Priority Queue

1. In the priority queue, we save the element along with its importance factor and remove the elements based on their importance factor.
2. Eg. In temples, there is a special VIP pass because the person having that ticket can take darshan fast. While the person who is not having the VIP pass has to wait for a long. So, here the important factor is the VIP ticket.
3. Two types
   i) Min: return key with smallest value
   ii) Max:returns key with biggest value.

Heaps

1. We cannot use balanced BST for making a priority as it has two issues

i) Balancing(Height of BST in the worst case is O(n))

j) Storing

2. Hence we design a new data structure. It has two properties.

i) Complete BT: all the levels should be filled except the last level. And the last level too must be filled from left to right only. So,this shows that in the worst case also CBT height is logn. The balancing issue is resolved by using this property.

j) Obey heap order property: if we travel level-wise to position then this will take O(n) operations. Hence we store CBT in an array but always visualize it as a Binary tree. For min-heap, the root node value should be minimum then its children

   a. ParentIndex = (child index-1)/2;
   b. ParentIndex = (2*i+1)(Leftchild),(2*i-1)(Rightchild)

3. Time Complexity:

i) Insert: O(h) where h=logn, insert happens in O(1), but we have to toggle it until it satisfies theheap order property which is called ap-heapify.

j) Delete:(h), h=logn, for min-heap we have to remove the element whose priority is minimum. And for max heap,we remove the element whose priority is maximum. In this,we first swap the value with theroot node,delete the last node and then toggle the value atthe root node until it satisfies the heap order property.This is called up-heapify. Hence it takes a lognvalue.

Identification of Heaps

1.k+smallest -> max heap

2.k+largest -> min heap, size of the heap will be k,suppose we want kth largest element then we will pop

3. heap qs are generally based on sorting

QS on Heaps

1. Kth largest
   https://leetcode.com/problems/kth-largest-element-in-an-array/description/
2. Top k frequent numbers
   https://leetcode.com/problems/top-k-frequent-elements/submissions/
3. Sort k sorted array
   https://www.codingninjas.com/codestudio/problems/nearly-sorted_982937
4. K closest numbers
   https://leetcode.com/problems/find-k-closest-elements/
5. Frequency Sort
   https://leetcode.com/problems/sort-array-by-increasing-frequency/submissions/
6. K closest points to origin
   https://leetcode.com/problems/k-closest-points-to-origin/submissions/
7. Connect ropes to minimize cost
   https://www.codingninjas.com/codestudio/problems/connect-n-ropes-with-minimum-cost_630476?leftPanelTab=1
8. Sum of Elements between k1 smallest and k2 smallest numbers