Unit I Heaps

Heap

max/min element in O(1)

Topological sort → Scheduling.

queue:

circular

priority queue → ascending / descending.

→ whenever priority needed.

→ scheduling tasks

Implement PQ → heap.

Deque → double ended queue

Input restricted queue

→ Insertion only one end.

→ Deletion both ends.

Output restricted queue

→ Deletion in one end, insertion at both ends.

Advantage:

Time Complexity reduced.

UNIT-1:	Operations
Heaps	insert
Min-Max	delete
Binomial	searching
Fibonacci	min/max → Extract / Find
Leftist	size
Deaps	04/07 Updating single element
	merging.

Unit -2

Splay - principle of locality

10% data ⇒ used for 90% time
90% data ⇒ used for 10% time

- B tree
- B+ tree.
- B* tree

} database application.

{ K-D tree → search in k dimension data
Quad - tree → ~~[illegible]~~ Image processing

Segment tree

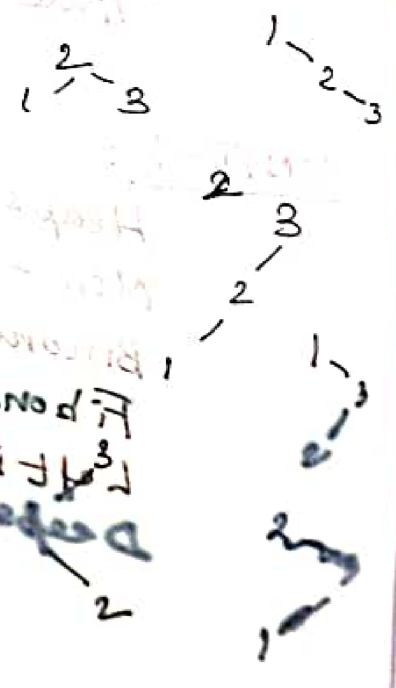
Unit -3

greedy
DP → sub problem interrelated
Backtracking → recursion
Branch and Bound
Divide and Conquer → sub problem independent

OBST
TSP
MCM

23

Matrix Chain Multiplication



Backtracking

N - queen
No queen attack each other

H cycle unweighted

Graph colouring
→ Decision Yes/No
→ Optimization

Unit IV

String Matching → application:
search engine
DNA matching

KMP
Rabin karp.

Unit V

Geometric techniques.

NP < S, R } Vertex cover — set of v. all edges covered.
clique. → Subgraph - complete graph

CT1 - 20 } → might be MCQ
CT2 - 20
10 marks - Assignment
End Sem - 50

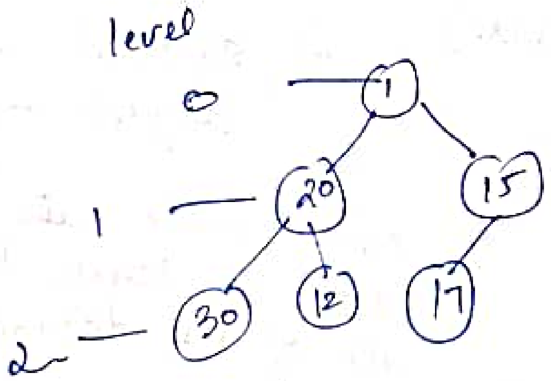
23/1/23

min - max heap:

Strictly BT - except leaves, all nodes have 2 children

Complete BT

left - right fill

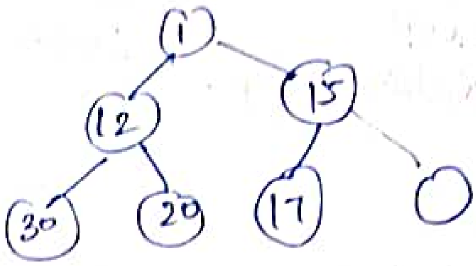


depth
max level.

ancestor
decendants.

min heap
parent → lesser value than children.

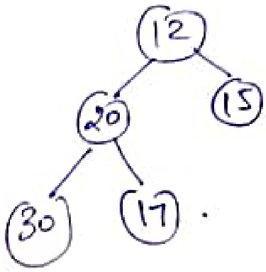
Insertion

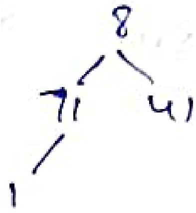


Insert
check property

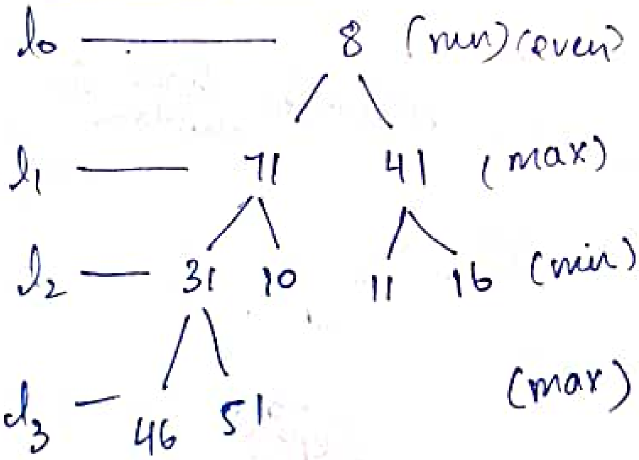
Deletion remove root.
bring next smallest child.

remove 1, 12, 20.
push 17.





Min-Max Heap



1) CBT, alternating min & max
min → even level
max → odd level.

2) smallest - l_0
largest - l_1

3)
* each node in min level is lesser than all of its descendants.
* each node in max level is larger than all of its descendants

max-min heap

max - l_0
min - l_1

Operations :

new node

min

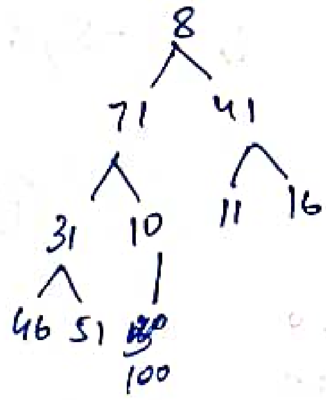
1) p < newnode.
   max @ p is violated.

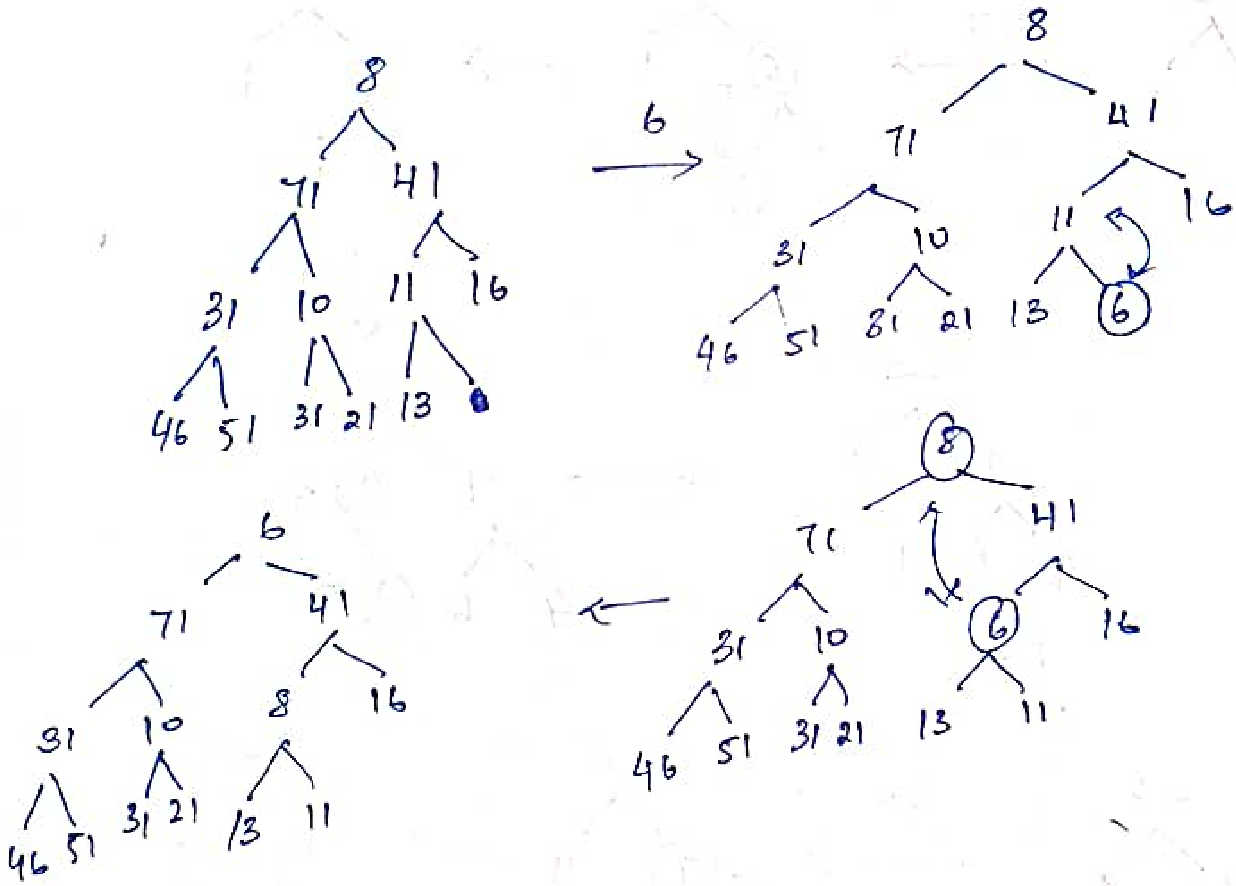
2) gp > newnode
   min @ gp is violated.

max

p > newnode
min @ p ✗

gp < newnode
max @ gp ✗

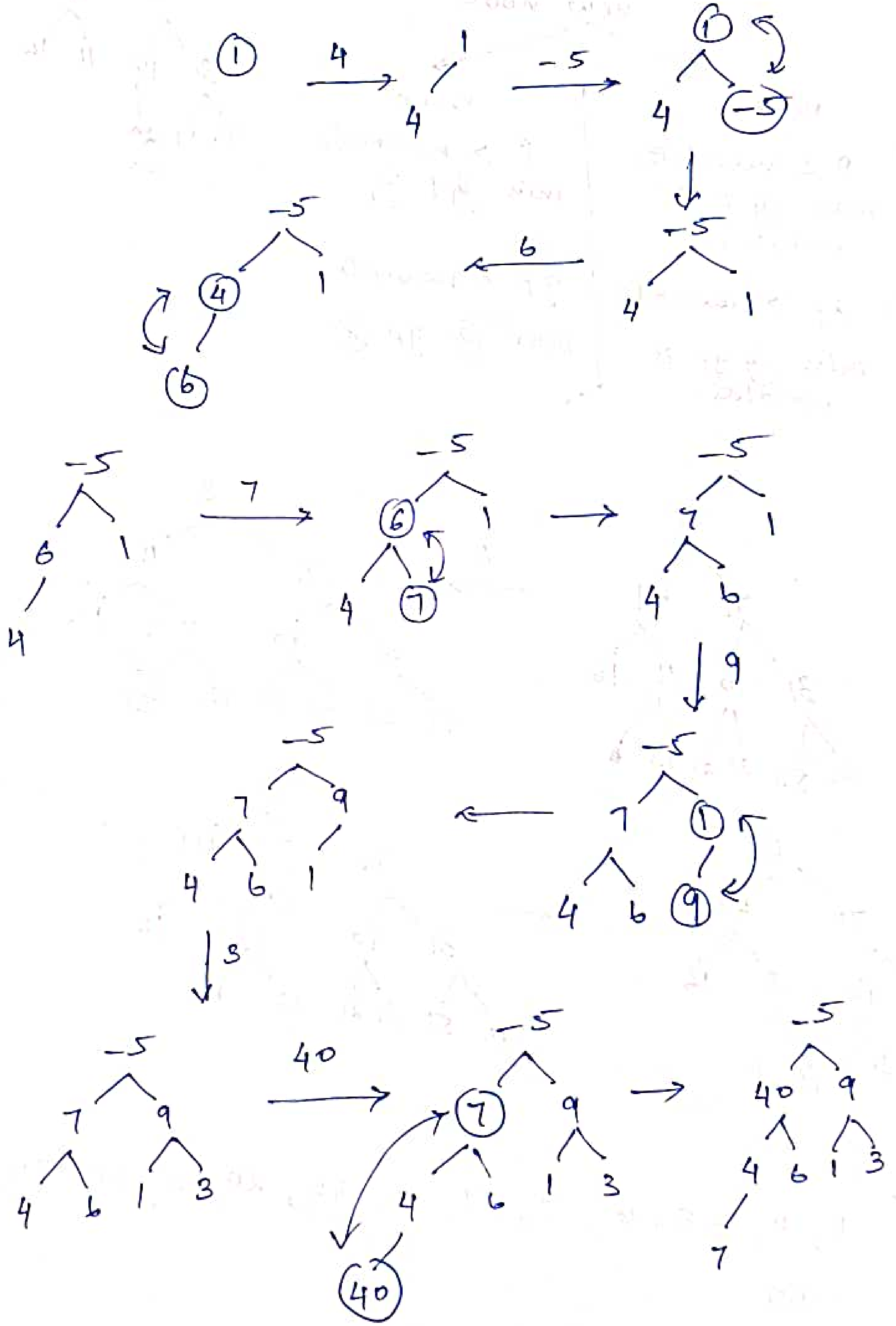


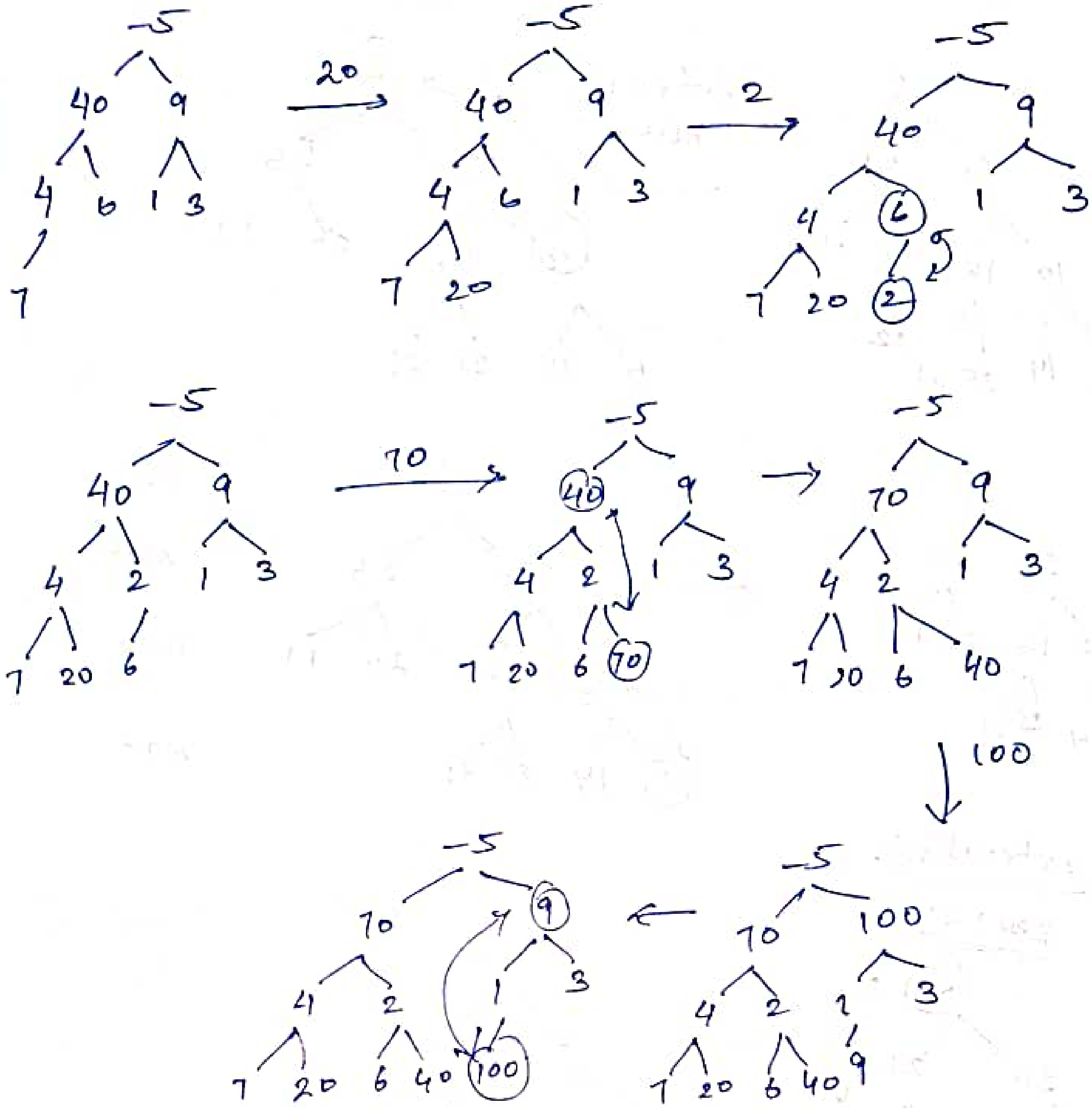


Q) 1, 4, -5, 6, 7, 9, 3, 40, 20, 2, ~~#~~, 70, 100

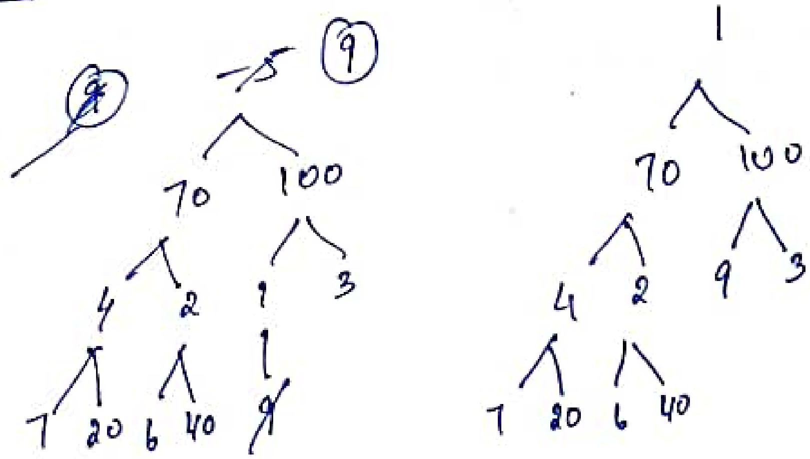
Construct min max heap

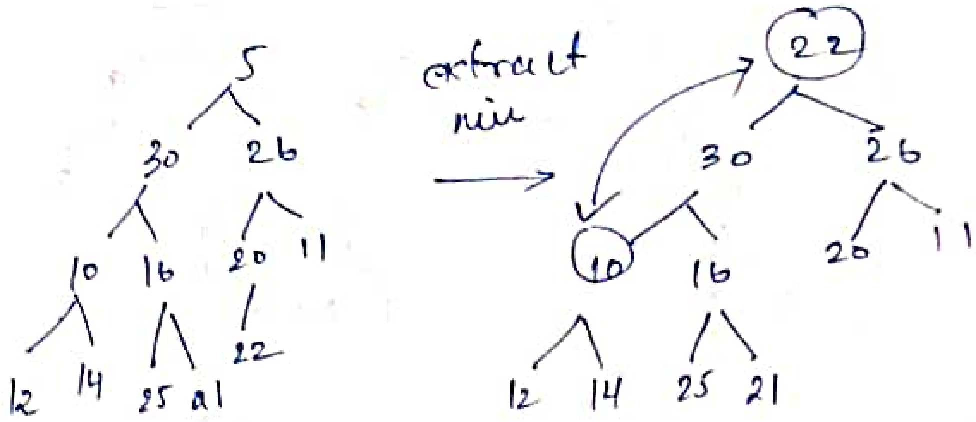
Q) 1, 4, -5, 6, 7, 9, 3, 40, 20, 2, 70, 100

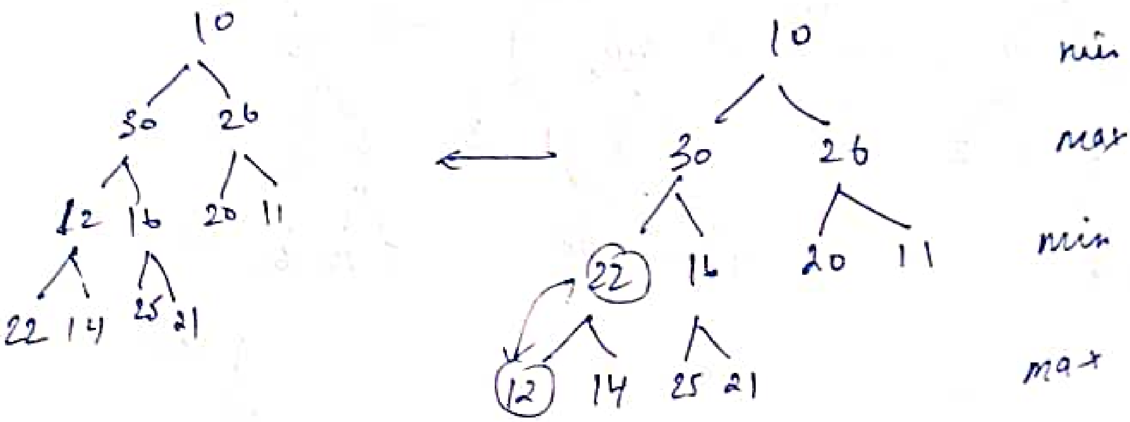




extract min   last node to root

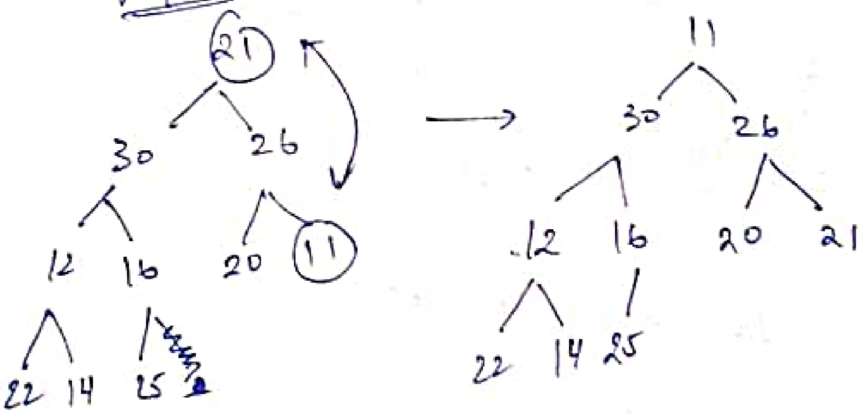






extract min

key : 21

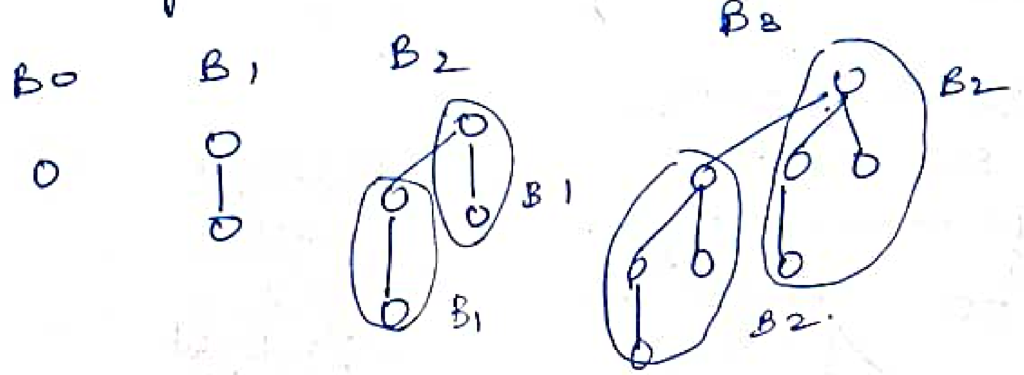


Binomial Heap

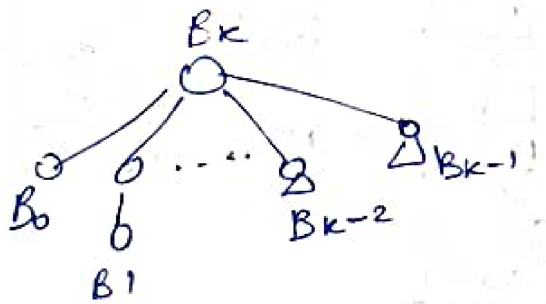
Collection of Binomial trees

1) BT_0 = 1 node
2) $BT_k = 2\,BT_{k-1}$

Root of 1 BT is linked as left most child of root of another BT



1) Consider BT_k, we have 2^k nodes.
2) Height → k.
3) ${}^kC_i$ nodes at depth i $\quad i = 0, 1 \ldots k$.
4) No. of nodes at i in B_k.
= (No. of nodes at i in B_{k-1})
+
(No. of nodes at $i-1$ in B_{k-1})
5) Degree of root → greater than degree of any node in BT.



Binomial Heap

→ min
→ max

Binomial Tree

(i) Min Heap.

(ii) atmost 1 BT of any order.

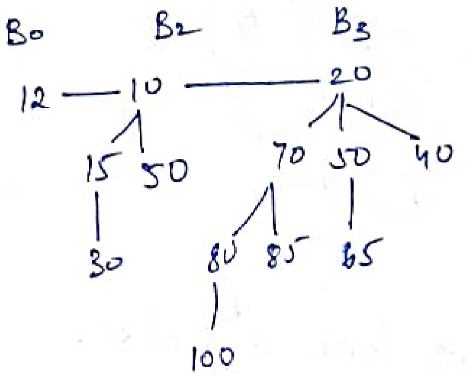
13

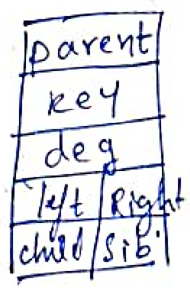
1101

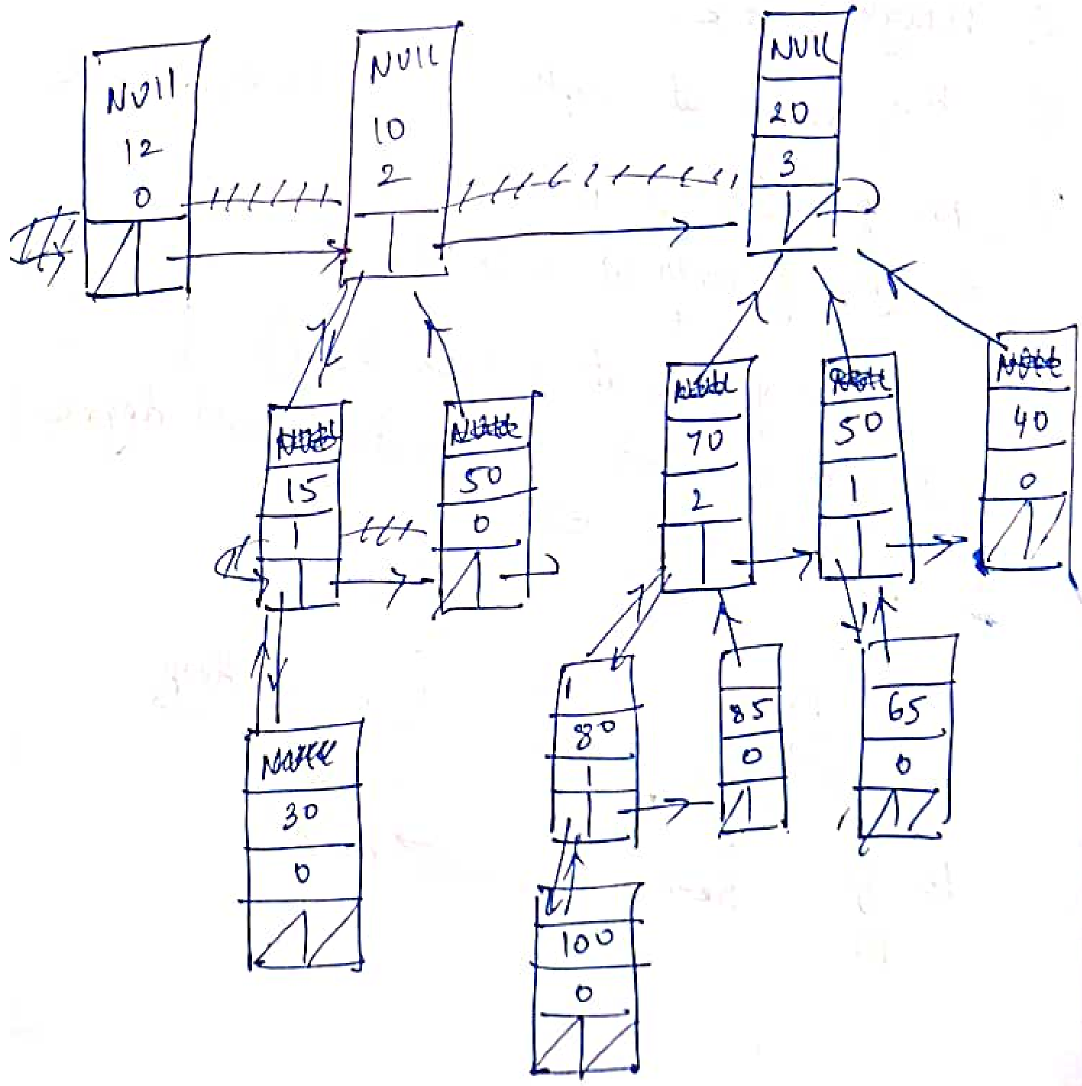
3 BT (B_0 B_2 B_3)

8 1000 1BT of order : 3.

Root list B_0 — B_2 — B_3.







Operations	Binomial Heap	Binary heap
1) Finding min	$\lfloor \log_2 N \rfloor$	$O(1)$
2) Union (H_1, H_2)	$O(\log N)$	$O(N)$

Merge:

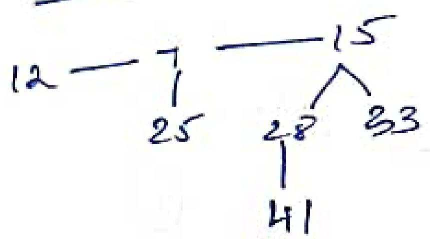
case 1: ptr next degree not same proceed.

case 2: ptr, next, next → next are same proceed.

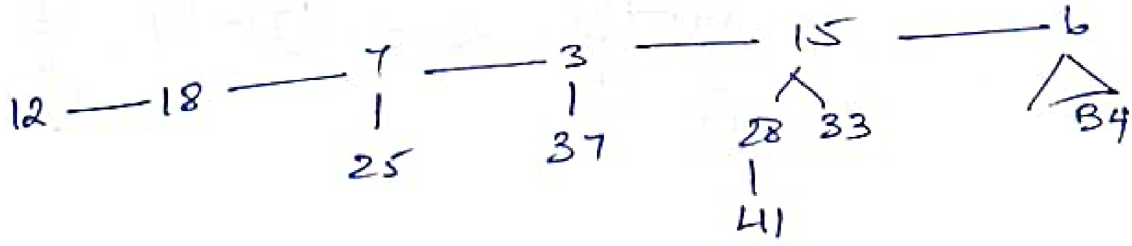
case 3: key of $\frac{\text{ptr}}{\text{root}}$ < key of $\frac{\text{next}}{\text{leftmost child of ptr}}$.

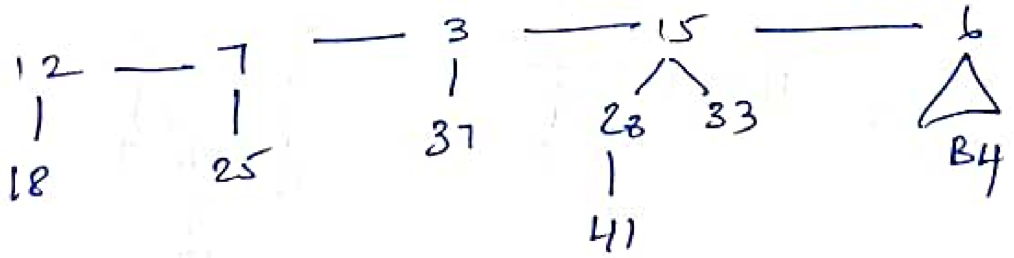
case 4: key of $\frac{\text{ptr}}{\text{left child}}$ > key of $\frac{\text{next}}{\text{root}}$.

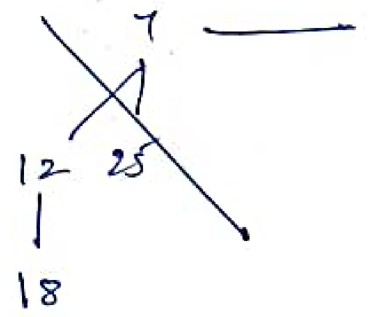
H1:

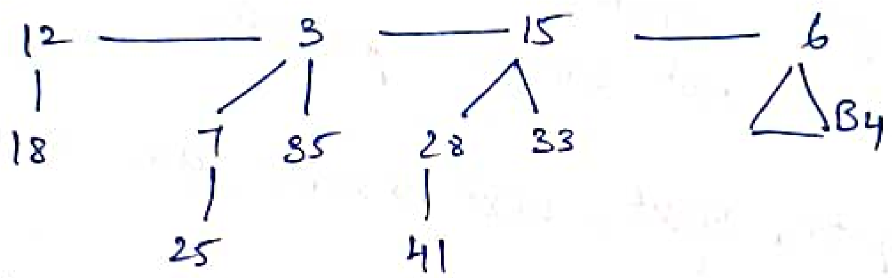


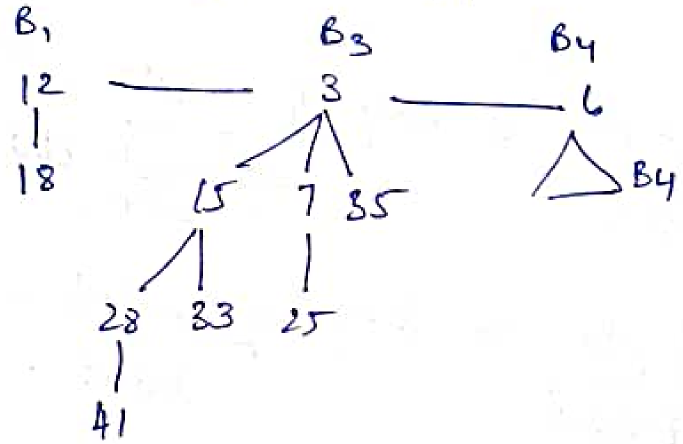
H2:







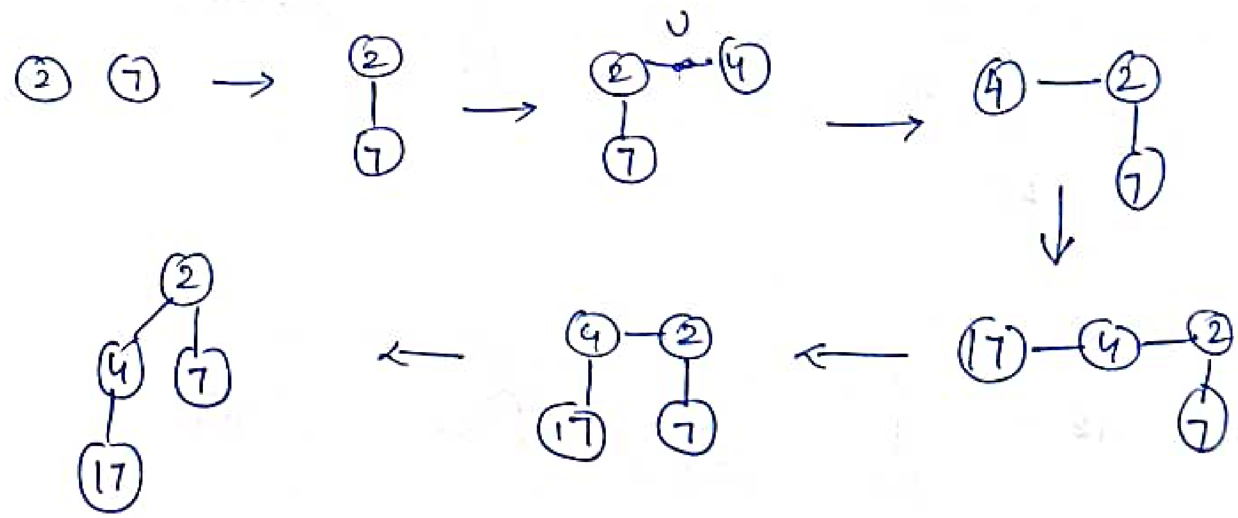


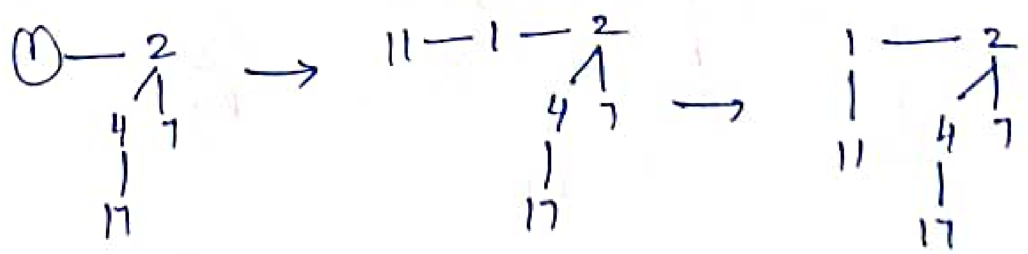


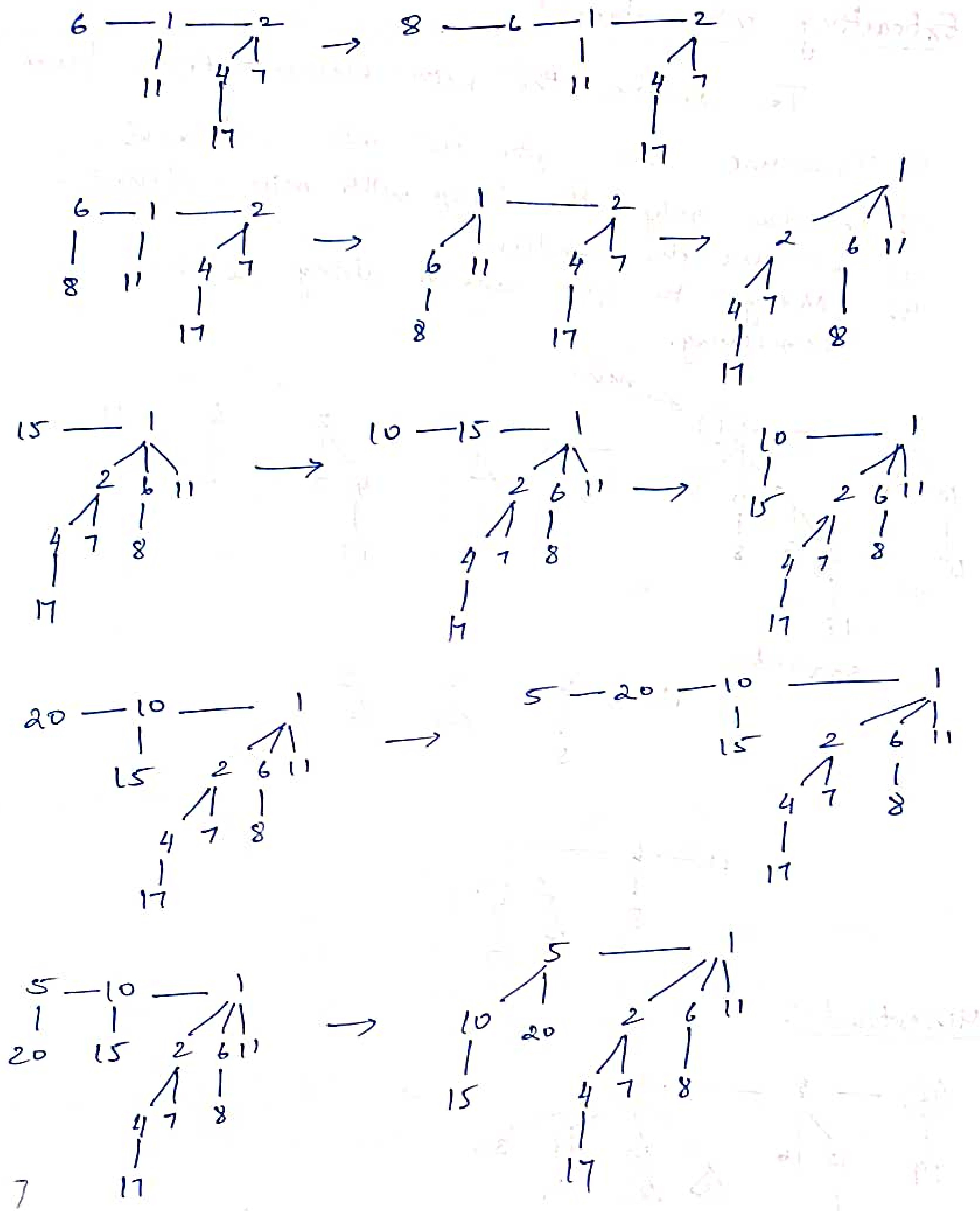
11010

16 + 8 + 2
= 26 nodes

Insertion: 7, 2, 4, 17, 1, 11, 6, 8, 15, 10, 20, 5





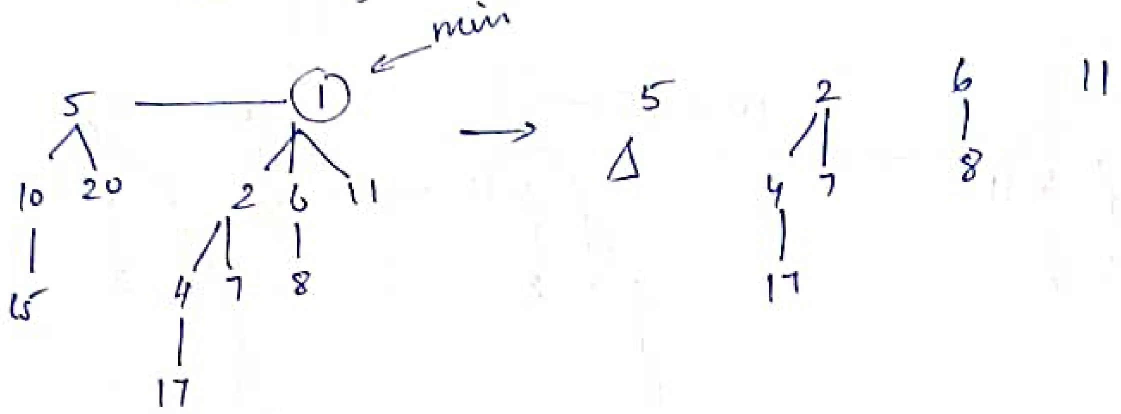


H/W: 13, 15, 7, 8, 11, 9, 5, 7, 3, 7

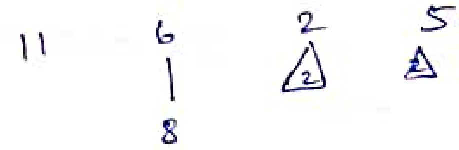
Extracting min element:

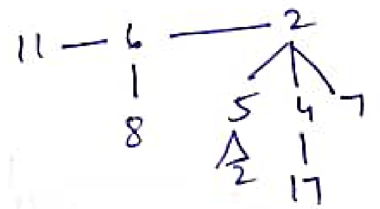
To delete the min element from heap.

1) Traverse the ptr to min element.
2) Consider only the heap with min. element.
3) reverse the children
4) Merge to get union along with remaining.

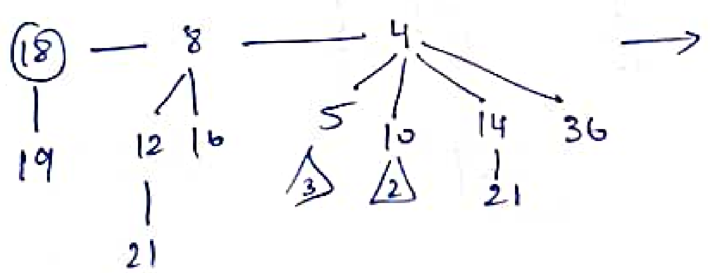


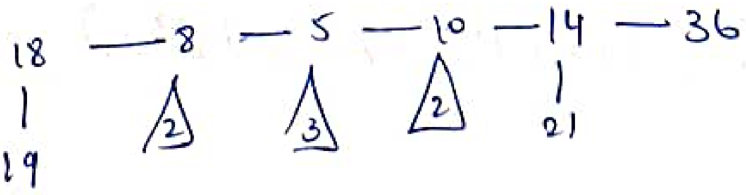
sorted

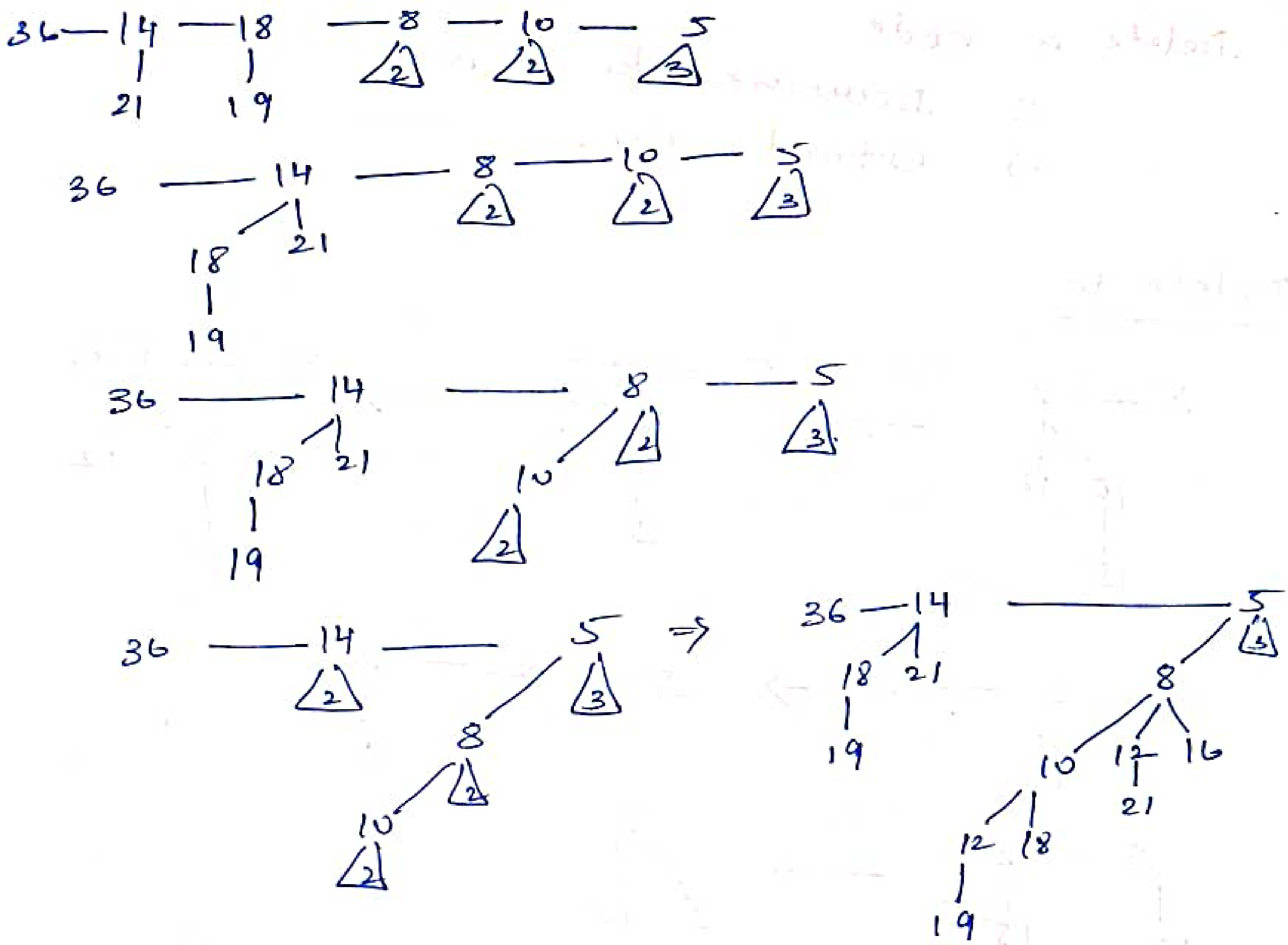




Minextract:



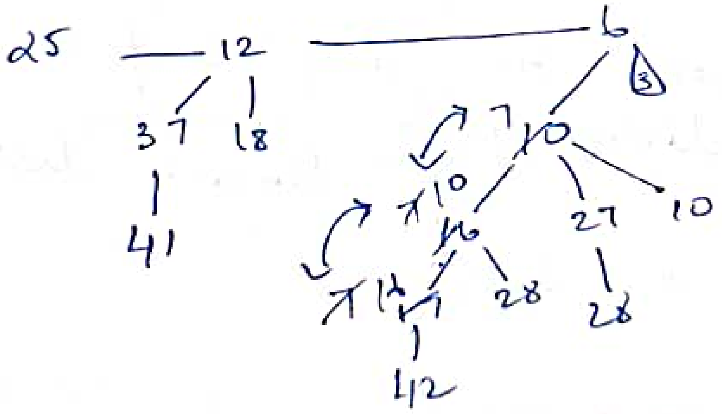




Decrease a key value: heap – H
original val – x
new val – k.

1) check if $k < x$.
2) check if min heap property is satisfied and swap where necessary.

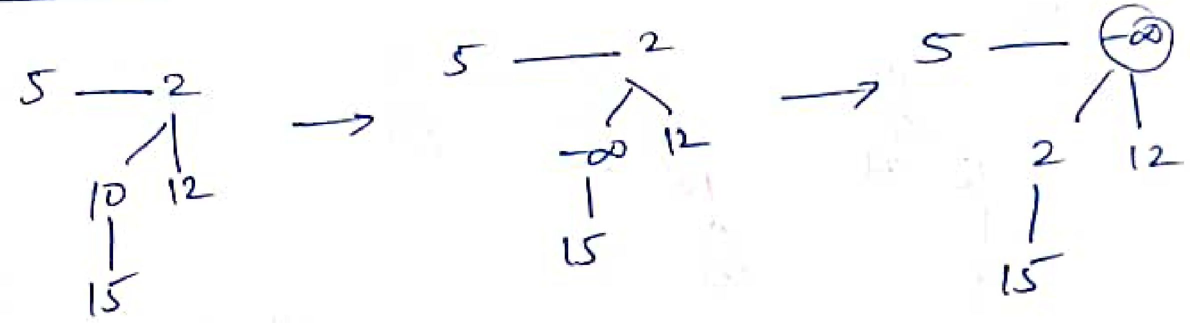
Increase key value
swap and verify downwards.

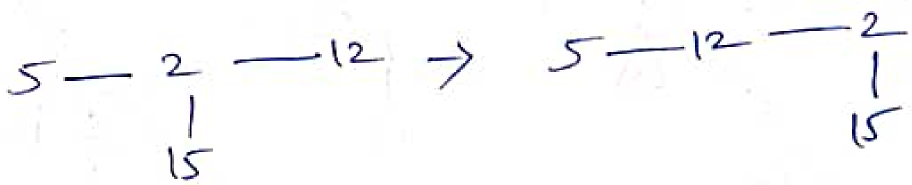


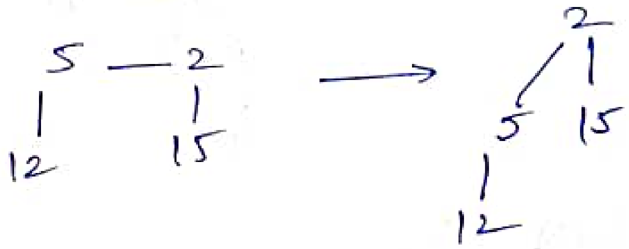
Delete a node

1) Decrease to $-\infty$.
2) extract min.

Delete 10

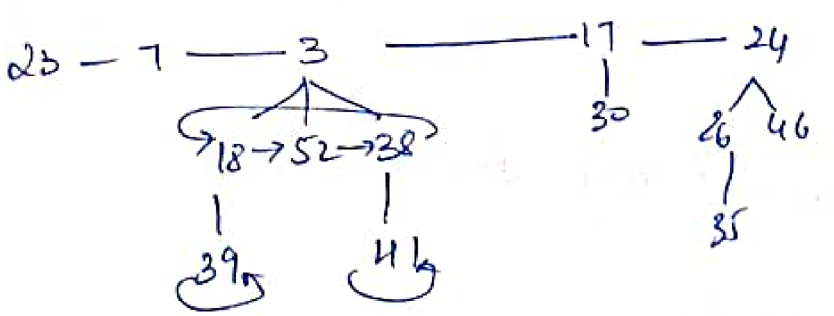






Fibonacci Heap:

Can be any shape
subtrees are unordered.



Features of node

P(x)
child(x)
deg(x)
min [H]
n [H]
mark [x]
to delete.

child(x) → can point to any 1 of children
[children are linked with circular doubly linked list]

Root list is also a CDLL

Fibonacci Heap

tree
any structure.

P[x]
child [x]
deg [x] → number of children
min [H] →
n [H] → number of node in heap.
mark [x] → T/F
for deletion operation.

Root list

Binomial heap
pointer to first element.

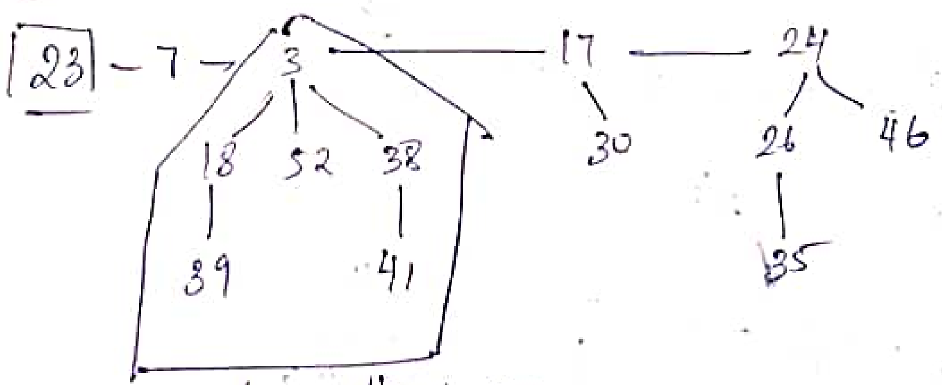
Fibonacci

0, 1, 1, 2, 3, 5, 8.
0 1 2 3 4 5 6.

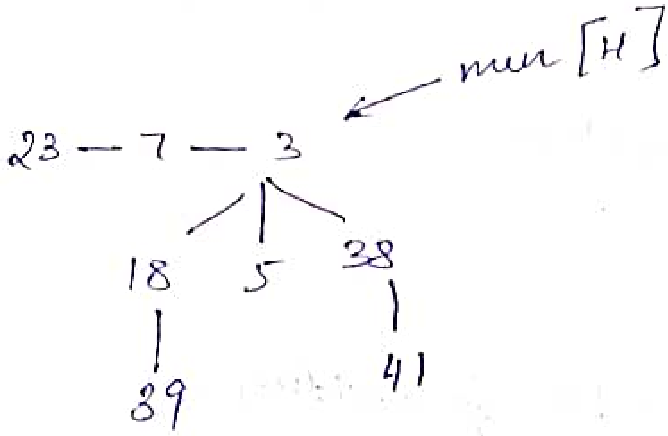
(*) Tree of order n has atleast F_{n+2} number of nodes.

order ≠ no. of node

order = 0 order = 3



has atleast 5 nodes.



By default new node inserted at left side of min element.

~~12, 15, 3,~~

12, 50, 3, 15, 20

12

50 — 12

50 — 3 — 12

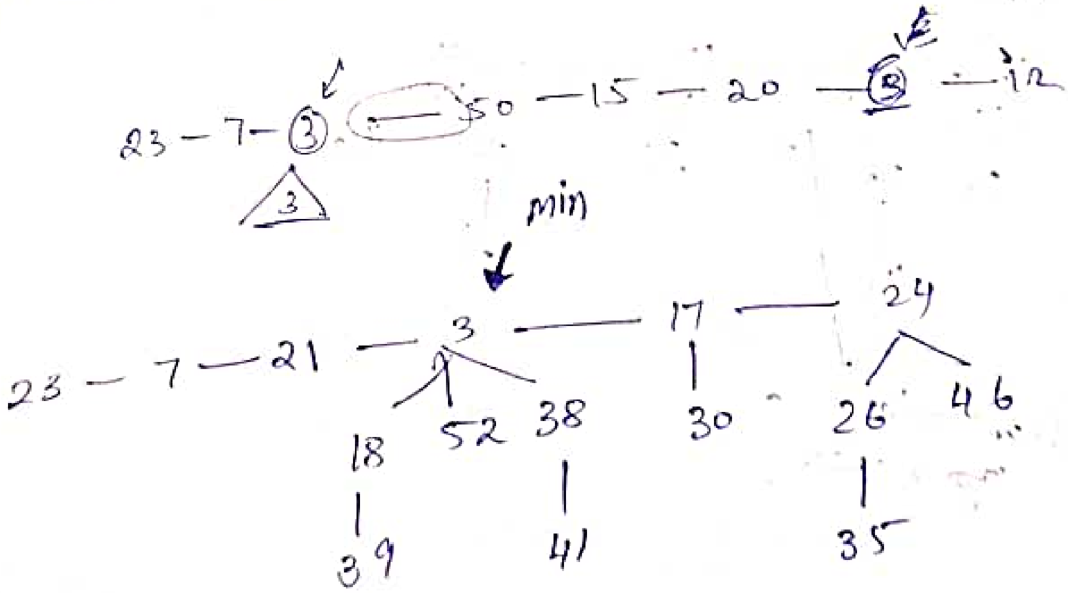
50 — 15 — 3 — 12

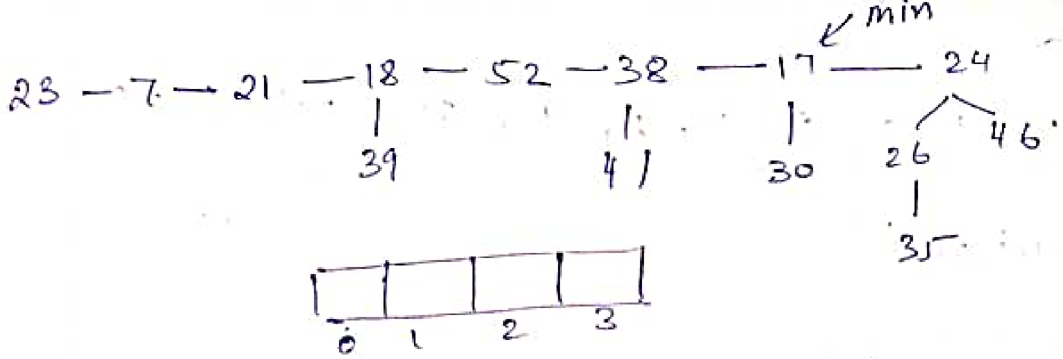
50 — 15 — 20 — 3 — 12

lazy operation

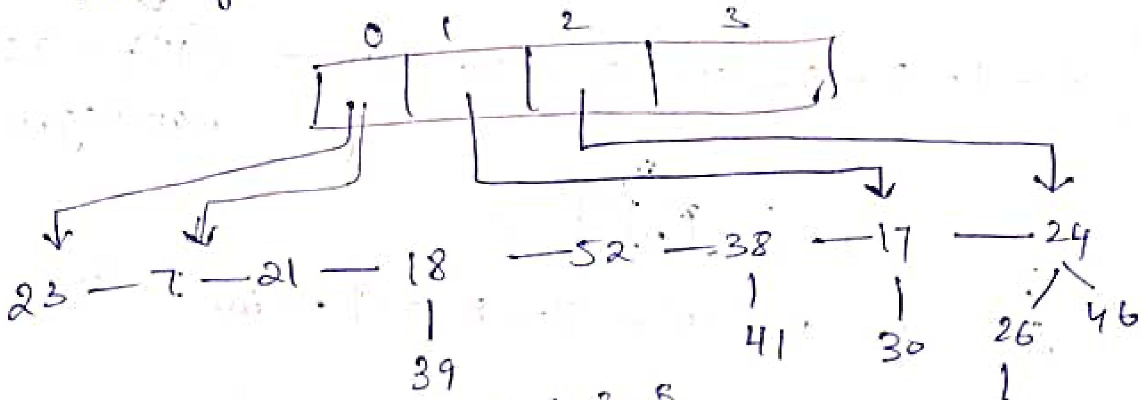
posponding the operation

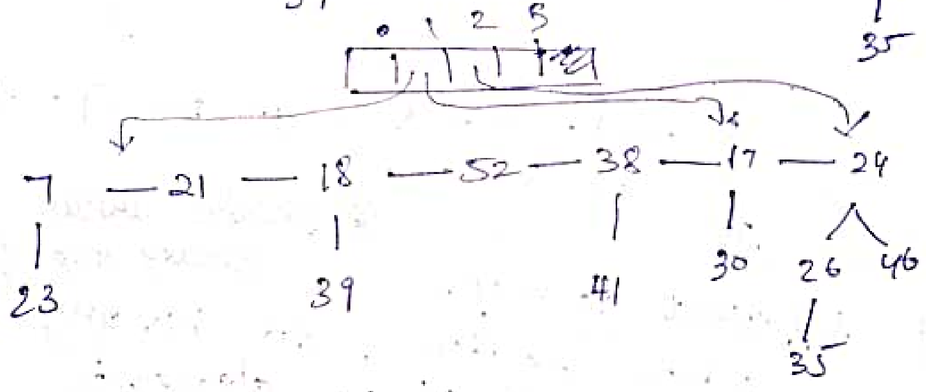
2) Union of 2 F.H.



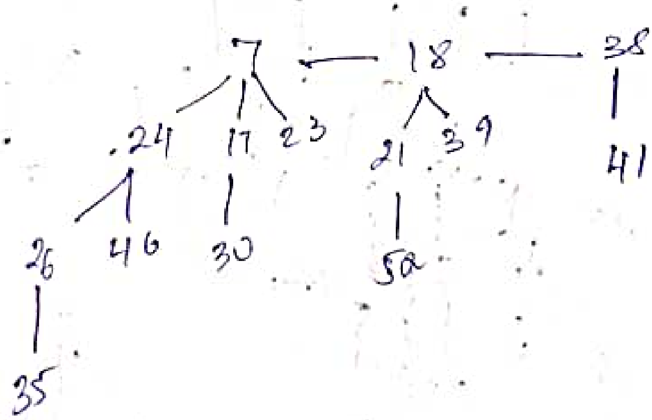


min pointer → to next node.





- create array size 1 + max(degree)
- create mapping
- then merge.



degree - Number of children.

Order - height of Tree.

5, 1, 3, 5, 7, 8, 9, 20, -15, 25

extract_min

5 — 1 — 3 — 5 — 7 — 8 — 9 — 20 — -15 — 25

max degree = 0

5 — 1 — 3 — 5 — 7 — 8 — 9 — 20 — 25

5 — 1 — 3 — 5 — 7 — 8 — 9 — 20
|
25

* create array of size. Binary rep of nodes.

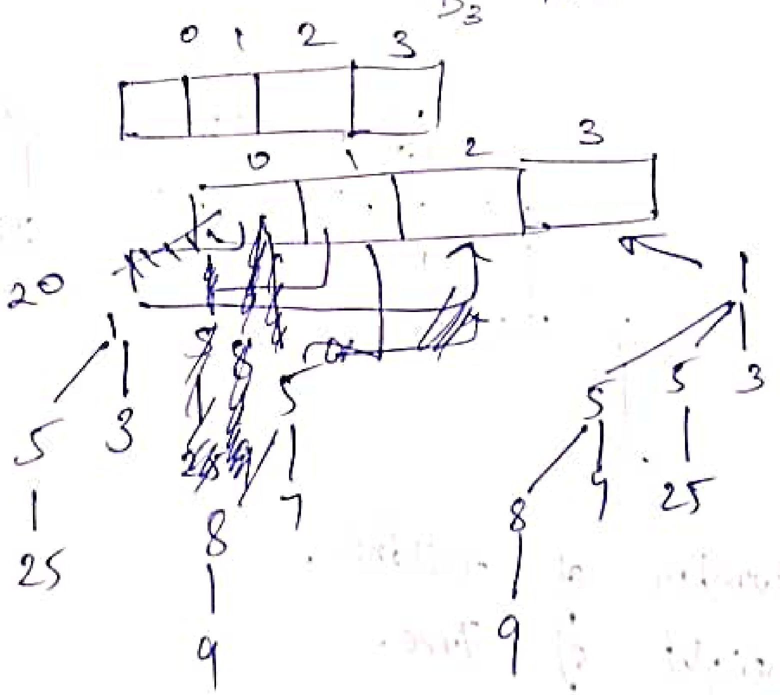
Not sufficient to map.

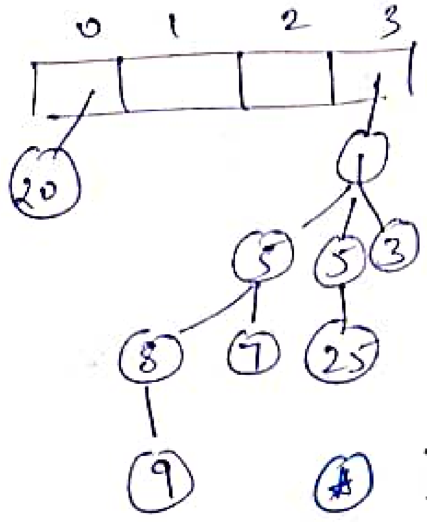
after deletion, do binary representation of remaining elements.

9 elements :

3	2	1	0
1	0	0	1

B_3 B 0

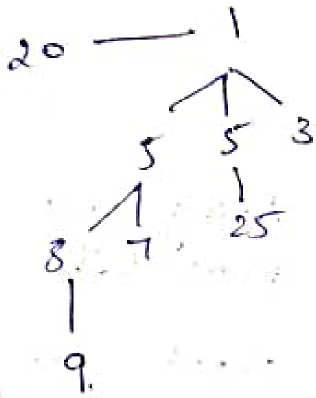




Binomial heap
Fibonacci } same.

(A) I I I I I E

(B). I I I E I I



Both heaps are different.

1/2/23

Decreasing key value.

Case 1: Not violated.

Case 2: Violated if p[x] is unmarked.

→ cutoff x and p[x]

→ move x to the root list

→ mark p[x]

Case 3: violated and p[x] marked.

→ cutoff x and p[x].

→ move x to root list

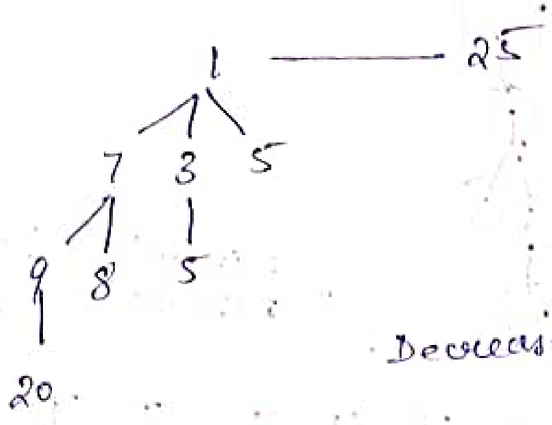
→ cut off p[x] of p[p[x]].

→ p[x] to root list

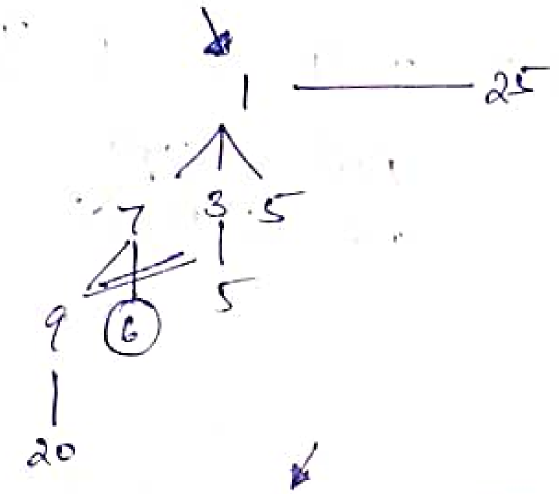
→ if p[p[x]] — unmarked.

mark p[p[x]]

else repeat p[p[x]] → x. (case 3)

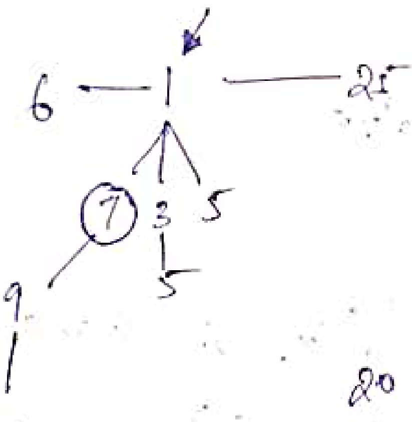


Decrease $8 \Rightarrow 6$.



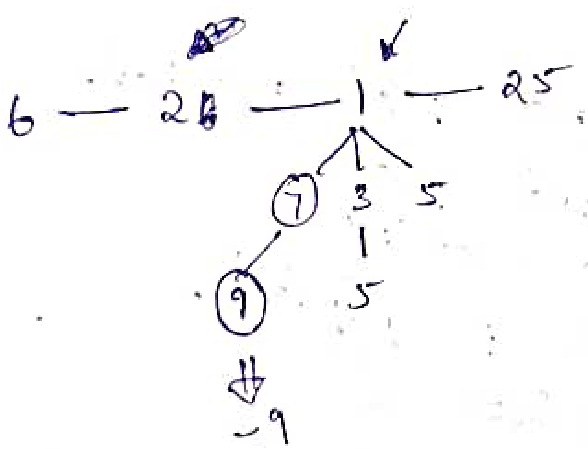
Bring any node to root list

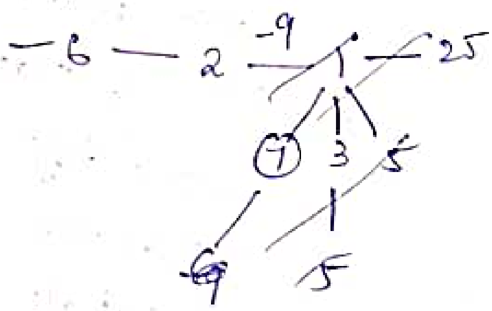
check for the min pointer.

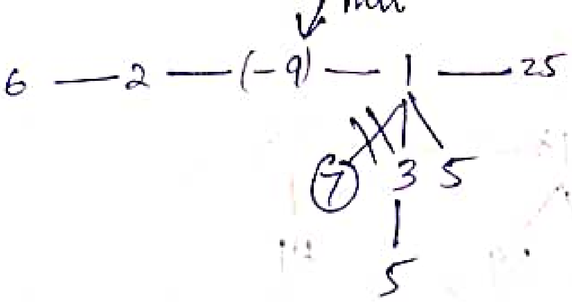


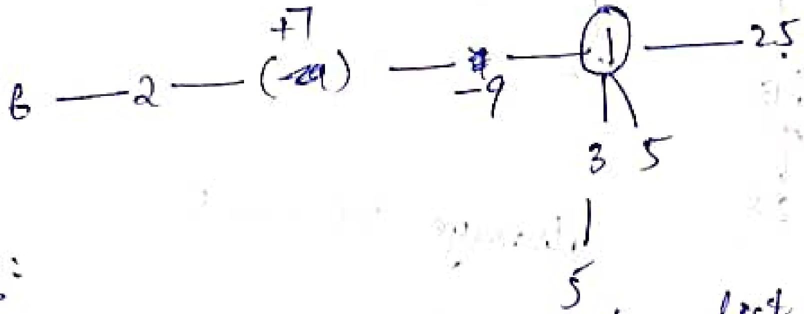
$20 \Rightarrow 2$.

$20 \Rightarrow 2$.



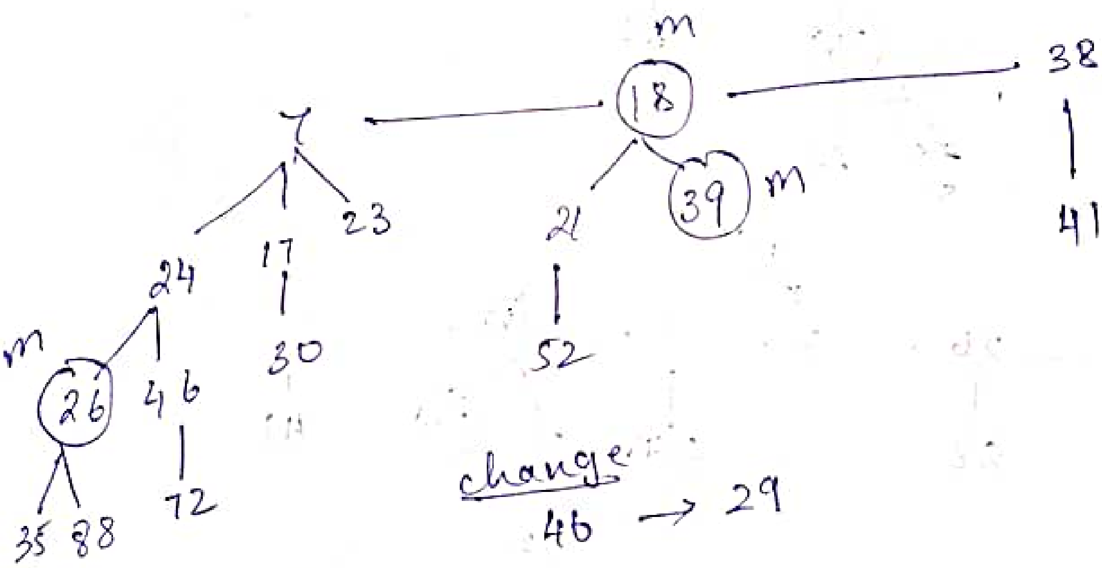




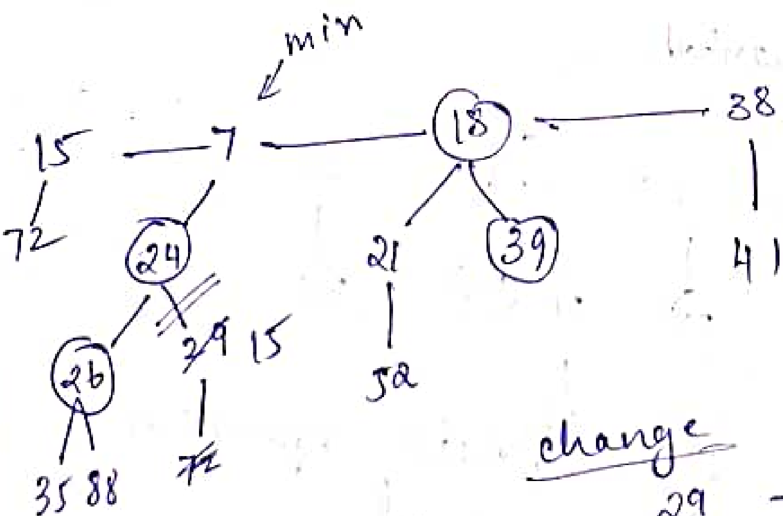


Case 3 :

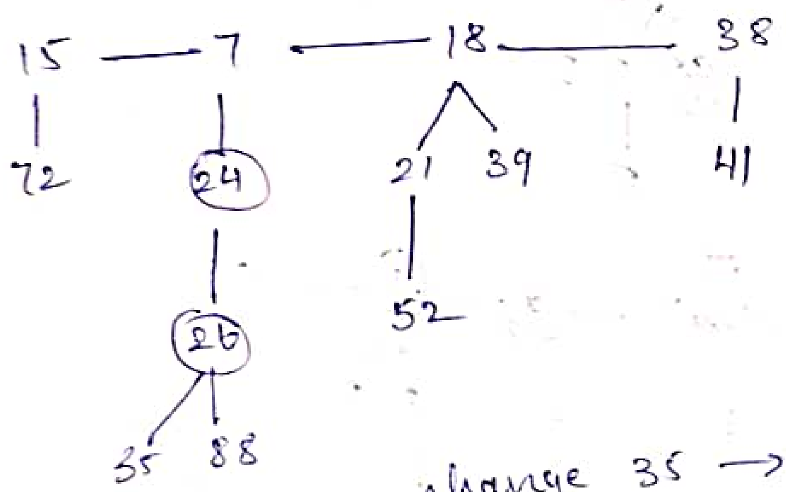
while moving p[p x] to root list,
if marked, unmark it.



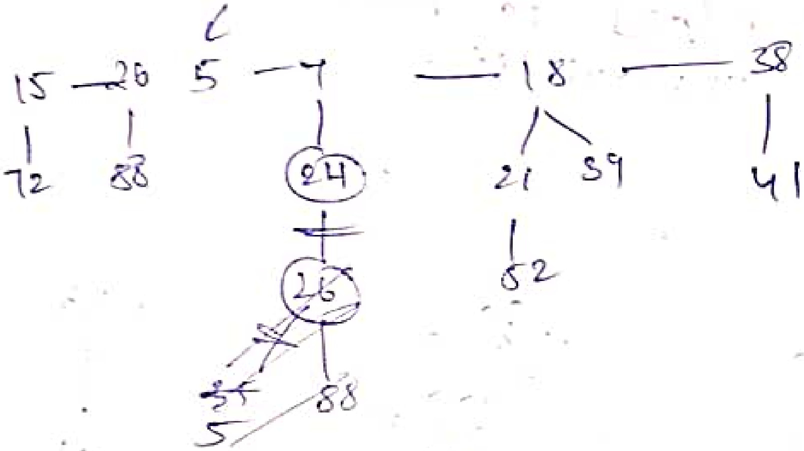
change
46 → 29

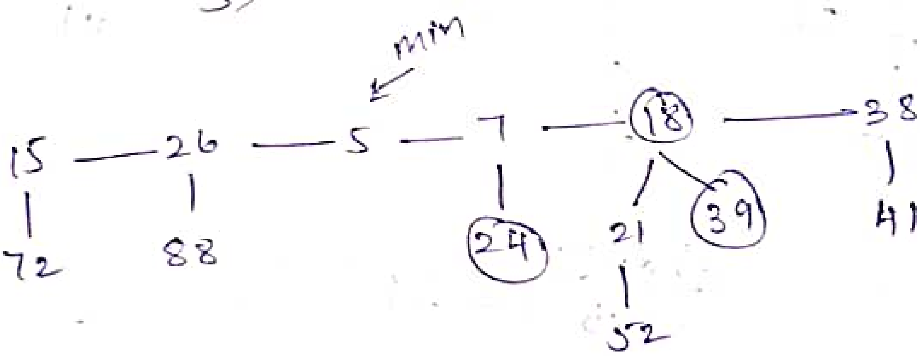


change
29 → 15

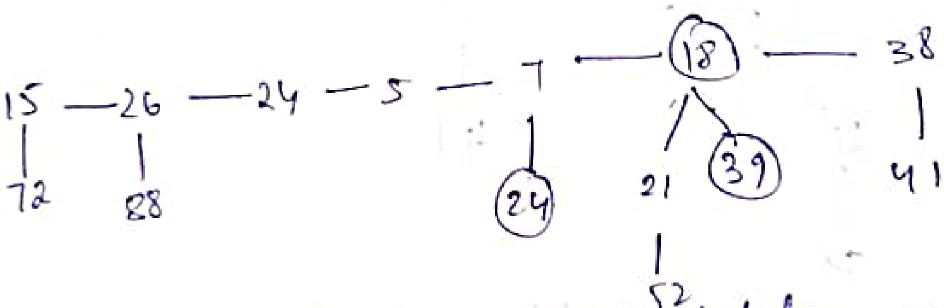


change 35 → 5





24 → marked



frequent extract min and delete operation fibonacci ~~non~~ heap is not preferred.

Amortized

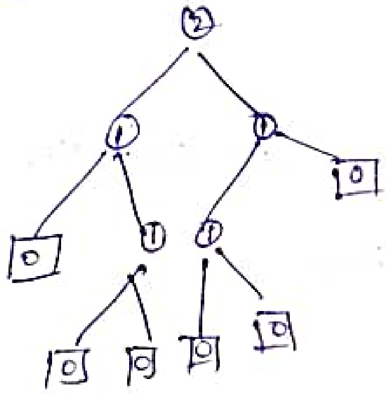
Most of operation : O(1)

- Leftist Heap → priority queue

double ended priority queue → binomial, fibonacci heap.

Leftist heap → Height biased LT → min / max
→ Wt biased LT → min / max.

extended binary tree.



$\delta(x)$

defined for all nodes.

Length of the shortest path from x to the extended node in its subtree.

Extended node $s(x) = 0$

Internal node $1 + \min\{s[L], s[R]\}$

(*) Extended BT is Height biased leftist Heap.

Iff at every node $\delta(L) \geq S(R)$

3/2/23

Operation

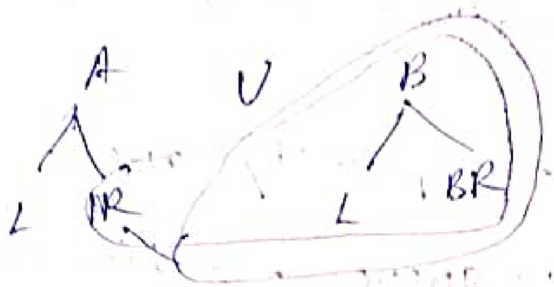
Insertion

Deletion

(*) Merging.

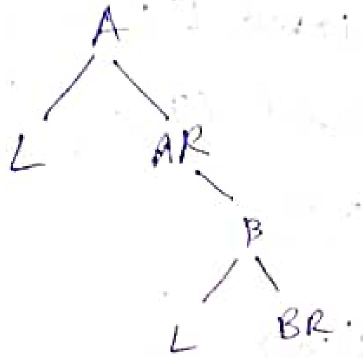
extract min/max

Max height biased Leftist tree



Assume $A > B$:

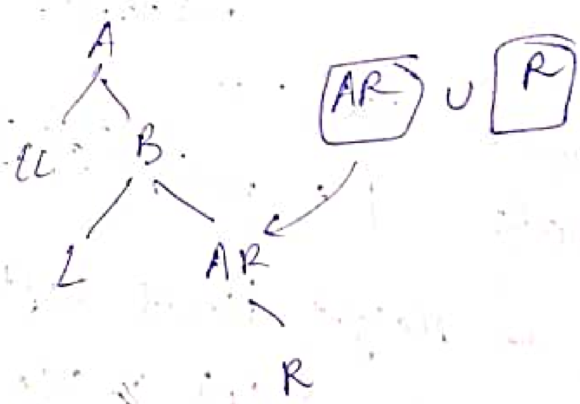
If $R > B$



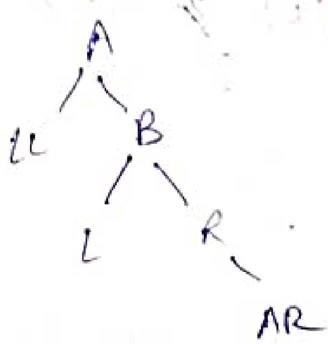
If $S(L) \leq S(R)$ condition not satisfied.
swap left and right subtree.

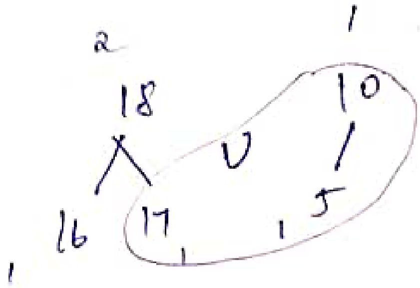
If $B > AR$

$AR > R \Rightarrow$

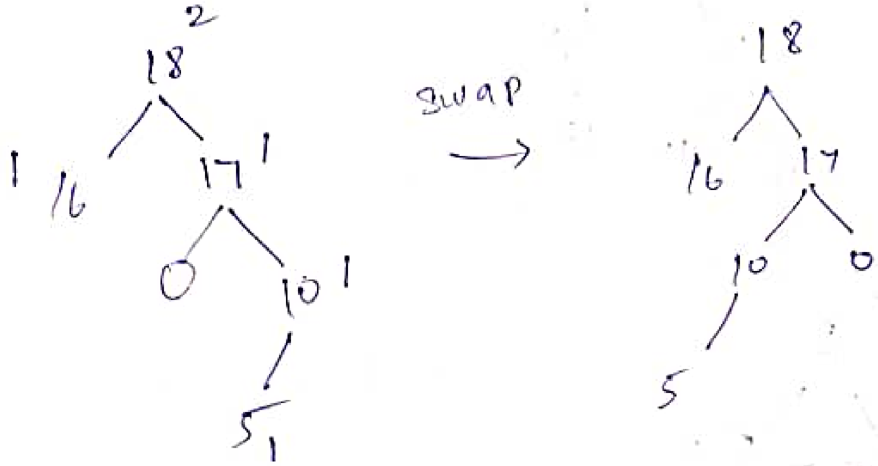


$R > AR$

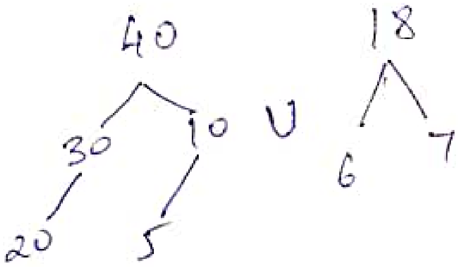


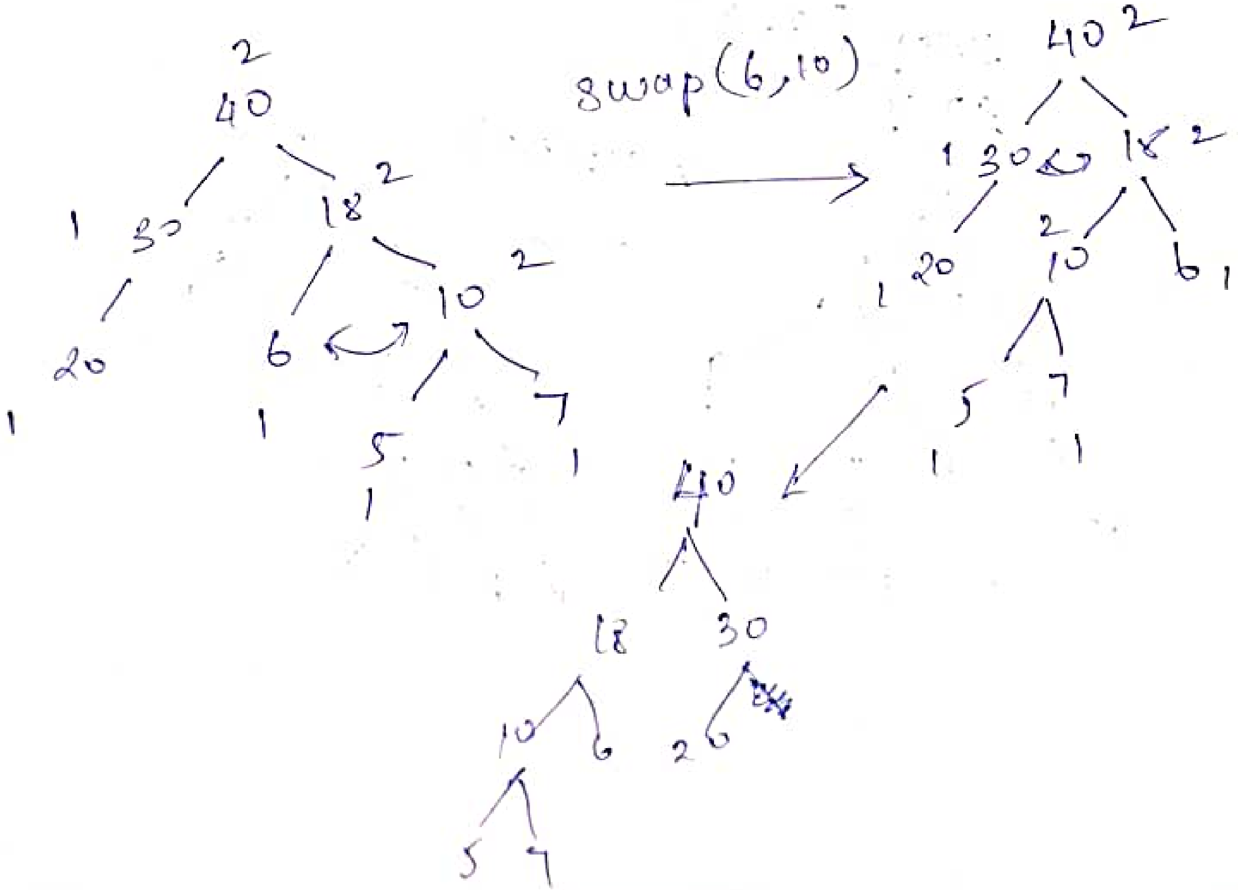


go to right subtree

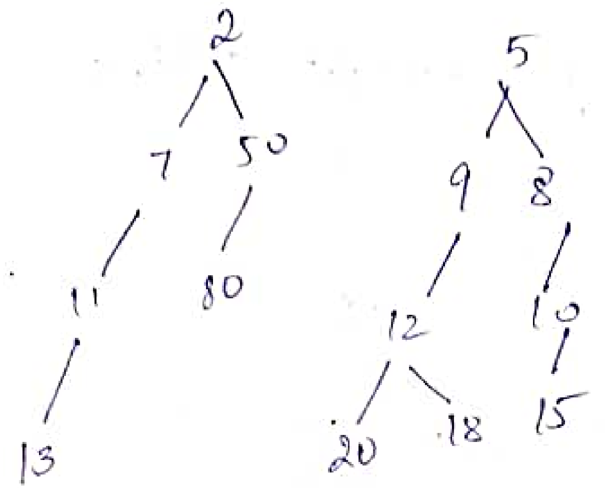


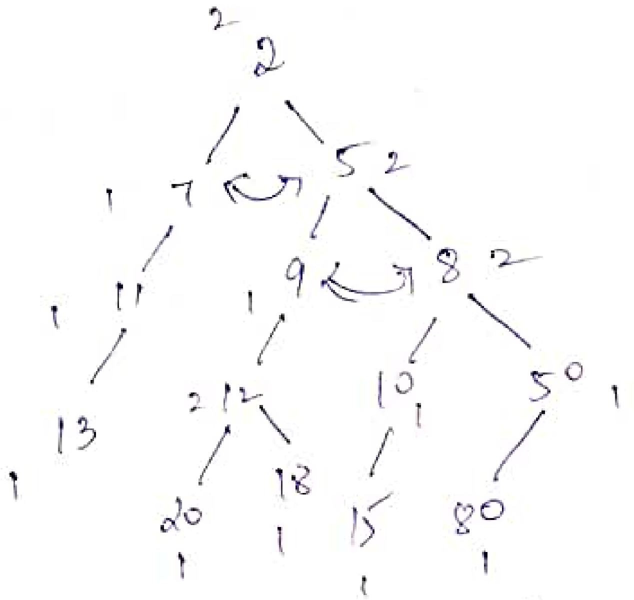
Merge:





Min Height Biased Leftist tree

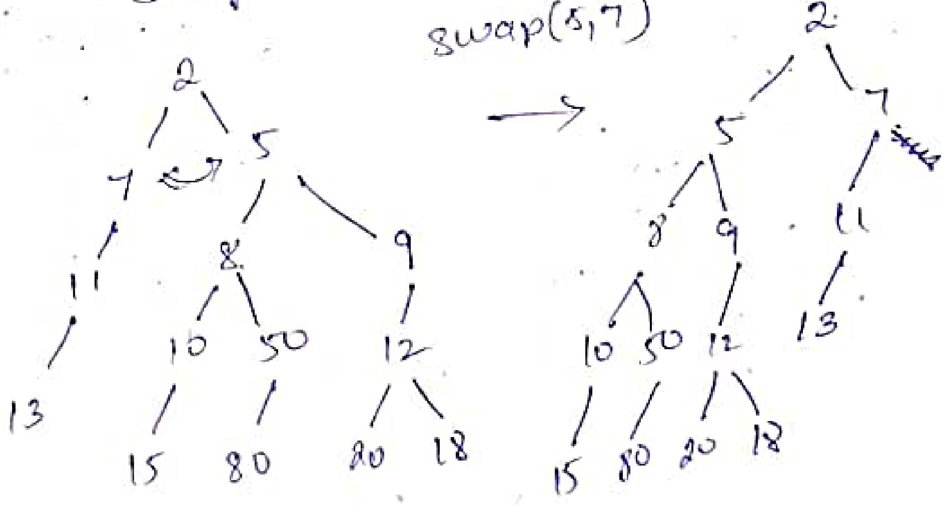




swap (9, 8)

~~swap (7, 5)~~

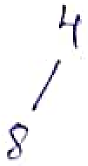
swap(5,7)



{4, 8, 10, 9, 1, 3, 5, 6, 11}

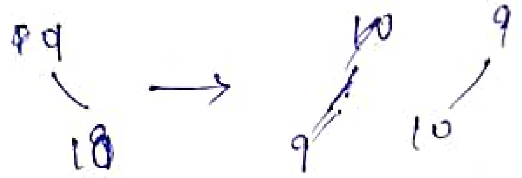
min HBLT:

8, 4

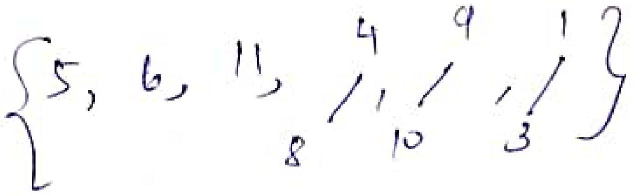


{10, 9, 1, 3, 5, 6, 11}

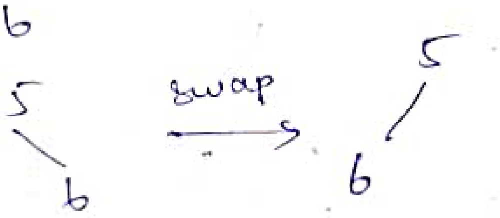
10 9

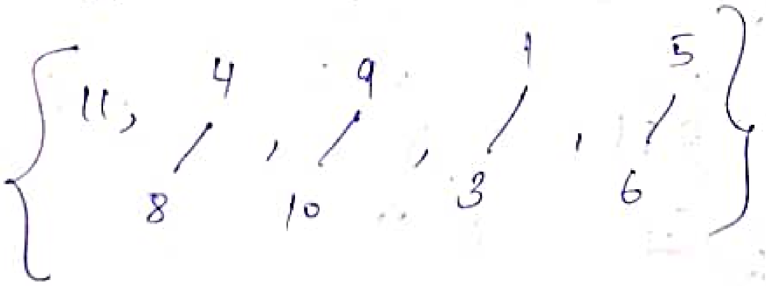


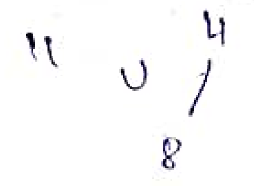
1, 3

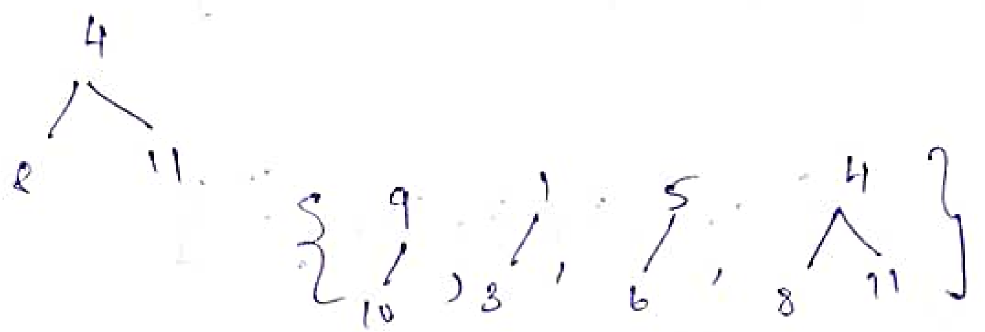


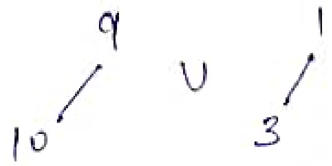
5, 6

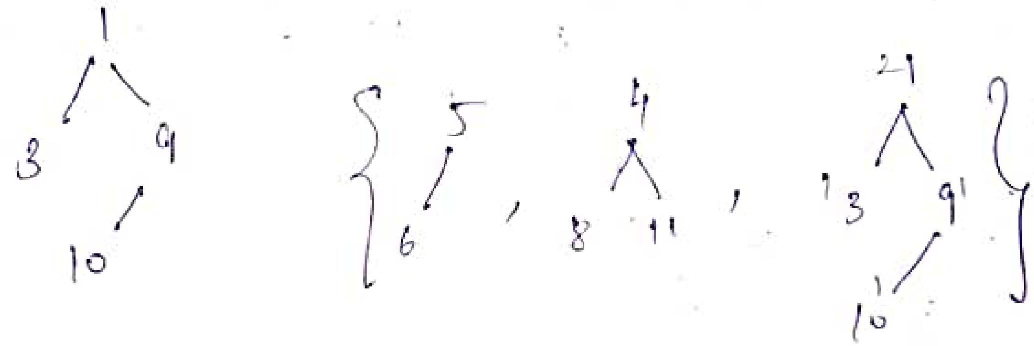


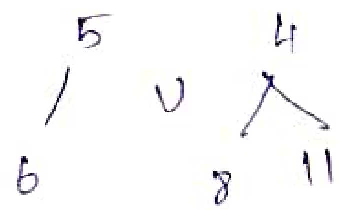


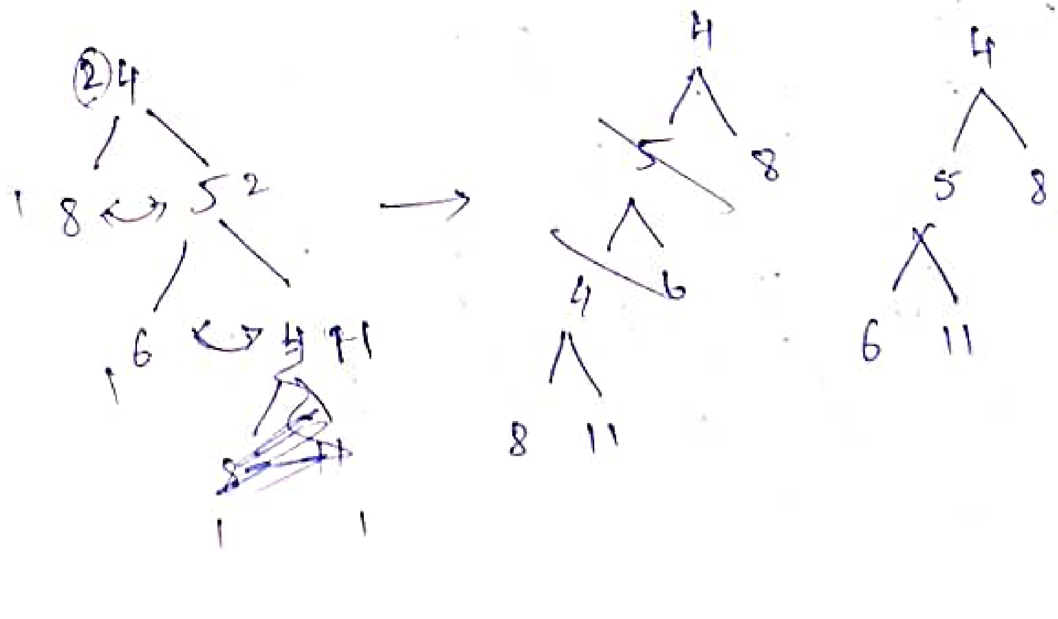


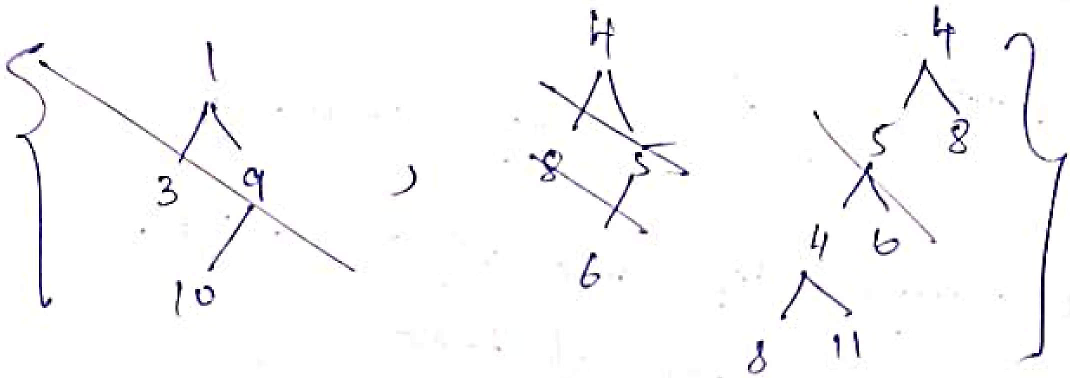


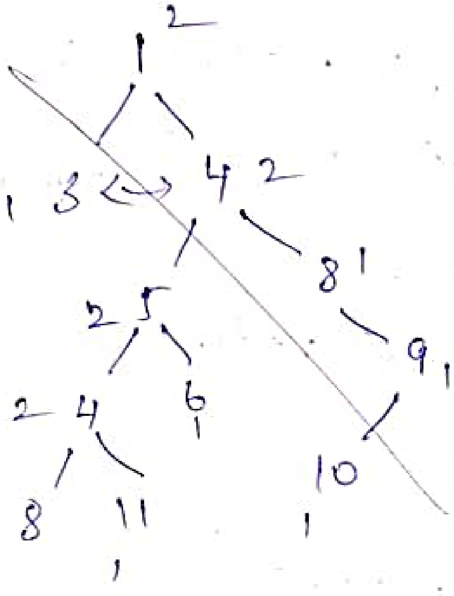


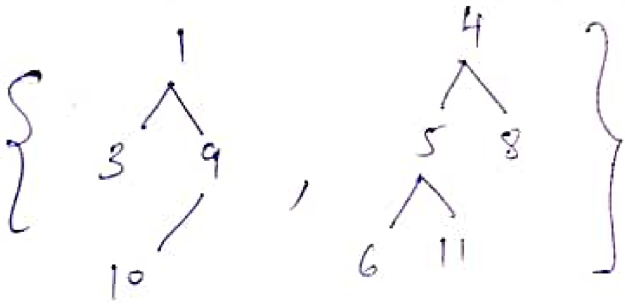


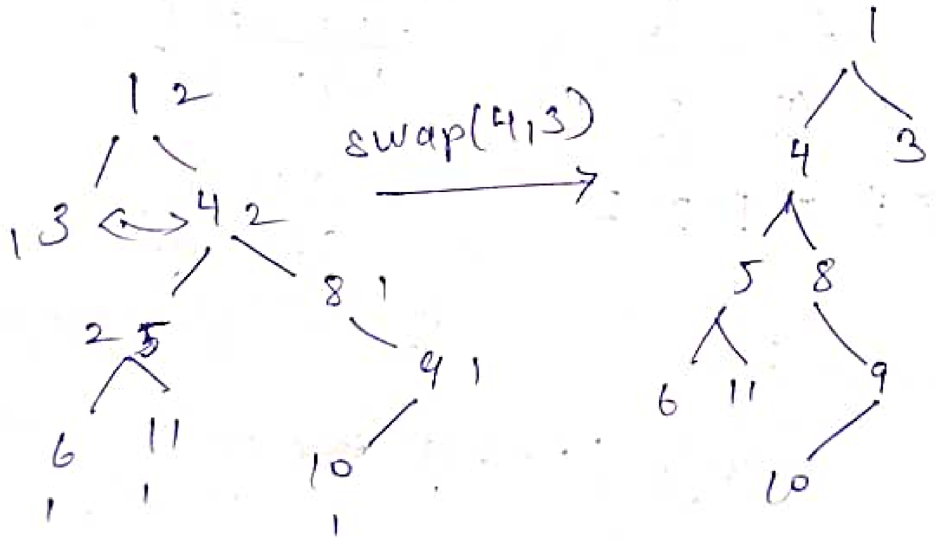











swap(4,3)

delete node

assume pointer to node.

* merge left and right child.

traverse top of tree.

Conditions:

→ If $S(l) < S(r)$: swap.

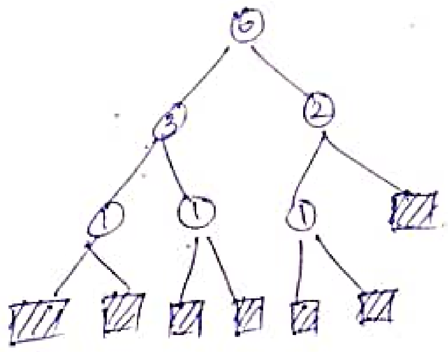
→ If $S(x)$ no change, continue. stop process

Weight Biased LT :-

No. of Internal nodes in subtree root as x.

$$S(x) = 1 + S(L) + S(R)$$

Condition: $S(L) \geq S(R)$



Only one traversal req. ?

But. S(L) S(R) to be stored

no need of parent pointer

* Application:

Singly / Doubly ~~ended~~ ended priority queue.

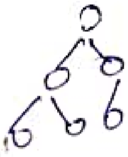
In HBLT → parent pointer required.

WBLT → S(L) S(R) to be stored.

6/2/23

Deaps

complete binary tree.
CBT
+
Correspondence property

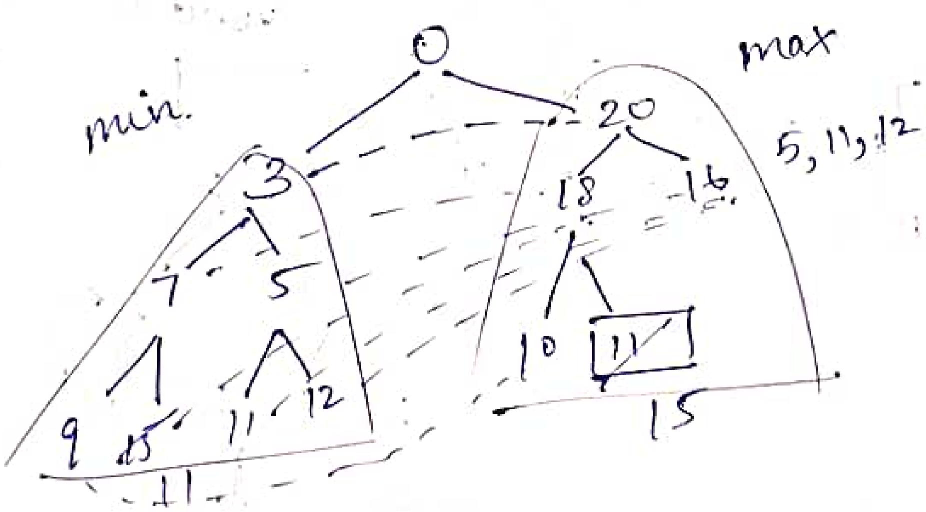


(i) x y - for each node in left subtree, there should be corresponding y node in right subtree

(ii) left (min)
right (max)
$x \le y$

There is no y for x - parent node is correspondent node of x considered.

Double ended priority queue.
Root - empty



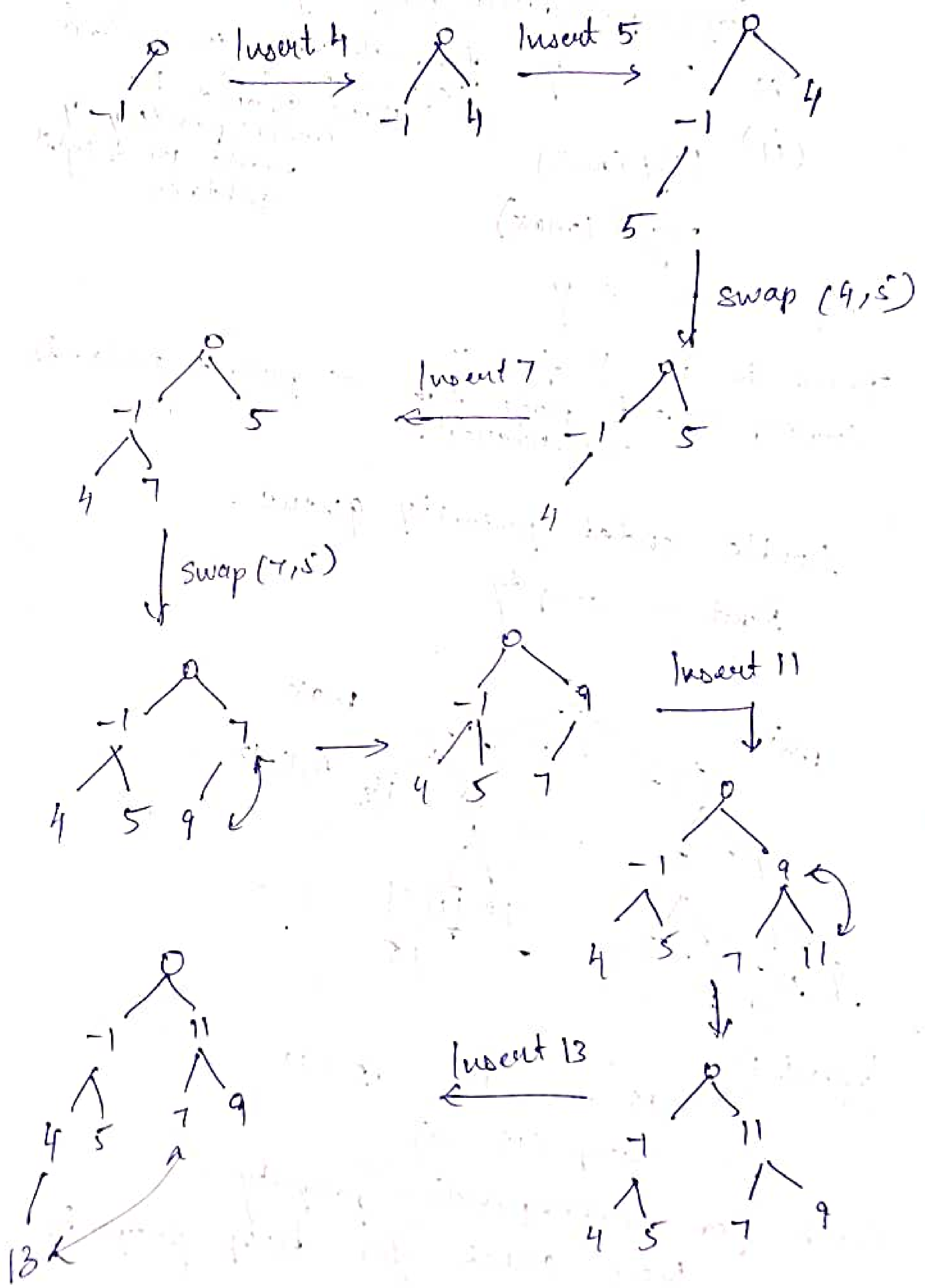
Insert 11

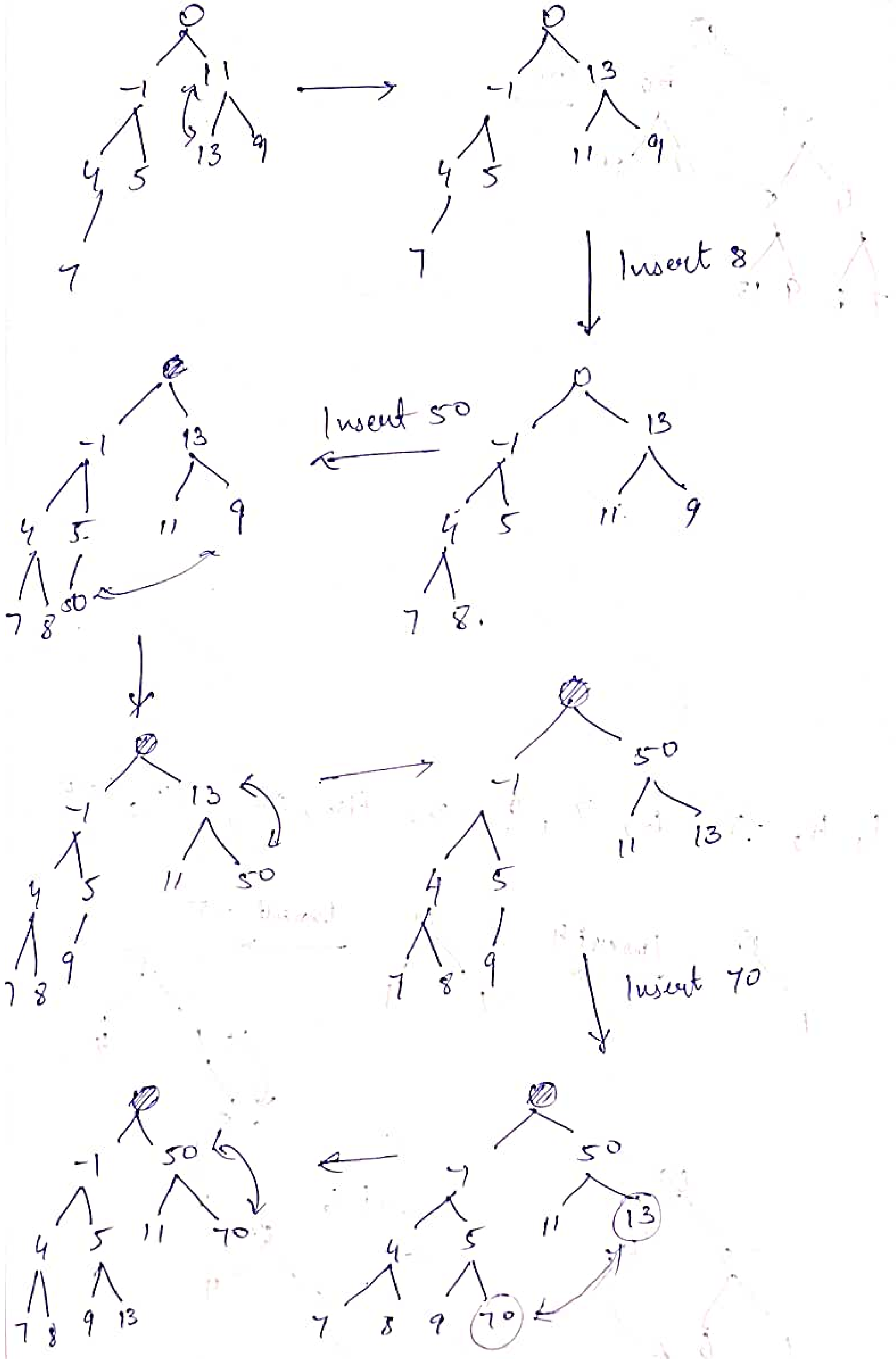
15 ⟶ 11. 15 > 11
swap (15, 11)

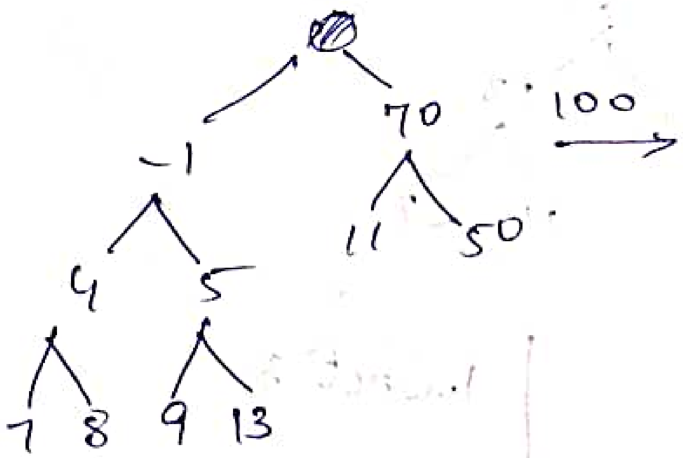
check for corresponding property
then check for heap property

Insertion

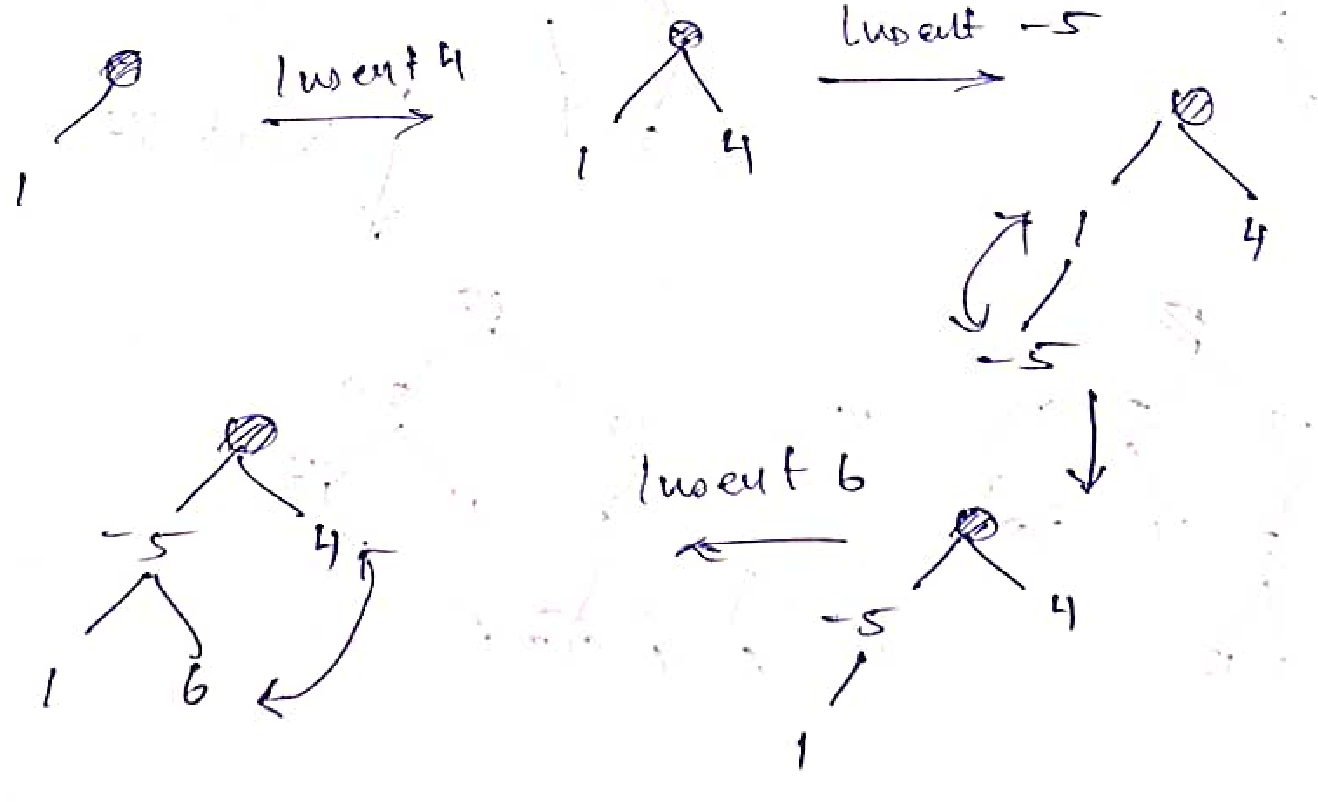
⇒ −1, 4, 5, 7, 9, 11, 13, 8, 50, 70, 100, 2, 4.

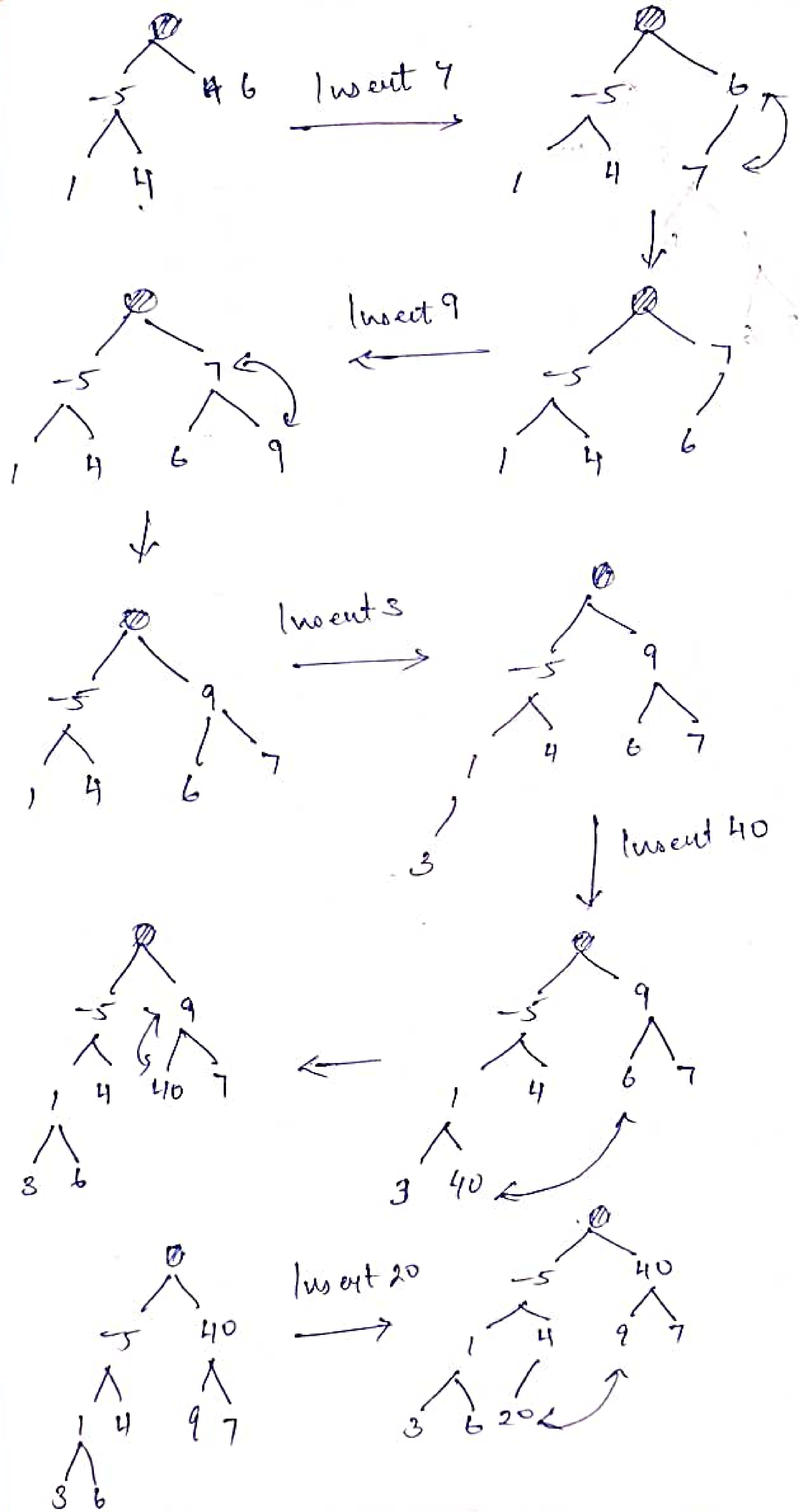


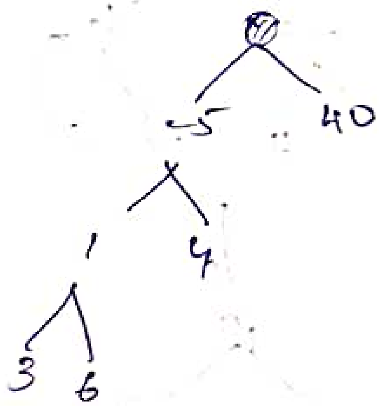

Insert 8
Insert 50
Insert 70



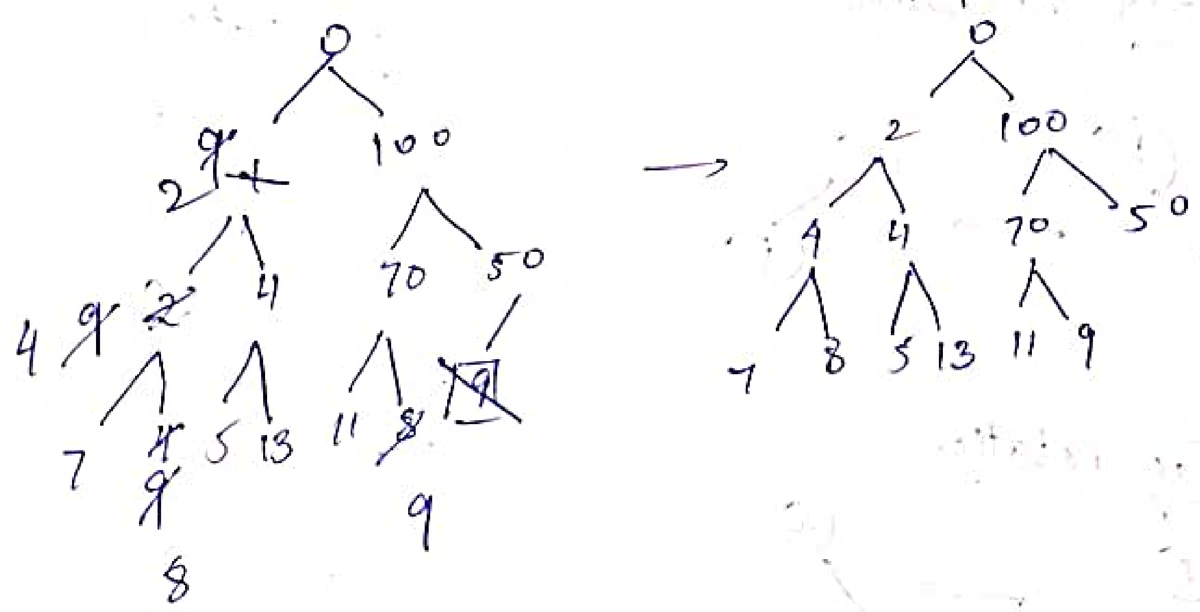
1, 4, -5, 6, 7, 9, 3, 40, 20, 70, 100



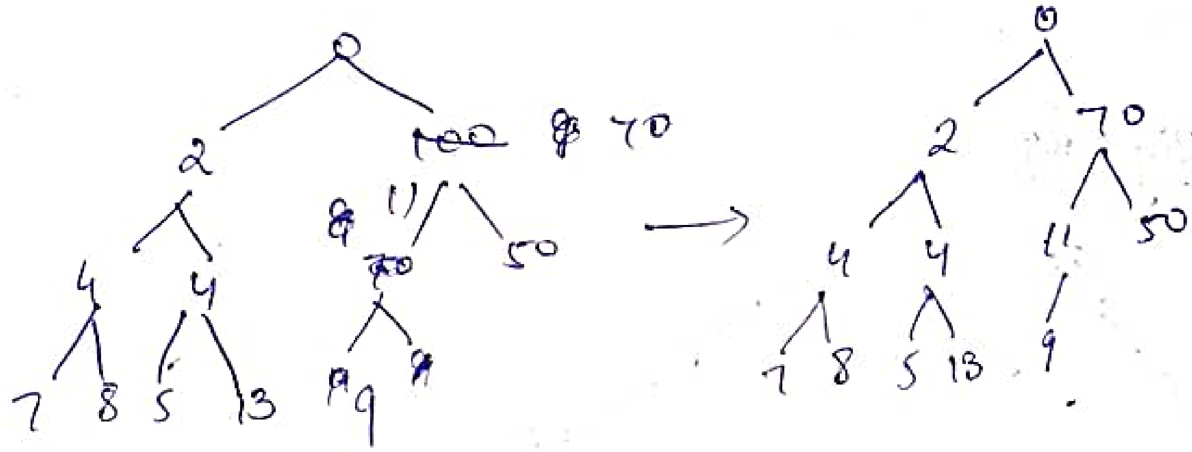

Insert 7
Insert 9
Insert 3
Insert 40
Insert 20


-5
40
1
4
3
6

Delete -1



Delete 100



Compare each heap.

Splay tree

BST

Splaying: sequence of rotations of node to root.

Not tightly balanced.

few data more frequently used.

Storing at root or near root.