Date: / /

1- Inserting 35

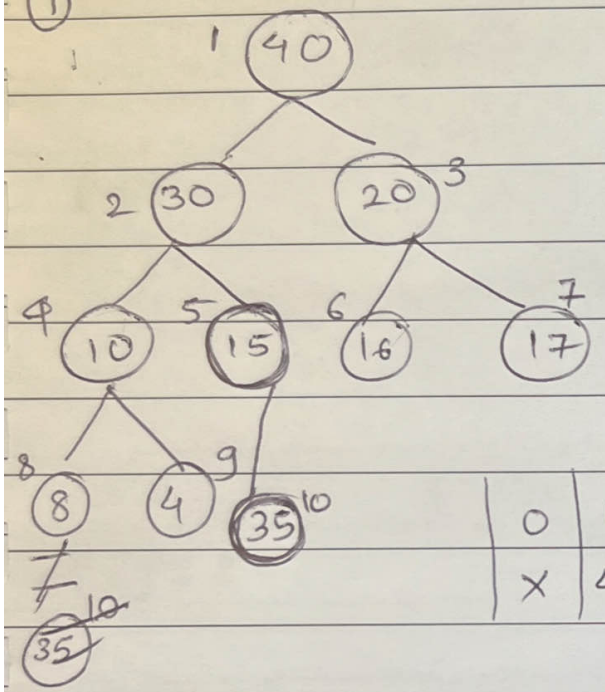
Input : 40, 30, 20, 10, 15, 16, 17, 8, 4.

Array :-

0	1	2	3	4	5	6	7	8	9	10
X	40	30	20	10	15	16	17	8	4	35

Added 35

(1).

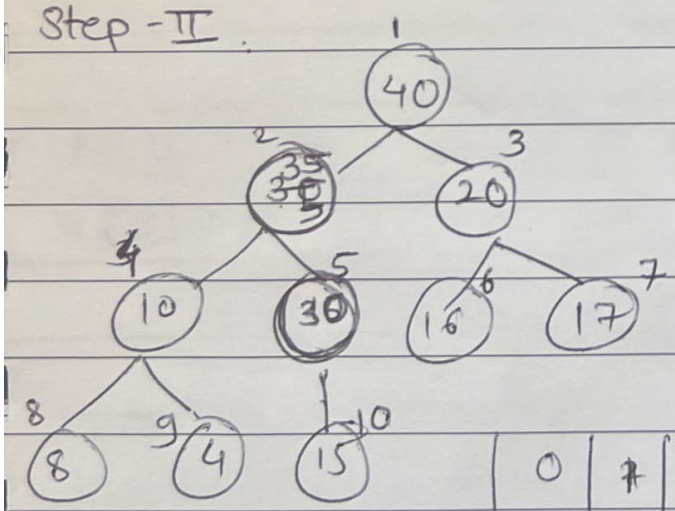


Since,
35 is greater than 15 (parent), will swap.

After swapping array :-

0	1	2	3	4	5	6	7	8	9	10
X	40	30	20	10	35	16	17	8	4	15

Step - II.



Again.
35 is greater 30 (parent) so we will swap.

After swapping :-

0	1	2	3	4	5	6	7	8	9	10
X	40	35	20	10	30	16	17	8	4	15

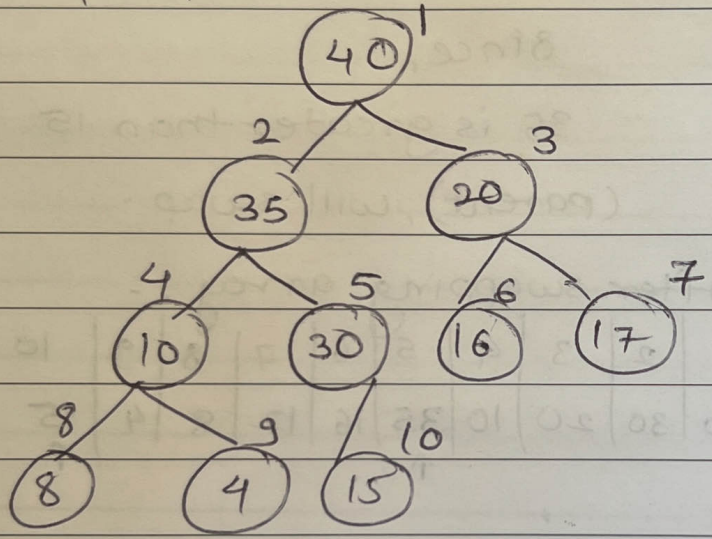
Date: / /

Since, 40 (parent) is greater than 35, there will no more swaps.

Final array looks like:

0	1	2	3	4	5	6	7	8	9	10
X	40	35	20	10	30	16	17	8	4	15

Heap representation :-



Date: / /

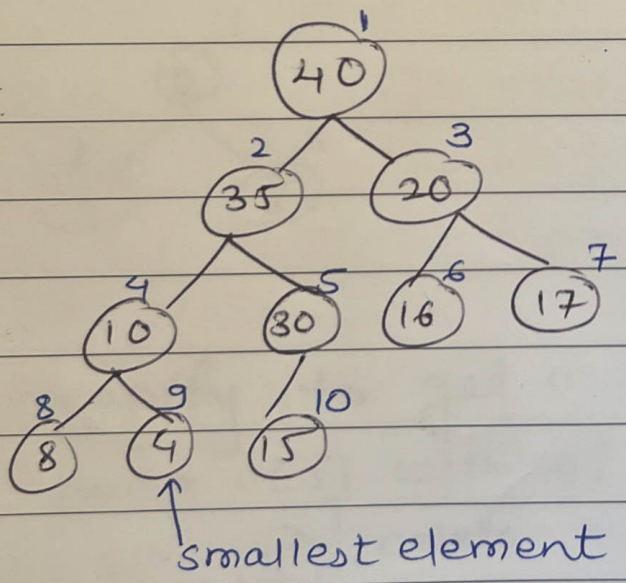
(2) · In a MAX heap containing n, numbers, the smallest element can be found in :-

complexity : O(n)

- In a MAXHEAP, the smallest element can be found on the last level of heap or the second last level of the heap.
- The time complexity to reach to last level is O(n)

For. ex.

A = {40, 35, 20, 10, 30, 16, 17, 8, 4, 15} - From 1st que.



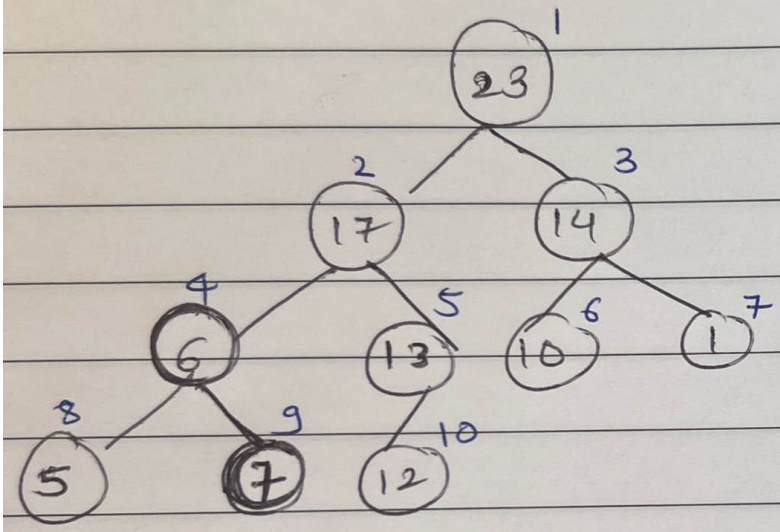
- To find 4, we need to traverse to all the all the levels of the heap.

(3) Is the array with values Date: / /

23, 17, 14, 6, 13, 10, 1, 5, 7, 12 a MAX HEAP-?

ANS:- Let's put it in heap representation:-

1	2	3	4	5	6	7	8	9	10
23	17	14	6	13	10	1	5	7	12



→ This is not a MAX-heap.
Since, MAX-heap should have root node values greater than its leaf node.

- At a[4] which 6, it has two children
  a[8] = 5
  a[9] = 7
  ∴ a[9] = 7 is greater than it's parent node. It is **not a** MAX-HEAP

4.

Date: / /

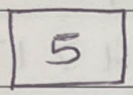
Buiding a Heap.

Top to Bottom.

$A = \{5, 3, 17, 10, 84, 19, 6, 22, 9\}$.

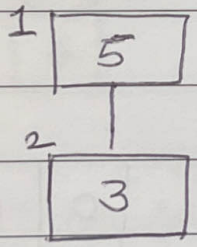
i	0	1	2	3	4	5	6	7	8	9
	X	5	3	17	10	84	19	6	22	9

$i = 1$.
$a[1] = 5$.



1
5

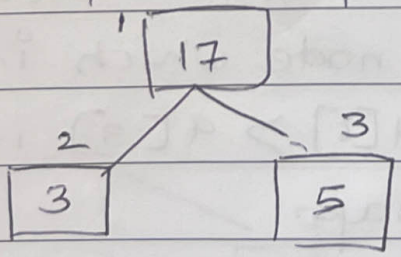
$i = 2$ since 3 is less than 5.
$a[2] = 3$.



New array :

0	1	2
X	5	3

$i = 3$ since 17 is greater than
$a[3] = 17$. 5, will swap.

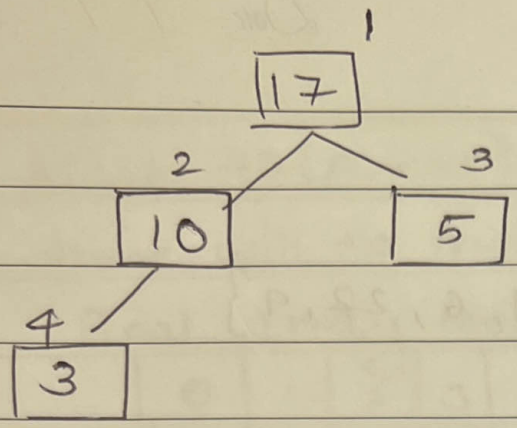


New array :

0	1	2	3
X	17	3	5

$i = 4$ since 10 is greater than 3
$a[4] = 10$ will swap

Date: / /



New array:-

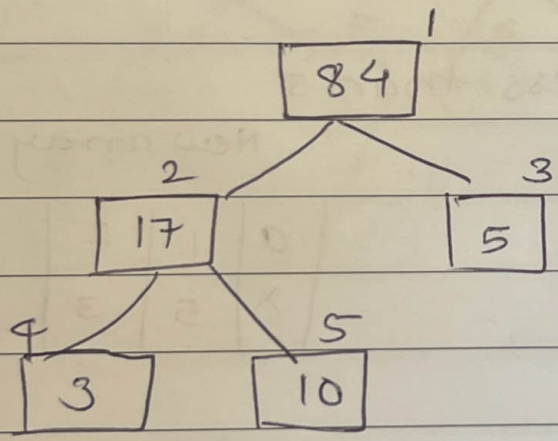
0	1	2	3	4
X	17	10	5	3

$i = 5$

$a[5] = 84$

- Since 84 is greater than 10, will swap with 10 first.
- Then will compare the parent node which is 17, $84 > 17$ so will swap 84 with 17.

∴



New array:

0	1	2	3	4	5
X	84	17	5	3	10

$i = 6$

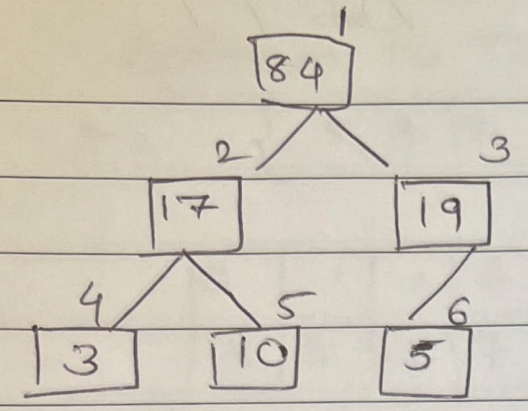
$a[6] = 19$

: For 6th index, will check with ~~internal~~ internal node which index 3. $a[3] = 5$

∴ Since $a[6] > a[3]$ i.e. $19 > 5$ will swap.

MATRIKAS

Date: / /

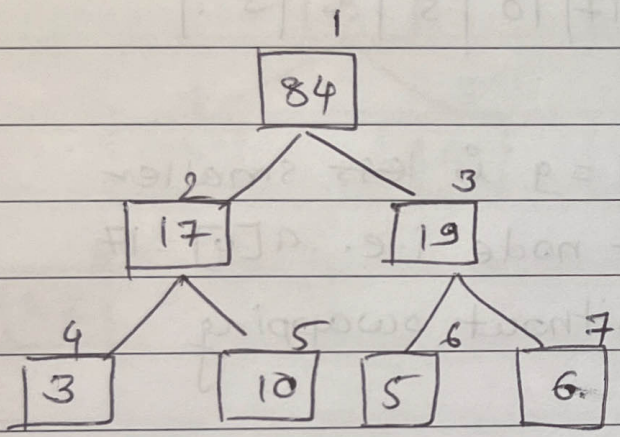


New array

0	1	2	3	4	5	6.
X	84	17	19	3	10	5

$i = 7$
$a[7] = 6$

since 6 is smaller than parent node at index 3, will keep as is.



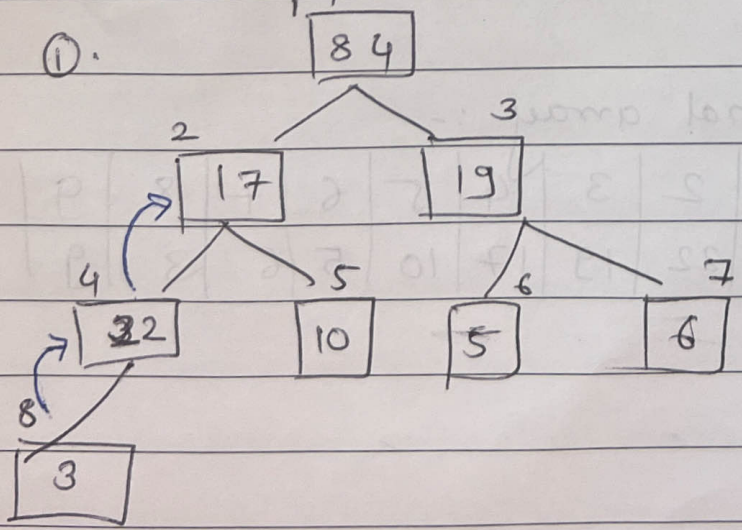
New array :.

0	1	2	3	4	5	6	7
X	84	17	19	3	10	5	6

$i = 8$
$a[8] = 22$

We will have leaf node at $i = 4$ which $a[4] = 3$, smaller than parent, so we will swap.

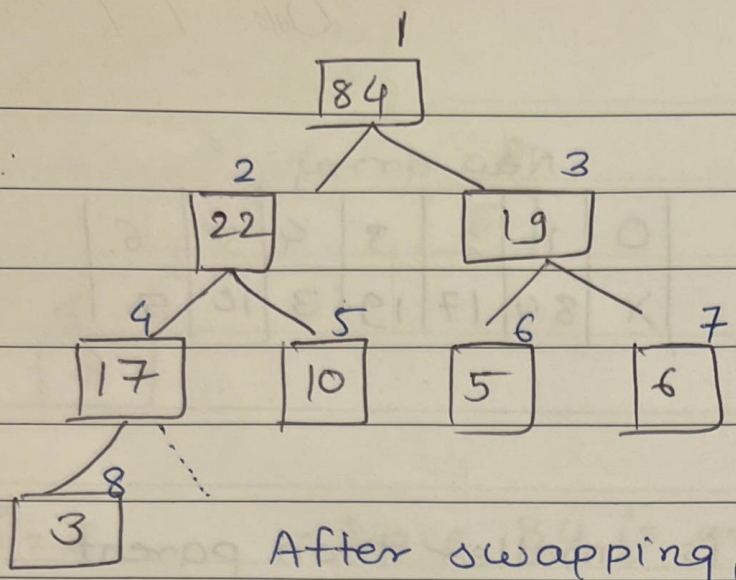
①.



Also,
$a[2] < a[4]$
will swap.

Since, parent can't be smaller than leaf node

Date: / /

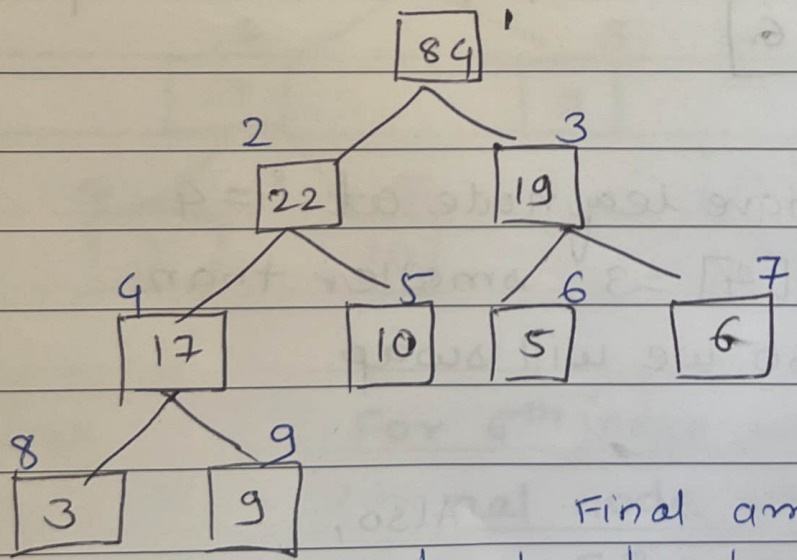


After swapping, twice final array:

0	1	2	3	4	5	6	7	8
X	84	22	19	17	10	5	6	3

i = 9
a[9] = 9

since a[9] = 9 is ~~less~~ smaller than parent node i.e. a[4] = 17
will put it, without swapping



Final array :-

0	1	2	3	4	5	6	7	8	9
X	84	22	19	17	10	5	6	3	9

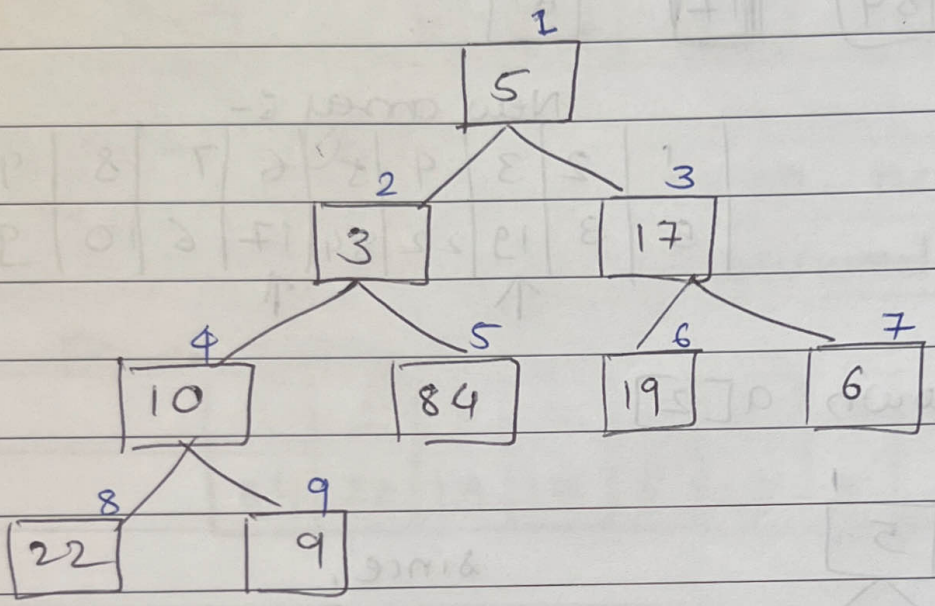
4. Building a heap

Date: / /

- Bottom to TOP.

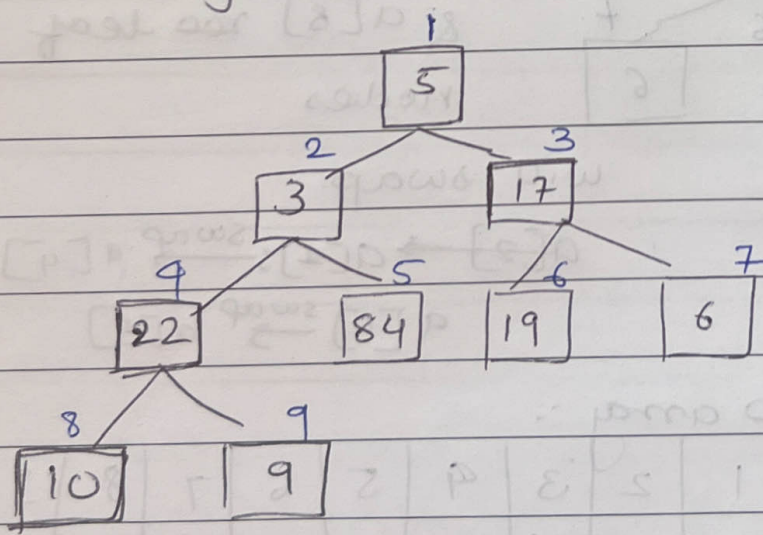
A = { 5, 3, 17, 10, 84, 19, 6, 22, 9 }

0	1	2	3	4	5	6	7	8	9
X	5	3	17	10	84	19	6	22	9.



Heapify a[4]

since a[4] > a[8]
we swapped.

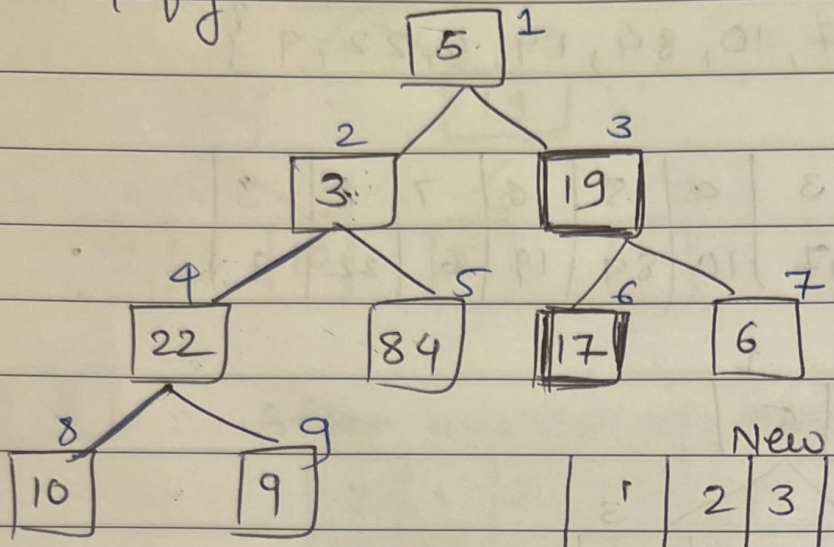


New array:-

0	1	2	3	4	5	6	7	8	9
X	5	3	17	22	84	19	6	10	9

Date: / /

② Heapify down a[3]

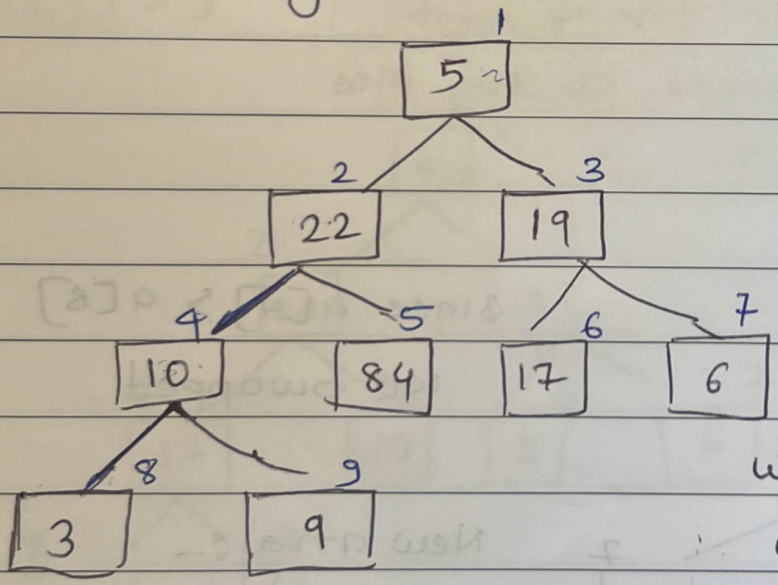


swapping a[3] with a[6]

New array :-

1	2	3	4	5	6	7	8	9
5	3	19	22	84	17	6	10	9

③ Heapify down a[2]



since, a[2] which was 3 smaller than a[4] & a[8] are leaf nodes will swap.

a[2] → a[2]: swap a[4]

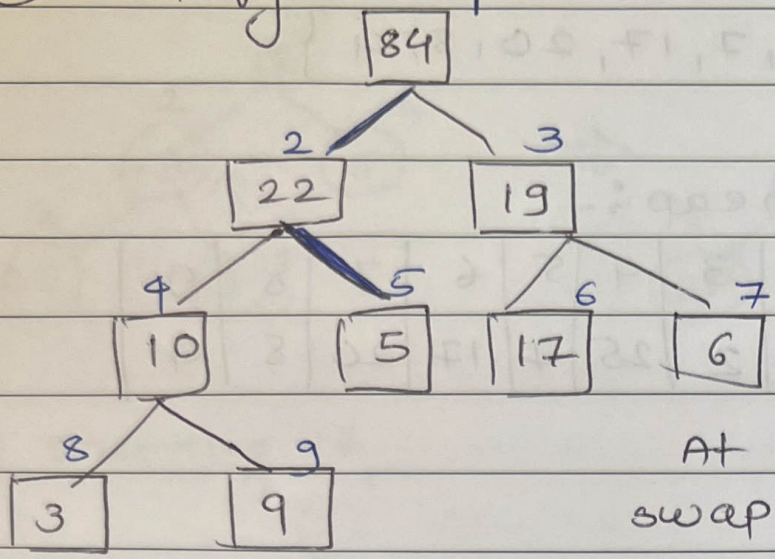
a[4] swap → a[8]

New array :-

1	2	3	4	5	6	7	8	9
5	22	19	10	84	17	6	3	9

Date: / /

④ Heapify down, a[i]



since,

~~22~~ a[2] > a[i]

will swap (22 & 5)

again, 84 is greater than 5, will swap

At last 84 > 22 will swap. Hence maxHeap achieved.

Final array :

1	2	3	4	5	6	7	8	9
84	22	19	10	5	17	6	3	9

5. Build minheap & maxheap:-

Date: / /

{5, 13, 2, 25, 7, 17, 20, 8, 4}

* Building minheap:-

A =

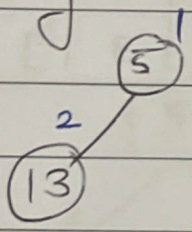
1	2	3	4	5	6	7	8	9
5	13	2	25	7	17	20	8	4

① Inserting 5

Array:

1
5

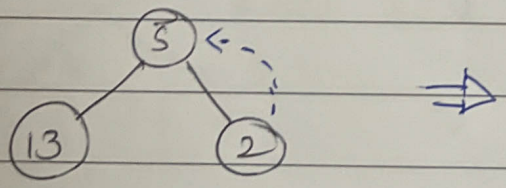
② Inserting 13

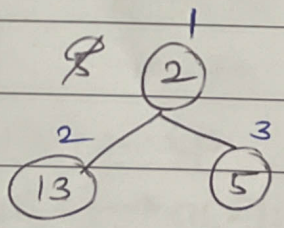


Array:-

1	2
5	13

③ Inserting 2

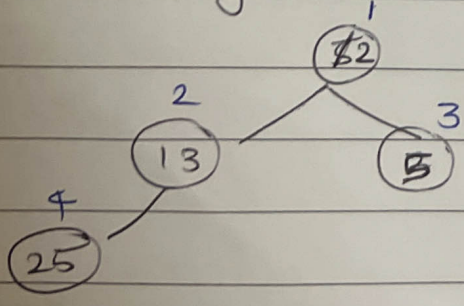




Array:

1	2	3
2	13	5

④ Inserting 25

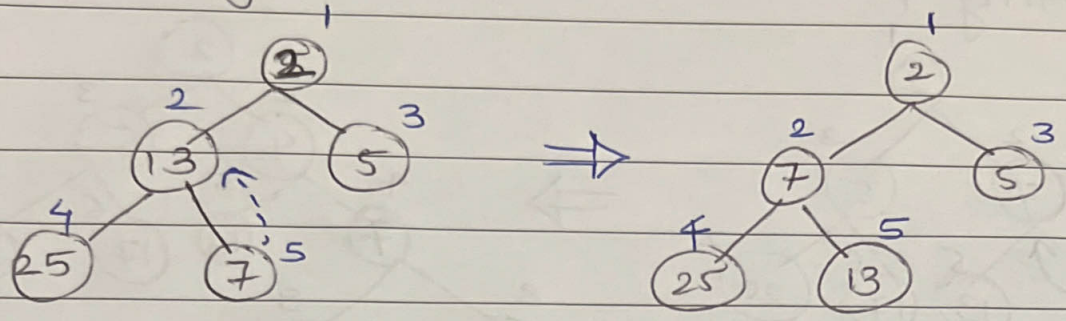


Array:

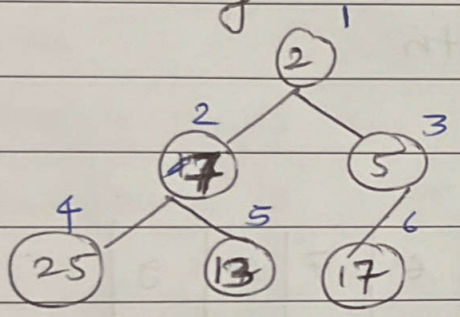
1	2	3	4
2	13	5	25

(5) Inserting (7)

Date: / /



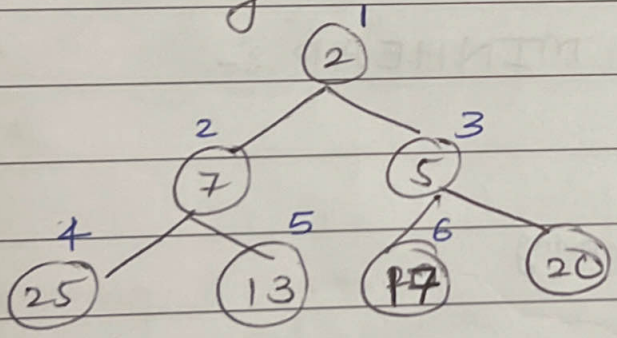
(6) Inserting 17



New arrng :-

1	2	3	4	5	6
2	7	5	25	13	17

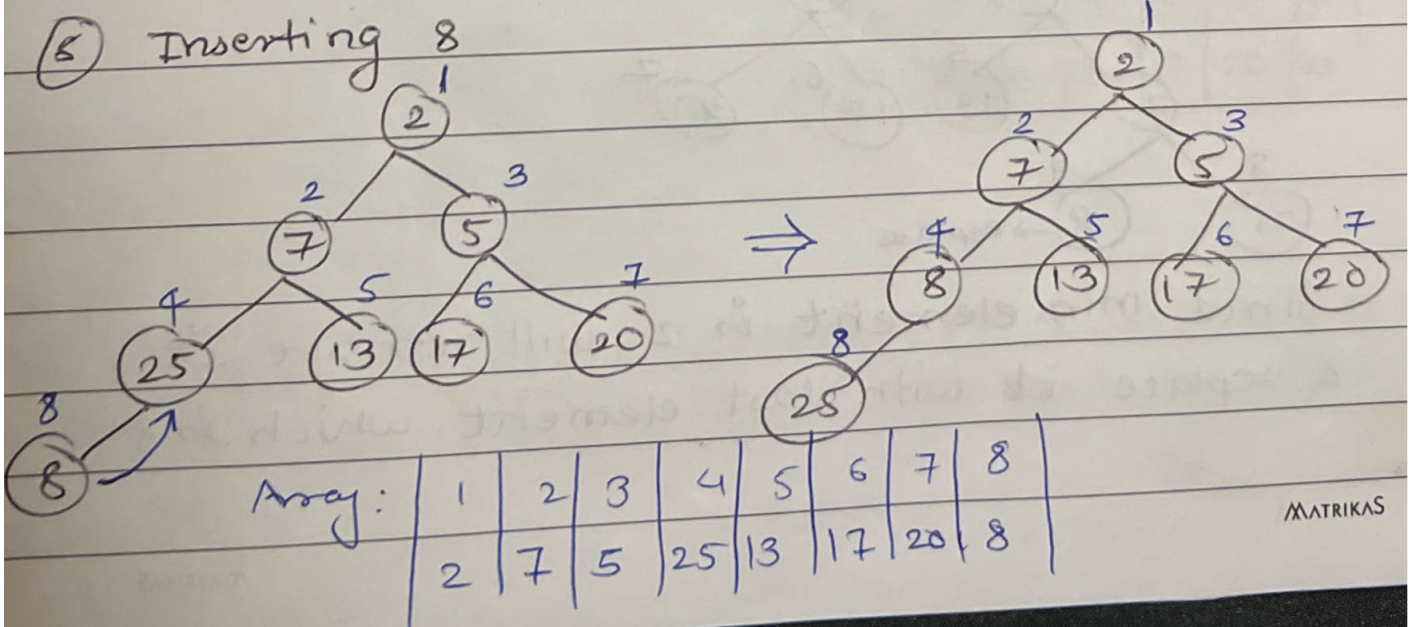
(7) Inserting 20



Array:

1	2	3	4	5	6	7
2	7	5	25	13	17	20

(8) Inserting 8



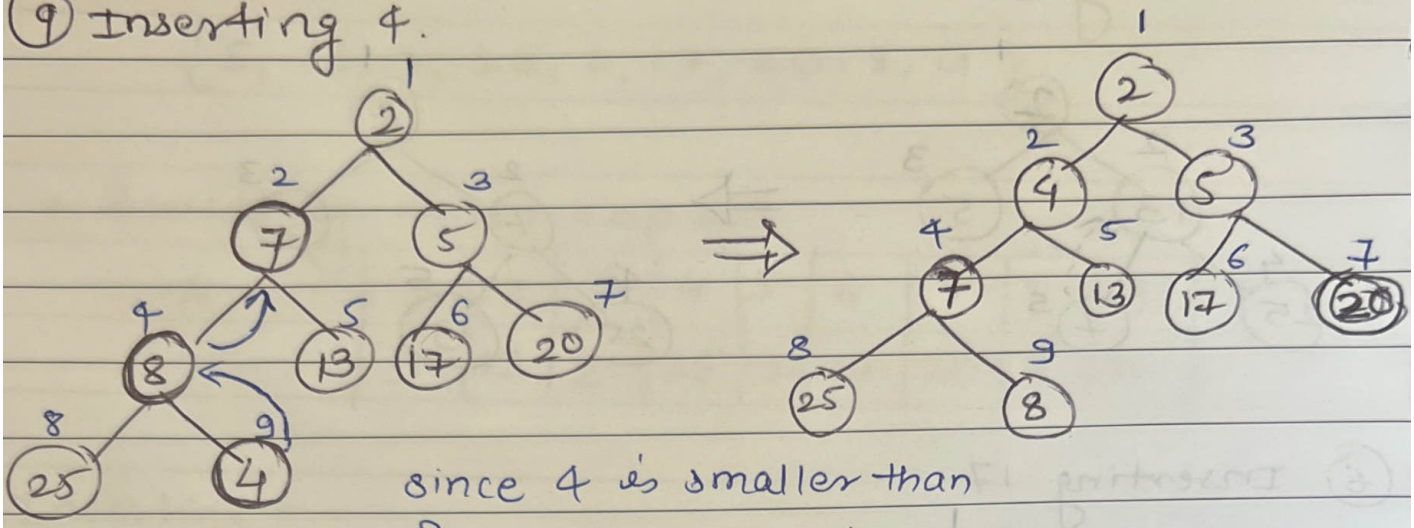
Array:

1	2	3	4	5	6	7	8
2	7	5	25	13	17	20	8

MATRIKAS

Date: / /

(9) Inserting 4.



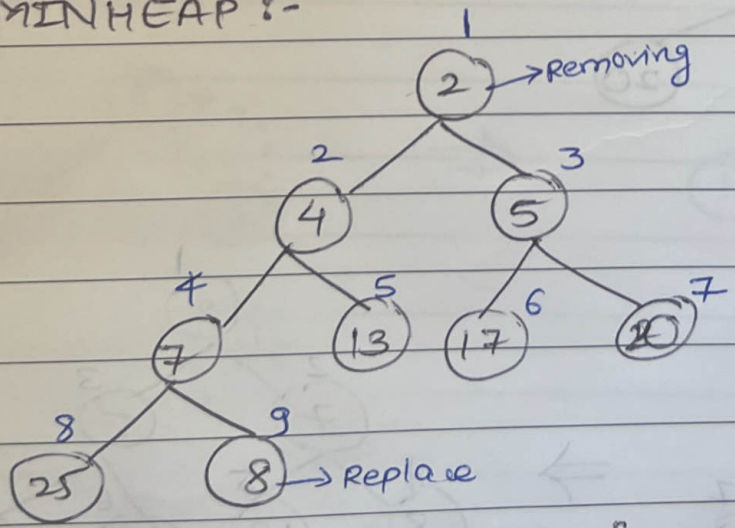
since 4 is smaller than 8 & 7 will swap with 8 first then 7

Final array :-

1	2	3	4	5	6	7	8	9
2	4	5	7	13	17	20	25	8

* Deleting min from MINHEAP :-

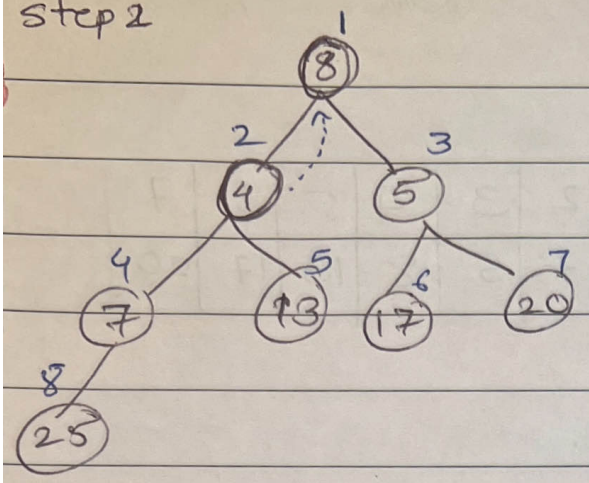
MINHEAP :-



Since min element is 2, will remove it & replace it with last element. which is 8

Date: / /

step 2

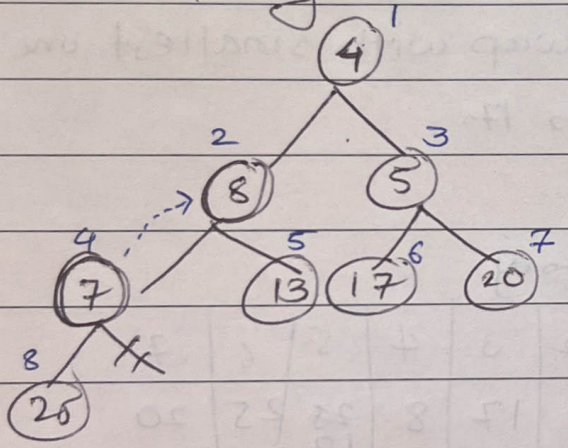


A =

1	2	3	4	5	6	7	8
8	4	5	7	13	17	20	25

Since, its a MINHEAP, Parent node cannot be greater than ~~child node~~ leaf node, (Heapify)

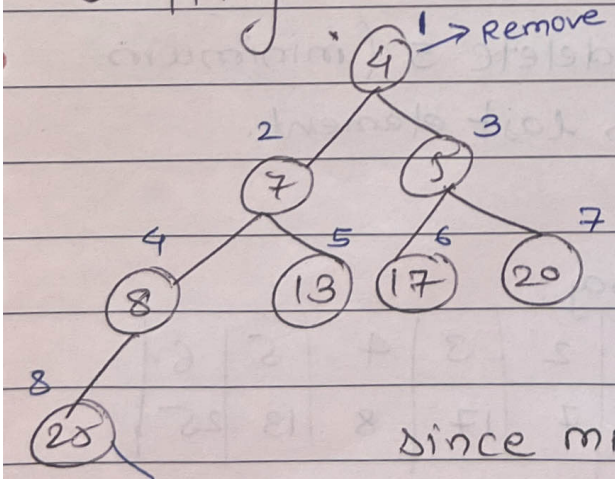
swapping 8 with 4.



A =

1	2	3	4	5	6	7	8
4	8	5	7	13	17	20	25

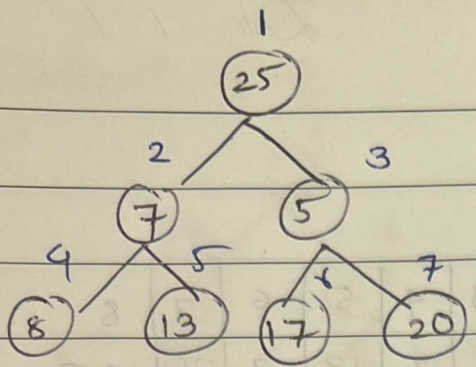
∴ swapping 8 with 7.



1	2	3	4	5	6	7	8
4	7	5	8	13	17	20	25

Since MIN element is 4, will delete 4, & replace it with last element.

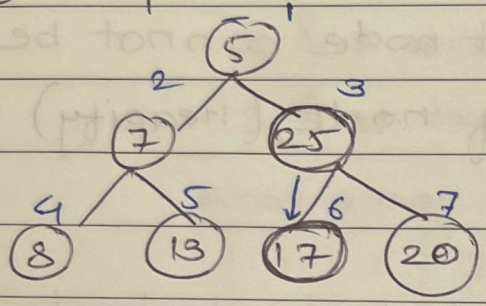
Date: / /



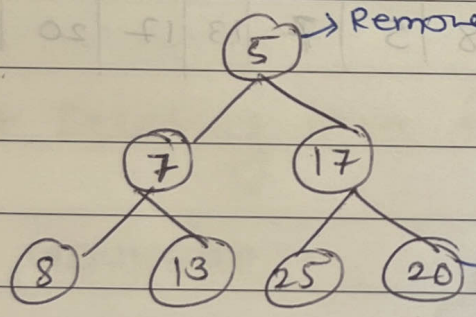
1	2	3	4	5	6	7
25	7	5	8	13	17	20

∴ Swapping to get MINHEAP.

(1) swap 25 with smallest element 5.



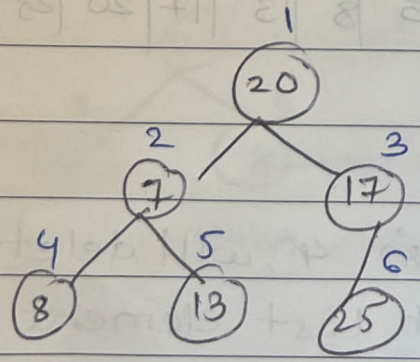
since, 25 is greater than it leaf nodes, will swap with smallest one which is 17.



Final array

1	2	3	4	5	6	7
5	7	17	8	~~25~~ 13	25	20

∴ Since, its MINHEAP, will delete 5 (minimum element) & replace it with last element.

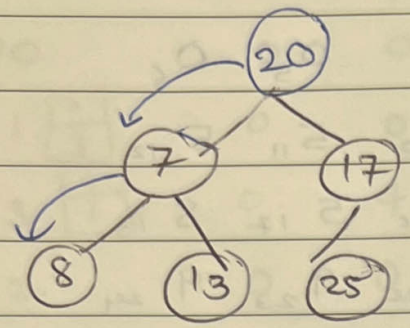


Array:

1	2	3	4	5	6
20	7	17	8	13	25

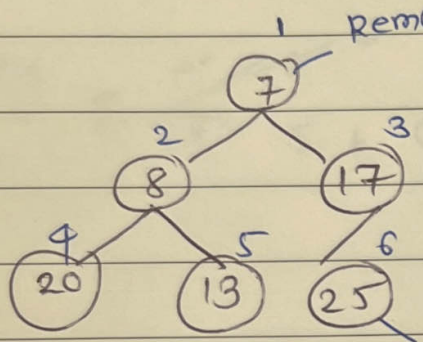
Date: / /

Heapify, to get minimum element at root.



① swap 20 with 7

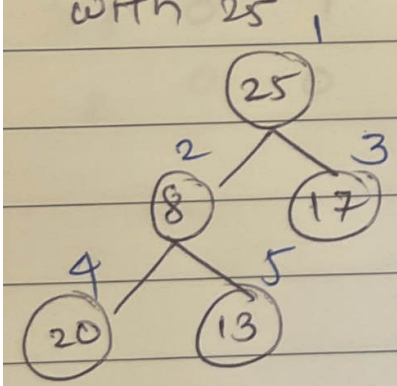
② swap 20 with 8.



since, all the leaf nodes are smaller than leaf node, its a MINHEAP

1	2	3	4	5	6		
7	8	17	20	13	25		

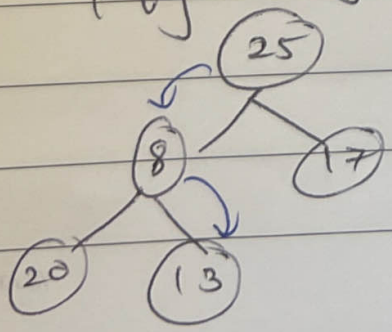
+ Remove minimum element i.e. 7 & replace it with 25



Array :-

1	2	3	4	5
25	8	17	20	13

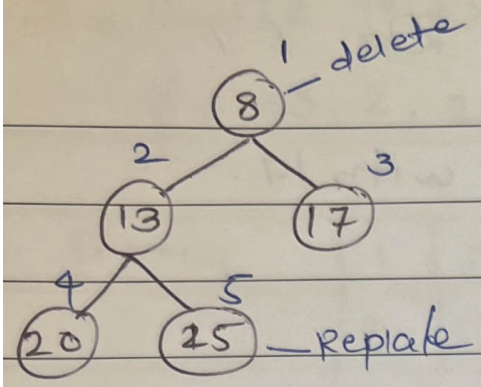
* Heapify, to get MINHEAP.



① swap 25 with 8

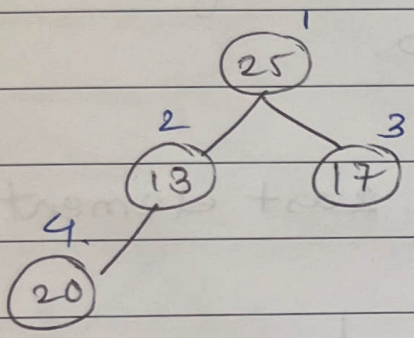
② swap 25 with 13.

Date: / /



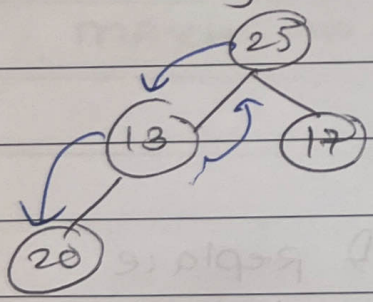
Since, all the leaf nodes are smaller than root node its MINHEAP.

* ~~Ref~~ Delete MIN element (8) & replace with last one.

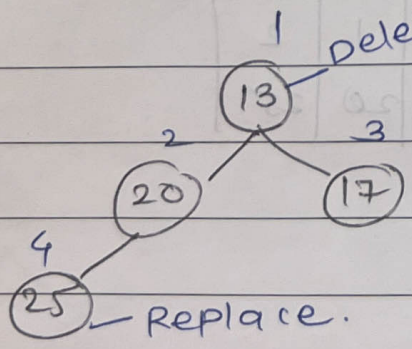


1	2	3	4
25	13	17	20

* Heapify to get MINHEAP :-



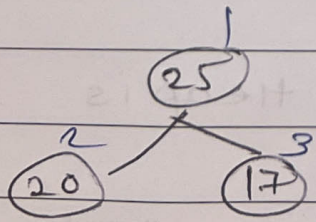
① swap 25 with 13
② swap 25 with 20.



Since, it's a MINHEAP

1	2	3	4
13	20	17	25

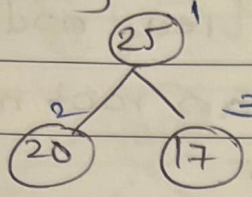
→



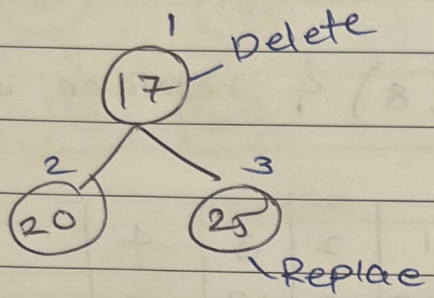
1	2	3
25	20	17

Date: / /

* Heapify to get MINHEAP :-

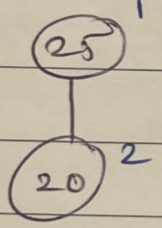


Replace 25 with 17.



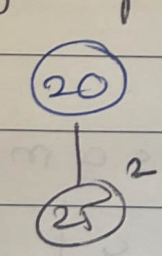
Since, root node is greater than leaf its MINHEAP.

- Delete 17 & Replace with last element



1	2
25	20

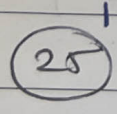
* Heapify to get MINHEAP



Delete 20 & Replace 25

1	2
20	25

Final :-



1
25

Delete last element 25 & Now Heap is Empty.

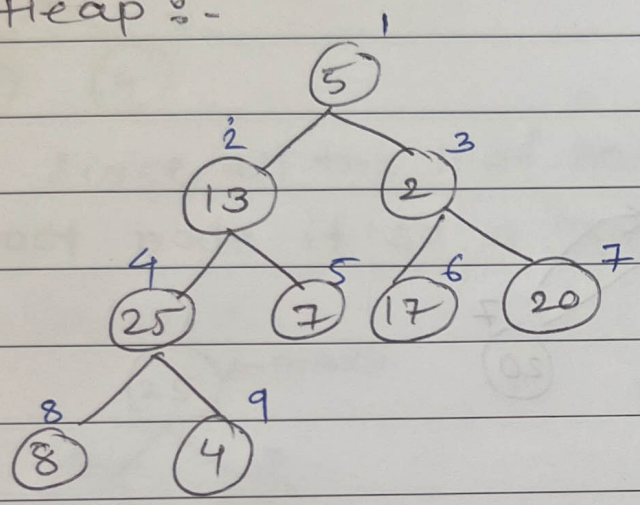
5 * Build MAXHEAP.

Date: / /

{5, 13, 2, 25, 7, 17, 20, 8, 4}.

Array:	1	2	3	4	5	6	7	8	9
	5	13	2	25	7	17	20	8	4

: Heap :-

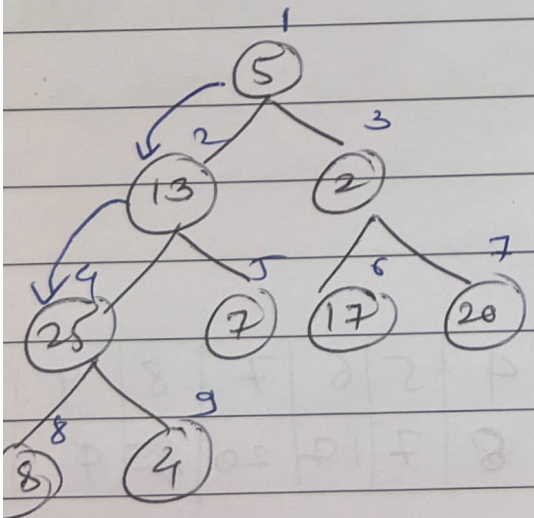


∴ MAXHEAP : has largest element as root node.

Starting from leaf node

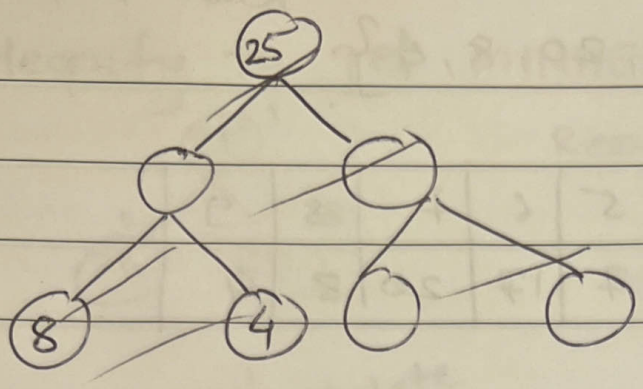
① swap 25 with 13 at a[4]

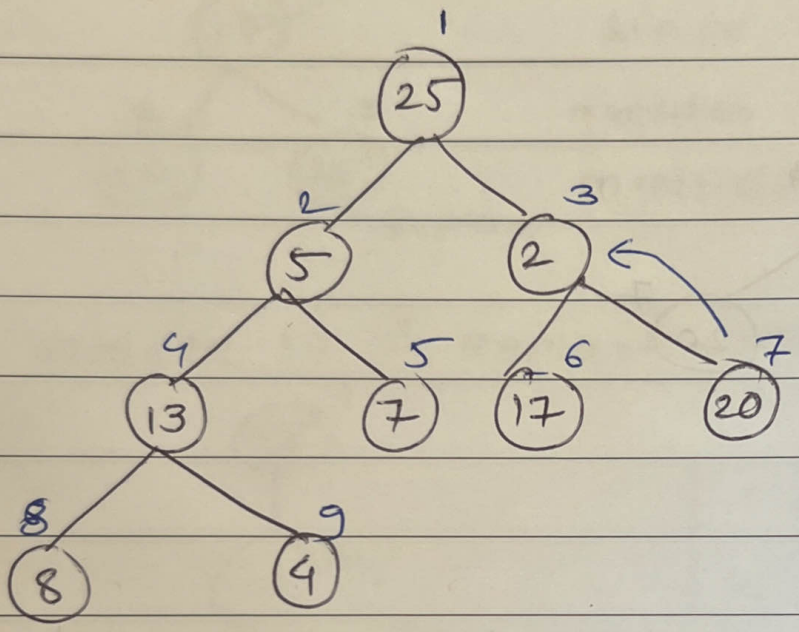
② swap 25 with 5 at a[1]



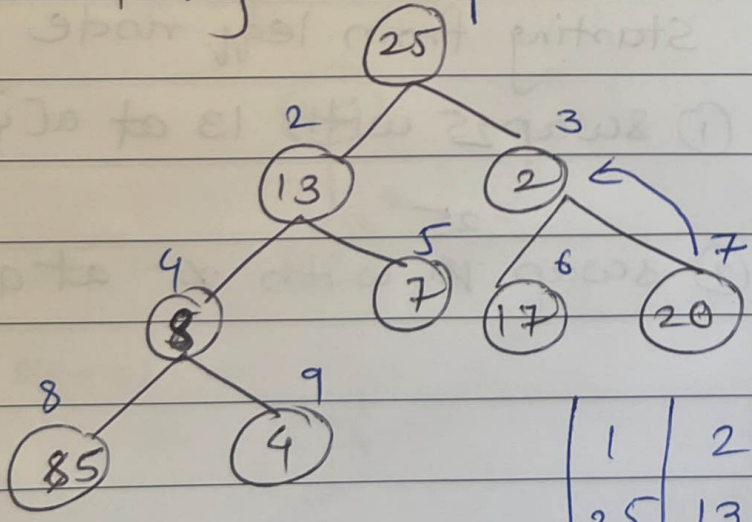
MATRIKAS

Date: / /





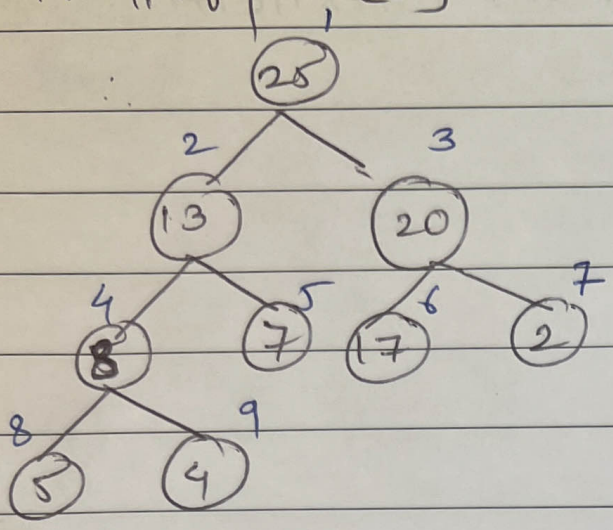
Heapify 4 : a[4]



1	2	3	4	5	6	7	8	9
25	13	2	8	7	17	20	85	4

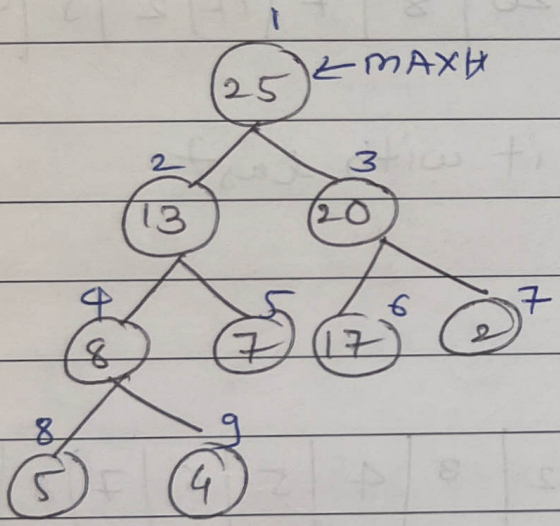
Date: / /

Heapify a[3]



1	2	3	4	5	6	7	8	9
25	13	20	8	7	17	2	5	4

Since, all the leaf nodes are smaller than root node it is a MAXHEAP.



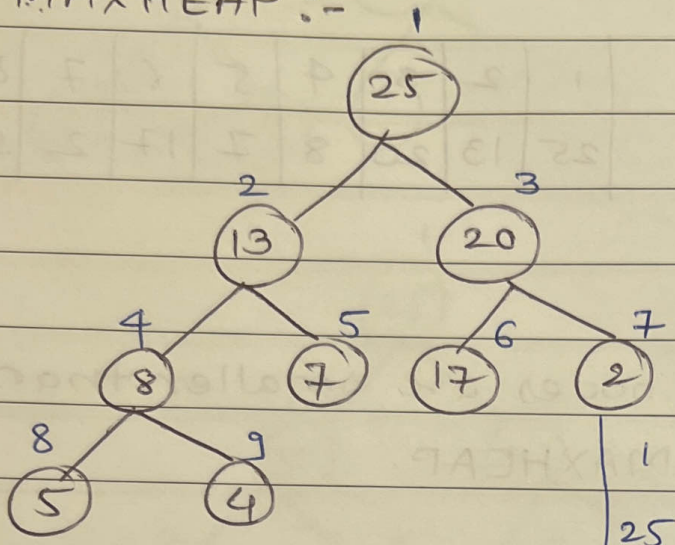
Final array :-

1	2	3	4	5	6	7	8	9
25	13	20	8	7	17	2	5	4

Date: / /

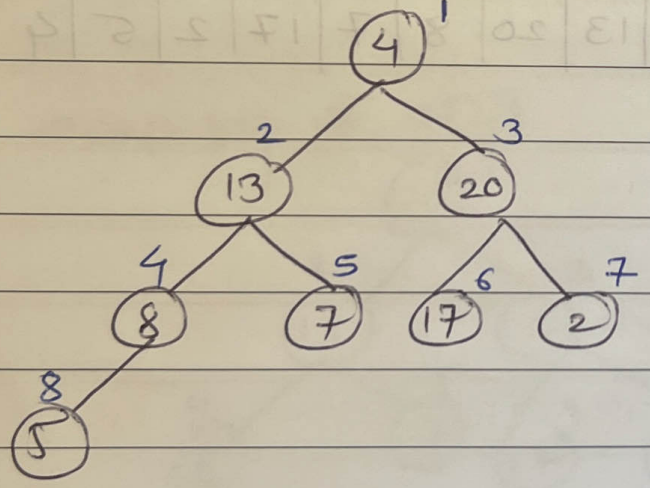
* Deleting MAX element from MAXHEAP.

MAXHEAP :-



1	2	3	4	5	6	7	8	9
25	13	20	8	7	17	2	5	4

(1) Delete, MAX → 25 & replace it with last element which is 4.



1	2	3	4	5	6	7	8
4	13	20	8	7	17	2	5

Heapify to get MAXHEAP.

∴ a[4] is already sorted.

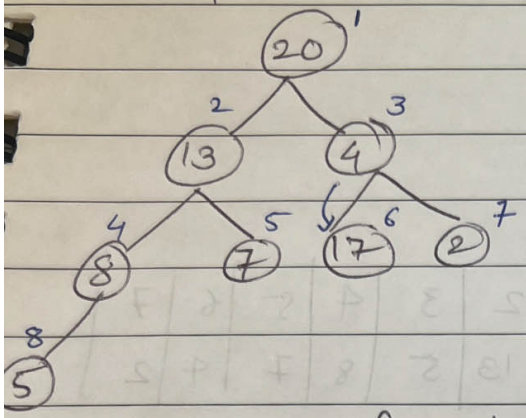
∴ a[3] ⇒ 20 already sorted } All the leaf nodes are smaller than root node.

∴ a[2] ⇒ 13 —11—

Date: / /

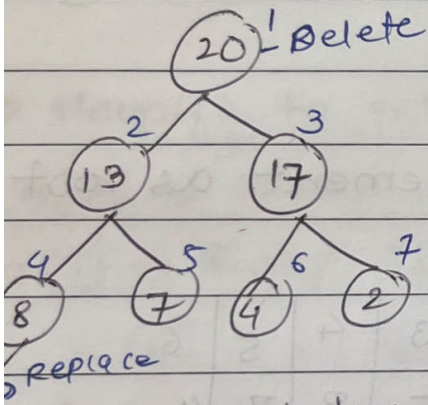
a[i] ⇒ 4.

Swap a[i] with a[3] to get MAXHEAP.



1	2	3	4	5	6	7	8
20	13	4	8	7	17	2	5

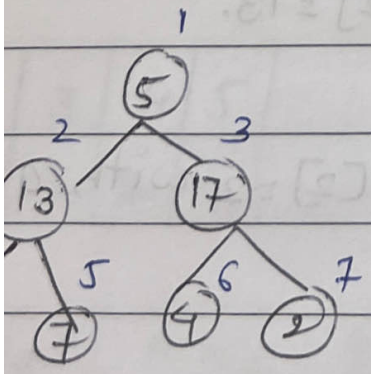
At a[3], leaf nodes are greater than root node, will swap.



1	2	3	4	5	6	7	8
20	13	17	8	7	4	2	5

Since, all the leaf nodes are smaller than root node, its MAXHEAP.

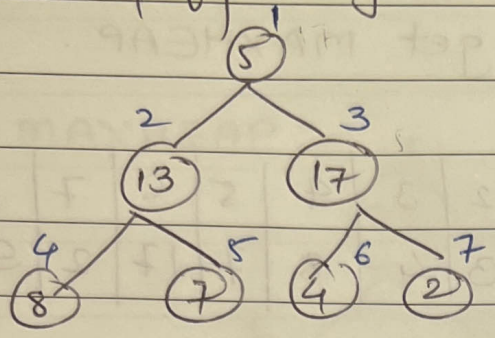
elete MAX i.e. 20 & replace with last element



1	2	3	4	5	6	7
5	13	17	8	7	4	2

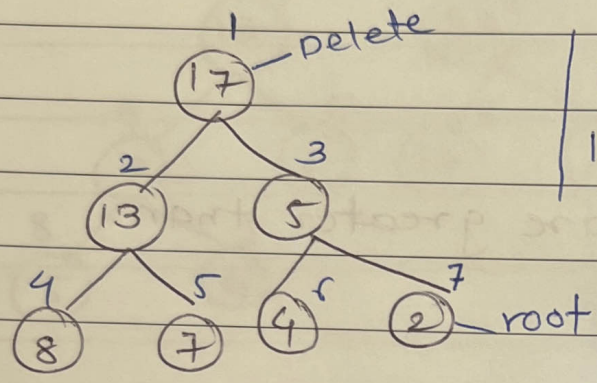
Date: / /

Heapify to get MAXHEAP.



swap a[3] = 17 with a[1] = 5

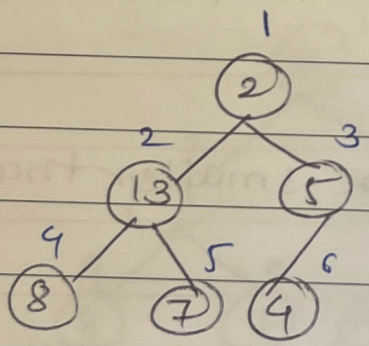
then,



1	2	3	4	5	6	7
17	13	5	8	7	4	2

- It's a MAXHEAP.

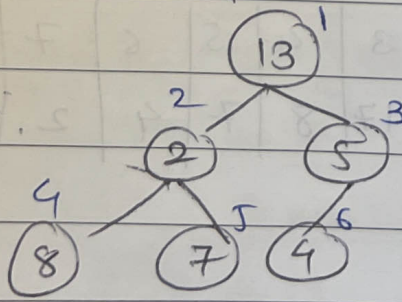
→ Delete 17 & place last element as root.



1	2	3	4	5	6
2	13	5	8	7	4

→ Heapify to get MAXHEAP.

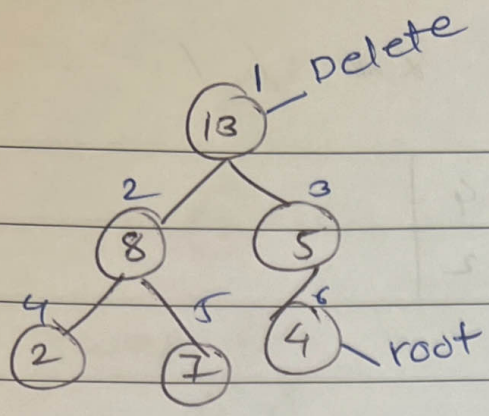
① swap a[1] = 2 with a[2] = 13.



Now,

swap a[2] = 2 with a[4]

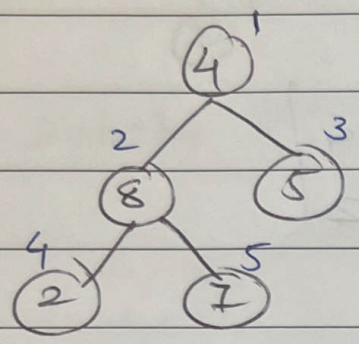
Date : / /



It's a MAXHEAP

1	2	3	4	5	6
13	8	5	2	7	4

* Delete 13 & replace with 4.

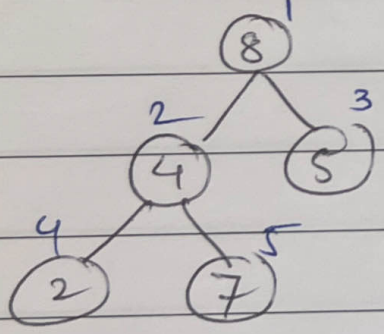


1	2	3	4	5
4	8	5	2	7

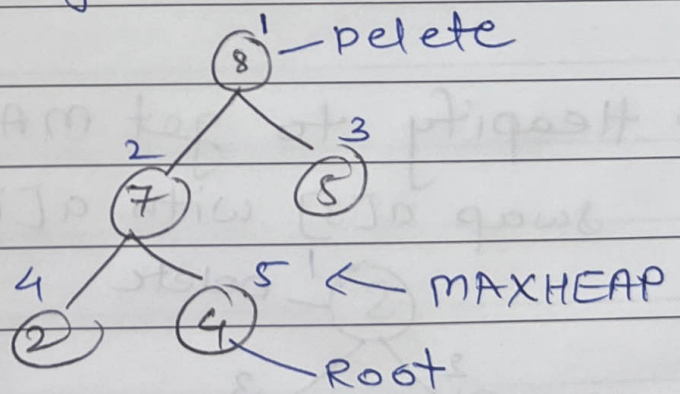
→ Heapify to get MAXHEAP.

a[2] ⇒ sortep

a[1] ⇒ Swap a[1] = 4 with a[2] which is 8



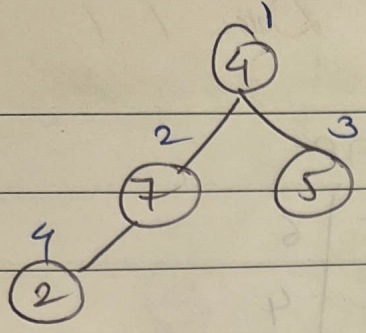
swap a[2] = 4 with a[5] = 7 to get MAXHEAP



1	2	3	4	5
8	7	5	2	4

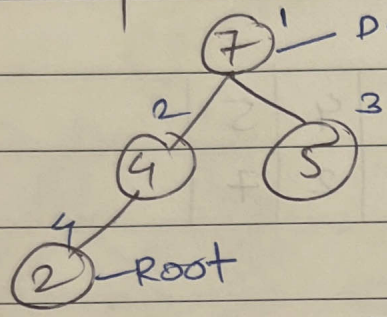
* Delete MAX = 8 & replace with 4

Date: / /



1	2	3	4
4	7	5	2

* Heapify to get MAXHEAP
  - swap a[2] = 7 with a[1] = 4

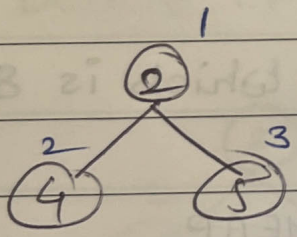


1	2	3	4
7	4	5	2

It's a MAXHEAP

→ Delete MAX = 7 & replace with 2.

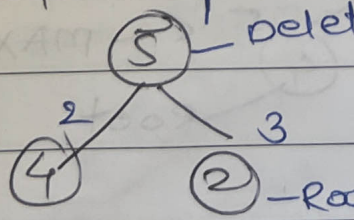
∴



1	2	3
2	4	5

* Heapify to get MAX HEP

swap a[3] with a[1]

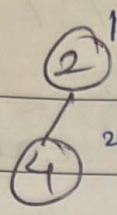


1	2	3
5	4	2

It's a MAXHEAP.

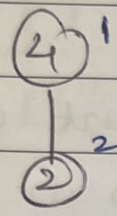
- Delete a[1] & Replace with 2

Date: / /



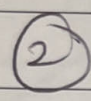
1	2
2	4

Swap for MAXHEAP



1	2
4	2

- Delete a[i] = 4.



1
2

- Final delete a[i] = 2 & MAXHEAP is empty Now.