Hashing & Heap Data Structure

Hashing / Dictionary — Python

→ (key, value)

↓

unique

map(102) = Priya

Map[key] = value

Map →

unique →

Roll Number

Hash table

Key	Value
101	[Address, Name, Score]
102	("Priya")
103	
104	
105	"Priya"
106	

Map(105) = "Priya"

0	1	2	3	4	5
10	20	30	40	70	90

arrays → O(1)

Random access

arrays[4] = 70

Real time

↳ { Employee
  { Students

Primary key

Hash table

key	value
0	88
1	100
2	62
3	73
4	
5	85
6	
7	77
8	47
9	89

100

Hash function

Division Modulo Hash function

77 62 85 47 89 73 88

77 % 10 = 7

Value % m = key

m = 10

collision

1) open addressing
   - Linear
   - Quadratic
   - Double
2) Chaining

Heap — Minheap / Maxheap

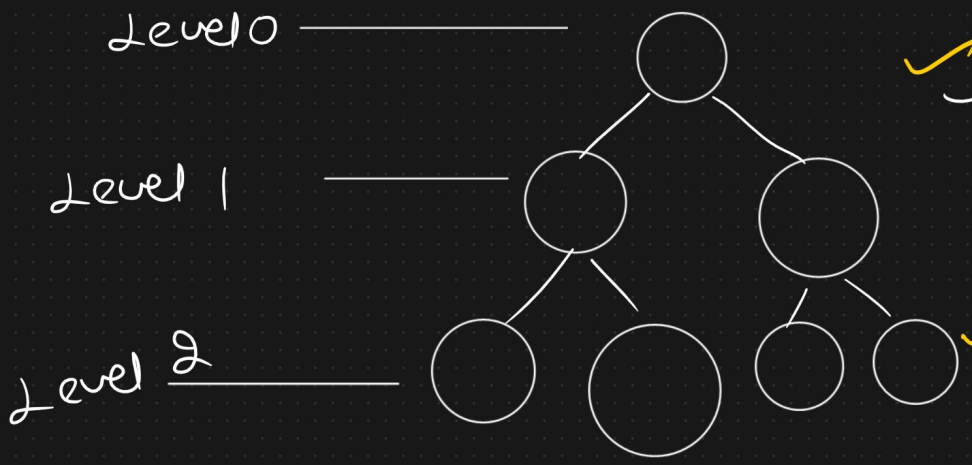
↳ (1) complete Binary tree

Tree → Non-Linear Data Structure

Binary Tree — 0 / 1 / 2

Atmost 2 child nodes

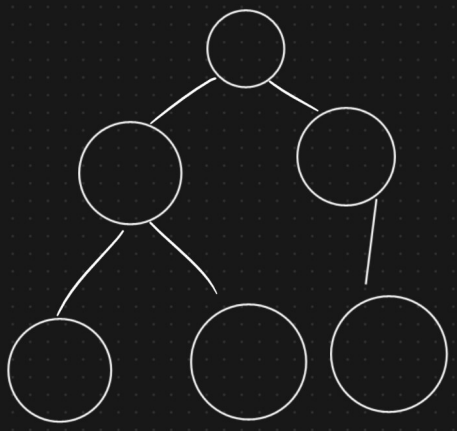
Complete Binary Tree

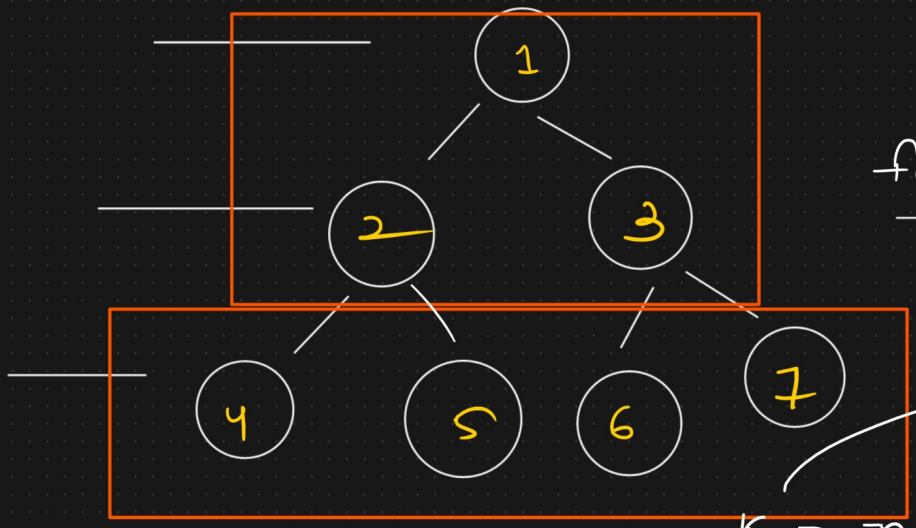


1) Left to Right side

2) Each level must be filled first before moving to the next level.

ACBT





full CBT

3

k = num of level

① number of node =

Property

$$2^k - 1 = 2^3 - 1$$

$$= 7$$

$$n = 2^k - 1$$

$$n+1 = 2^k$$

$$\log_2^{(n+1)} =$$

$$k\left(\log_2 2\right)$$

$$K = \log_2(n+1)$$

$$K = O(\log_2 n)$$

$$\log_2 2^K$$

$$K \log_2 2 = K$$

$\log_2 2 = 1$

O child node

2) No of leaf Node

↳ $\lceil n/2 \rceil$

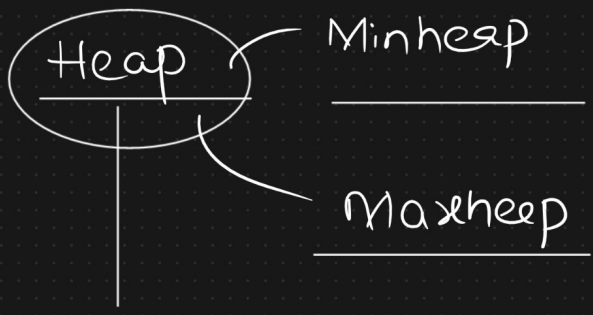
$\Rightarrow \lceil 7/2 \rceil$

$\Rightarrow 4$

(3) Num of non-leaf node

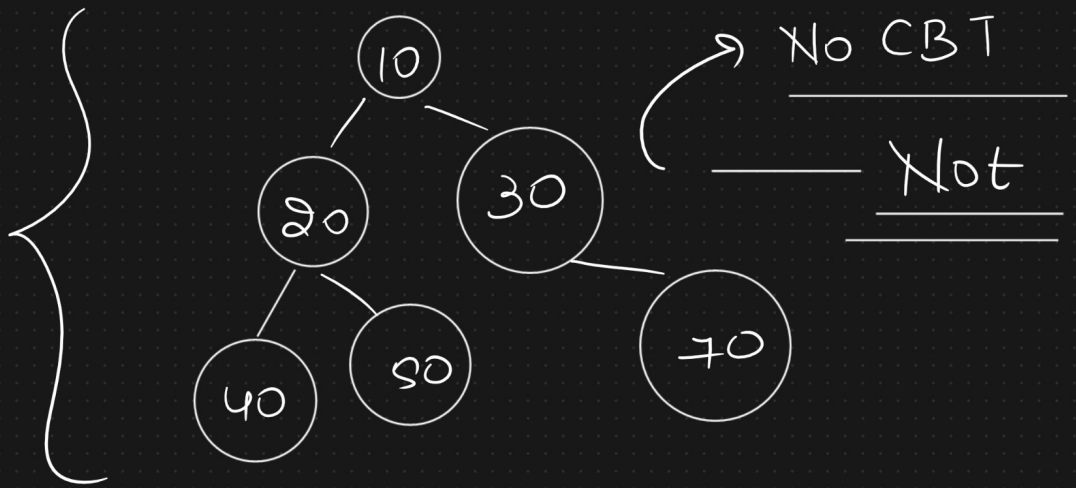
↳ $\lfloor n/2 \rfloor = \lfloor 7/2 \rfloor$

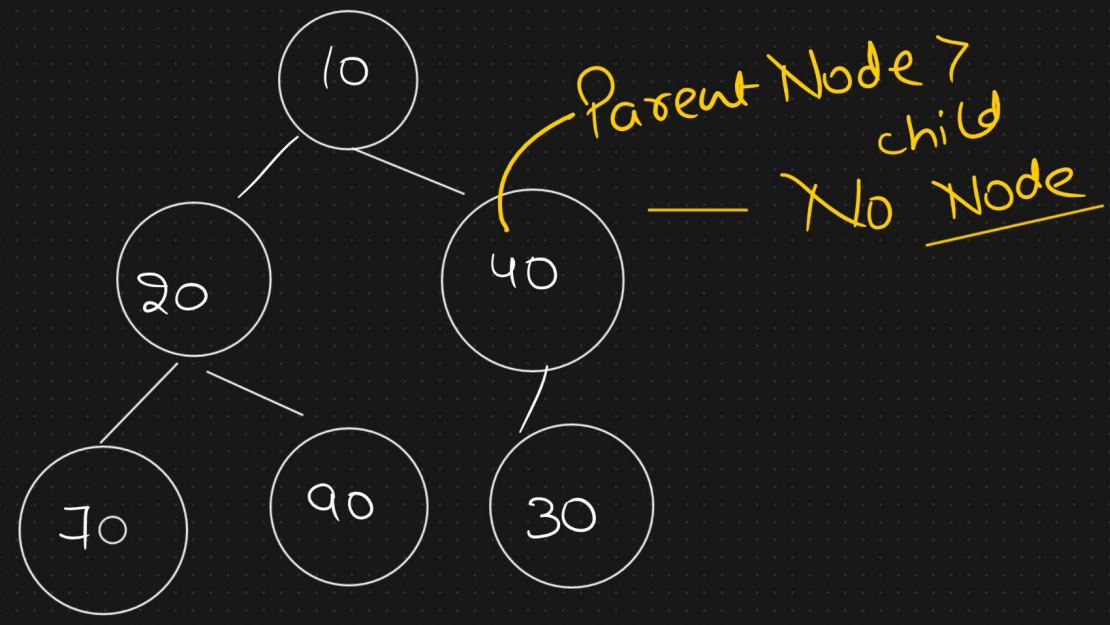
= 3

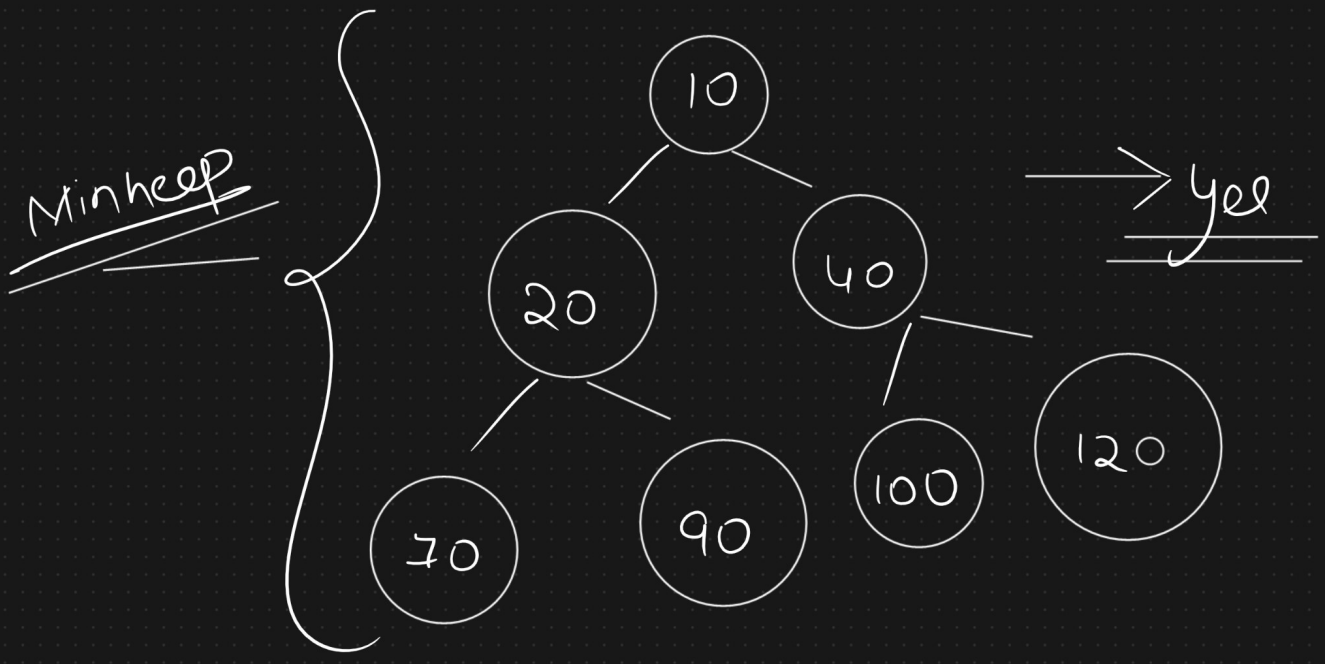


Minheap
1. Complete binary tree
2. Parent node value <= child node value

Maxheap
1. Complete binary tree
2. Parent node value >= child node value







Operations

- Insertion — $\log_2 n$
- Deletion — $\log_2 n$
- Build/create — $O(n)$

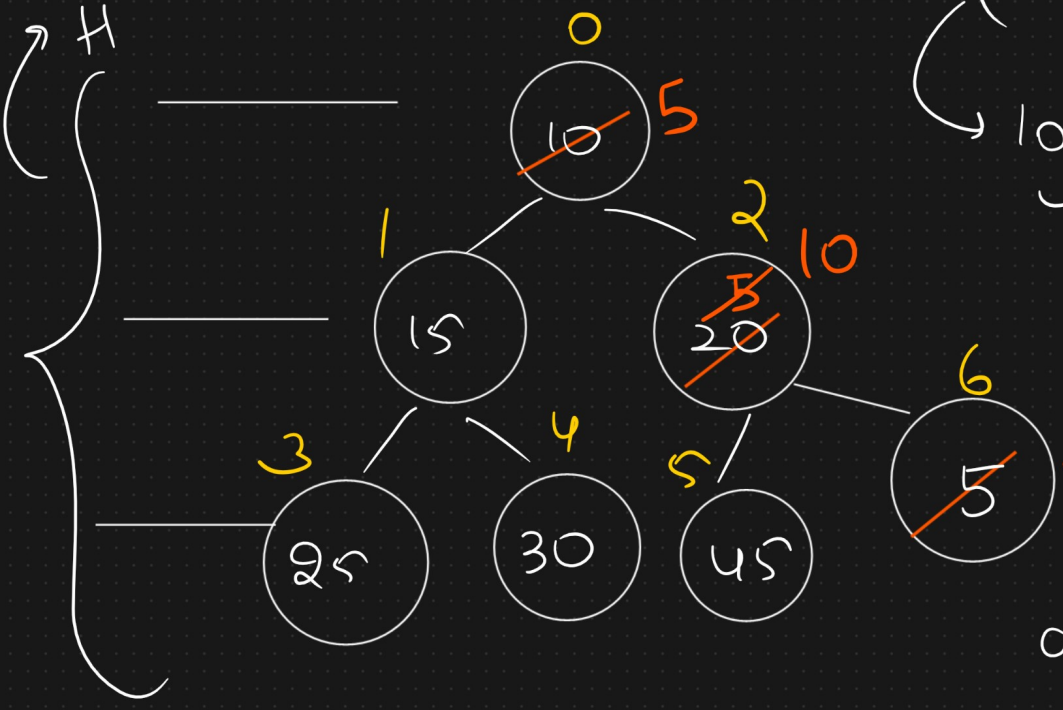
Insertion

{ 1 comparison
1 swap

$\log_2 n$

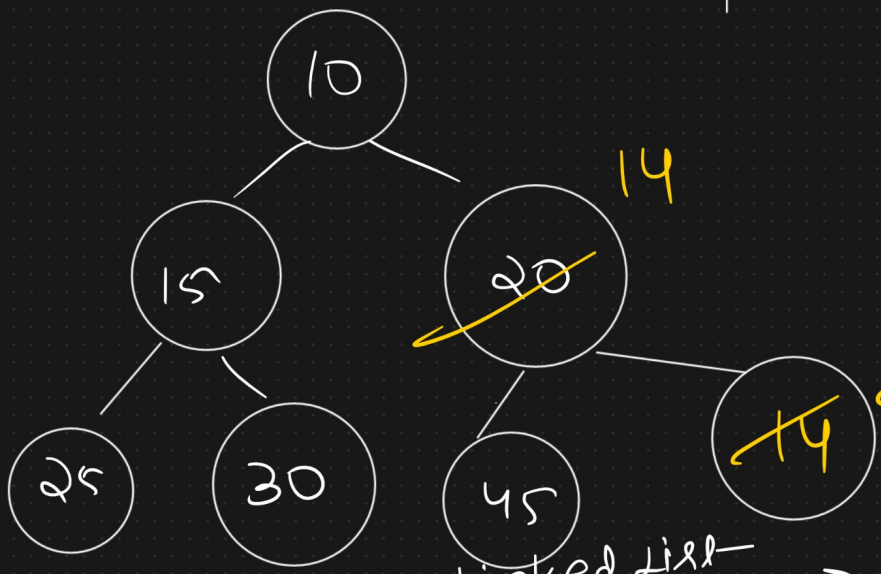
worst case

$O(\log_2 n)$



arr(6) = 5
arr(h) = 5

0	1	2	3	4	5	6
5 ~~10~~	15	10 ~~20~~	25	30	45	~~5~~ 20

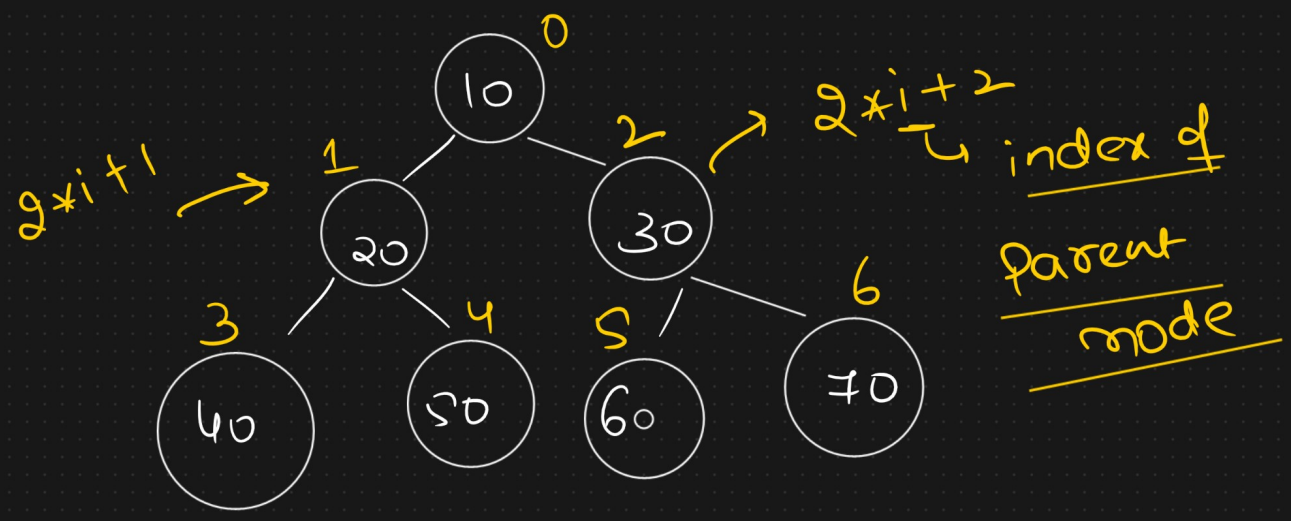


Linked list → Pointers

↳ space wastage

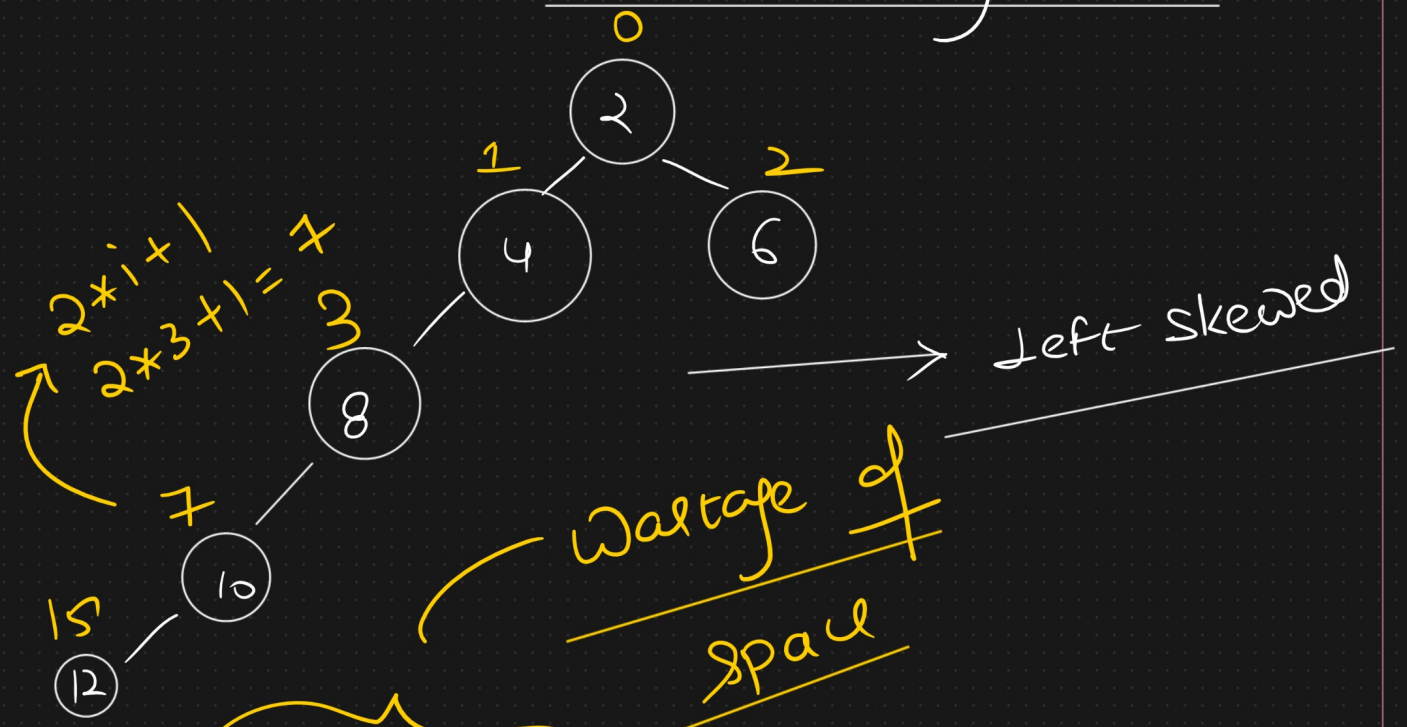
↳ Random access

Balanced

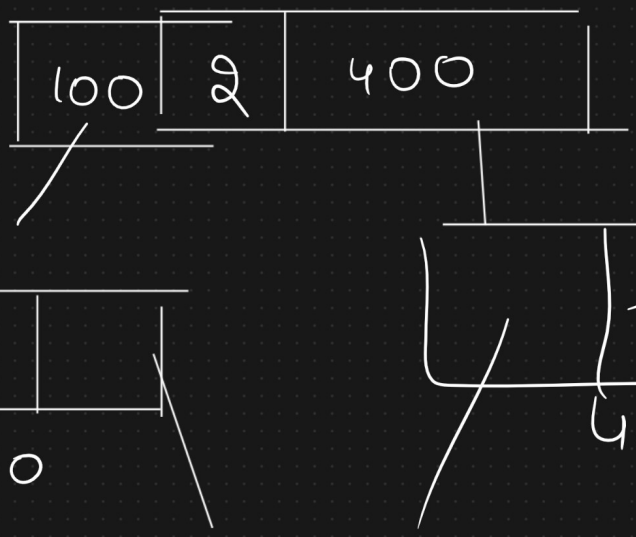


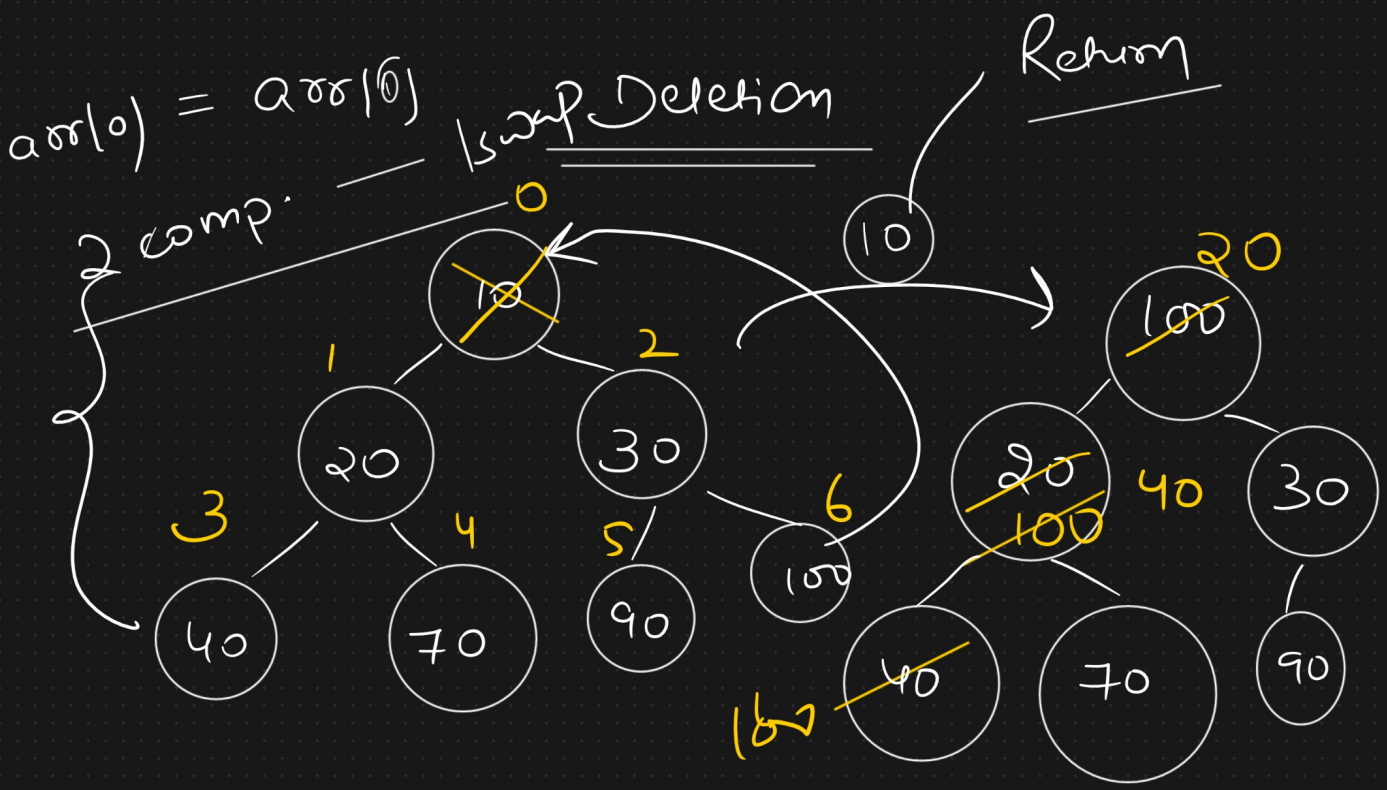
0	1	2	3	4	5	6
10	20	30	40	50	60	70

Skewed binary tree



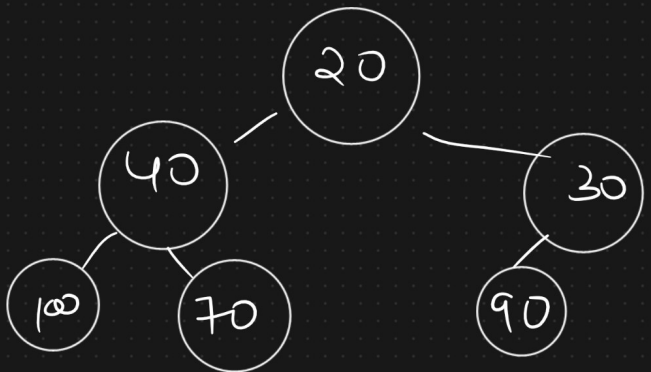
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	4	6	8				10								12


100
2
400
4
100
6
400


arr[0] = arr[6]
2 comp.
Swap Deletion
Return
0
10
1
20
2
30
3
40
4
70
5
90
6
100
20
100
20
40
30
100
40
70
90
100


(100)
(20, 30)


(100)
(40, 70)


20
40
30
100
70
90

$O(\log_2 n)$

$2 \log_2 n$

$\nearrow 2$

Time complexity

Comparison

$\searrow 1 \log_2 n$

$\searrow 1$

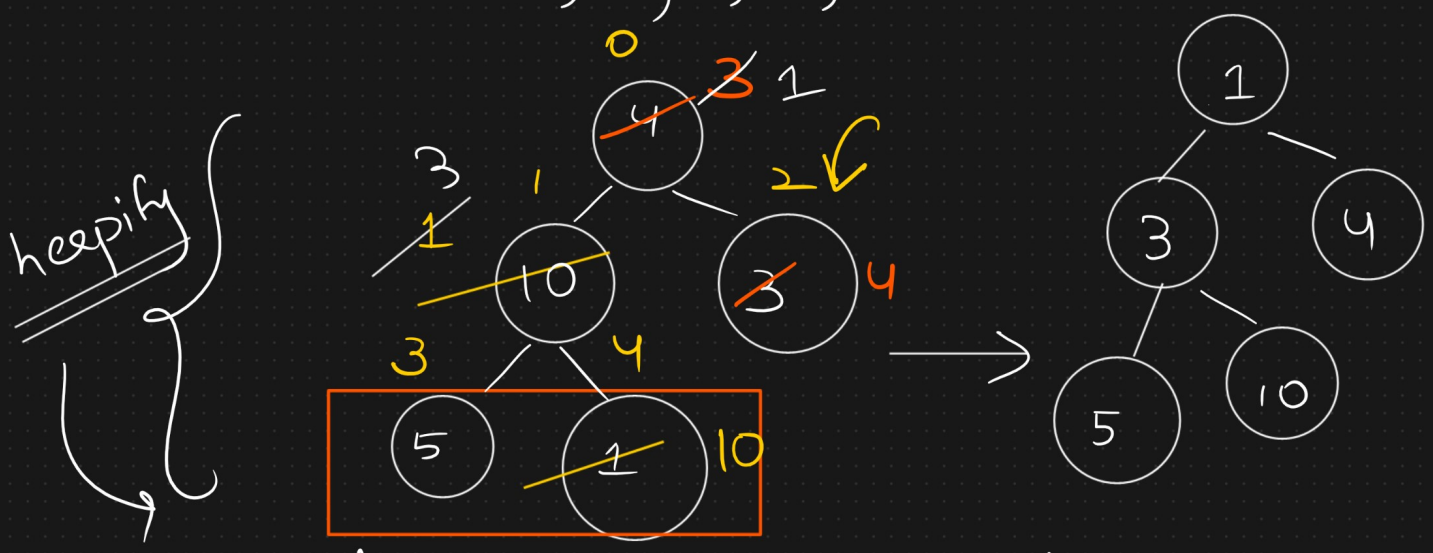
Swap

$O(\log_2 n)$

Build Minheap

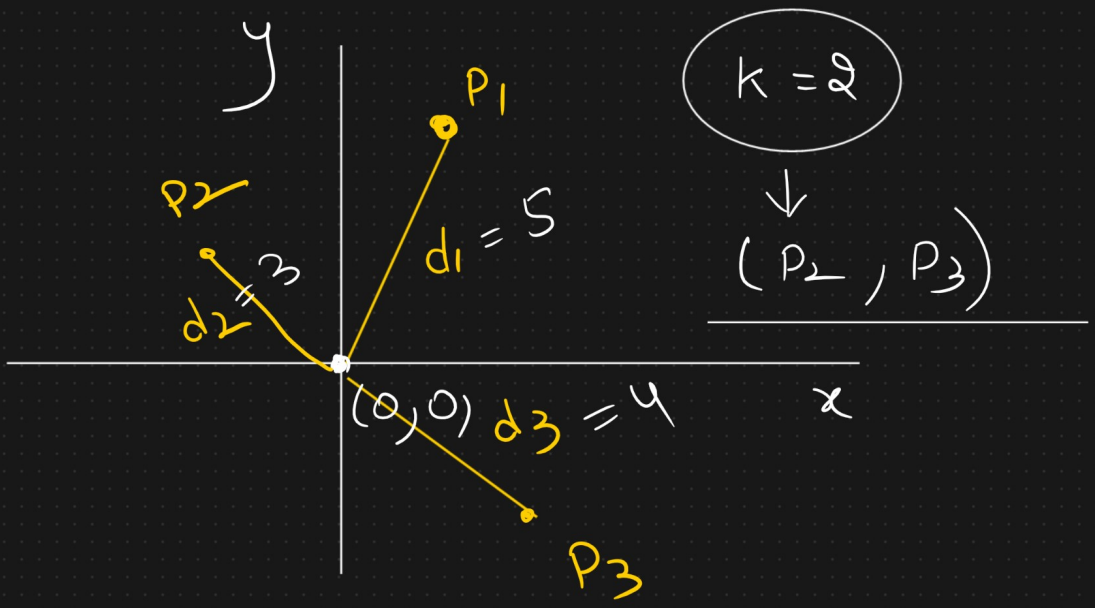
Minheap

4, 10, 3, 5, 1



Compute Mathematical Derivation

$\hookrightarrow O(n)$



Euclidean Distance

P_1 —— d —— P_2

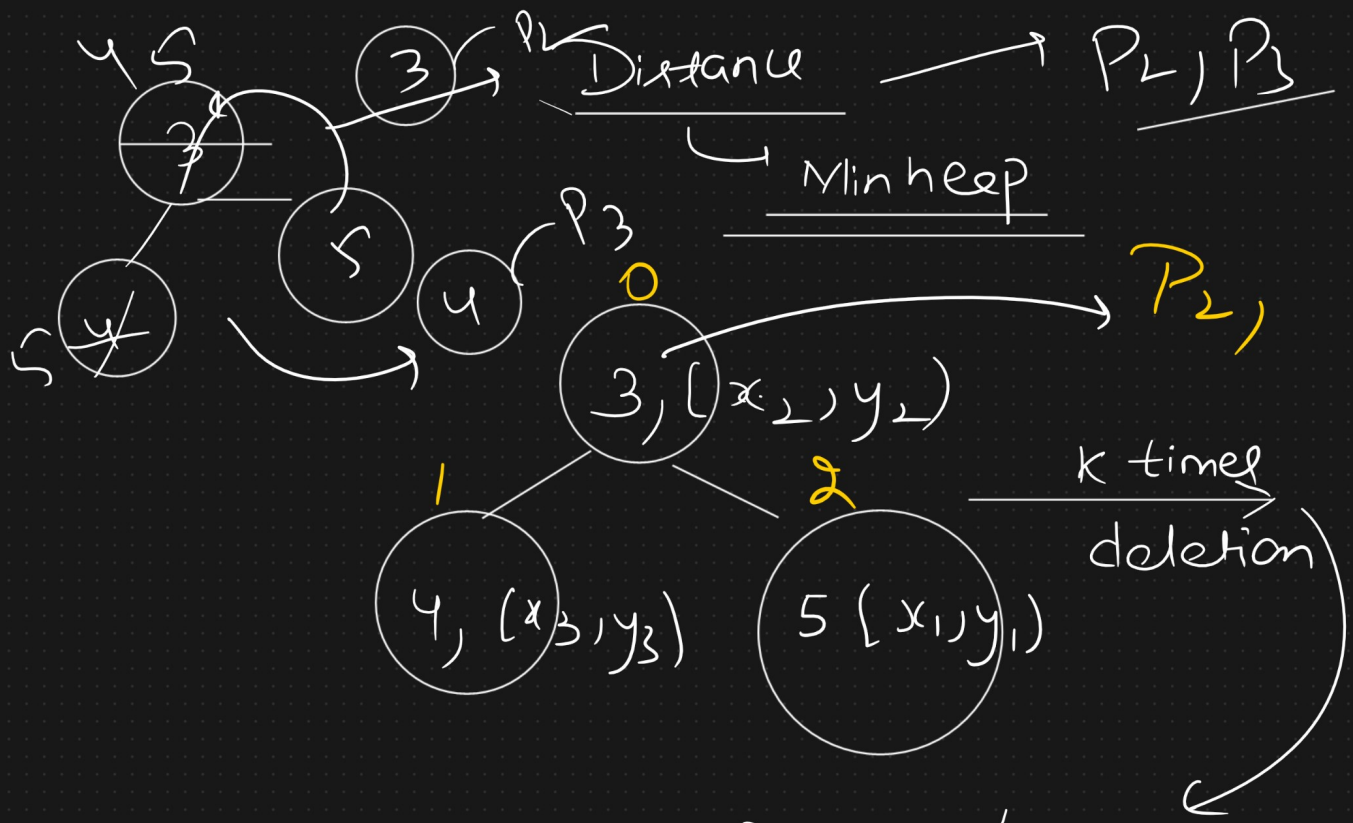
(x_1, y_1) (x_2, y_2)

$(0,0)$ (x, y)

$$d = \sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}$$

$$= \sqrt{x^2 + y^2}$$

$$= x^2 + y^2$$



top 2 min. / closest

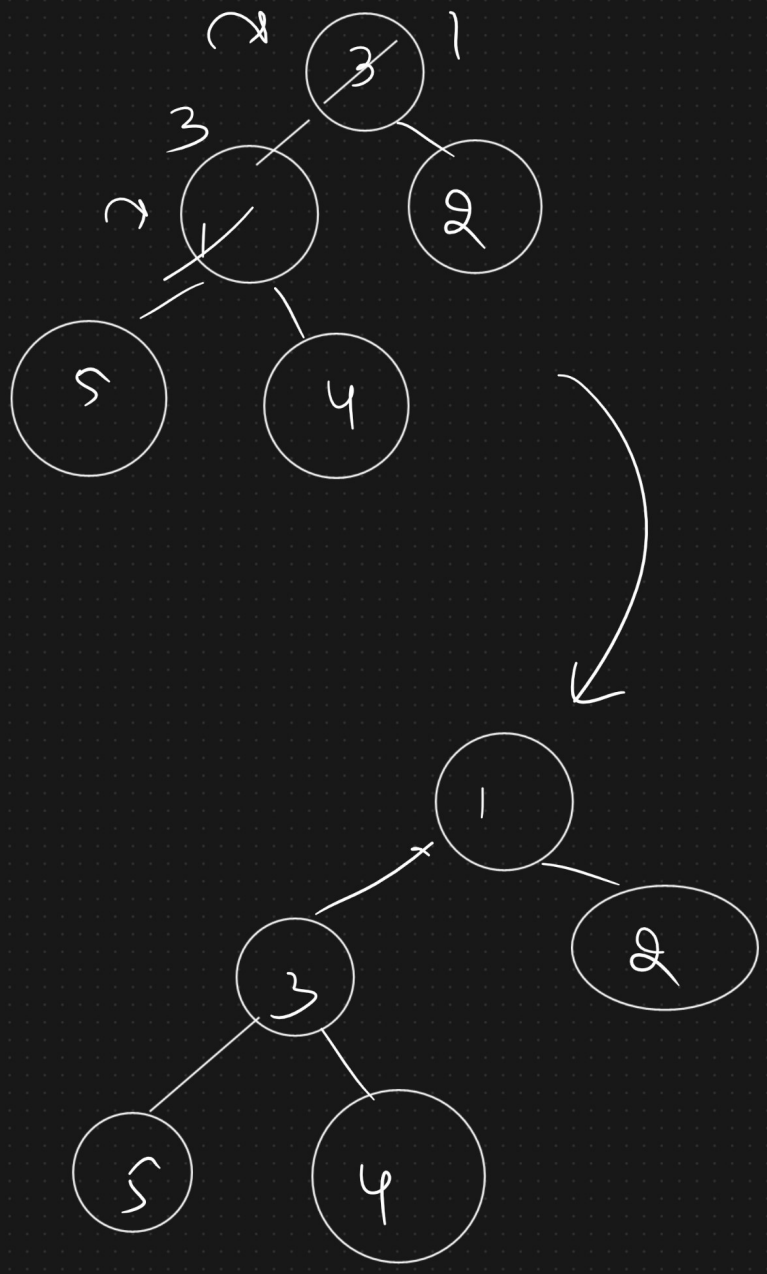
Points to origin

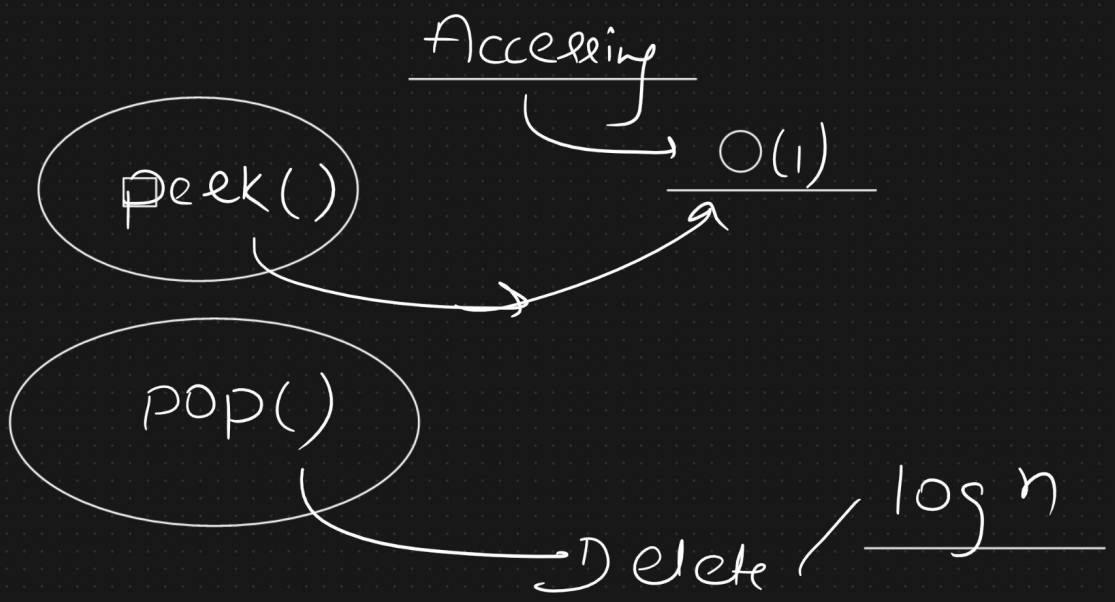
kth smallest / k closest-element

↳ Minima / Maxima

↳ Heap

3, 1, 2, 5, 4




Accessing
O(1)
peek()
pop()
Delete / log n