**PRIORITY QUEUE**

Data Structure so far

* Linked List
* Stacks and Queues
* Dynamic Arrays
* Trees(Generic + Binary)
* BST

What is Priority Queue ?

* Data can be entered in any order ,but can be retrieve only in priority order

Is it either LIFO or FIFO ?

What data structure we need to use to implement priority?

O(n) Insertion R min get Min

UA O(1) O(n) O(n)

(unsorted array)

SA O(n) O(n) O(1)

(sorted array)

O(n) Insertion Remove min get Min

UL O(1) O(n) O(n)

(unsorted Linked List)

SA O(n) O(1) O(1)

(sorted Linked List)

BST O(log(n)) O(log(n)) O(log(n))

(Binary Search Tree)

We don’t have any data structure who have good all kind operation , but rather there Is either insertion is good and retrieval is not or vice versa

Except BST , whose all operation is in average case

But BST is log(n) only if BST is balanced and to make BST balance as we need to do extra operation to make it balance

**Now Which data structure we need to use?**

Lets do something unique

Let store balanced BST in array in level order

1

2 3

4 5 6 7

In an array , it will stored as

1 2 3 4 5 6 7

How to find child and parent of a particular node

So

Child of any node at ith index of an array : 2\*i + 1 , 2\*i + 2

Parent of any node at ith index of an array : (i – 1)/2 , (i – 2)/2

On generalizing : ( i - 1 ) / 2

**There is a drawback as this approach is useful only when tree is balanced i.e complete binary tree**

Properties of Complete Binary tree

* Right child can be inserted only if its parent has left child.
* Next level can be started to fill only if its previous level is full to its capacity.
* Lets recall why need priority queue=> just retrieve data in priority order but can insert data in any order
* Example : a group of student => need to give award => so we will pick student with highest marks
* Now lets say now we just store student name and rank instead marks then we pick student with least value
* So priority is not defined with high value but priority can be defined with min value
* **There are two type of Priority Queue**
  + **Min Priority Queue**
    - Give output with least value
  + **Max Priority Queue**
    - Give output with maximum value

**HEAPS**

There are two types of heaps i.e Max Heap and Min Heap

**ALL HEAP MUST POSES PROPERTY OF COMPLETE BINARY TREE**

MIN HEAP

* Here , parent node must have less value then value of child nodes

MAX HEAP

* Here , parent node must have greater value then value of child nodes

With heaps => we can deduce that root will be min if we use Min Heap or max if we use Max Heap

Therefore we will use heaps in priority queue

**COMPLEXITIES OF OPERATIONS**

* + Here We can find min and max value in o(1) time
  + Here We can insert value in o(log(n)) time (due to heapify operation)
  + Here We can delete value in o(log(n)) time (due to heapify operation)
    - In heap deletion means is to delete root only as root has maximum priority
    - How to delete ?
      * Just paste last element of an array to root then perform heapify operation. As root will be greater now therefore we need to make heap.