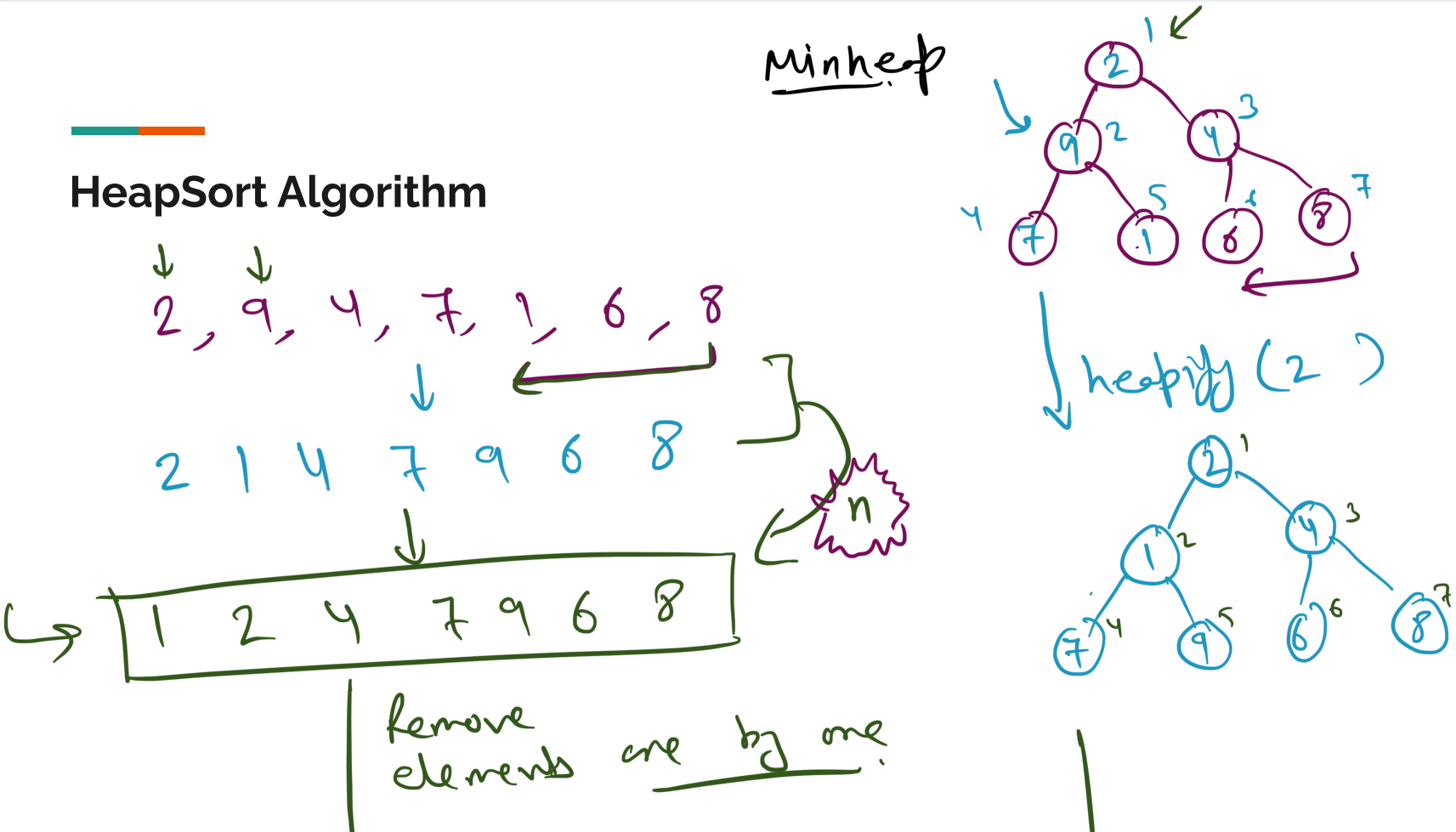
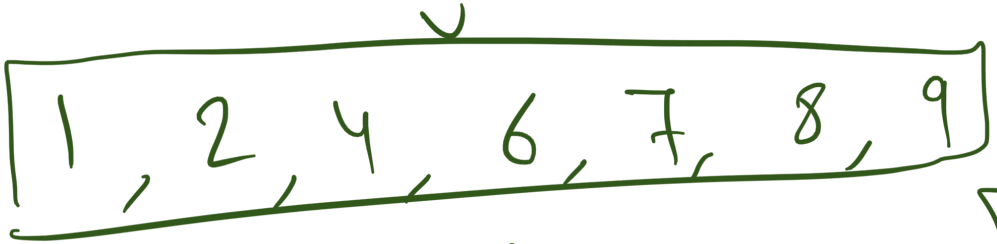
Priority Queue - I

HeapSort Algorithm

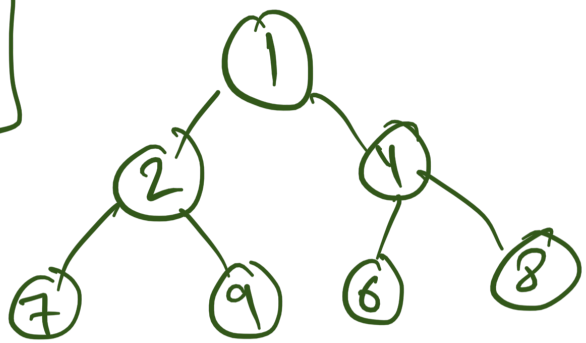


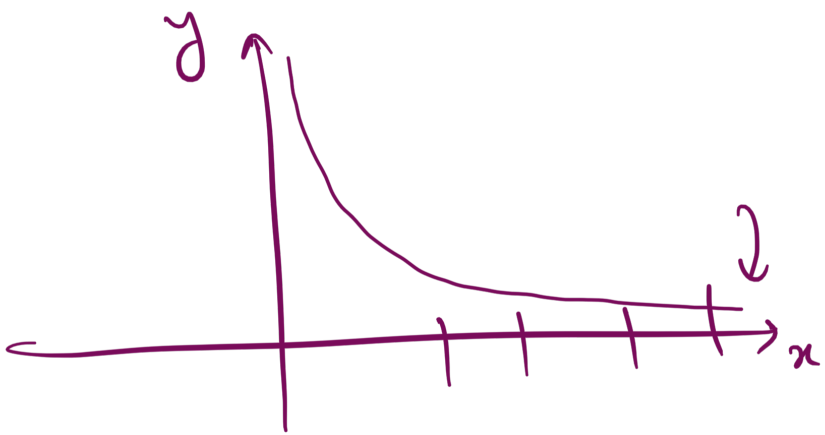


nlogn

Heap!

heapify(1)





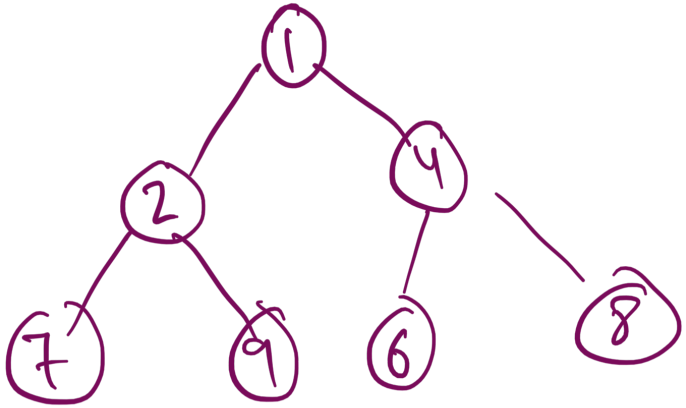
$y = \frac{1}{x}$

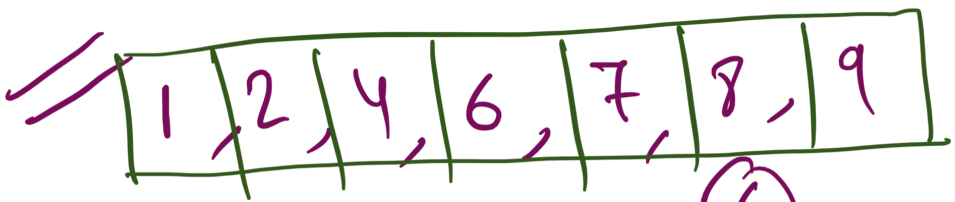
Insert element in a Heap → $\log n$

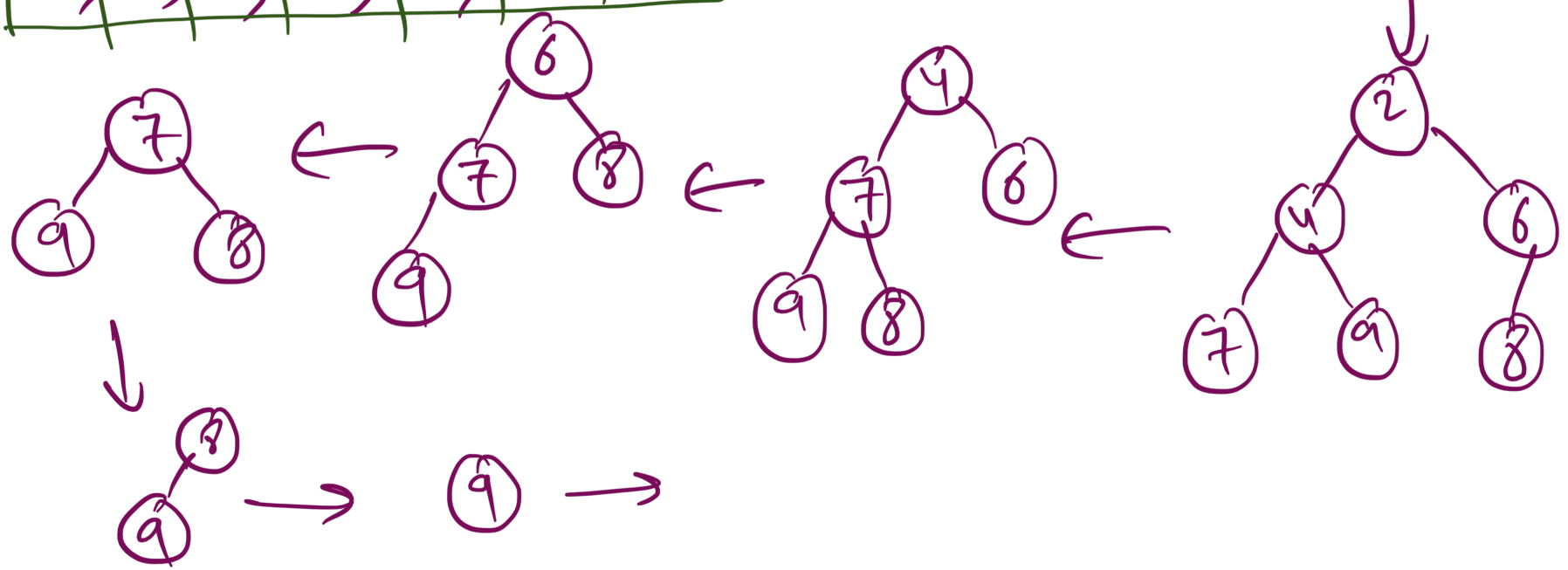
But to build a Heap

T.C → $O(n)$.

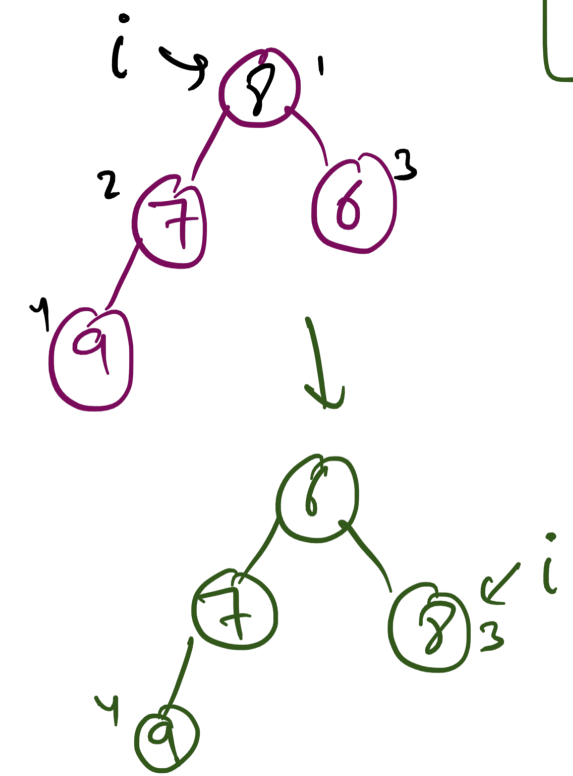

0
1
2
4
7
9
6
8


1
2
4
7
9
6
8


1
2
4
6
7
8
9


6
7
8
7
9
8
4
7
6
9
8
2
4
6
7
9
8
8
9
9

extractMin() ?



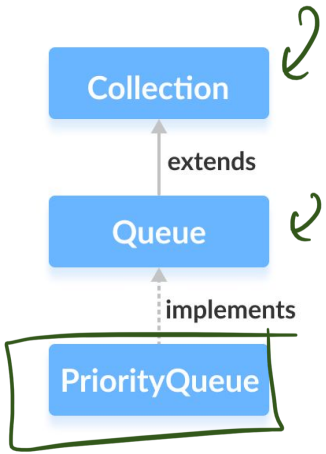
result = 4

smallest = 3

left = 6

right = 7

PriorityQueue in Java



The `PriorityQueue` class provides the implementation of all the methods present in the `Queue` interface.

add() / offer()

↳ peek() / element()

↳ poll() / remove()

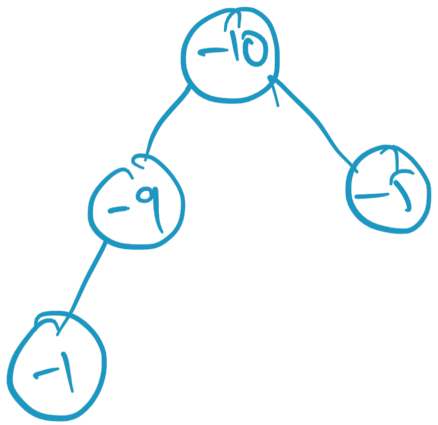
Min Heap

5, 10, 9, 1

Max Heap

- -5, -10, -9, -1

- (-10)



4th smallest

Find the kth Largest Element in an Array.

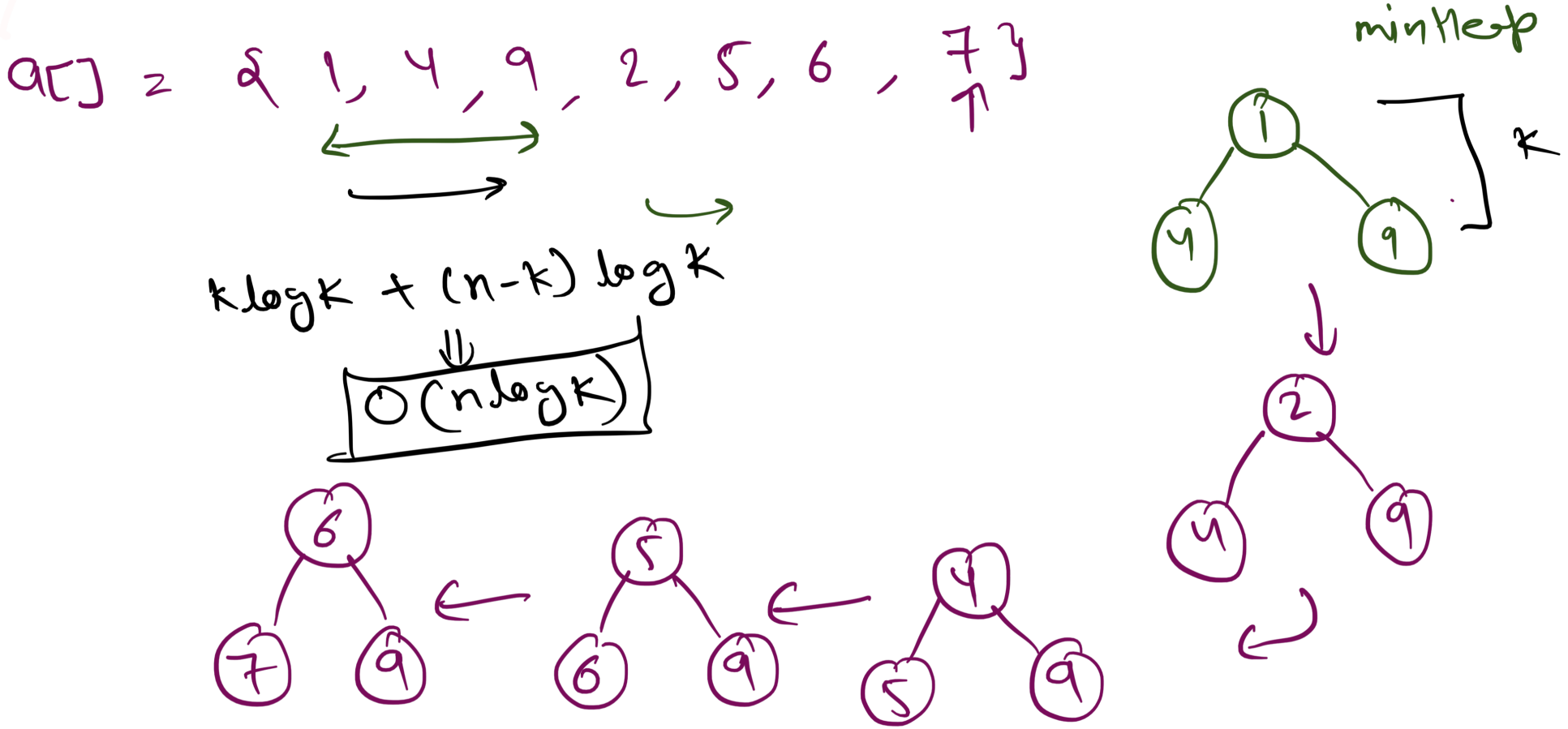
// a[] = { 1, 4, 9, 2, 5, 6, 7 }

3rd largest

Sort O(n log n)

{ 1, 2, 4, 5, 6, 7, 9 }

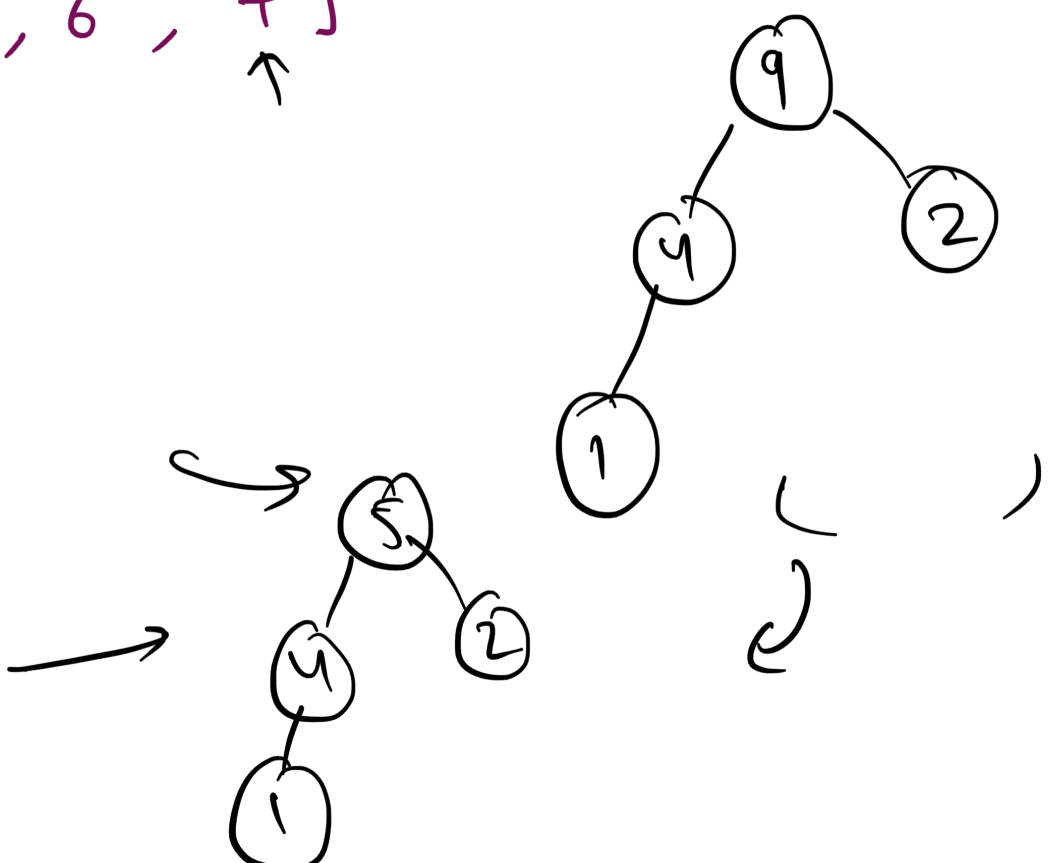
n-3 | 3


a[] = { 1, 4, 9, 2, 5, 6, 7 }
minHeap
k
klogk + (n-k) logk
O(nlogk)

Find the kth smallest element. k = 4

a[] = { 9, 1, 4, 9, 2, 5, 6, 7 }

Max Heap



Practice Problems

1. Maximum sum of at most two non-overlapping intervals in a list of Intervals | Interval Scheduling Problem
2. Split Array into K non-overlapping subset such that maximum among all subset sum is minimum
3. Maximize profit possible by selling M products such that profit of a product is the number of products left of that supplier