Priority Queue

- ADT: Abstract data type
- A queue where each object has an associated priority attached
- Operation
    - Insert: with a priority value
    - Remove: max/min
    - Get/Peek: max/min value

Can be implemented using simple sorted array, heap or using balanced BSTs

simple sorted array

- Insert O(N)
- Remove O(1)
- Peek: Min/max O(1)

HEAP

- Insert O(log N)
- Remove O(log N)
- Peek O(1)

Balanced BSTs

- Insert O(log N)
- Remove O(log N)
- Peek O(log N)

Practically Heap is the prefered implementation over BST as Heap uses an array which gives better locality of reference (caching) Hence better practical performance even though complexity is same.

In [ ]:

In [ ]:

Priority Queue in different languages

In [ ]:

Python

- min heap by default

In [10]:

```python
import heapq
h = [10,4,3,1,7]

heapq.heapify(h) # O(N Log N)
print(h)

print(h[0])

heapq.heappop(h)
print(h)

heapq.heappush(h, 2)

print(h)

```

```
[1, 4, 3, 10, 7]
1
[3, 4, 7, 10]
[2, 3, 7, 10, 4]
```

Strings and heap

In [13]:

```python

h = ["d", "c", "a", "b"]
heapq.heapify(h)
print(h)

print(heapq.heappop(h))
print(h)
```

```
['a', 'b', 'd', 'c']
a
['b', 'c', 'd']
```

In [ ]:

```python

```

Custom Objects and Heap

In [15]:

```python
# min heap example using age for priority
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age =  age

    def __lt__(self, other): # less than
        return self.age < other.age

    def __str__(self):
        return f"({self.name}, {self.age})"

    def __repr__(self):
        return self.__str__()

h = [Person("B", 10), Person("P", 13), Person("A", 19), Person("C", 15), 
heapq.heapify(h)

print(h)
```

```
[(B, 10), (D, 11), (A, 19), (C, 15), (P, 13)]
```

In [ ]:

```python

```

Python heap as max heap

In [16]:

```python
# max heap example using age for priority
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age =  age

    def __lt__(self, other): # less than
        return self.age > other.age

    def __str__(self):
        return f"({self.name}, {self.age})"

    def __repr__(self):
        return self.__str__()

h = [Person("B", 10), Person("P", 13), Person("A", 19), Person("C", 15), 
heapq.heapify(h)

print(h)
```

```
[(A, 19), (C, 15), (B, 10), (P, 13), (D, 11)]
```

In [17]:

```python
# min heap example using name for priority
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age =  age

    def __lt__(self, other): # less than
        return self.name < other.name

    def __str__(self):
        return f"({self.name}, {self.age})"

    def __repr__(self):
        return self.__str__()

h = [Person("B", 10), Person("P", 13), Person("A", 19), Person("C", 15), 
heapq.heapify(h)

print(h)
```

```
[(A, 19), (C, 15), (B, 10), (P, 13), (D, 11)]
```

In [ ]:

JAVA

default min heap

```java
PriorityQueue<Integer> pq=new PriorityQueue<>();

for(int i=5; i>= 1;i--)
{
  pq.add(i);
}

while(!pq.isEmpty())
{
  System.out.println(pq.poll());
}
```

default min heap

```
    int data[]={1,2,3,5,4};

    PriorityQueue<Integer> pq=new PriorityQueue<>();

    // copy data
    for(int i=0;i< data.length;i++)
    {
      pq.add(data[i]);
    }

    while(!pq.isEmpty())
    {
      System.out.println(pq.poll());
    }
```

max heap using comparator

```
    int data[]={1,2,3,5,4};

    PriorityQueue<Integer> pq=new PriorityQueue<Integer>((o1,o2)->  o
2-o1);
    for(int i=0;i< data.length;i++)
```

In [ ]:

```

```

C++

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

int main()
{
      int arr[]={1,2,3,4,5};

    priority_queue<int> pq(arr, arr+5);
    cout<<"Max priority queue: ";
    while(!pq.empty()){
      cout<<pq.top()<<endl;
      pq.pop();
    }


    priority_queue <int, vector<int>, greater<int> > pq1(arr,arr+5);
    cout<<"Min priority queue: ";
    while(!pq1.empty()){
    cout<<pq1.top()<<endl;
    pq1.pop();
    }

}
```

C#

min heap

```
PriorityQueue<string, int> queue = new PriorityQueue<string, int>();
queue.Enqueue("Item A", 1);
queue.Enqueue("Item B", 2);
queue.Enqueue("Item C", 3);
queue.Enqueue("Item D", 5);
queue.Enqueue("Item E", 4);

while (queue.TryDequeue(out string item, out int priority))
{
    Console.WriteLine($"Popped Item : {item}. Priority Was : {priorit
y}");
}
```

Javascript

In [ ]:

```
```

- heapify -> convert a sequence/array to satisfy heap property: O(N)
- heappop: assumes data is in heap format/heapified: O(log N)
- heappush: assumes data is in heap format/heapified: O(log N)
- peek: O(1)

In [ ]:

```
```

Question: Kth largest/smallest element
https://leetcode.com/problems/kth-largest-element-in-an-array/
(https://leetcode.com/problems/kth-largest-element-in-an-array/)

```
Solution-1:
        Sort
        Get N-k index element
        TC: O( n log n)
        SC: O(1)

Solution-2:
    All data in max heap
    Heap pop k times
    TC: O(N log N) + O(K log N)
    SC: O(N)

Solution-3:
    Maintain a min heap of k elements: O(K log K)
    Loop over the remaining elements: if heap.min() < curr : replace
O ( (N-K) log K )
    Get heap.min() -> kth largest element    O(1)
    TC: O(k log k) + O( (n-k) log K) + O(1) => O(n log k)
    SC: O(k)

Solution-4: BST
    Put all data in bst
    Reverse Inorder traversal to get Kth element
    TC: O(n log n)
    SC: O(N)

Solution-5:
    Count: Count frequency of each number [3, 2, 1, 1, 4, 5, 3] ->
1:2 2:1 3:2 4:1 5:1 [1,1,2,3,3,4,5] -> O(N)
    Bucket: 10,11,21,20,45,42,15   : [0-9]:[]  [10-19]:[10,11,15]  [2
0-29]:[20,21] [30-39]:[] [40-49]:[42,45]

Solution-6:
    Quick sort based solution
    Avg: O(N)
    Worst: O(N^2)
```

Java: Heap solution

```
class Solution {
    public int findKthLargest(int[] nums, int k) {
        PriorityQueue<Integer> pq = new PriorityQueue<>();

        for(int i = 0 ; i < nums.length; i++){
```

```
class Solution {
public:
    int findKthLargest(vector<int>& nums, int k) {
       priority_queue<int,vector<int>,greater<int>> q(nums.begin(),nums.begin()+k);
        for(int i=k;i<nums.size();i++)
        {
            if(q.top()<nums[i])
             {
                 q.pop();
                 q.push(nums[i]);
             }
        }
        return q.top();
    }
};
```

~~~
```Python
def findKthLargest(self, nums: List[int], k: int) -> int:
       return heapq.nlargest(k,nums)[-1]
```
~~~

In [ ]:

Question: Merge k sorted lists
[https://leetcode.com/problems/merge-k-sorted-lists/ (https://leetcode.com/problems/merge-k-sorted-lists/)](https://leetcode.com/problems/merge-k-sorted-lists/)

K lists.
Each of size N.

```
k lists of length n

Solution-0:
    Join all linked list into 1 list
    Sort the single linked list O(nk log nk)

Solution-1: Merge lists sequentially till left with 1 list [k-1 merges]
    O(2n) + O(3n) + ... O(kn)
     O(n) (2 + 3 + 4... k)
        O(n) k(k+1)/2 => O(n) ((k^2)/2 + k/2) => O(n) O(k^2) => O(n * k^2)


              l1 l2 l3 l4 l5 l6 l7 l8
                 12  3 4 5 6 7 8
                 123  4 5 6 7 8
                    ....
                     12345678

Solution-2: Merge lists pairwise till left with 1 list [k-1 merges]
        O(k*n) * log k => O(nk log k)

                1 2 3 4 5 6 7 8
                 12  34  56 78 -> n*k
                   1234   5678 -> n*k
                     12345678

Solution-3: Sorting

Solution-4: Heap
```

In [ ]:

```

```

Sort nearly k sorted array

Given a k–sorted array that is almost sorted such that each of the n elements may be misplaced by no more than k positions from the correct sorted order. Find a space-and-time efficient algorithm to sort the array.

For example,

Input:

arr = [1, 4, 5, 2, 3, 7, 8, 6, 10, 9] k = 2

Output:[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

In [ ]:

```

```

In [ ]:

```

```

Question Skyline

[https://leetcode.com/problems/the-skyline-problem/ (https://leetcode.com/problems/the-skyline-problem/)](https://leetcode.com/problems/the-skyline-problem/)

In [ ]:

```
- Note down the problems
- SD, DSA, Framework
```

In [ ]:

```
- salary
- work
- people, culture
- work life
```

In [ ]:

```
Upskill:
- Get someone's experience in your CV
- Make your own personal project

- Tech ?
  - Java
  - Spring


Specific                 vs   Generic
(Lang, framework etc.)
- Start a project
- Deploy on server

```

In [ ]:

```

```