordered by a priority value. The item with the *highest priority* is always the *next* to be removed from the queue. (Highest Priority In, First Out: *HPIFO*)
- Highest priority can be either the minimum or maximum value in the queue
- We will assume the highest priority is the minimum
- So our priority queue will be least – first - out

PQ Operations

- Priority Queue operations:

  1 – **isEmpty**: return if the queue is empty or not

  2 – **Add** : insert elements into the queue

  3 – **PeekMin** : return the smallest element in the queue

  4 – **RemoveMin** : remove the smallest element in the queue

PQ Implementations

- The problem with a priority queue is in finding efficient implementations which allows fast enqueing and dequeing.
- Representing a PQ using:

  1 – Unsorted array

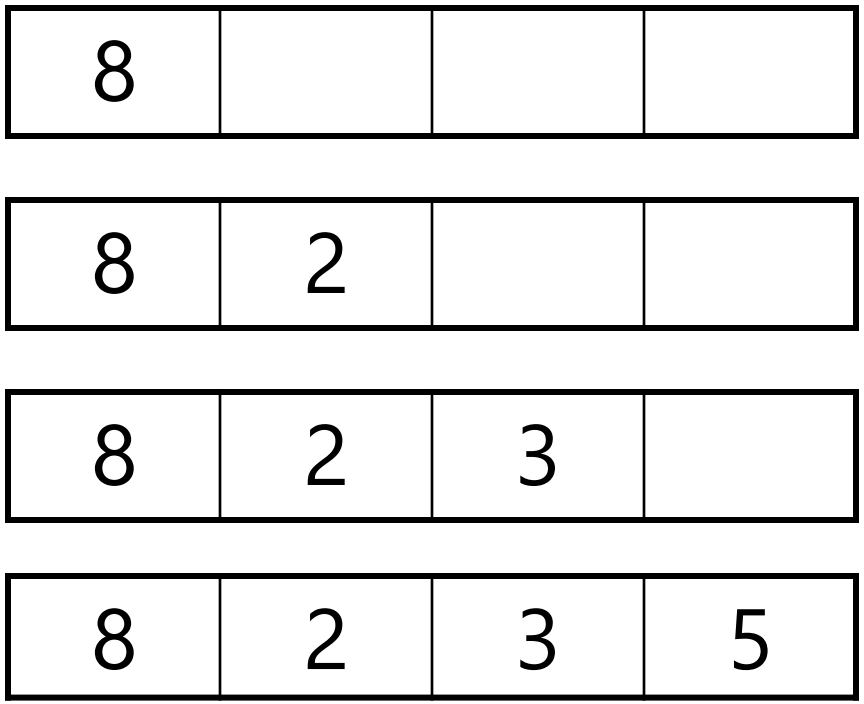
  2 – Sorted array

  3 – LinkedList

  4 - Heaps

Representation # 1

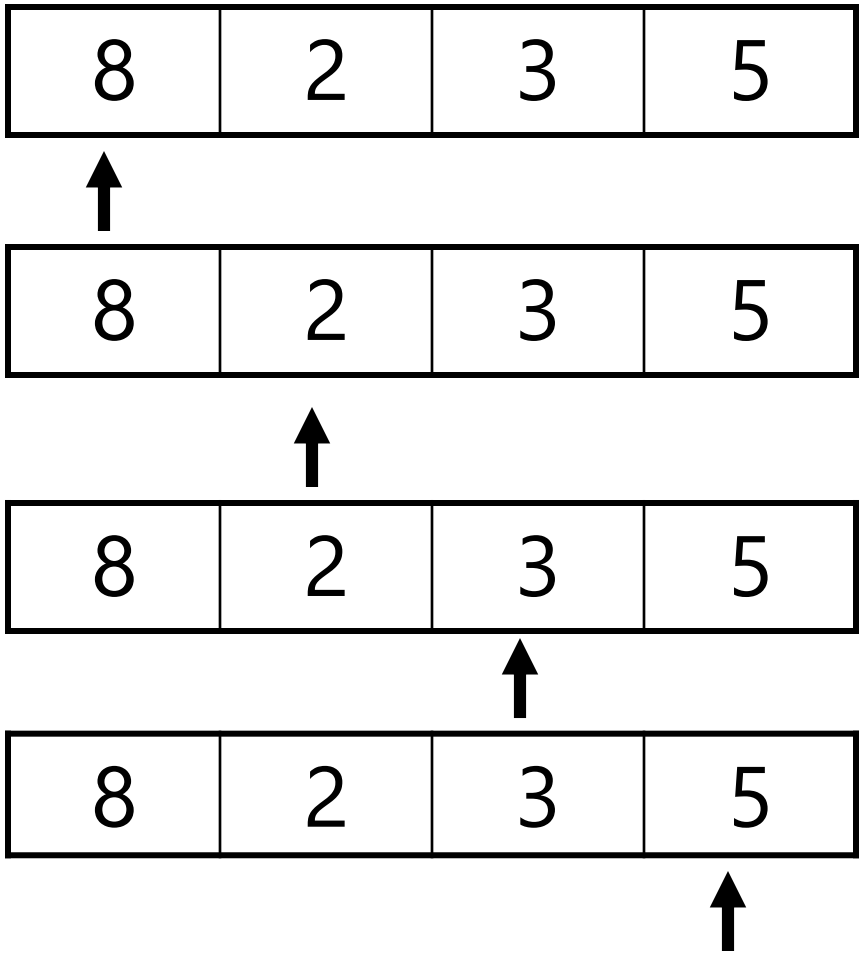
- Representing PQ using **unsorted array**
- **Add ➔** 8 , 2 , 3 , 5



Insert at specified index will cost ➔ 1

Representation # 1

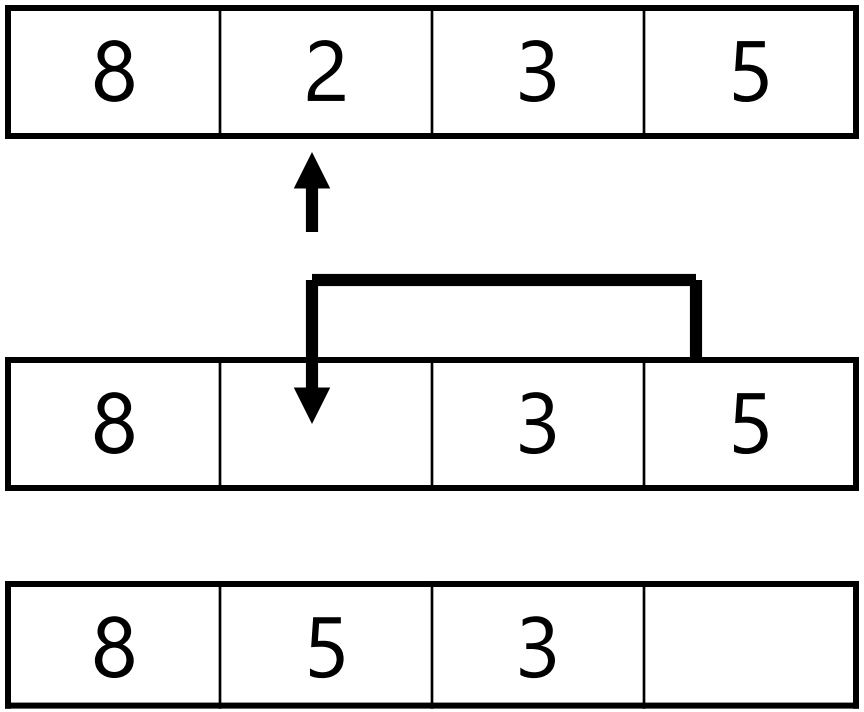
- **peekMin➔** 8 , 2 , 3 , 5



Search all the elements for the minimum value will cost ➔ n

Representation # 1

- **removeMin➔** 8 , 2 , 3 , 5



No need for shifting here

Find the minimum value will cost ➔ n

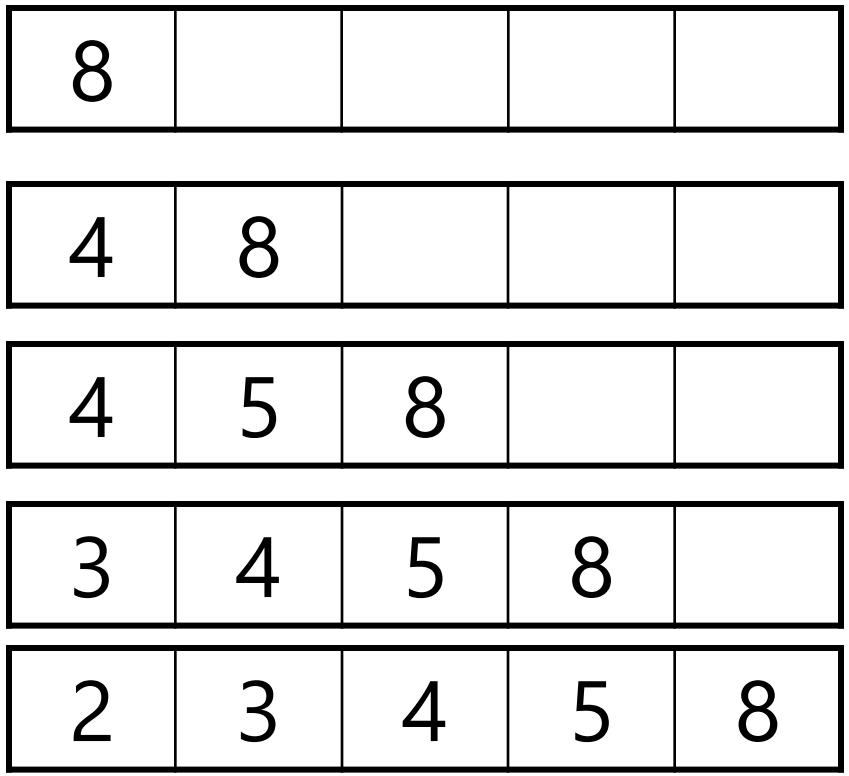
Remove minimum will cost ➔ 1

Complexity Analysis

Operation	Running time
Add	O (1)
peekMin	O (n)
removeMin	O (n)

Representation # 2

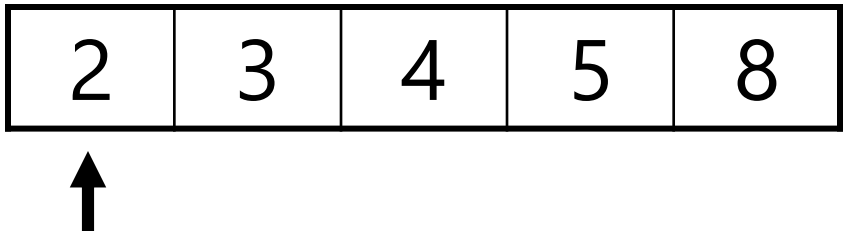
- Representing PQ using **Sorted Array in Increasing Order**
- **Add ➔** 8 , 4 , 5 , 3 , 2



Maintain order by shifting elements will cost ➔ n

Representation # 2

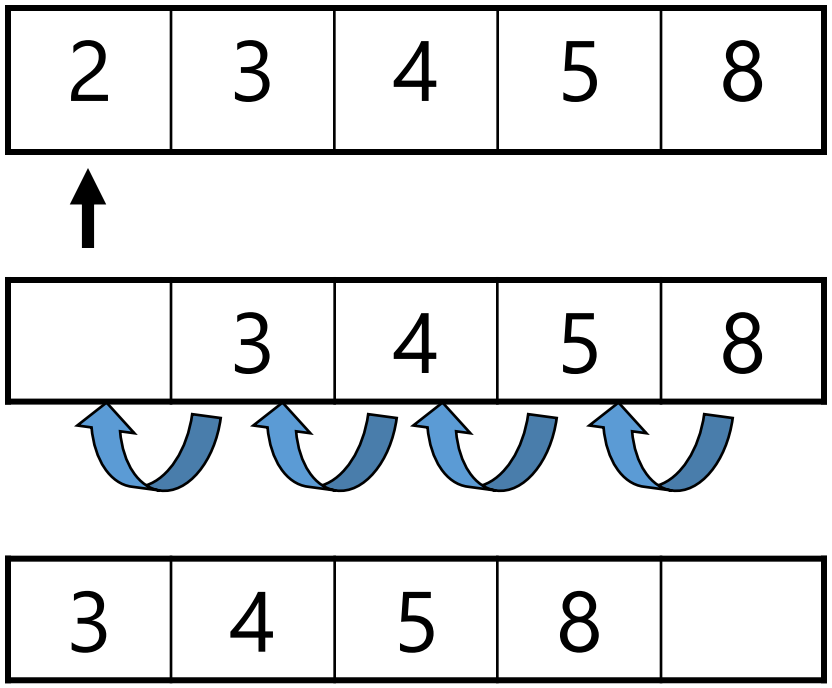
- **peekMin➔** 8 , 4 , 5 , 3 , 2



Return first value will cost ➔ 1

Representation # 2

- **removeMin➔** 8 , 4 , 5 , 3 , 2



Need for shifting here

Find the minimum value will cost ➔ 1

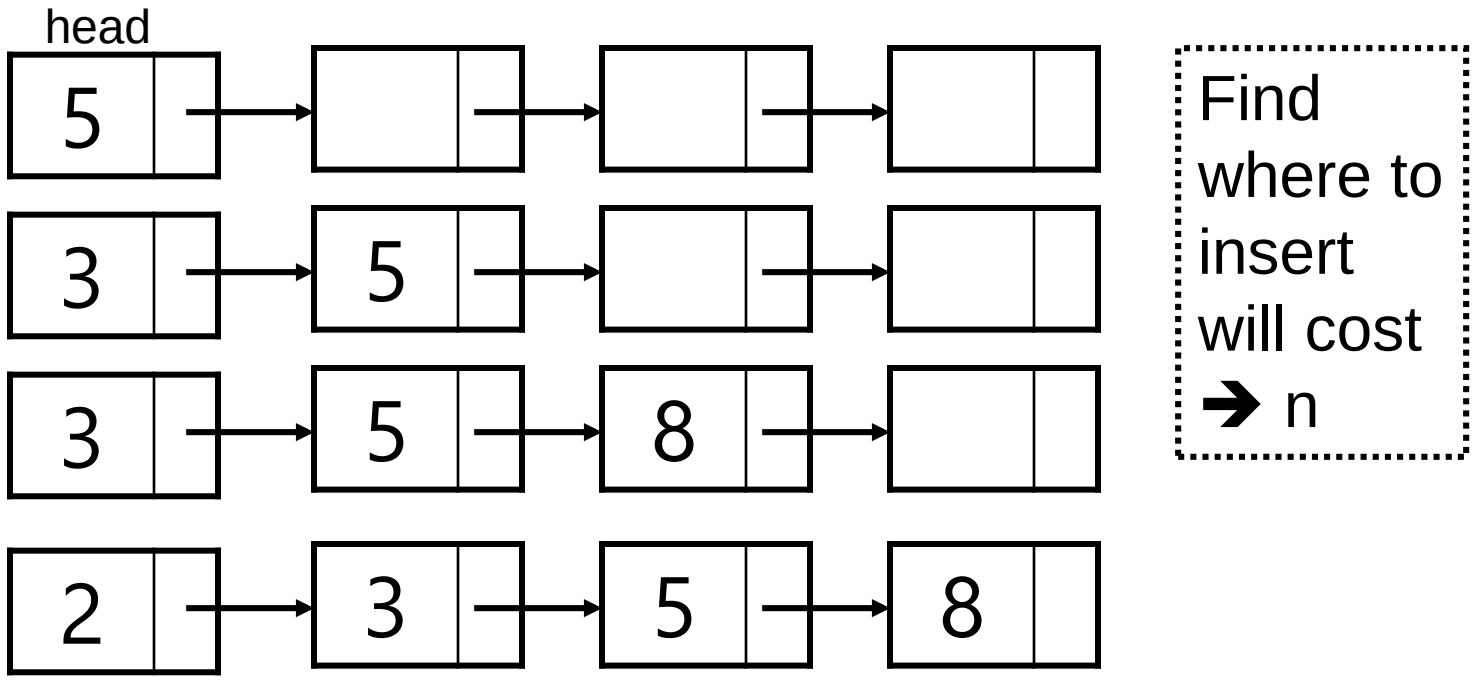
Remove minimum by shifting all values will cost ➔ n

Complexity Analysis

Operation	Running time
Add	O (n)
peekMin	O (1)
removeMin	O (n)

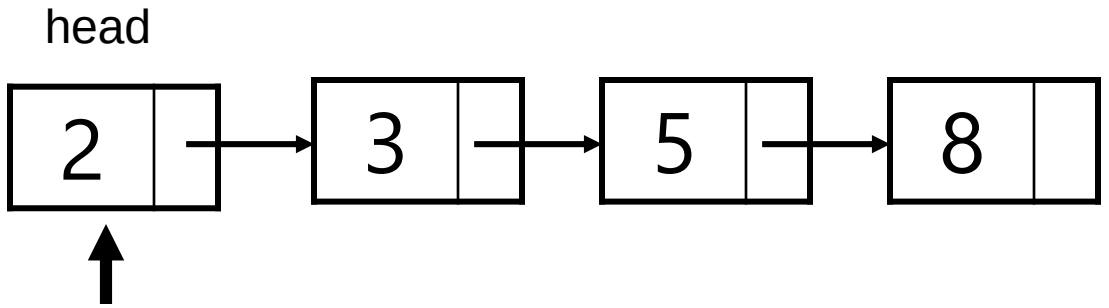
Representation # 3

- Representing PQ using **linked list in increasing order**
- **Add** ➔ 5 , 3 , 8 , 2



Representation # 3

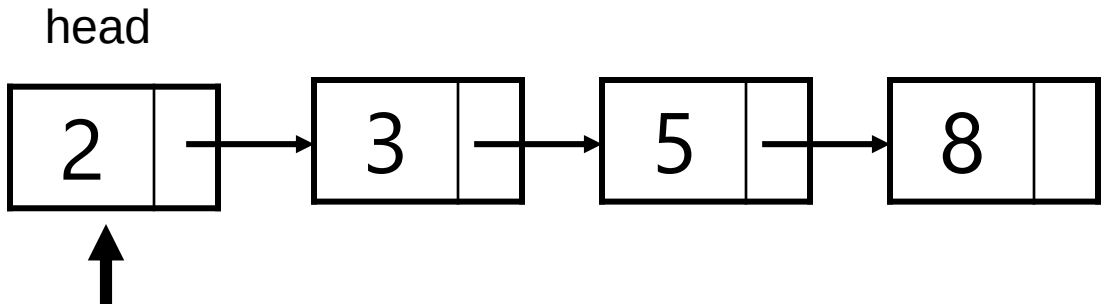
- Representing PQ using **linked list in increasing order**
- **Add** ➔ 5 , 3 , 8 , 2

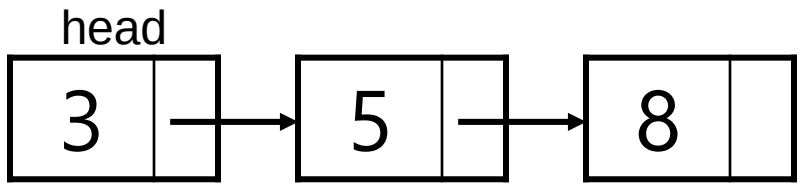


Return value at the first node will cost ➔ 1

Representation # 3

- Representing PQ using **linked list in increasing order**
- **Add ➔** 5 , 3 , 8 , 2





Remove value at the first node will cost ➔ 1

Complexity Analysis

Operation	Running time
Add	O (n)
peekMin	O (1)
removeMin	O (1)

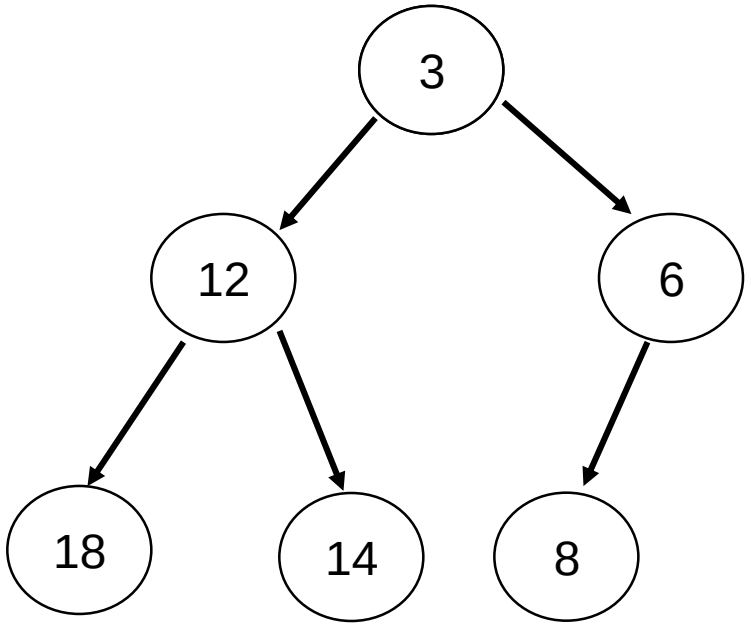
Representation # 4 (Heaps)

- Using heaps
- Heap is a complete binary tree with structural property and heap property.
- Heap must have 2 properties:

  1 – Structural property : all levels except the last are full and last level is left filled.

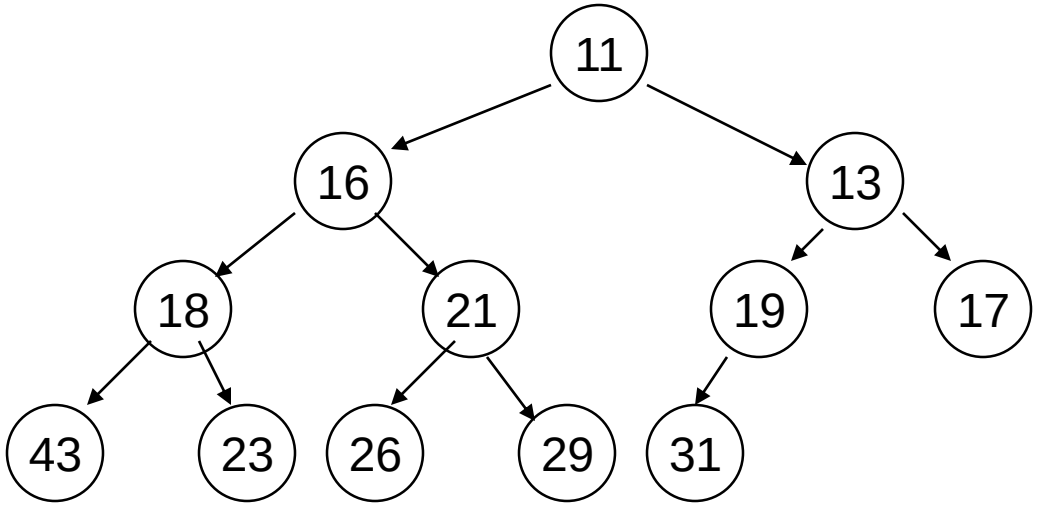
  2 – Heap property : value of nodes is large than its parents (priority of nodes is less than its children)

heap



1) H = $\lfloor \log n \rfloor = \lfloor \log 6 \rfloor = 2$
2) Parent (i) = $\lfloor i / 2 \rfloor = \lfloor 5 / 2 \rfloor = 2$
3) Left (i) = 2i = 2*2 = 4
4) Right (i) = 2i+1 = 2*2+1 = 5
5) Minimum value will always be at root

heap



- Structural preperty ➔ valid
- Heap property ➔ valid ➔ this is heap

Priority Queue Using heaps

- Implement a PQ using heap
- Use an array starting at position 1 , where each item in the array corresponds to one node in the heap.
- Depending on heap in the previous slide :

	11	16	13	18	21	19	17	43	23	26	29	31			

Priority Queue

Cont.

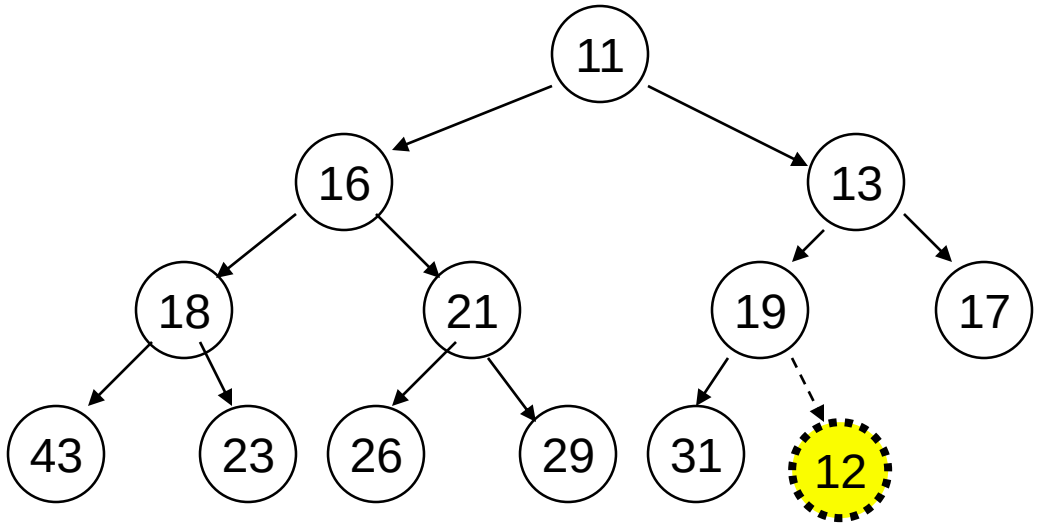
- There are 3 operations :
  1) add
  2) peekMin
  3) removeMin

peekMin

- peekMin
- Since the minimum value will always be at the root of heap or at index 1 of the array so it will cost O (1) to return it.

Add

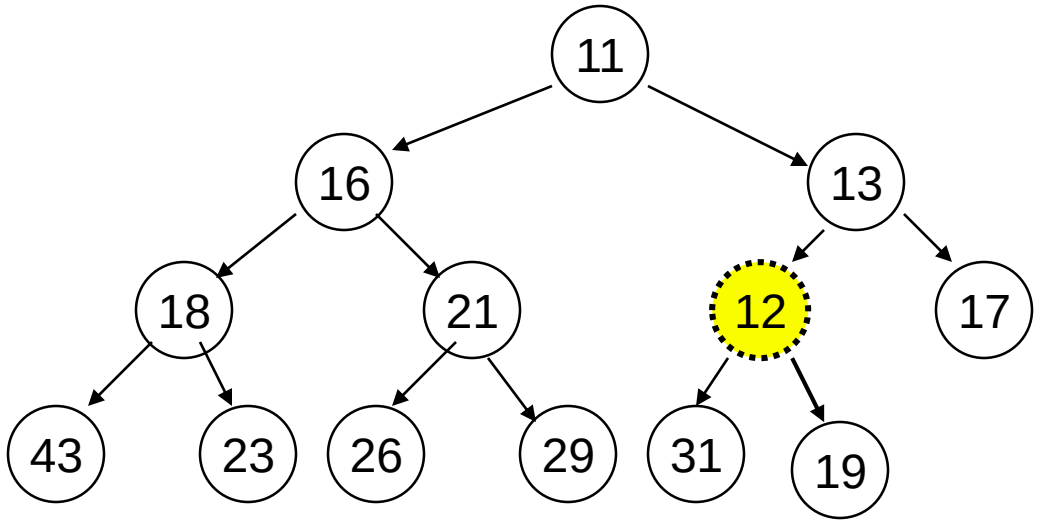
- Add (12) to the heap



1) Insert 12 at right of 19

Add

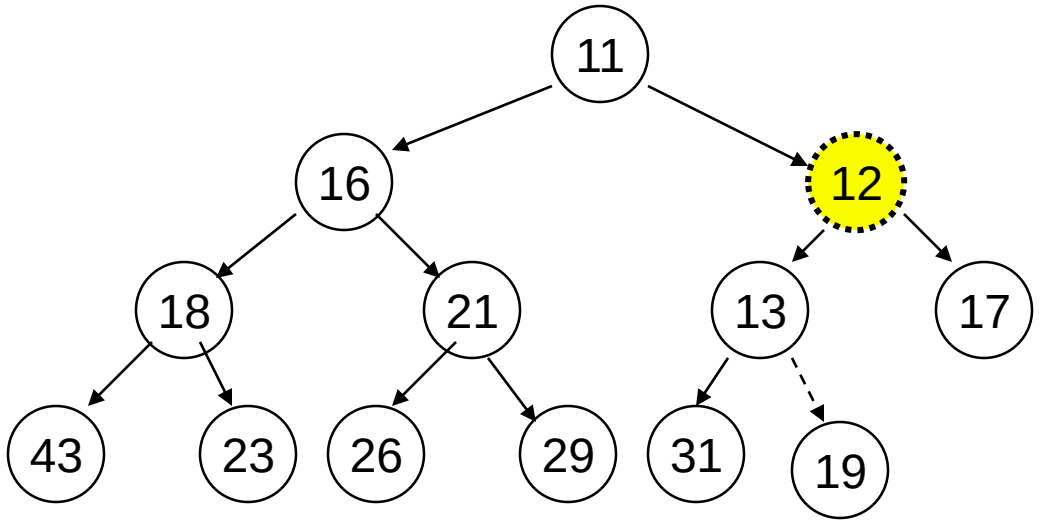
- Add (12) to the heap



2) Swap 12 and 19

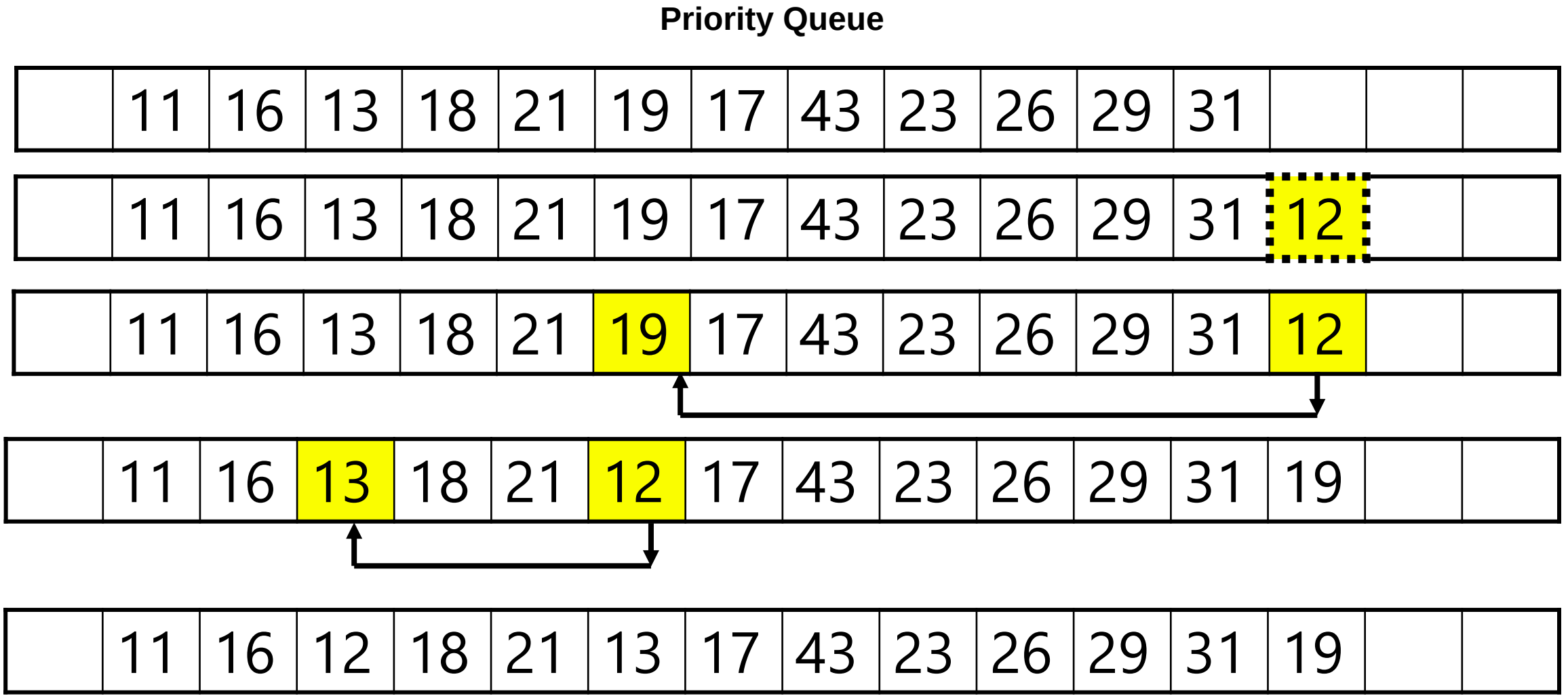
Add

- Add (12) to the heap



3) Swap 12 and 13

Add


Priority Queue
11 16 13 18 21 19 17 43 23 26 29 31
11 16 13 18 21 19 17 43 23 26 29 31 12
11 16 13 18 21 19 17 43 23 26 29 31 12
11 16 13 18 21 12 17 43 23 26 29 31 19
11 16 12 18 21 13 17 43 23 26 29 31 19

Running time for Add operation

- Height = log (n)

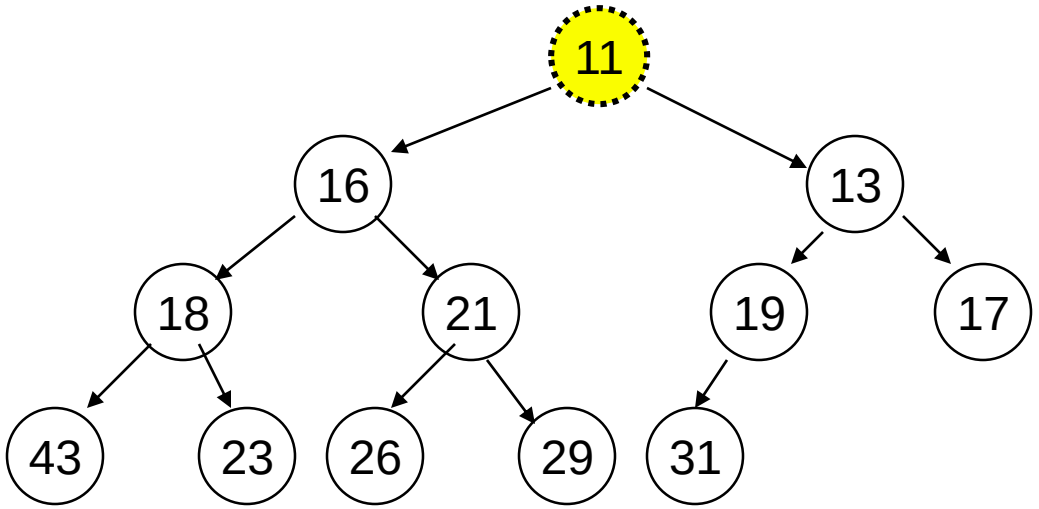
  1) inserting the element at the right most leaf in the tree will cost O (1)

  2) swapping process that maintain the heap property will cost

  O (log n). Because in the worst case the swapping will begin from the bottom of the tree (leaf) to the root.

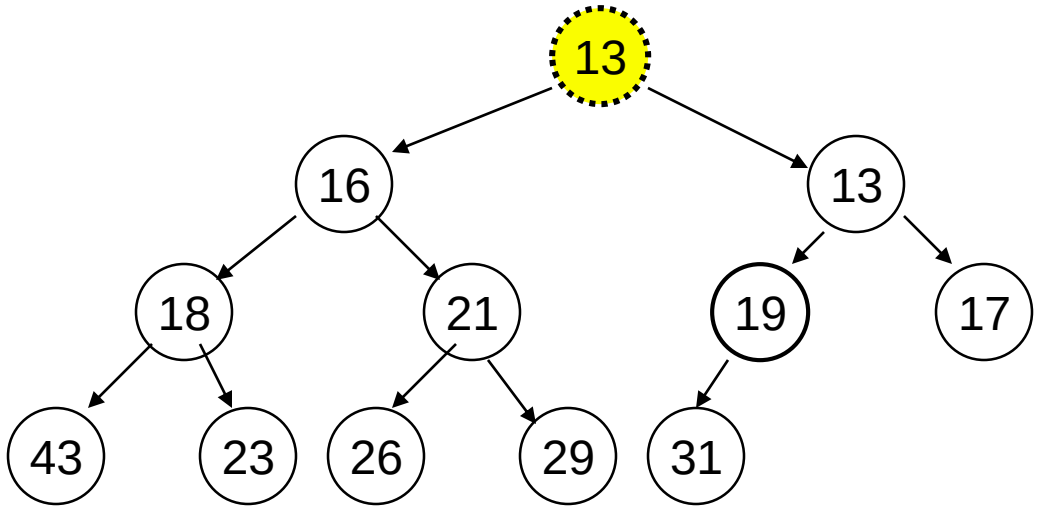
removeMin

- Remove 11



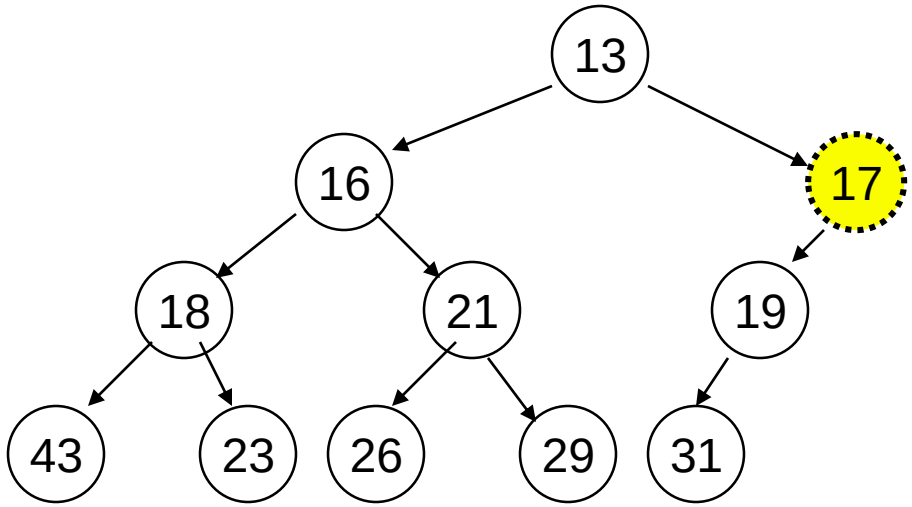
1) Remove 11

removeMin



2) Put 13 at the root
(minimum between 13 and 16)

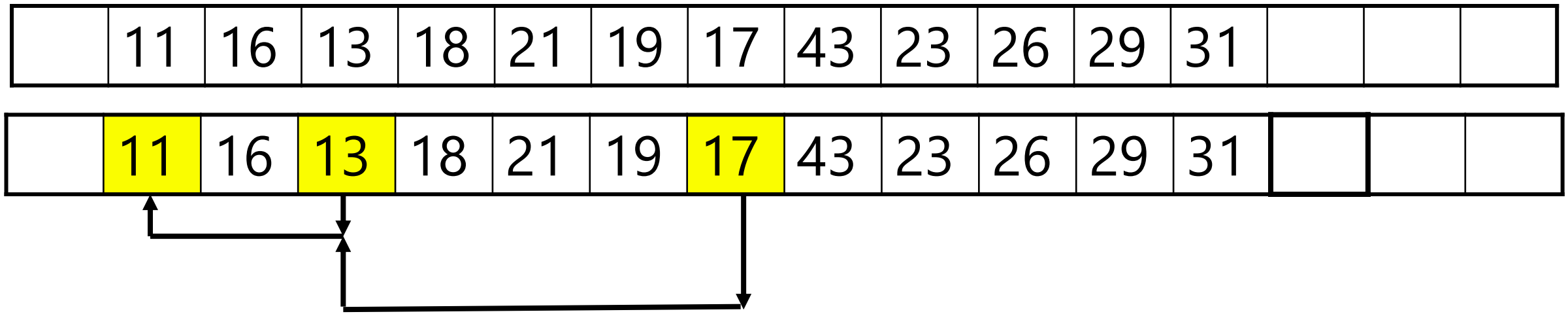
removeMin

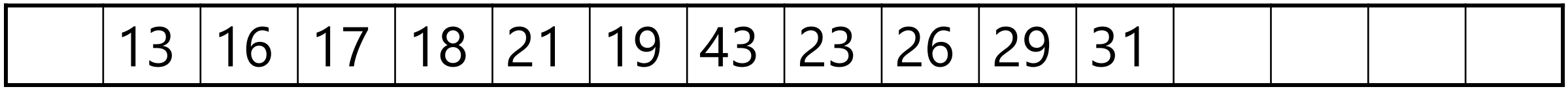


3) Modify place of 17

Add

Priority Queue





Running time for Add operation

- Height = log (n)

  1) removing the minimum value at the root will cost O (1)

  2) modifying the position of elements to maintain the heap property will cost

  O (log n). Because in the worst case this process will begin from the root to the leaf.

Complexity Analysis

Operation	Running time
Add	O (log n)
peekMin	O (1)
removeMin	O (log n)

APPLICATIONS OF PQ

- PQ used to manage limited resources such as bandwidth on a transmission line from a network router.
- College admissions process for students.
- PQ used in process scheduling in operating systems
- The airline company keeps a priority queue of a standby passengers waiting to get a seat.

End