CS 2700 : PROGRAMMING & DATA STRUCTURES.

PRIORITY QUEUE ADT.

GOALS:

- WHAT IS A PRIORITY QUEUE?
- IMPLEMENTATION
- APPLICATION

PRIORITY QUEUE.

Printer Queue

J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
(1)	(1)	(100)	(2)	(2)	(1)	(4)	(1)

- Jobs are processed in queue order. (FIFO)

In Practice

J_3 : book print

J_6 : flight ticket.

PRIORITY QUEUE.

- Each element has a $\underbrace{\text{priority}}_{\text{part of input}}$ assigned

- Assumptions :
  - priority values are distinct
  - lower value $\Rightarrow$ higher priority

PRIORITY QUEUE. (ADT)

- Each element has a $\underbrace{\text{priority}}_{\text{part of input}}$ assigned

Operations supported:

- Insert (enqueue)
- DeleteMin (dequeue)
  ↳ remove element with highest priority.

PRIORITY QUEUE. (ADT)

- Each element has a $\underbrace{\text{priority}}_{\text{part of input}}$ assigned
- < key , value> pair is inserted

  ↳ priority
- We will insert only keys (illustration)
- retrieve element with highest priority

  (and delete it)

PRIORITY QUEUE

I 10, I 20, I 5 , I 6 , I 13

D , D , I 2 , I 12, D

Write down output of "D" operations

D : 5

D : 6

D : 2

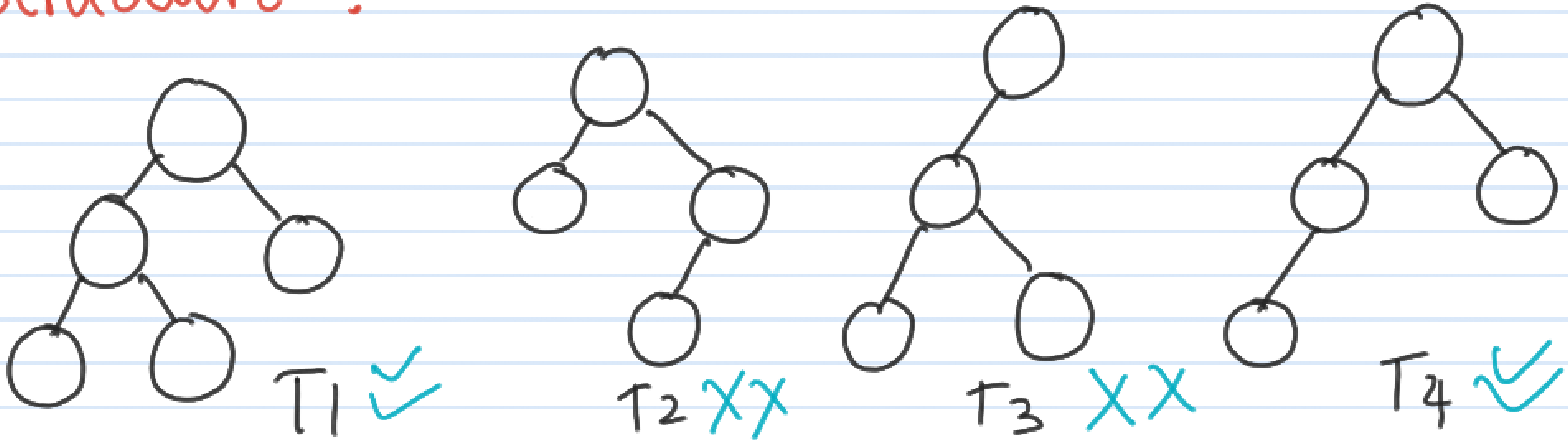
PRIORITY QUEUE (IMPLEMENTATIONS)

	INSERT	DELETE MIN.
- LINKED LIST	$O(1)$	$O(n)$
- SORTED LINKED LIST	$O(n)$	$O(1)$
- BINARY SEARCH TREE	$O(n)$	$O(n)$
- AVL TREE	$O(\log n)$	$O(\log n)$

PRIORITY QUEUE (IMPLEMENTATION)

Binary heap : Special Binary Tree

Structure :



Heap order : Parent key < child key

PRIORITY QUEUE (IMPLEMENTATION)

Binary heap :

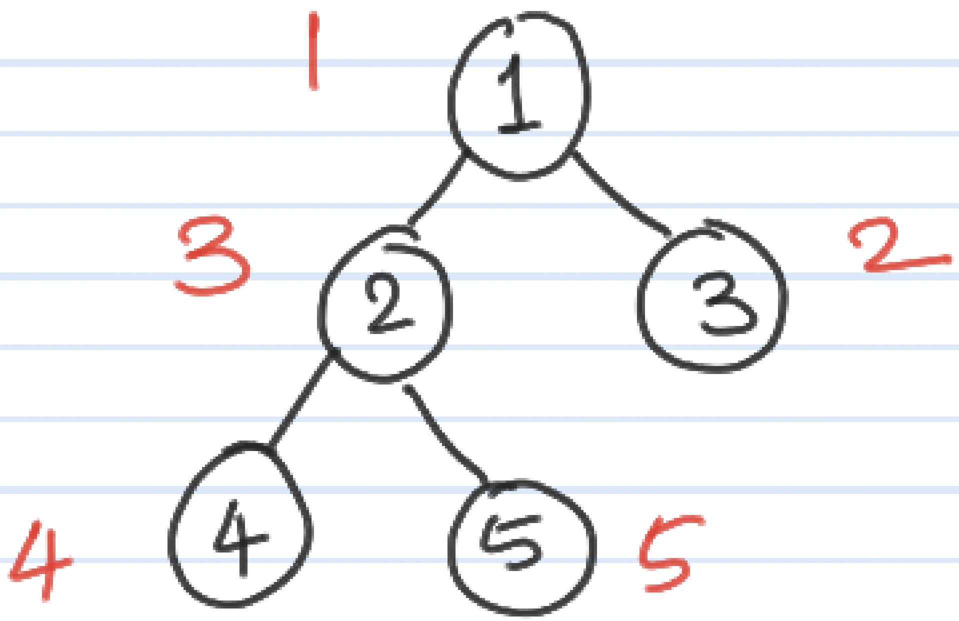
Special Binary Tree

- Complete binary Tree with possibly last level not full
- Last level : left to right
- Heap order

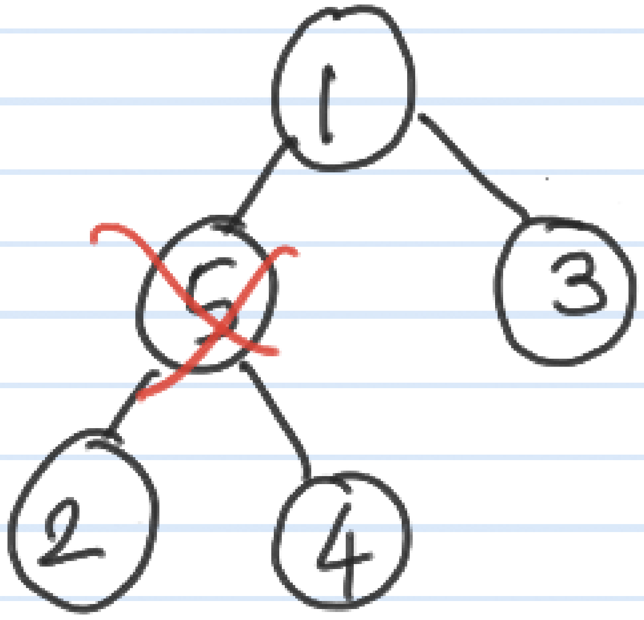
1, 2, 3, 4, 5

1 2, 5, 3, 4

1 3 2 4 5



T_1

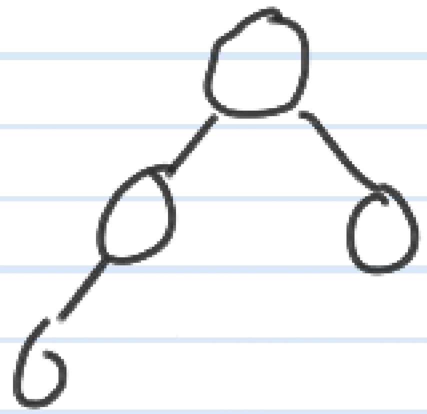


T_2

PRIORITY QUEUE (IMPLEMENTATION)

Binary heap : Special Binary Tree

- Complete binary Tree with possibly last level not full
- Last level : left to right

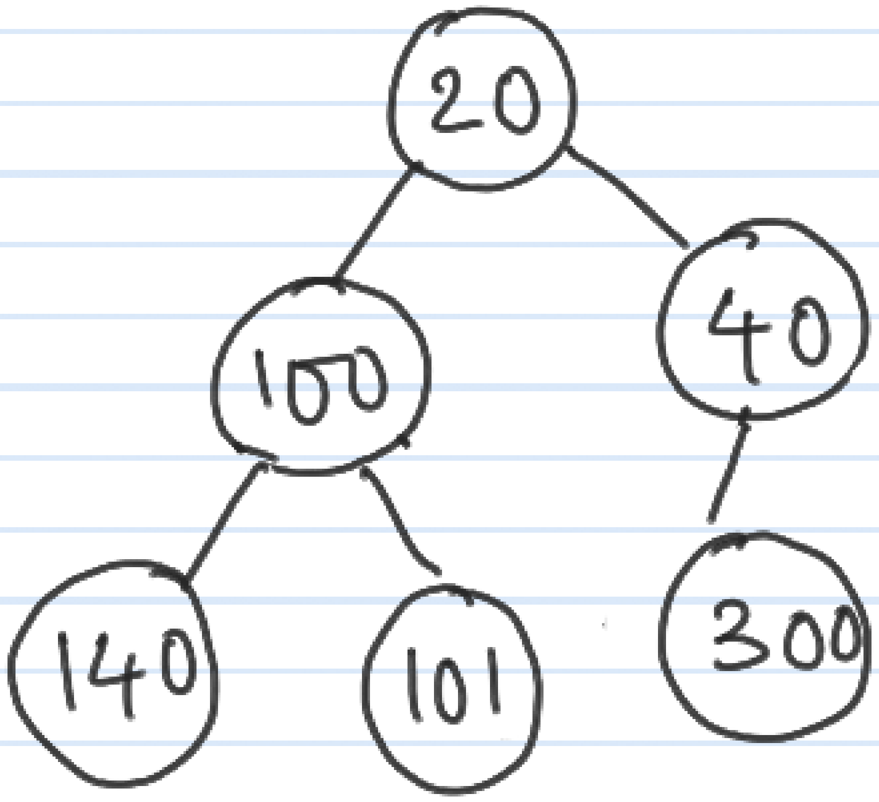


- Heap order
- # of nodes in a heap of height h

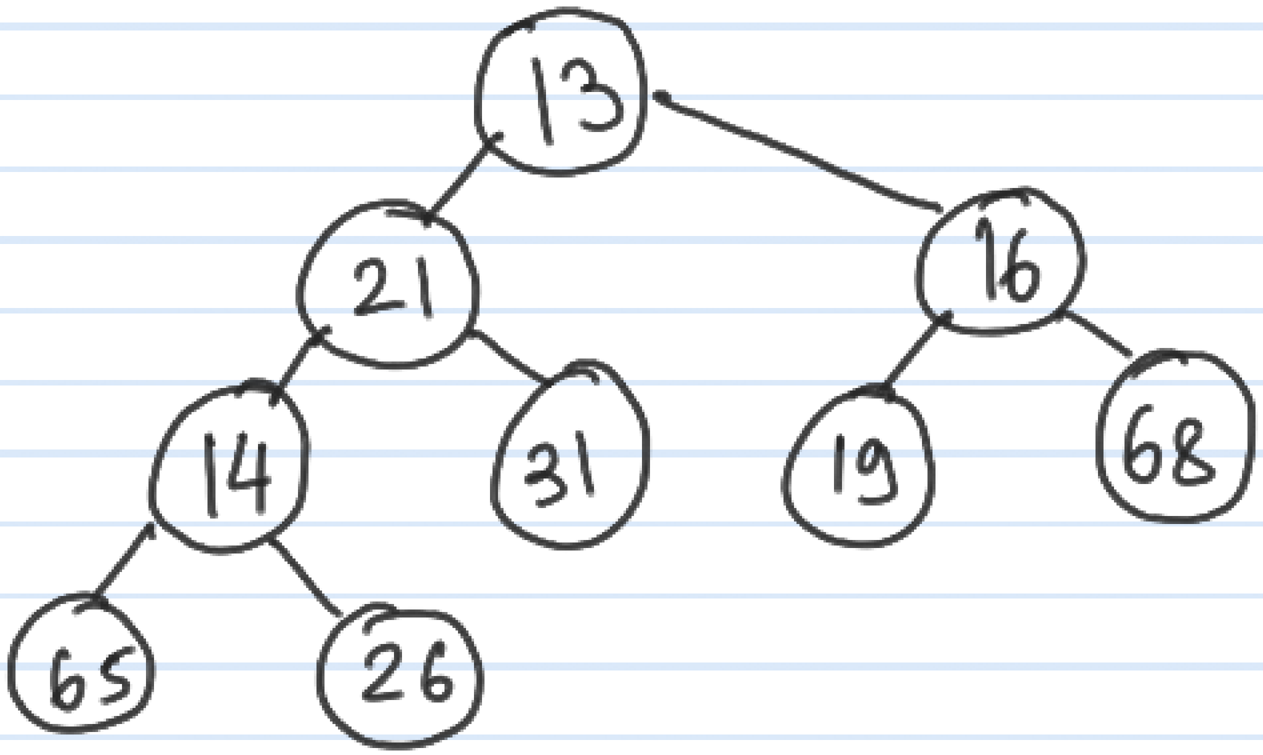
$h=2$: $2^2 \dots 2^3-1$ $[4, \dots 7]$
$h=3$: $2^3 \dots 2^4-1$

general case : $(2^h) \dots (2^{h+1}-1)$

BINARY HEAP (MIN Heap)



T_1



T_2

Structure

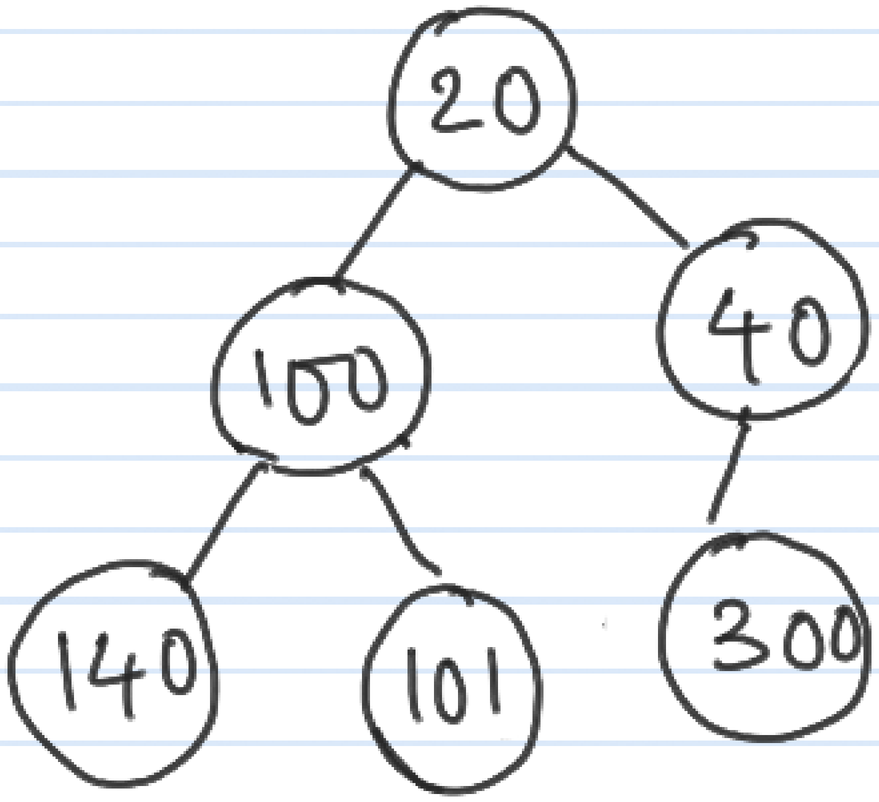
Heap order

Heap Order vs BST property

BINARY HEAP (MIN Heap)

Structure

Heap order



T_1

• Conveniently represented as an array

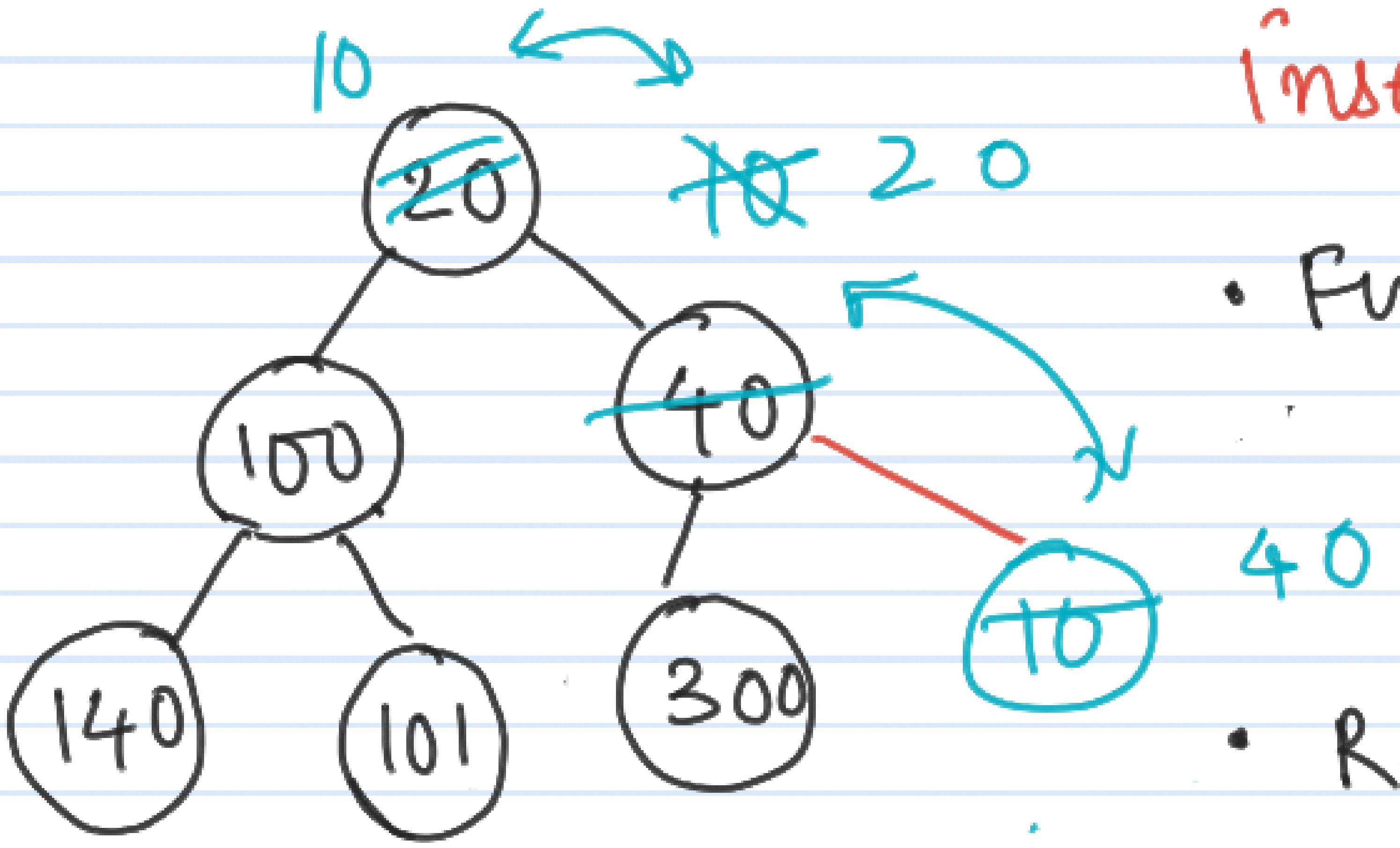
	20	100	40	140	101	300	
0	1	2	3	4	5	6	

index i

Parent :

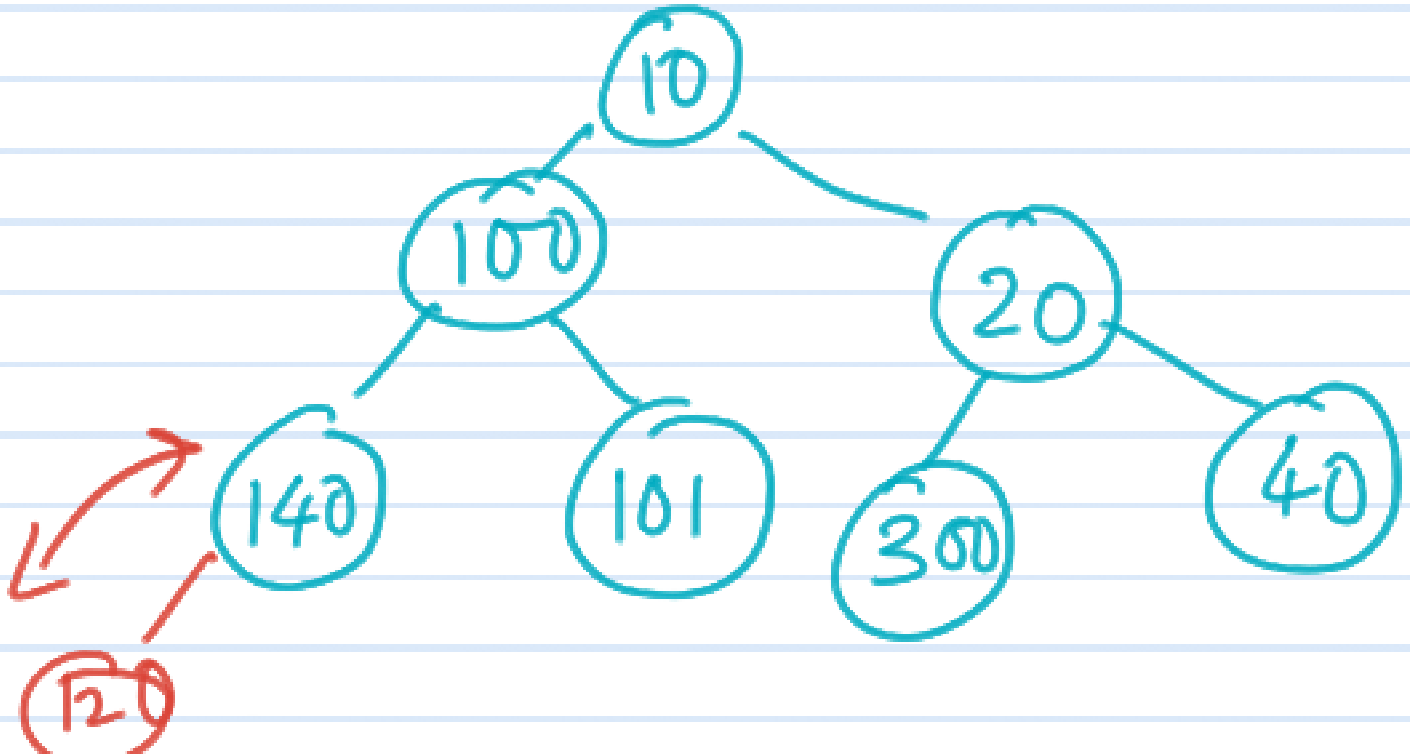
children :

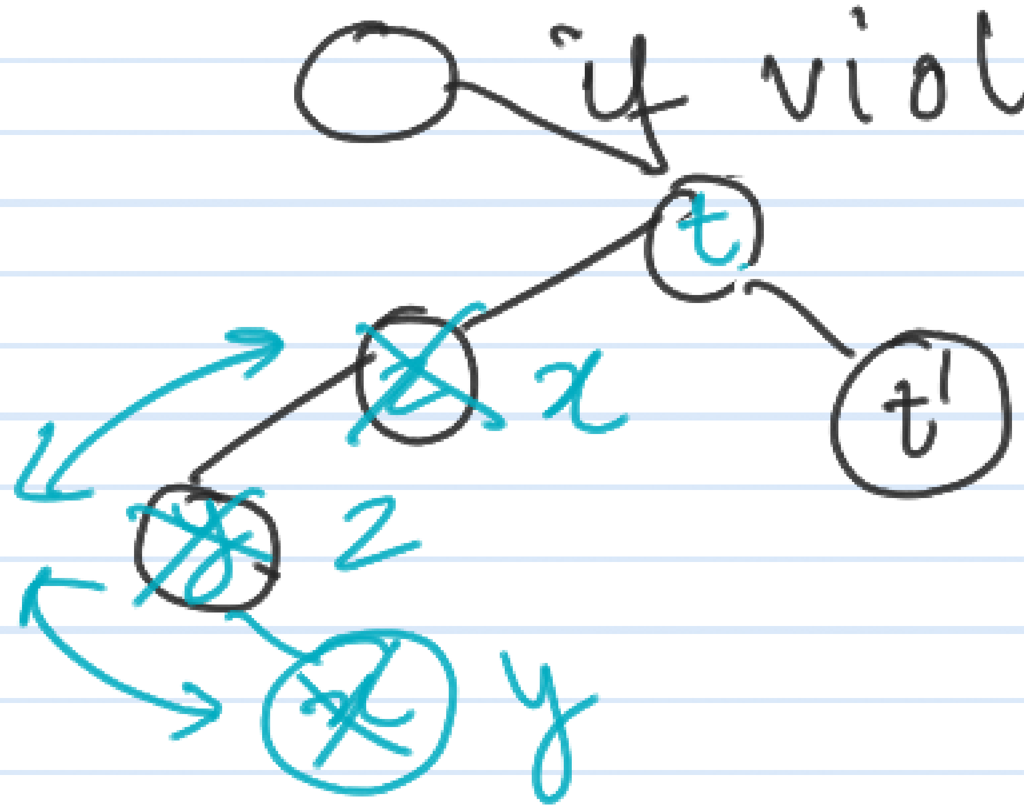
BINARY HEAP (MIN Heap)



insert (120)

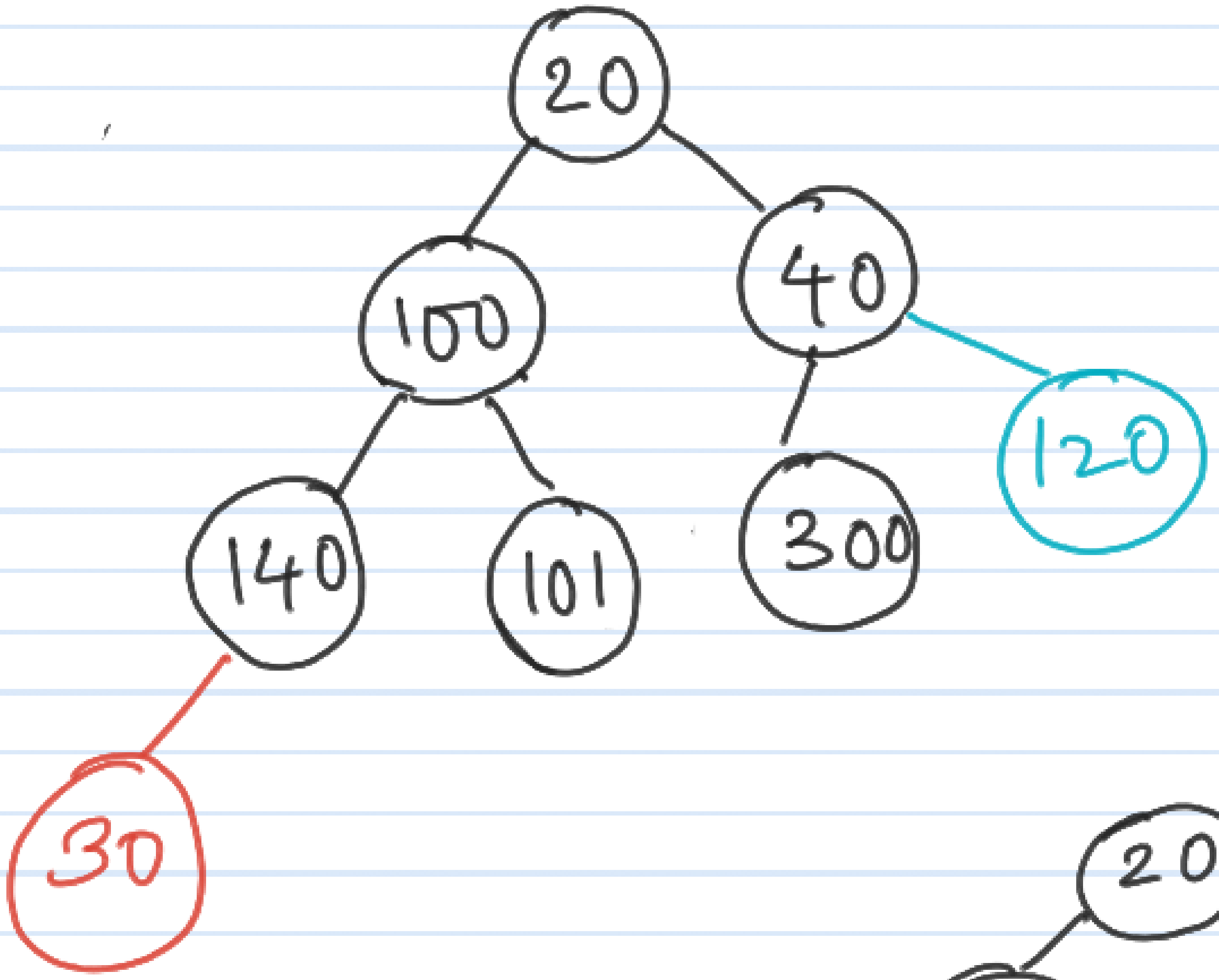
- Find next available hole (location)
- Restore heap order if violated.

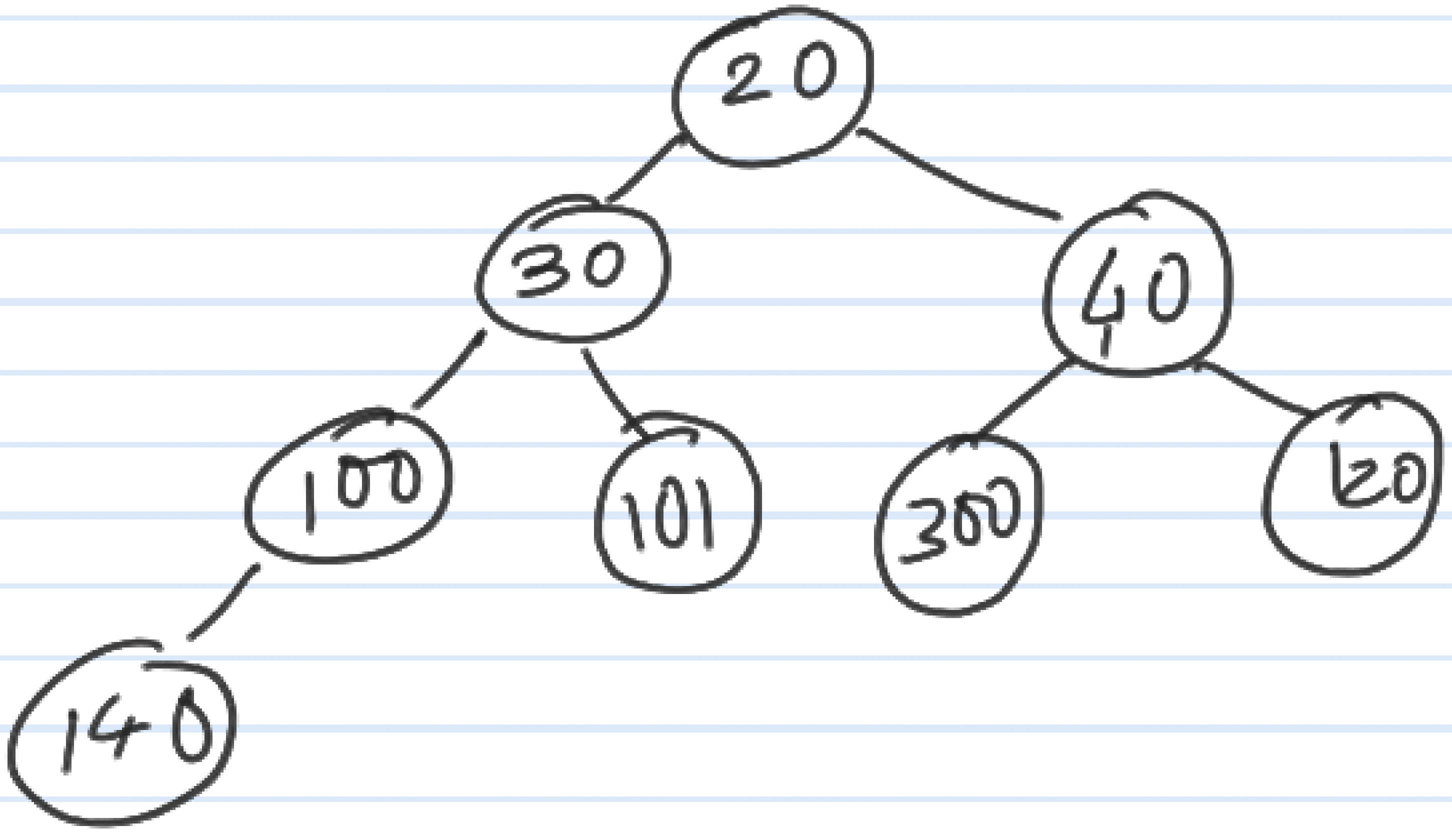




BINARY HEAP (MIN Heap)

- insert (120)
- insert (30)

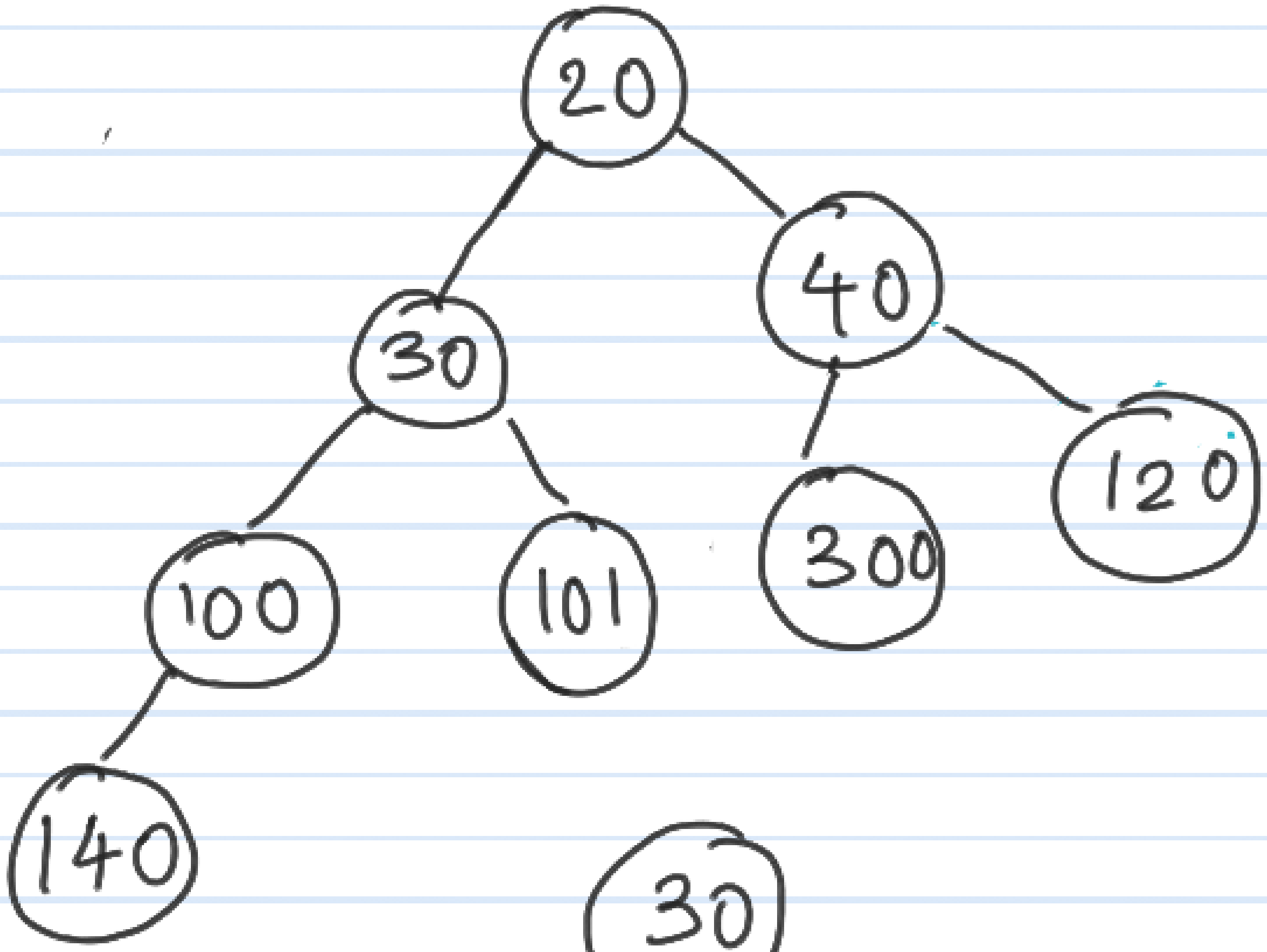


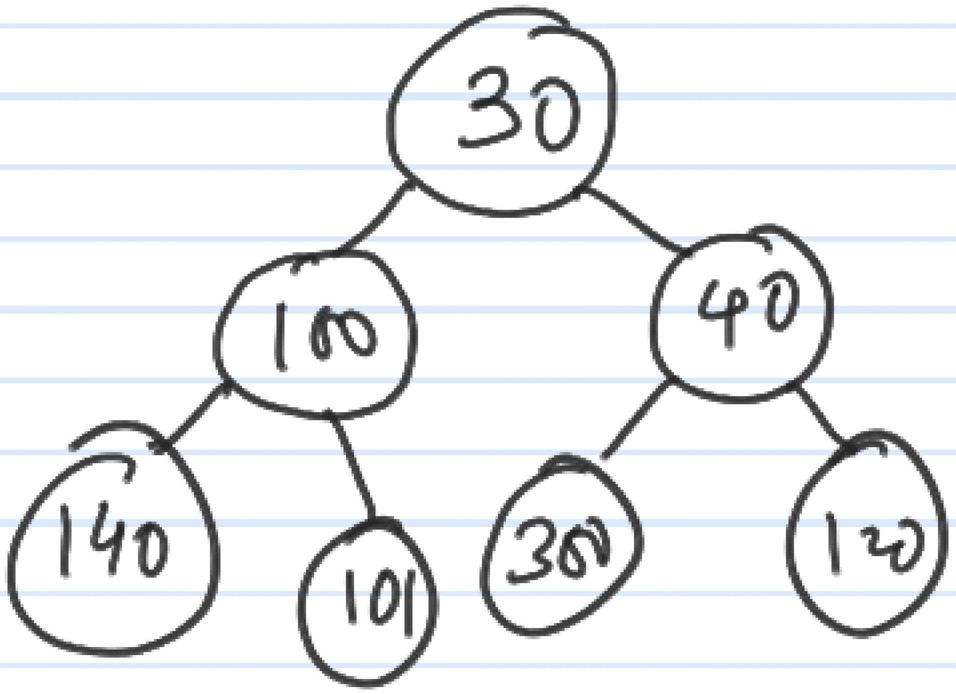


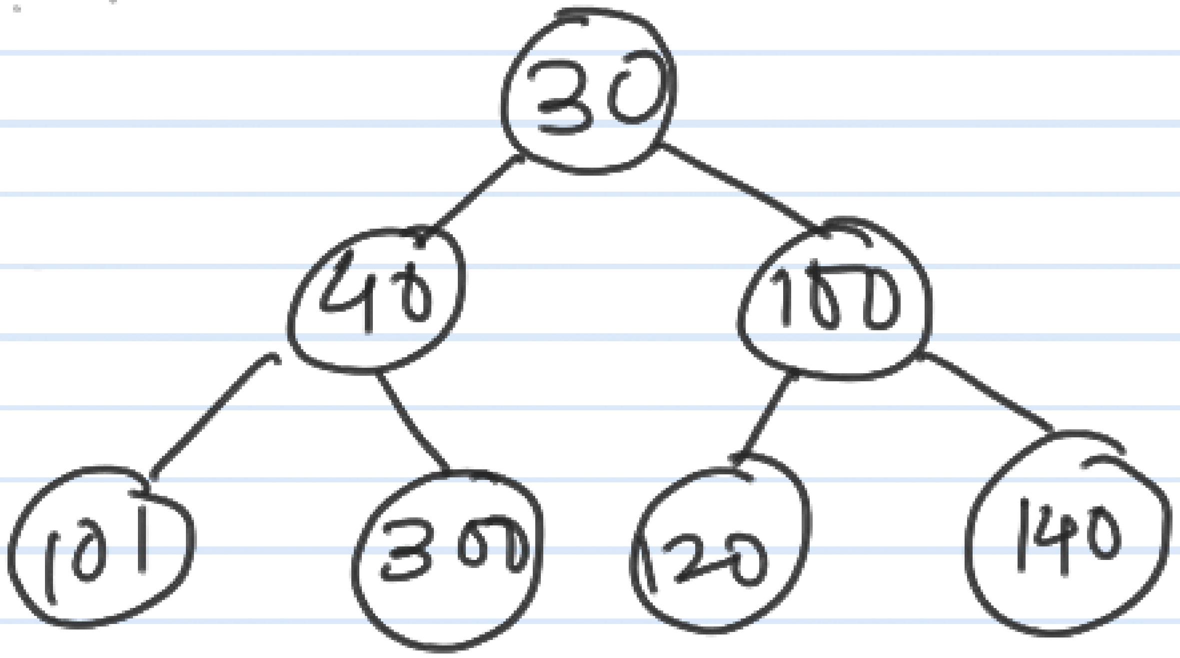
- Restore heap order
- element moves up (percolate up)

BINARY HEAP (MIN Heap)

– Delete Min

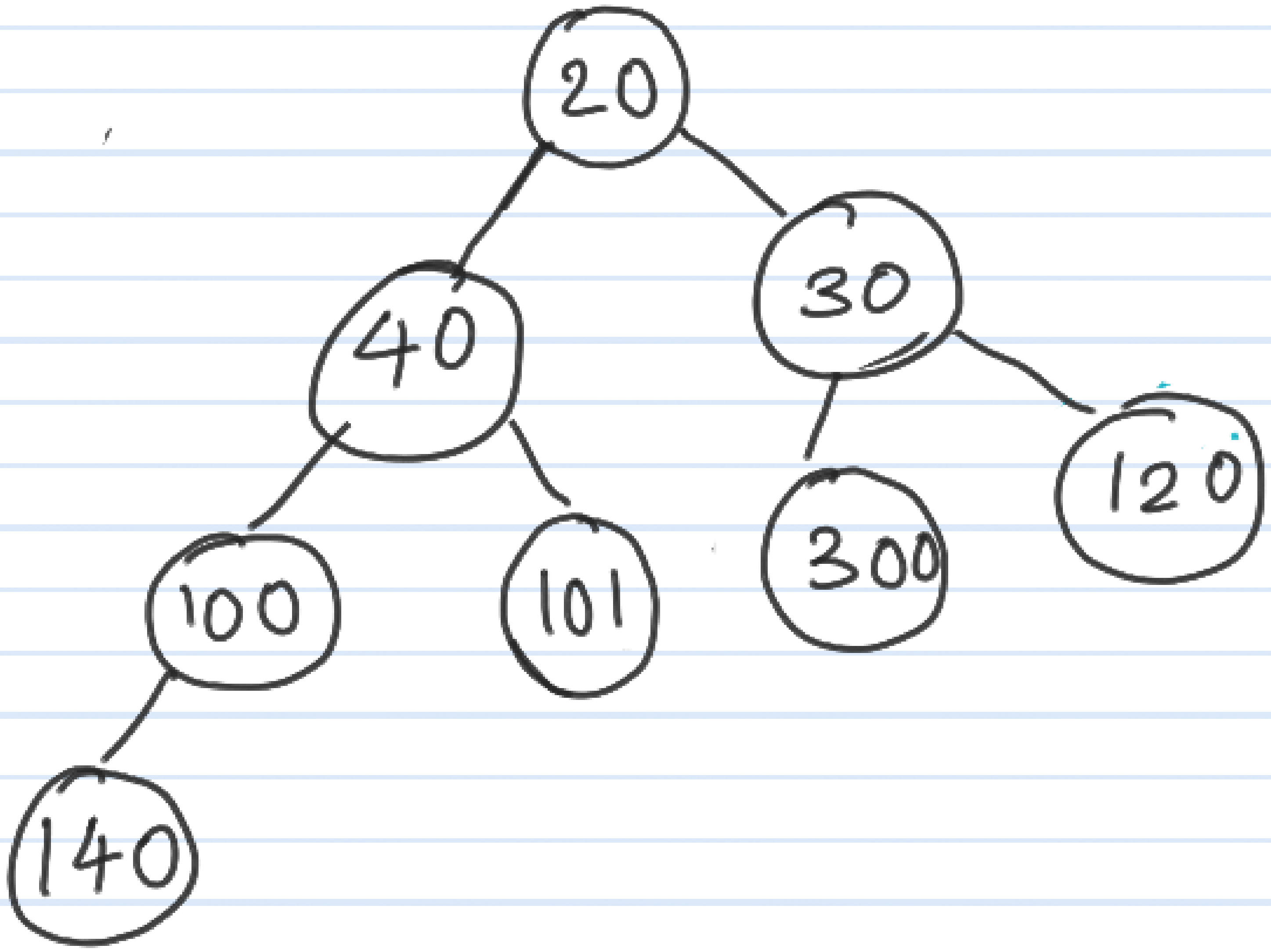




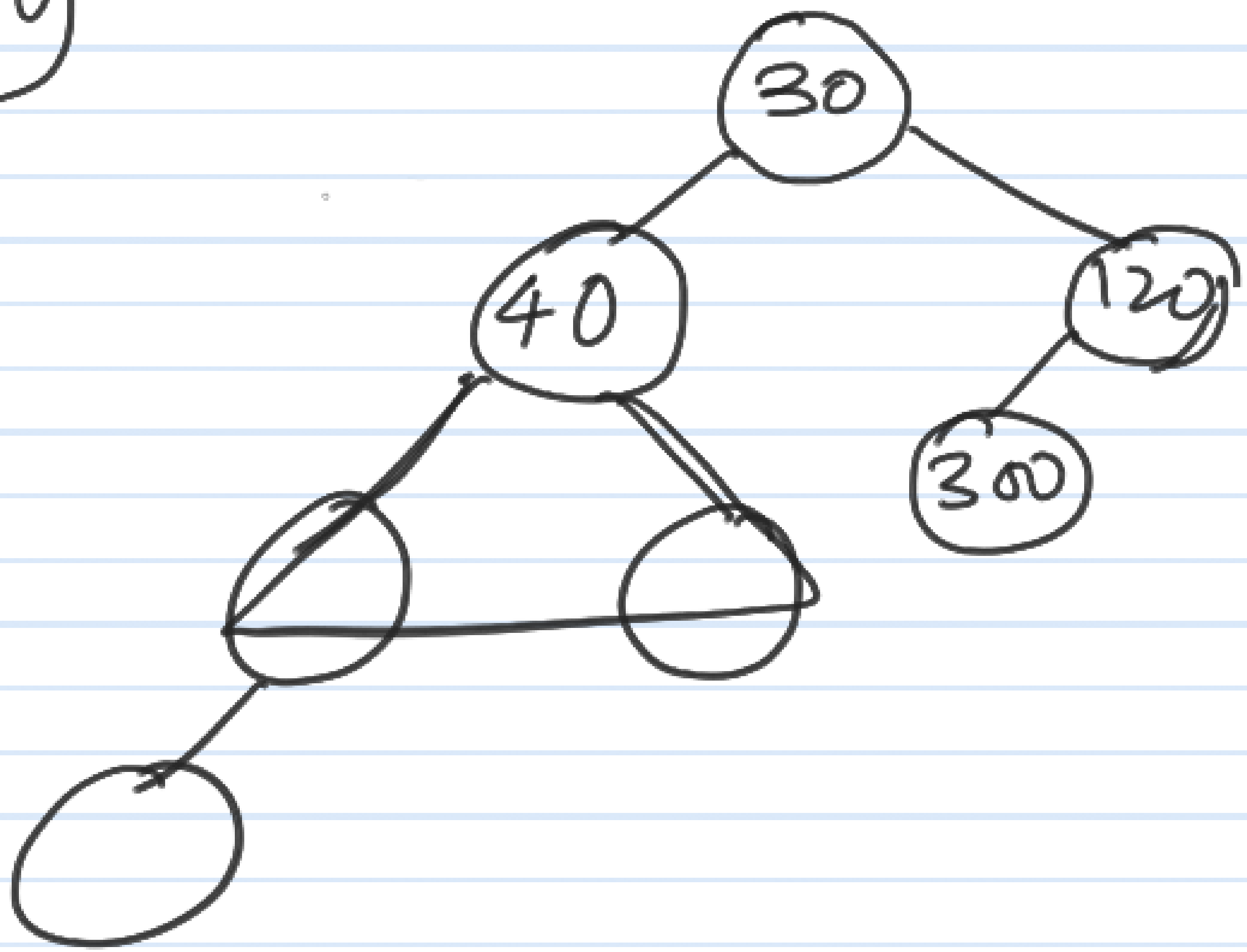


Shape after deletion

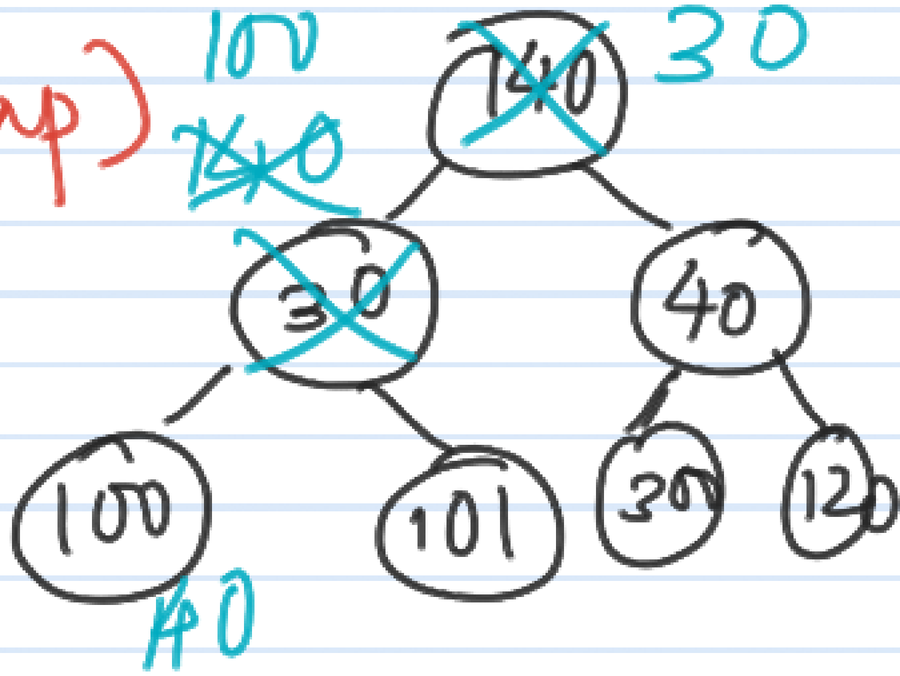
BINARY HEAP (MIN Heap)



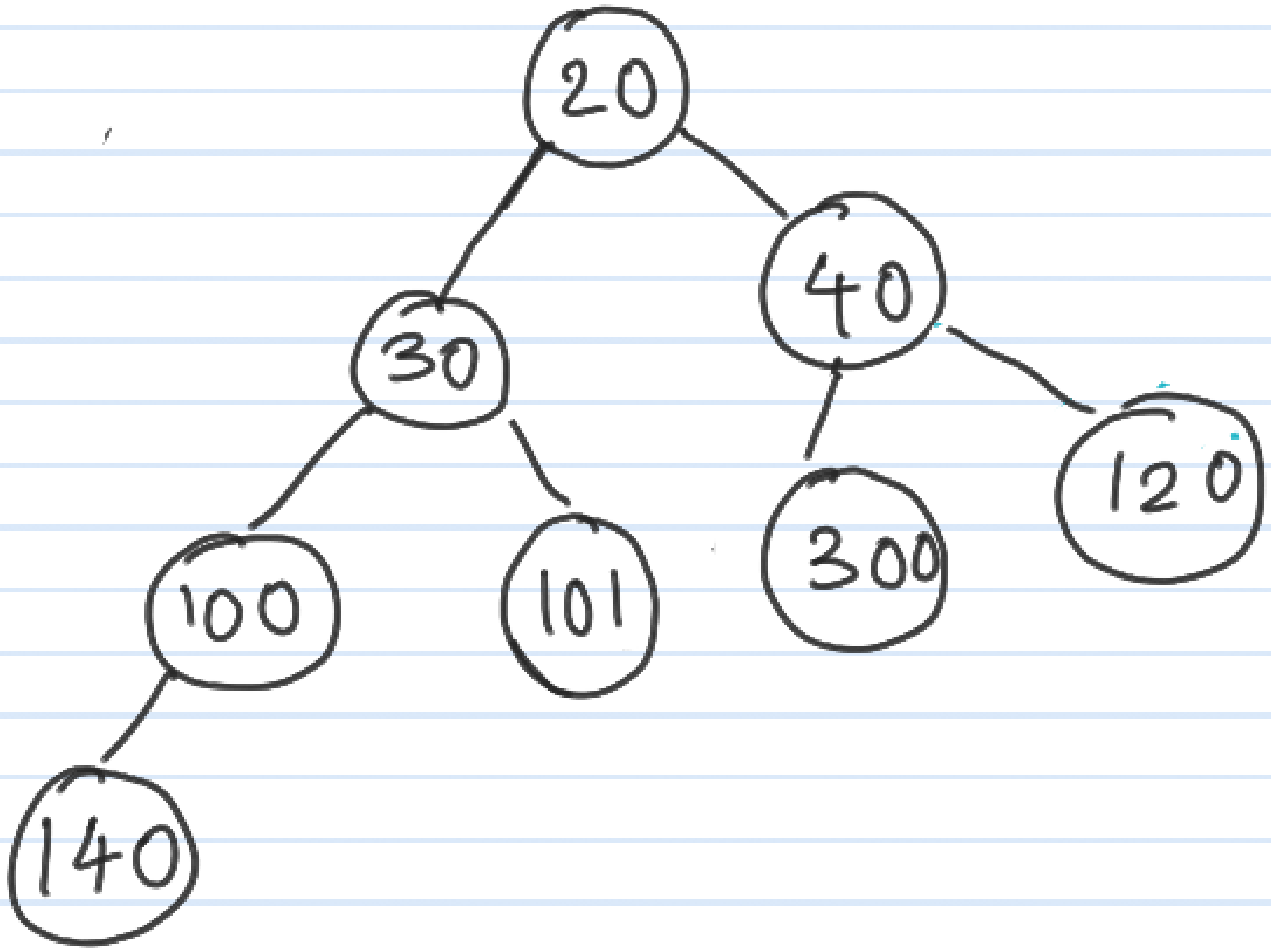
– Delete Min

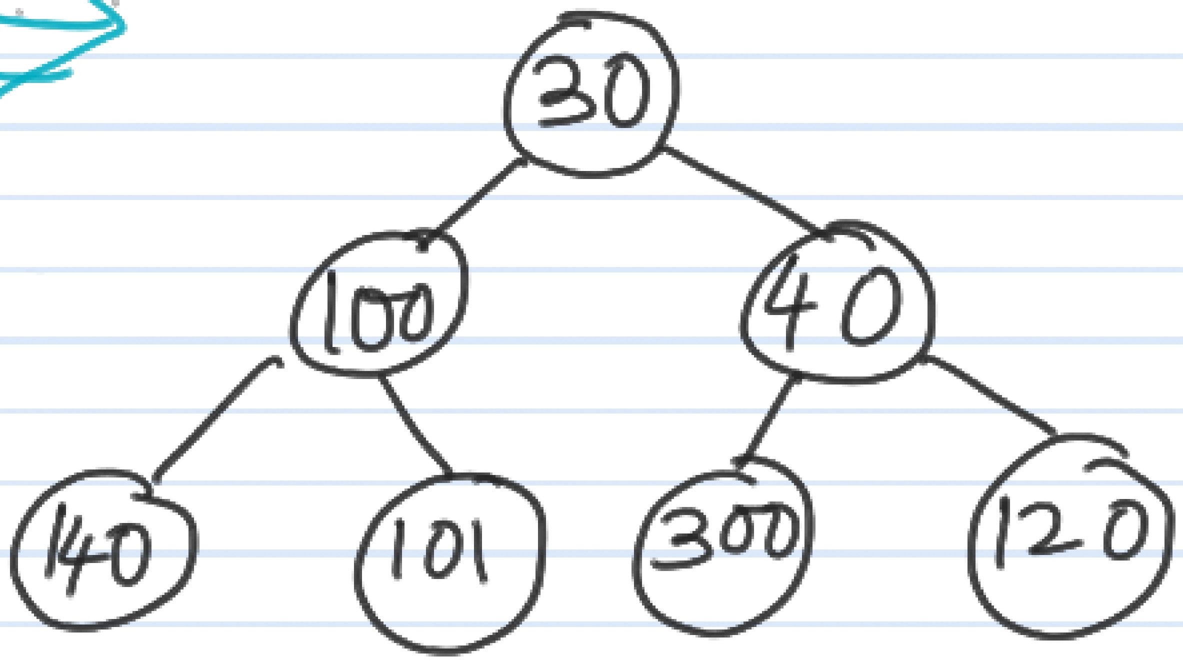


BINARY HEAP (MIN Heap)



- Delete Min

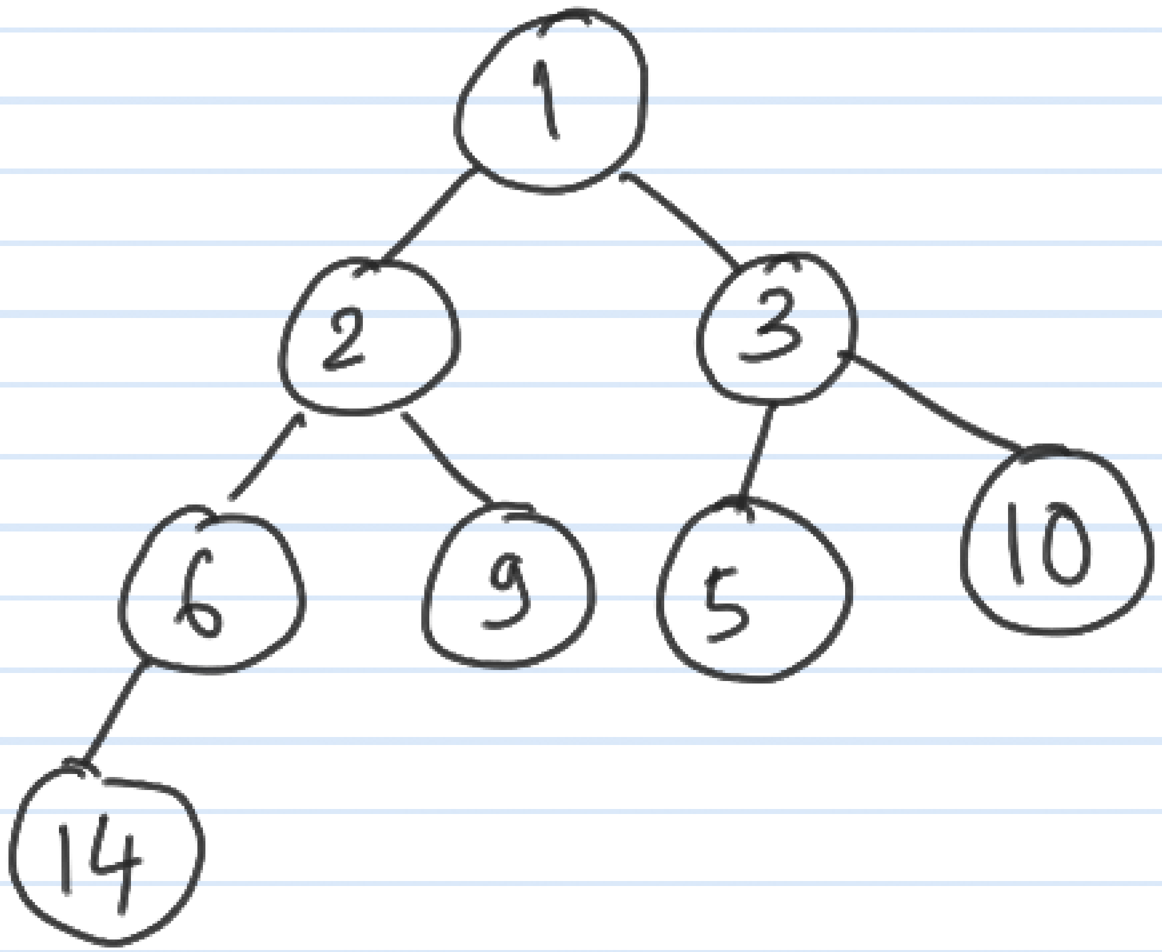




Shape after deletion

- Remove element in root
- Replace root with last elem.
- Percolate down.

BINARY HEAP : EXAMPLE

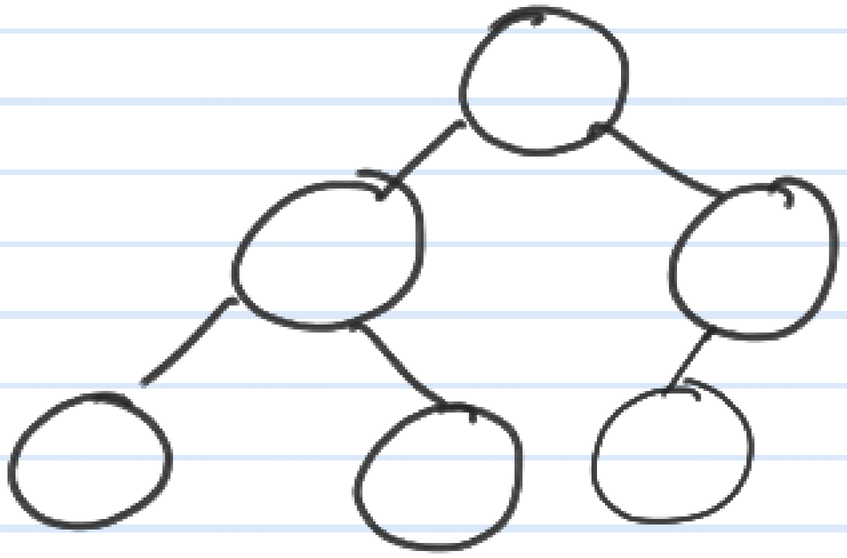


delete min

insert(13)

delete min

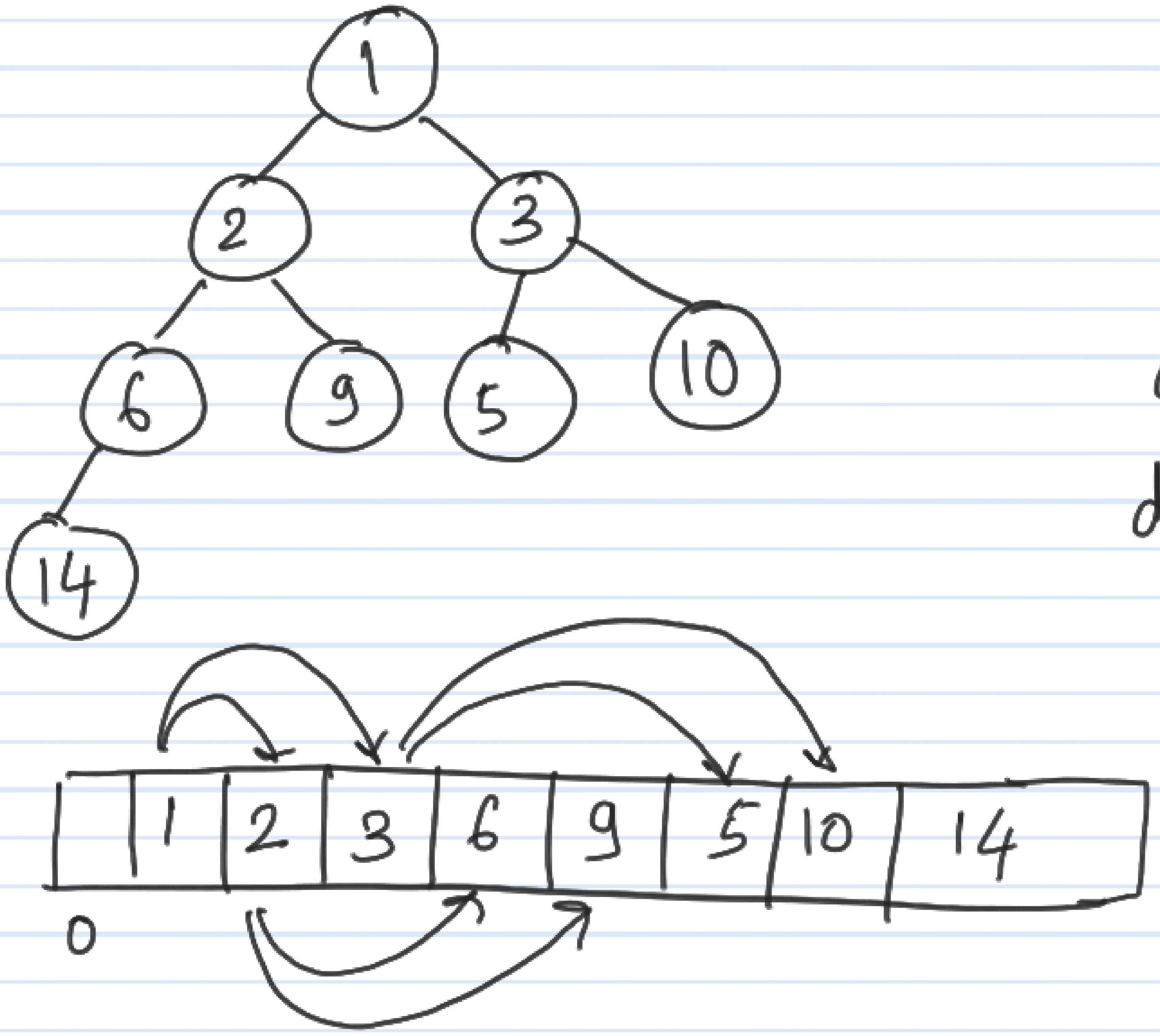
delete min



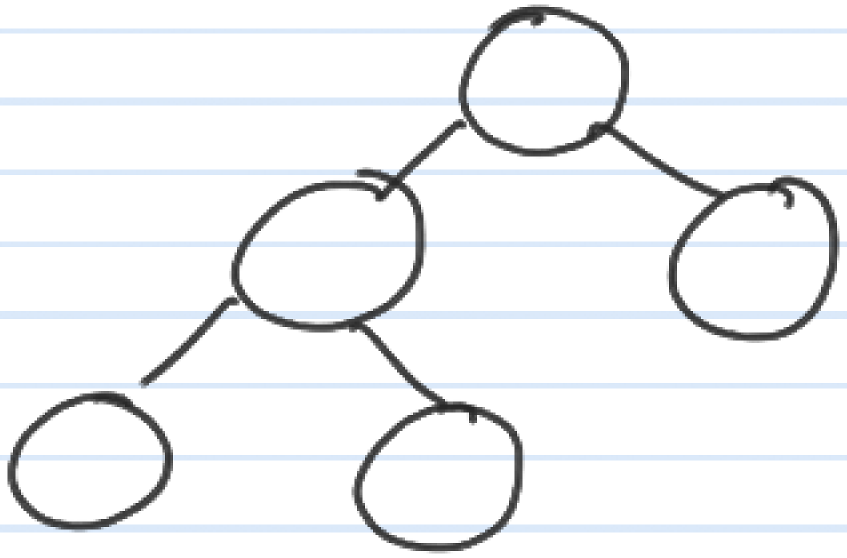
5, 6, 10, 13, 9, 14

6, 9, 10, 13, 14, 9

BINARY HEAP : IMPLEMENTATION



delete min
insert(13)
deletemin
deletemin



insert (e)