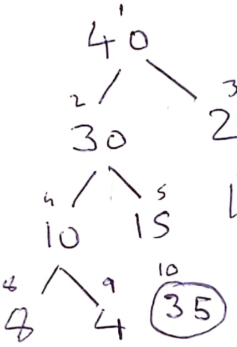
Hw 7

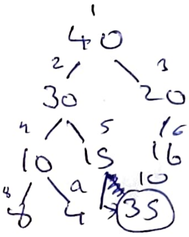
1. [40, 30, 20, 10, 15, 16, 17, 8, 4]

1 2 3 4 5 6 7 8

[40, 30, 20, 10, 15, 16, 17, 8, 4, 35



->



-> to insert (append the element at the end) and check its father.

Since 35 > 15 i.e:- element at 10th index is greater than element at 5th index

(i//2) -> to find father element

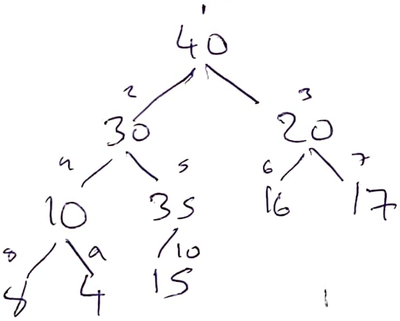
-> 10//2=5

∴ we swap 35 and 15 :-

1 2 3 4 5 6 7 8 9 10

[40, 30, 20, 10, 35, 16, 17, 8, 4, 15]

tree



5//2 -> 2

∴ father of 35 -> 30

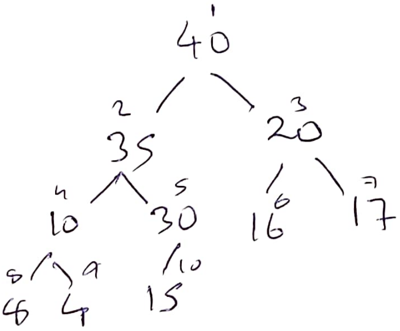
is 30 > 35 -> false

∴ swap 30 and 35

1 2 3 4 5 6 7 8 9 10

[40, 35, 20, 10, 30, 16, 17, 8, 4, 15]

tree



∴ this is the new heap after 35 is inserted

2. Complexity → O(n)

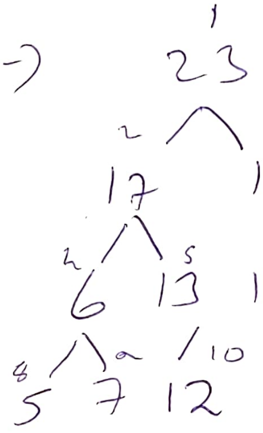
why → if the Min element is anywhere in the tree (i.e in any node) then we need to check both left and right sub-trees to find the minimum element.

∴ traversing the entire tree, each takes O(n) time Complexity

1 2 3 4 5 6 7 8 9 10

3. [23, 17, 14, 6, 13, 10, 1, 5, 7, 12] is max heap ?

Tree →



No, it is not a Max heap

explaination :- root 23 > left node 17 → true

23 > Right node 14 → true

17 is > left 6 → true

17 > Right 13 → true

14 > left 10 → true

14 > 1 → true

6 > left 5 → true

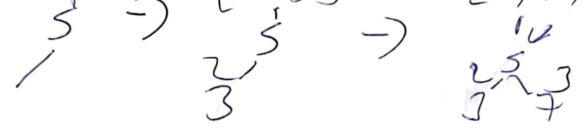
→ But, 6 > Right 7 → False

Since Parent (6) is lesser than Right Child node (7), this array is not a Maxheap

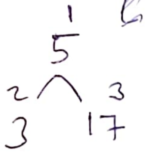
1 2 3 4 5 6 7 8 9

4. [5, 3, 17, 10, 84, 19, 6, 22, 9]

Build Maxheap :- [5] → [5, 3] → [5, 3, 7]



1. Top to Bottom



check if left node is greater than root

:- i * 2 → 1×2 → 2th index = 3

∴ is 5 > 3 → true

Consider 17 because 17 > 3

check if right node is greater than root

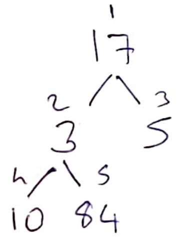
:- (i*2)+1 → 1 * 2 + 1 → 3nd index = 17

∴ is 5 > 17 → False

∴ Swap 5 and 17

1 2 3 4 5 6 7 8 9

[17, 3, 5, 10, 84, 19, 6, 22, 9]



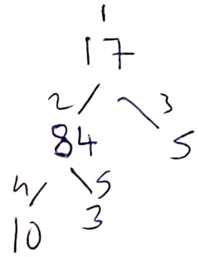
Consider 84 because 84 > 10

is 84 > 3 → true (i.e 5//2 → 2nd index = 3)

∴ Swap 3 and 84

1 2 3 4 5 6 7 8 9

[17, 84, 5, 10, 3, 19, 6, 22, 9]

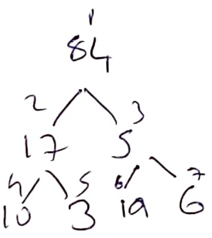


is 84 > 17 → true (i.e 2//2 → 1st index = 17)

∴ Swap 17 and 84

1 2 3 4 5 6 7 8 9

[84, 17, 5, 10, 3, 19, 6, 22, 9]

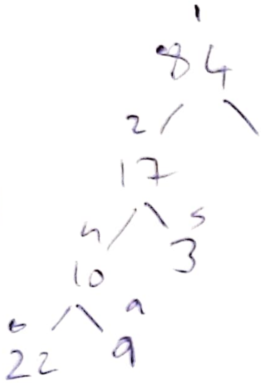


Consider 19 because 19 > 6

∴ is 19 > 5 → true (6//2 → 3rd index = 5)

Swap 5 and 19

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

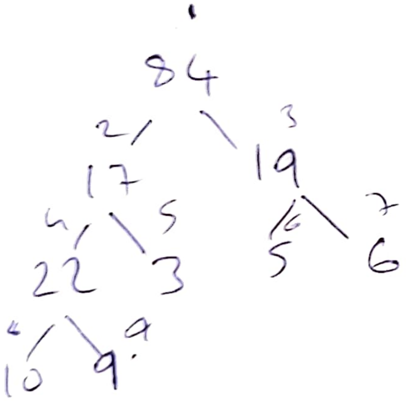


Consider 22 since 22 > 9

is 22 > 10 → true (8//2 → 4th index = 10)

∴ swap 10 & 22

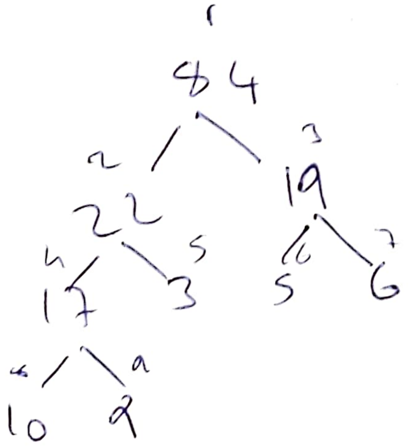
1 2 3 4 5 6 7 8 9
[84, 17, 19, 22, 3, 5, 6, 10, 9]



~~Consider~~ is 22 > 17 → true

∴ swap 22 and 17

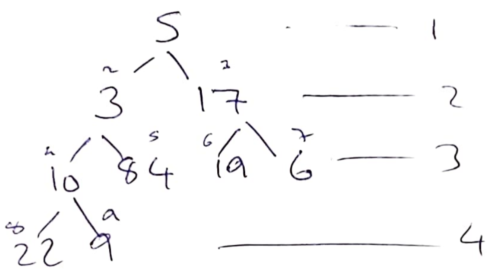
1 2 3 4 5 6 7 8 9
[84, 22, 19, 17, 3, 5, 6, 10, 9]



∴ the Maxheap is built from Top to bottom

4.2 Bottom to top → [5, 3, 17, 10, 84, 19, 6, 22, 9]

Create a tree :-



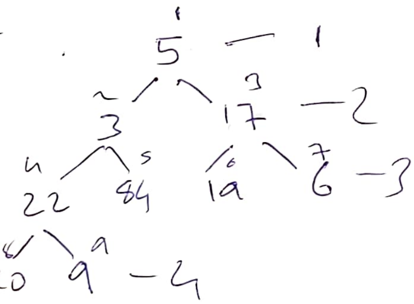
Start from 4

Consider 22 because 22 > 9

is 22 > 10 → true (8//2 → 4th index = 10)

∴ Swap 22 and 10

[5, 3, 17, 22, 84, 19, 6, 10, 9]



now check - 3

consider 84 since 84 > 22

is 84 > 3 → true (5//2 → 2 = 3)

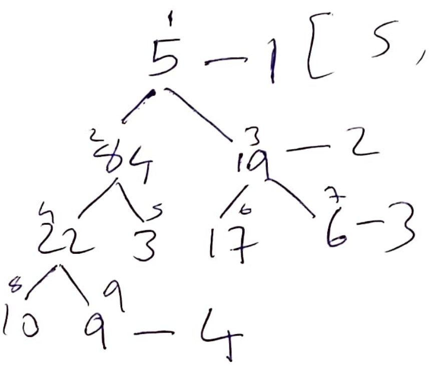
∴ Swap 84 and 3

also checks for Right node

consider 19 since 19 > 6

is 19 > 17 → true (6//2 → 3 → 17)

∴ Swap 19 & 17



[5, 84, 19, 22, 3, 17, 6, 10, 9]

now Check - 2

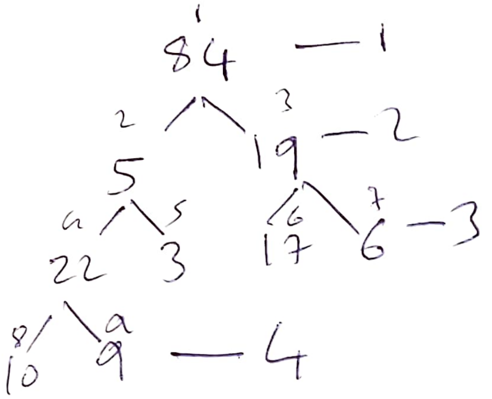
Consider 84 since 84 > 19

is 84 > 5 → true (2//2 → 1 = 5)

∴ swap 84 & 5

1 2 3 4 5 6 7 8 9

[84, 5, 19, 22, 3, 17, 6, 10, 9]



in the step, check now if the father is greater than child if not, swap

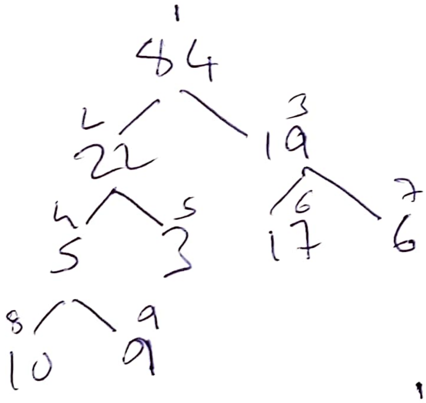
is 5 > 22 (left node)
False (i*2)

∴ swap 5 and 22

1 2 3 4 5 6 7 8 9

[84, 22, 19, 5, 3, 17, 6, 10, 9]

(keep on ~~Again~~ checking till it satisfies)

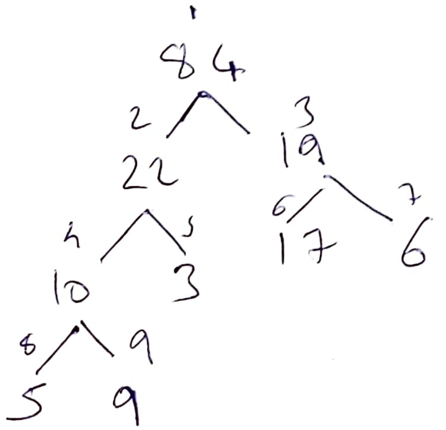


Now is 5 > 10 → False
(4*2=8 → 10)

∴ Swap 5 and 10

1 2 3 4 5 6 7 8 9

∴ [84, 22, 19, 10, 3, 17, 6, 5, 9]

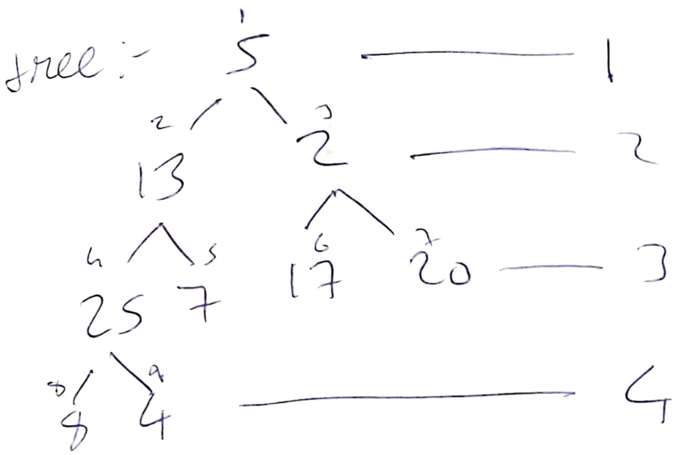


∴ this is the final heap after bottom up approach

5. [5, 13, 2, 25, 7, 17, 20, 8, 4]

Build Max heap using bottom-up

tree :-



Start from 4

consider 8 because 8>4

is 25>8 → true

So do nothing

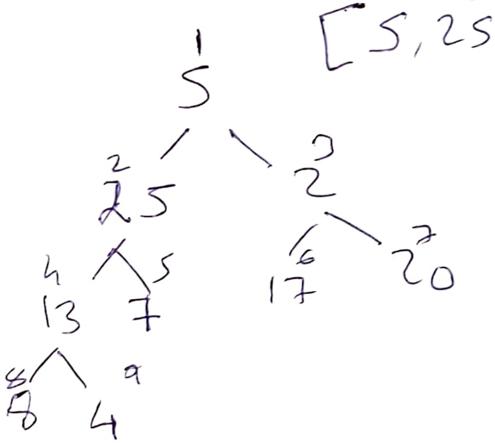
check - 3

for left node → consider 25 since 25 > 7 :-

is ~~25~~ 13 > 25 → False

∴ Swap 13 and 25

$[5, \overset{2}{25}, \overset{3}{2}, \overset{4}{13}, \overset{5}{7}, \overset{6}{17}, \overset{7}{20}, \overset{8}{8}, \overset{9}{4}]$



is 13 greater than its child nodes → true

So do nothing

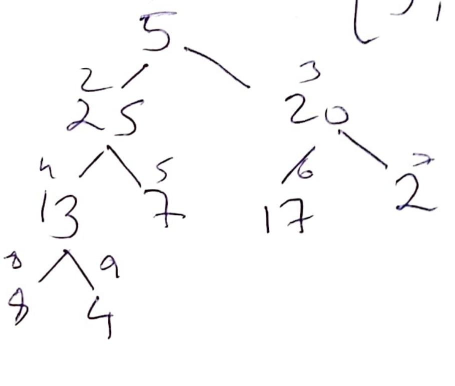
for right node → is 2 > 20 → False

consider 20 (20>17)

∴ Swap 2 and 20

$[\overset{1}{5}, \overset{2}{25}, \overset{3}{20}, \overset{4}{13}, \overset{5}{7}, \overset{6}{17}, \overset{7}{2}, \overset{8}{8}, \overset{9}{4}]$

Now check - 2

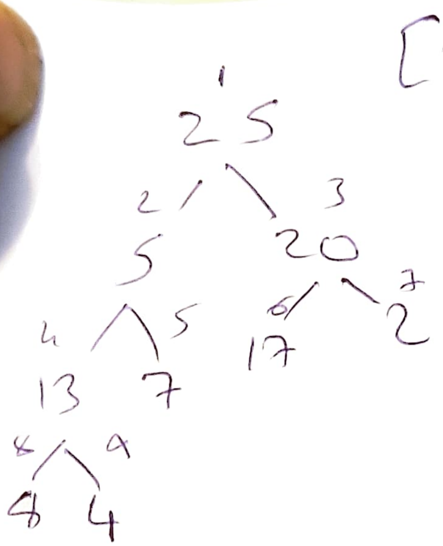


consider 25 since 25>20

is 5 > 25 → False

∴ Swap 5 and 25

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

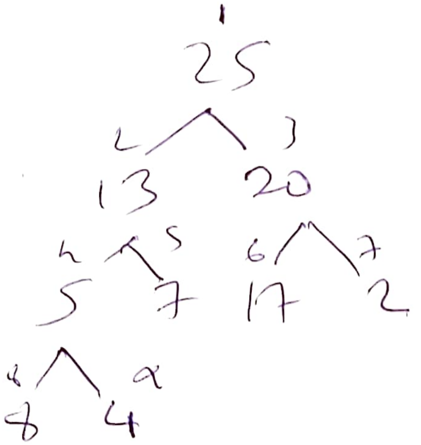


check if 5 is greater than its child nodes → False

5 > 13 → False

∴ Swap 5 and 13

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

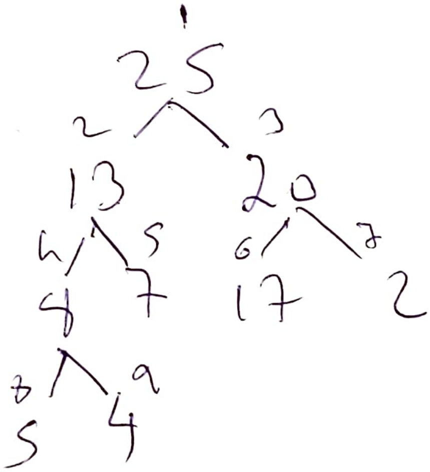


again check is 5 is greater than its child nodes → False

5 > 8 → False

∴ Swap 5 and 8

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

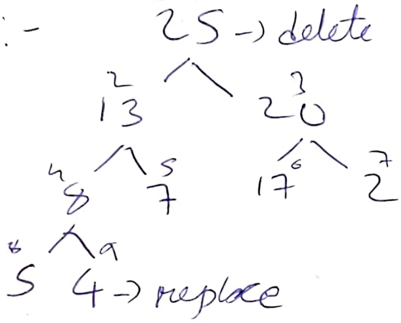


∴ This is the final Max heap built from bottomup.

To keep on deleting Max from Max heap:-

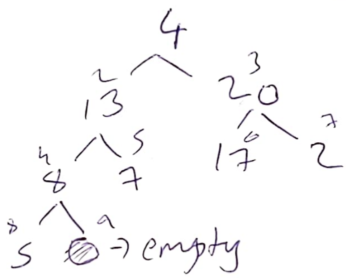
[25,13,20,8,7,17,2,5,4].

tree :-



Need to delete 25 Since its the Max. Replace 25 with the last element -> 4

[4,13,20,8,7,17,2,5]



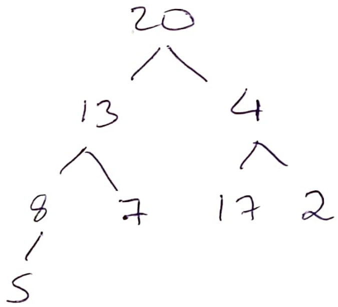
Now check if root is bigger than both its child nodes.

is $4 > 13$ -> false

is $4 > 20$ -> false

Since $20 > 13$, swap 4 and 20

[20,13,4,8,7,17,2,5]



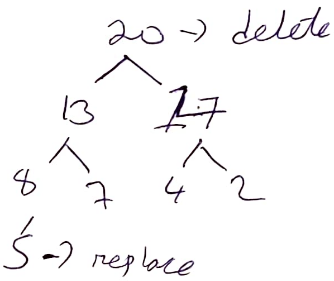
Again check is 4 is greater than its child nodes

is $4 > 17$ -> false

is $4 > 2$ -> true

Since 4 is lesser than 17, swap 4 and 17

[20,13,17,8,7,4,2,5]

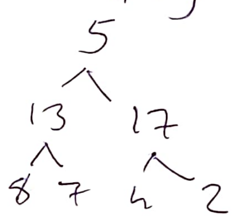


Now all elements are in heap. Next repeat the process.

delete Max -> 20

replace last node -> 5

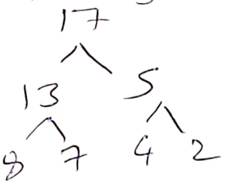
[5,13,17,8,7,4,2]



Since 5 is lesser than its child nodes -> $5 < 13$ and $5 < 17$

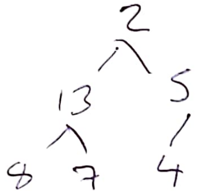
and $17 > 13$ -> Swap 5 & 17

[17,13,5,8,7,4,2]



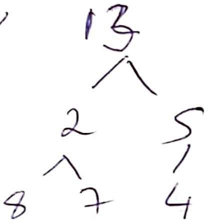
all Elements are in heap.
next remove 17
replace ~~8~~2 (farthest right node)

[2,13,5,8,7,4]



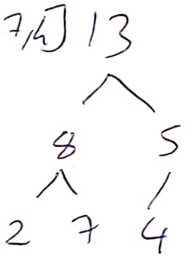
Since 2 is lesser than its child nodes and since 13 > 5,
Swap 2 and 13

[13,2,5,8,
7,4]



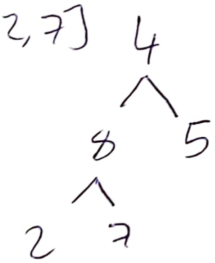
Since 2 is lesser than its child nodes and since 8 > 7,
Swap 8 and 2

[13,8,5,2,7,4]



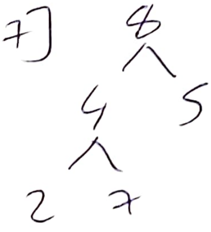
Now, delete 13 -> Max
replace by 4

[4,8,5,2,7]



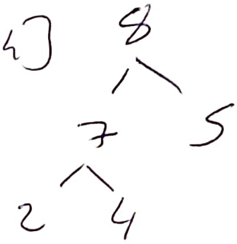
Since 4 is lesser than 8 (left children)
Swap 4 and 8

[8,4,5,2,7]



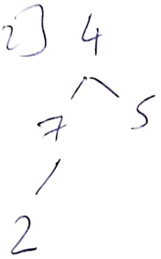
Since 4 is lesser than 7 (right Child)
Swap 4 and 7

[8,7,5,2,4]



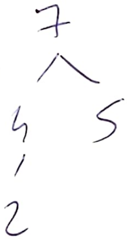
Now, delete 8 -> Max
replace 4

[4,7,5,2]



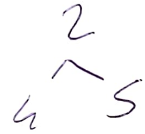
Since 4 is lesser than 7 and 7 > 5, swap 4 & 7

[7,4,5,2]



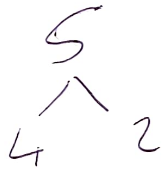
Now, delete 7 and replace by 2

[2,4,5]



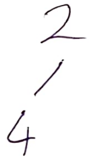
Since 2 is lesser than both its child nodes and since 5 > 4, swap 2 and 5

[5,4,2]



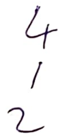
Now, delete 5 → max replace by 2

[2,4]



swap 2 and 4

[4,2]



delete 4 and replace by 2

[2]

delete 2 → max

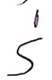
[ ] empty heap

∴ Successfully deleted max until maxheap is empty

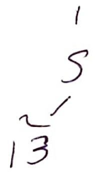
Build Minheap from Top-bottom

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

1
Insert [5]

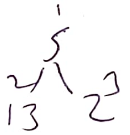


1 2
[5, 13]



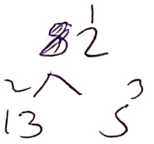
Don't do anything since 5 < 13 → true

[5, 13, 2]

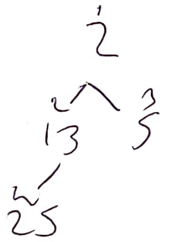


5 < 2 → False
right node is the smallest
∴ swap 5 and 2

[2, 13, 5]

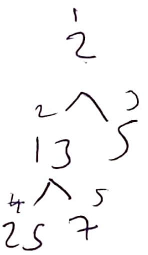


1 2 3 4
([2, 13, 5, 25]



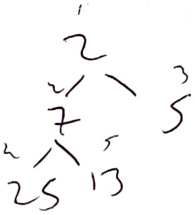
Don't do anything
13 < 25 → true

1 2 3 4 5
[2, 13, 5, 25, 7]

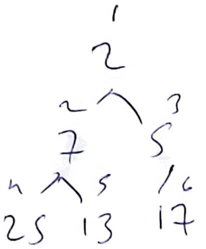


Since 7 is lesser than 13
Swap 13 and 7

1 2 3 4 5
[2, 7, 5, 25, 13]

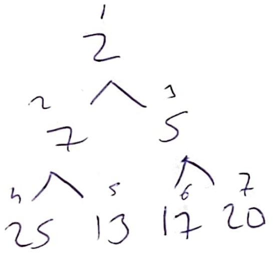


[2,7,5,25,13,17]



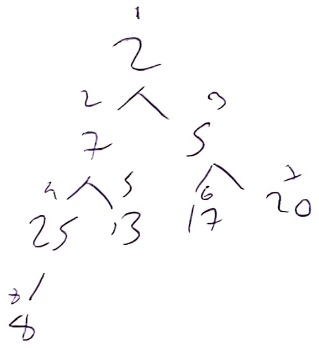
Don't do anything
Since 5 is lesser than
17

[2,7,5,25,13,17,20]



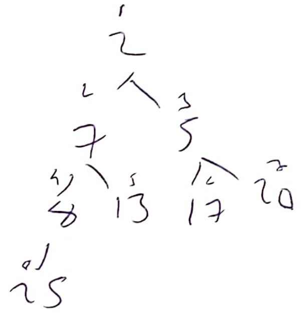
Don't do anything since
5 < 17 and 5 < 20

[2,7,5,25,13,17,20,8]



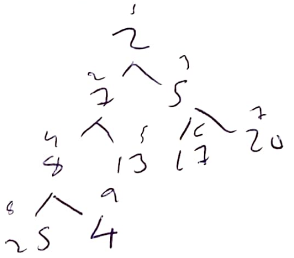
Since 8 is lesser than 25
swap 8 and 25

[2,7,5,8,13,17,20,25]

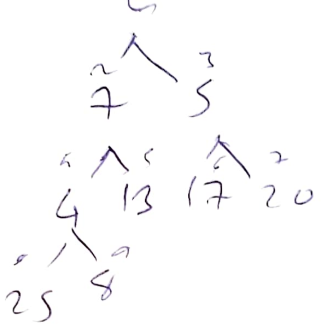


Don't do anything since
7 < 8 and 7 < 13

[2,7,5,8,13,17,20,25,4]

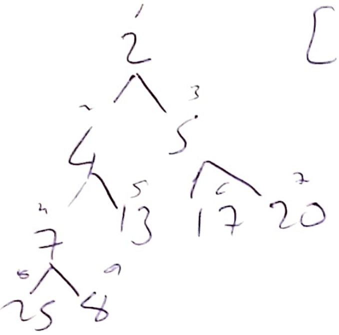


Since 4 is lesser than 8
swap 4 and 8



Since $4 < 7$, Swap 4 and

[2, 7, 5, 4, 13, 17, 20, 25, 8]



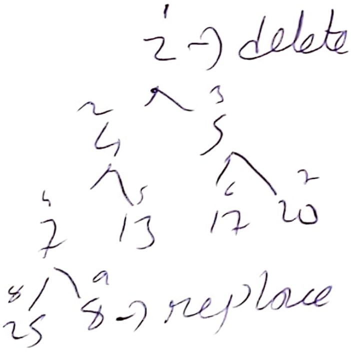
[2, 4, 5, 7, 13, 17, 20, 25, 8]

Don't do anything since $2 < 4$ and $2 < 5$

∴ Successfully Built Min heap

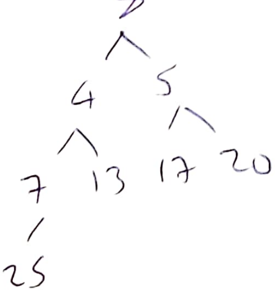
To delete Min from Min heap

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

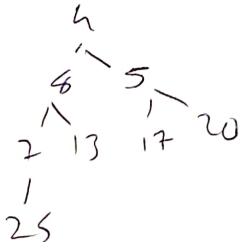


delete 2 -> Min
replace it by last node -> 8

[8, 4, 5, 7, 13, 17, 20, 25]

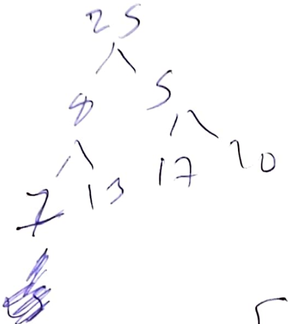


Since $8 > 4$ and $8 > 5$
and since $4 < 5$ ->
swap 4 and 8



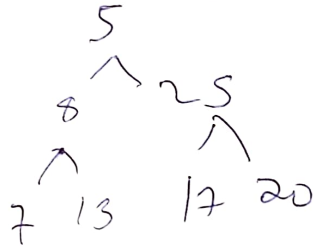
Now delete 4 -> Min
replace last node -> 25

[25, 8, 5, 7, 13, 17, 20]



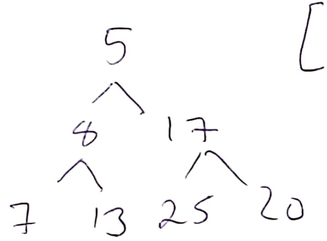
Since 25 > 8 and 25 > 5
and Since 5 < 8,
Swap 5 and 25

[5, 8, 25, 7, 13, 17, 20]



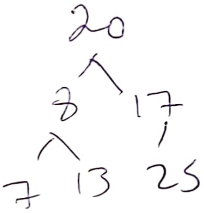
Since 25 > 17 and 25 > 20
Since 17 < 20, Swap 17 & 25

[5, 8, 17, 7, 13, 25, 20]



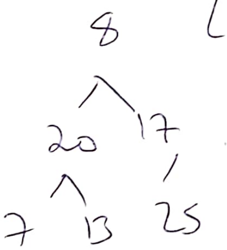
~~Don't~~ Now delete 5 → Min
replace by 20 → last node

[20, 8, 17, 7, 13, 25]



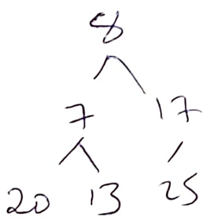
Since 20 > 8 and 20 > 12
Since 8 < 17, Swap 20 & 8

[8, 20, 17, 7, 13, 25]



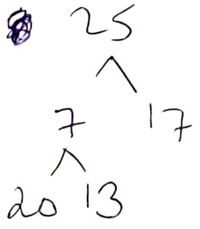
Since 20 > 7 and 20 > 13
Since 7 < 13, Swap 20 & 7

[8, 7, 17, 20, 13, 25]



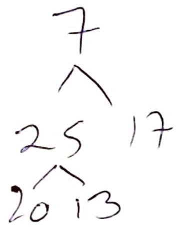
Now delete 8 → Min
replace by 25 → last node

[25, 7, 17, 20, 13]



Since 25 > 7 & 25 > 17
Swap 25 and 7

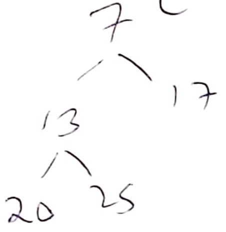
[7, 25, 17, 20, 13)



Since 25 > 20 & 25 > 13

Since 13 < 20, swap 25 & 13

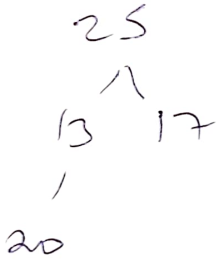
[7, 13, 17, 20, 25]



Now delete 7 → Min

replace by → 25 → last node

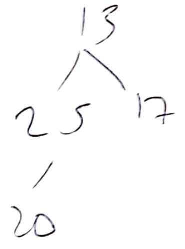
[25, 13, 17, 20)



swap 13 and 25

Since 25 > 13

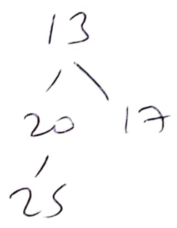
[13, 25, 17, 20)



swap 25 and 20

Since 25 > 20

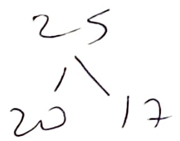
[13, 20, 17, 25)



Now delete 13 → Min

replace by 25 → last node

[25, 20, 17]

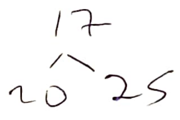


Swap 17 and 25

Since 25 > 17

It is least

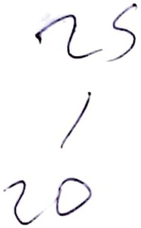
{17, 20, 25)



Now delete 17 → Min

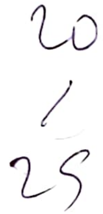
replace by 25

(25, 20)



Swap Since 20 < 25

(20, 25)



delete 20 -> Min

replace 25 -> last node

[25]

-> delete 25 -> Min

[ ] 0 -> empty node

∴ Successfully deleted Min from Minheap until empty