PW SKILLS

→ data structure

Heaps → priority queue

Lecture 1

What and Why?

SKILLS

Array

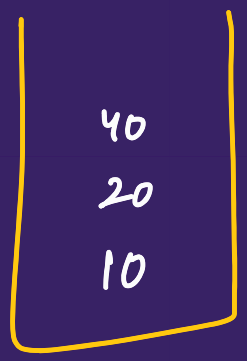
LL

Stack

Queue

Hashmap

Heap



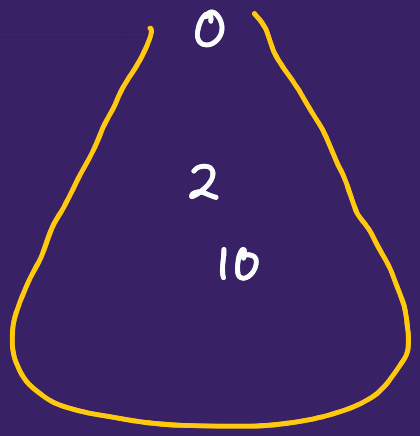
stack

push → O(1) (add)

pop → O(1) (remove)

peek → O(1) (get)

1



min heap

✓ add (2)

✓ add (10)

✓ add (1)

✓ remove()

✓ add (0)

What and Why?

PW SKILLS

Heap → add : the element is added to heap & the minimum element is at top T.C. = O(log n)

remove : the top (min^m) ele is removed & the min^m element is at top T.C. = O(log n)

peek : returns the top (min^m) element T.C. = O(1)

size : returns the size

2 Types of heaps

PW SKILLS

- Minheap : Java Collection Framework (Builtin)

If we are adding 'n' elements one by one in a heap then →

$$\text{no. of ops} \approx \log(1) + \log(2) + \log(3) + \ldots \log(n) = \log(n!)$$

$$\approx n \log n$$

$$\text{T.C.} = O(n \log n)$$

2 Types of heaps

PW SKILLS

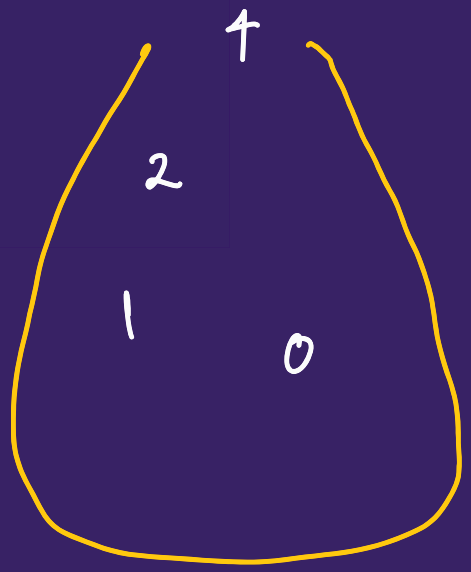
Minheap v/s Arraylist → vala heap isme add me T.C. = O(nlogn)

add(2)	arr = {2}
add(10)	arr = {2,10} → {10,2}
add(1)	arr = {10,2,1} → {10,2,1}
remove()	arr = {10,2}
add(0)	arr = {10,2,0} → {10,2,0}
add(20)	arr = {10,2,0,20} → {20,10,2,0}

Priority Queue STL

PW SKILLS

2) Maxheap



10

- ✓ add(2)
- ✓ add(10)
- ✓ add(1)
- ✓ remove()
- ✓ add(0)
- ✓ add(4)

```
PriorityQueue<Integer> pq = new PriorityQueue<>(Collections.reverseOrder());
```

Problem Identification

PW SKILLS

1) K^{th} smallest / largest / closest / frequent / K-sorted array

Out of n, if we are working on 'K' elements

2) $O(n \log n) < O(n \log K) < O(n)$

3) Minimise, Maximise, Continuous sorting

3) For K smallest → we use MaxHeap.

Ques:

PW SKILLS

Q1 : Find the kth smallest element in a given array.

arr = { 10, 2, 3, 8, -4, -2, 6 } K=3

M-1 : Brute Force

Selection Sort (K passes) then arr[k-1] T.C. = O(Kn)

M-2 : Builtin Sort → Quick Sort / Merge Sort

T.C. = O(n log n)

S.C. = O(log n) / O(n)

arr[k-1]

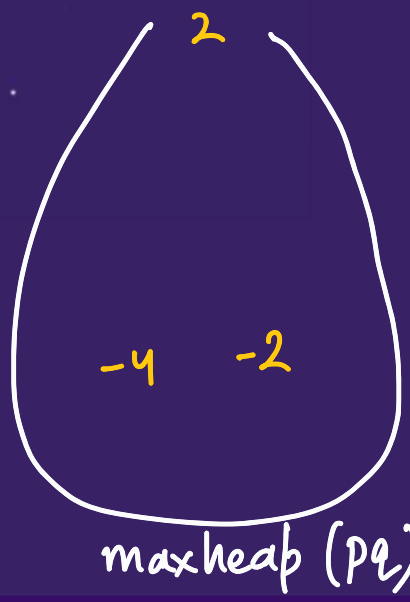
Ques:

PW SKILLS

Q1 : Find the kth smallest element in a given array.

arr = { 10, 2, 3, 8, -4, -2, 6 } K = 3

Restrict the size of maxheap to 'K'



if (pq.size() > K) pq.remove()

Ques:

PW SKILLS

Q1 : Find the kth smallest element in a given array.

```
for (int ele : arr){
    pq.add(ele);
    if (pq.size() > k) pq.remove();
}
sout(pq.peek());
```

add → n log k

remove → (n-k) log k

total = (2n-k) log k

T.C. = O(n log k)

Ques:

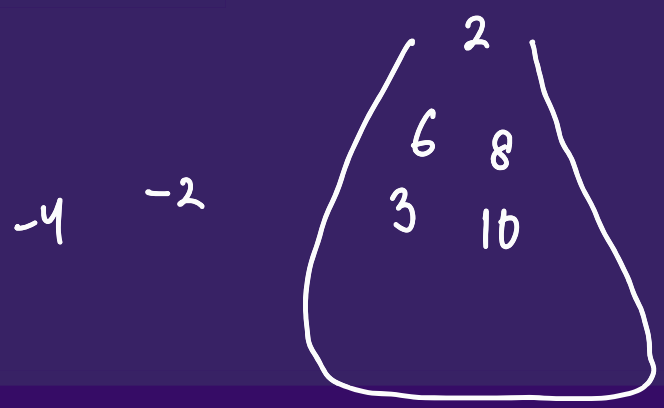
PW SKILLS

Q1 : Find the kth smallest element in a given array.

arr = { 10, 2, 3, 8, -4, -2, 6 } K = 3

M-3 : Using heap / pq But which heap ?

If we use minheap ; then



add → n log n

remove → K log n

T.C. = O(n log n)

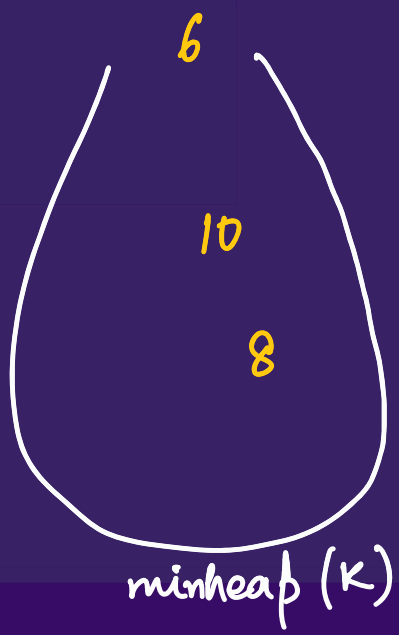
S.C = O(n)

Ques:

PW SKILLS

Q2 : Find the kth largest element in a given array.

arr = { 10, 2, 3, 8, -4, -2, 6 } K = 3



Time Complexity → O(n log K)

Auxiallary Space → O(K)

[Leetcode 215]

Ques: [Good interview Question]

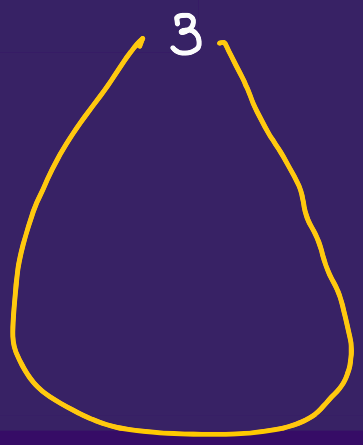
PW SKILLS

Q3 : Sort a 'k' sorted array (sort a nearly sorted array).

arr = {6,5,3,2,8,10,9} k=3 | arr = {3,1,4,2,5} k=2

6 5 3 2 8 10 9 k=3

ans = { 2 3 5 6 8 9 10 }



minHeap (k)

Ques:

PW SKILLS

Q4 : K Closest Points to Origin

arr = { {3,3}, {5,-1}, {-2,4}, {1,0}, {3,2} } K=2

$\sqrt{18}$ $\sqrt{26}$ $\sqrt{20}$ $\sqrt{1}$ $\sqrt{13}$

→ K smallest distances → Maxheap

ans = { {1,0}, {3,2} }

[Leetcode 973]

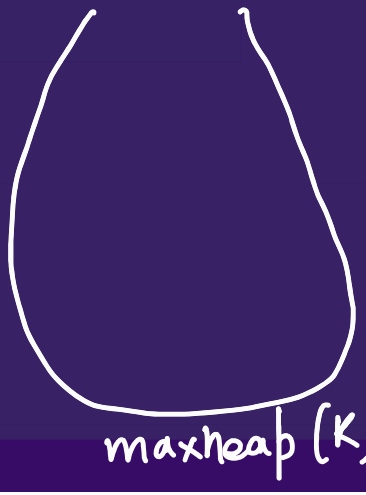
Ques:

PW SKILLS

Q4 : K Closest Points to Origin

arr = { {3,3}, {5,-1}, {-2,4}, {1,0}, {3,2} } K=2

d 18 26 20 1 13



13, {3,2}
1 {1,0}
20, {-2,4}
26 {5,-1}
18, {3,3}

map < Distance^2, Coordinate[] >
int arr

ans = {{3,2}, {1,0}}

[Leetcode 973]

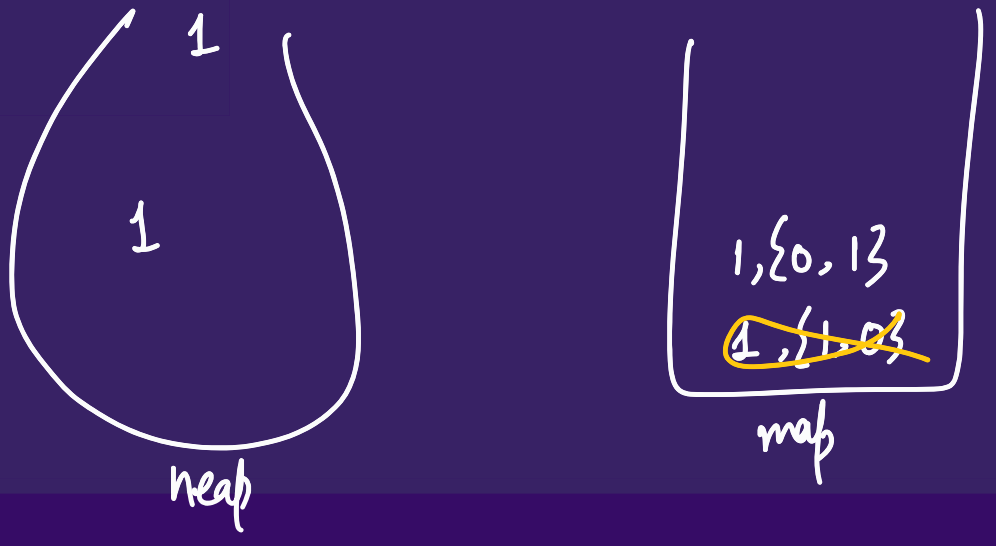
Ques:

PW SKILLS

Q4 : K Closest Points to Origin

arr = { {1,0}, {0,1} } K=2

1 1



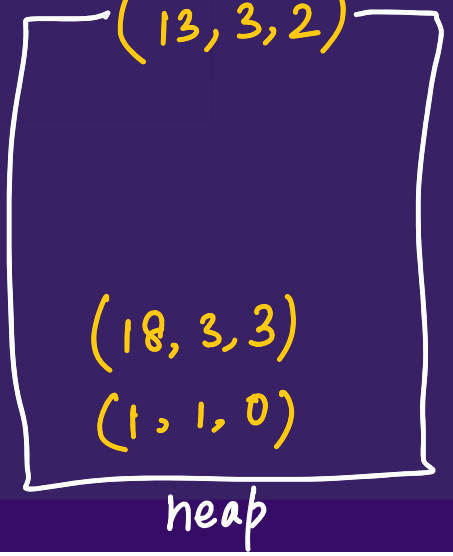
[Leetcode 973]

Ques: Custom Comparator pq<triplet> PW SKILLS

Q4 : K Closest Points to Origin M-2

arr = { {3,3}, {5,-1}, {-2,4}, {1,0}, {3,2} } K=3

18 26 20 1 13



If I am inserting an Object in PQ, pls use Comparable

[Leetcode 973]

Ques:

PW SKILLS

Q5 : Find K Closest Elements → Comparable

arr = { 7, 8, 18, 4, 9, 6 } K = 3 x = 8

1 0 10 4 1 2

arr = { 4, 5, 6, 7, 8, 9 } x = 2 , K = 4

2 3 4 5 6 7

[Leetcode 658]

Ques:

PW SKILLS

Q5 : Find K Closest Elements → Comparable

arr = { 1, 2✓, 3✓, 4✓, 5✓, 6, 7 } x = 4, K = 4

3 2 1 0 1 2 3

Closeness → a, b, x

a is closer to x if |x-a| < |x-b|

& if |x-a| == |x-b| then if a < b then

a is closer

[Leetcode 658]

Ques:

PW SKILLS

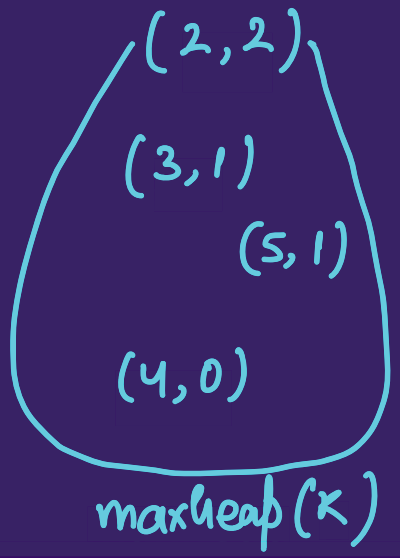
Q5 : Find K Closest Elements → Comparable

arr = { 1, 2, 3, 4, 5, 6, 7 }
3 2 1 0 1 2 3

x = 4, K = 4

Maxheap

K closest/ smallest



pq < Pair >

↓

ele, |x-ele|

T.C. = O(n log K)

A.S = O(K)

[Leetcode 658]

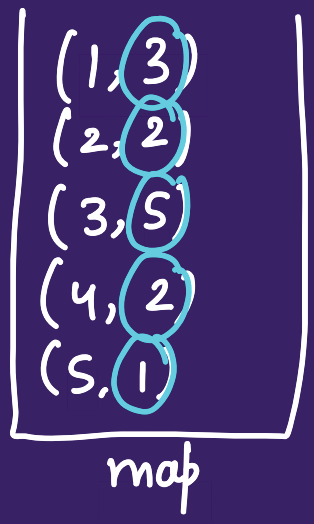
Ques:

PW SKILLS

Q6 : Top K Frequent Elements → K largest (frequency) Elements

arr = { 1, 2, 1, 3, 1, 3, 3, 3, 2, 3, 4, 5, 4 } K=3

return those 'k' elements which occur the most in given array



In the ques :

It is given, each distinct ele has a unique frequency

Space → n, K, K → O(n)

Time → O(n log K)

[Leetcode 347]

Homework:

PW SKILLS

Q : Sort Array by Increasing Frequency.

[Leetcode 1636]

Ques:

PW SKILLS

Q7 : Last Stone Weight

arr = {2, 7, 4, 1, 8, 1}

↓

{2, 4, 1, 1, 1}

{1, 1, 1, 2}

{1, 1, 1}

{1}

→ ans

At every step we are taking 2 largest stones

↓

maxheap

Brute Force

↓

Use Arraylist & Sorting

Time → $n \log n + (n-1)\log(n-1) + (n-2)\log(n-2) + \dots$

$\sum_{r=n}^{1} r \log \approx n^2 \log n$

r → n to 1

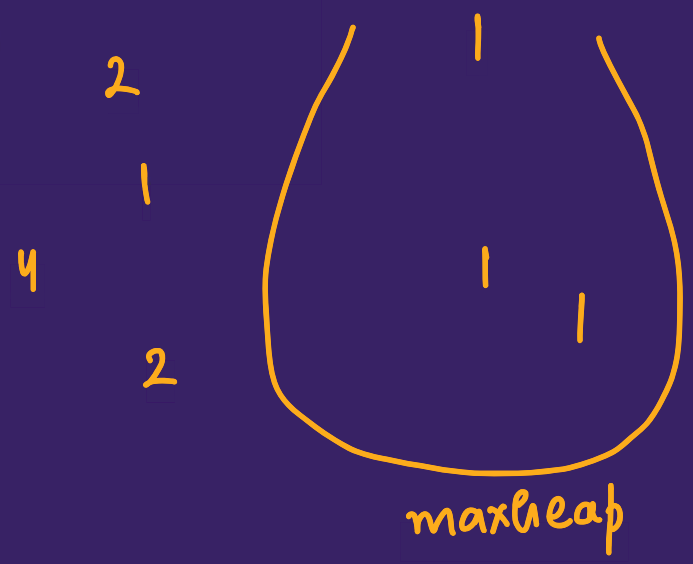
[Leetcode 1046]

Ques:

PW SKILLS

Q7 : Last Stone Weight

arr = { 2, 7, 4, 1, 8, 1 }



8 7

here, we have at max 'n' elements in heap. so insertion, removal of each element will take 'log n'

T.C. = O(n log n)

arr = { 4, ~~10~~, ~~10~~, 2 }

6 → ans

[Leetcode 1046]

Ques: I can connect 2 ropes at a time & the cost of joining is the sum of length of ropes

PW SKILLS

Q8 : Minimum Cost to Connect all Ropes

arr = {2, 7, 4, 1, 8}

I have 'n' ropes of some lengths

2 7 4 1 8

2 15 4 1

2 19 1

21 1

cost = 15 + 19 + 21 + 22

= 77 max

2 7 4 1 8

3 7 4 8

7 7 8

14 8

cost = 3 + 7 + 14 + 22

= 46 min

T.C. = O(nlogn)

S.C. = O(n)