Priority Queues [PQ]

- Supports operations such as Insert, DeleteMin and DeleteMax [returns & removes the maximum element], decreasekey [very rare]
- Used to find minimum/max element among a collection of elements.
- In PQ, the order in which the elements enter the queue may not be the same in which they were processed.
- Eg: Job scheduling, which is prioritized instead of serving in first come first serve.
- Also called an Ascending PQ, if the item with the smallest key has the highest priority (i.e., delete smallest element always).
- Also called descending PQ, if item with largest key has the highest priority (delete max element always).

Operations

1) Insert (Key, data) - Inserts data with key to PQ. Elements are ordered based on key

2) DeleteMin / DeleteMax
3) GetMin / GetMax

Auxiliary PQ operations

1) K^{th} - Smallest / K^{th} - largest - Returns the K^{th} - smallest / K^{th} - largest key in PQ.
2) Size - Returns no. of elements in PQ.
3) Heap sort - Sorts the elements in PQ based on priority (key).

PQ Applications

1. Data Compression - Huffman coding algorithm
2. Shortest path algos - Dijkstra's algo.
3. Minimum spanning tree algos - Prim's algo.
4. Event driven simulation - customers in a line
5. Selection problem - Finding K^{th} smallest element.

Implementations

1) Unordered Array impl.
   Insertions complexity: O(1).
   Deletion Complexity: O(n).

2) Unordered list impl.
   Insertions → O(1)
   Deletions → O(n)

3) ~~Ordered Array Impl~~ -

~~Insertion ⇒ O(n)~~

~~Deletion ⇒ O(1)~~

4) Ordered list Impl.

Impl.	Insertion	Deletion
1) Unordered Array Impl	O(1)	O(n)
2) Unordered list Impl	O(1)	O(n)
3) Ordered Array	O(n)	O(1)
4) Ordered list	O(n)	O(1)
5) Binary Search Trees	O(log n)	O(log n)
6) Balanced BST	O(log n)	O(log n)
7) Binary heap Impl.	O(log n)	O(log n)

~~Binary heap~~ → search, Implementation, deletion ⇒ O(log n)

→ getMax, getMin ⇒ O(1)