Design & Analysis of Algorithms

NAME: Muhammad Ahmed Amjad

CLASS: BSIT - 4B

ENROLLMENT: 01-135232-051

SUBMITTED TO: Maam MADIHA JAVEED

ASSIGNMENT - 1

Sort the following array using Insertion, Merge and Heap Sorts.

0	1	2	3	4	5	6	7	8	9	10	11	12
55	6	89	3	34	65	71	23	57	92	46	100	0

Insertion Sort

starting from Index 1, as 55 has no element on its left, so its sorted.

55	6	89	3	34	65	71	23	57	92	46	100	0

FIRST CYCLE

55 →	89	3	34	65	71	23	57	92	46	100	0

	55	89	3	34	65	71	23	57	92	46	100	0

KEY "6"

int temp [6] → key

SECOND CYCLE

6	55		3	34	65	71	23	57	92	46	100	0

KEY "89"

int temp [89]

→ As 55 < 89 so no shift.

→ 89 will go back to its original position

THIRD CYCLE

6	55	89	34	65	71	23	57	92	46	100	0

AS 3 < 89 , 3 < 55 , 3 < 6 SO key "3" will shift to 0 index.

	6	55	89	34	65	71	23	57	92	46	100	0

int temp 3

key "3"

4th CYCLE

3	6	55	89	65	71	23	57	92	46	100	0

key "34"

- 34 < 89
- 34 < 55

but

34 > 6 and 34 > 3

3	6	34	55	89	65	71	23	57	92	46	100	0

int temp 34

5th CYCLE

3	6	34	55	89	71	23	57	92	46	100	0

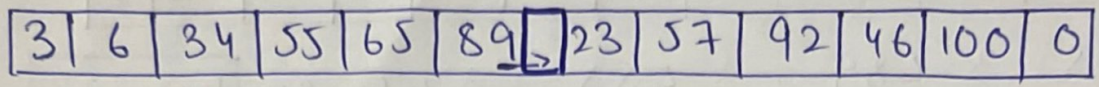
key "65"

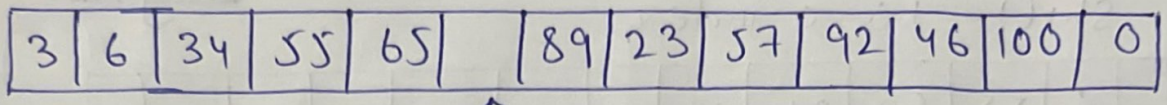
- 65 < 89 but 65 > 55

3	6	34	55		89	71	23	57	92	46	100	0

int temp 65

6th CYCLE



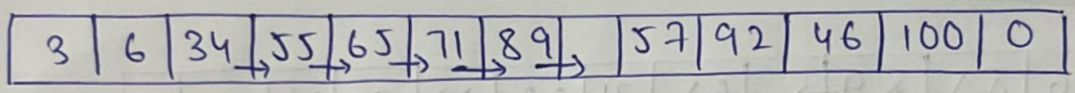


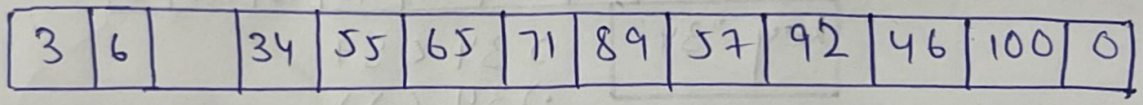
Key "71"

int temp 71

∴ 71 < 89 but greater than 65, 55, 34 & 3.

7th CYCLE

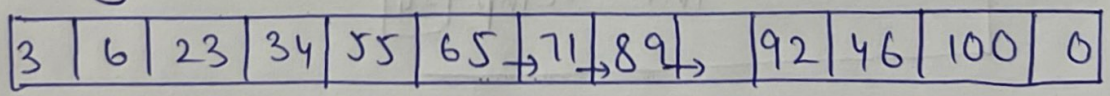


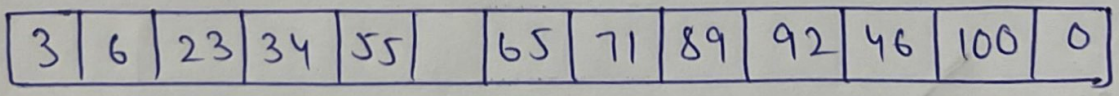


int temp 23

Key "23"

8th Cycle

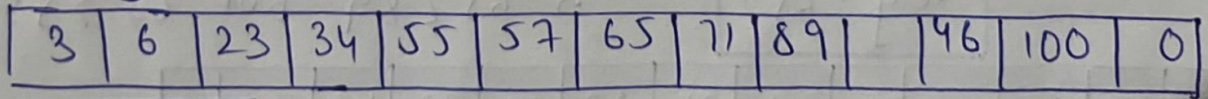




int temp 57

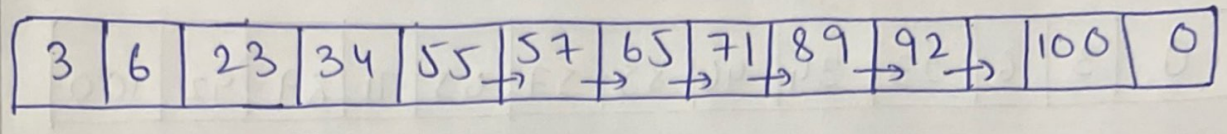
Key "57"

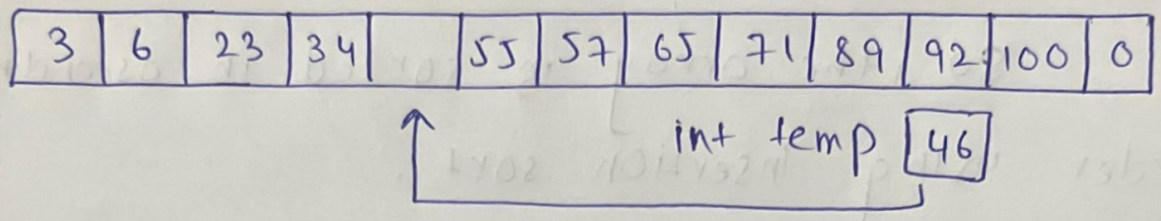
9th Cycle



Key "92" is greater than every element to its left, so it's in correct position and will remain there.

10th Cycle

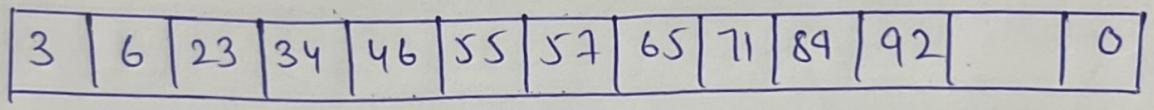




key "46"

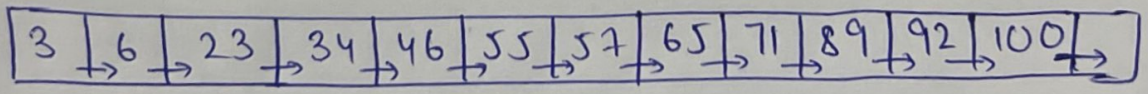
46 < 92
(shifted to right)
46 < 89 , 46 < 71
46 < 65 , 46 < 57 , 46 < 55
(shifted to right)

11th Cycle

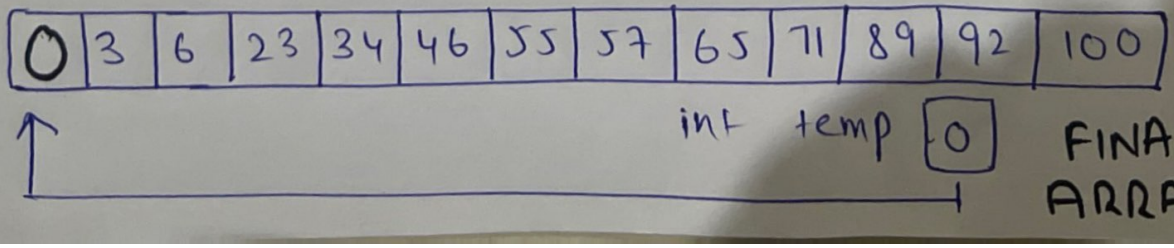


key "100" > 92
(element will not shift)

12th cycle



key "0" is less than every element to its left. So it will shift to left-most index.

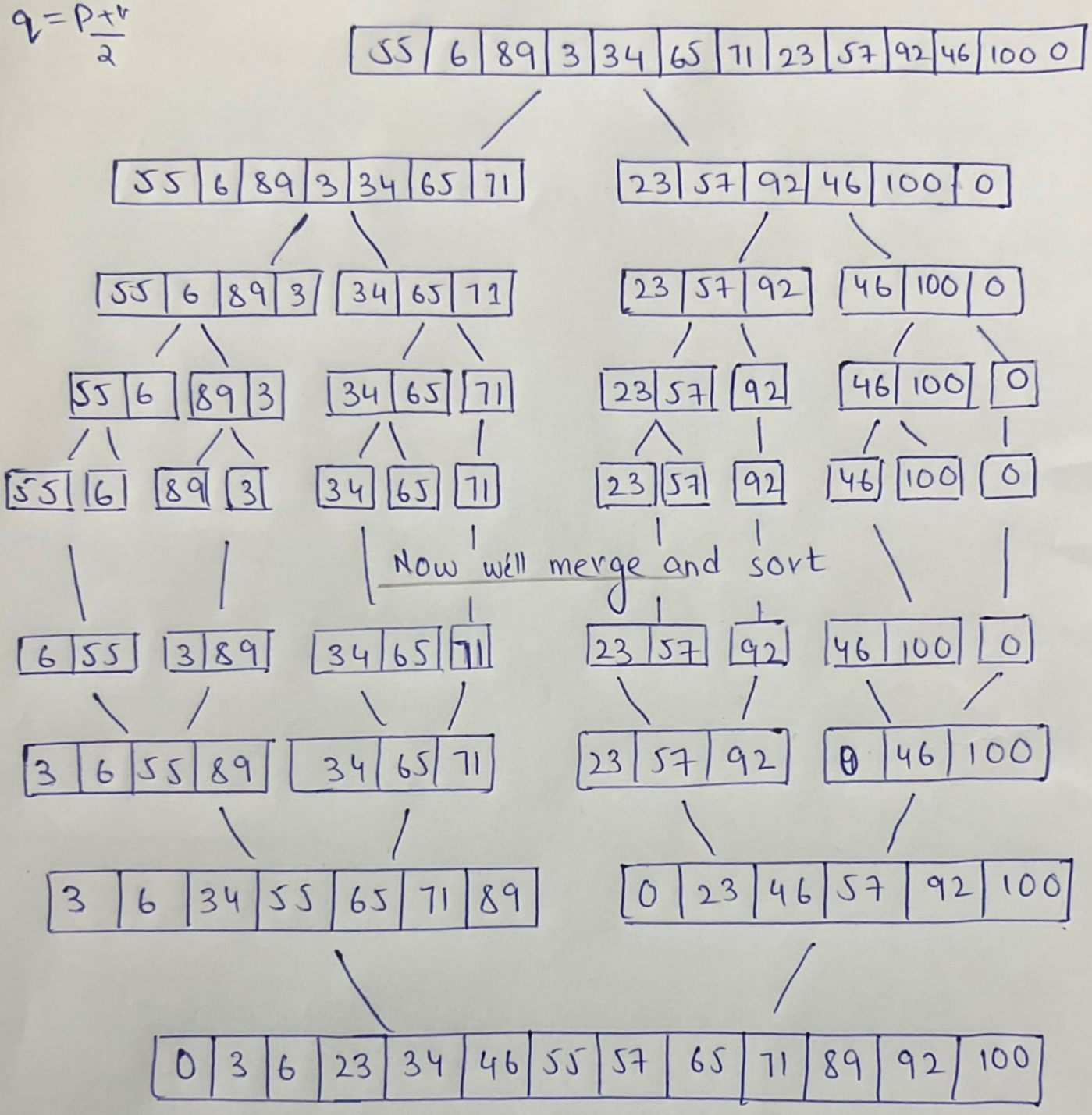


0	3	6	23	34	46	55	57	65	71	89	92	100

So, the array is sorted in ascending order using Insertion sort.

MERGE Sort

$q = \frac{p+r}{2}$

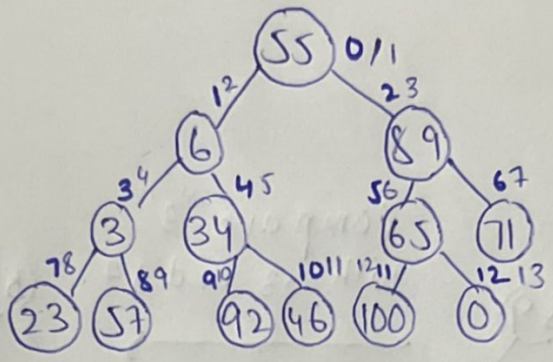


It work on divide and conquer technique, to first split then remerge the array.

HEAP SORT

55	6	89	3	34	65	71	23	57	92	46	100	0

Firstly make a tree using the above array



start from index 0 or index 1.

start Max-heapify in bottom up manner

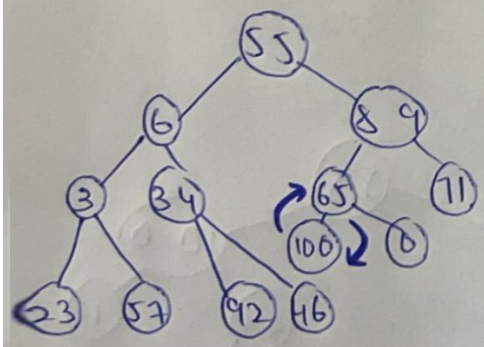
lenght [A] /2 = 13/2 = 6

index 6 where 65 is stored.

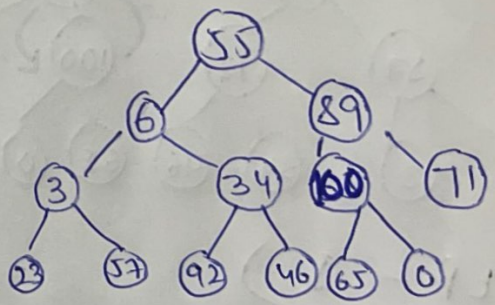
↓ or

start from the lowest right level

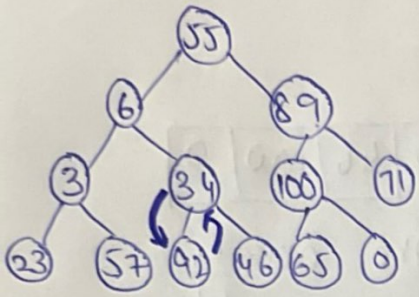
STEP-1



first compare 65 with 100 and 0.

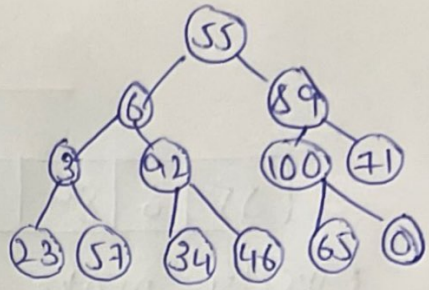


STEP-2

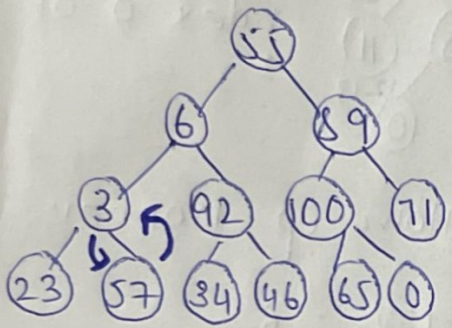


Compare 34 with 92 and 46.

As 92 is greater than 34 it will be swapped.

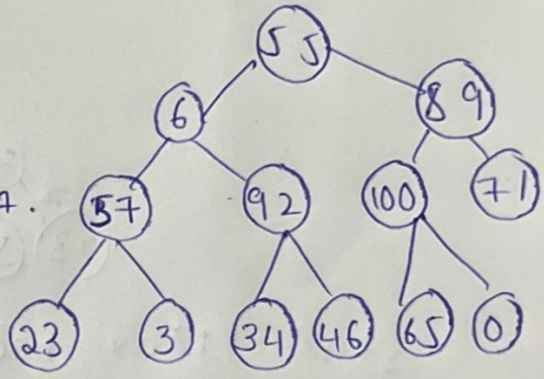


STEP - 3

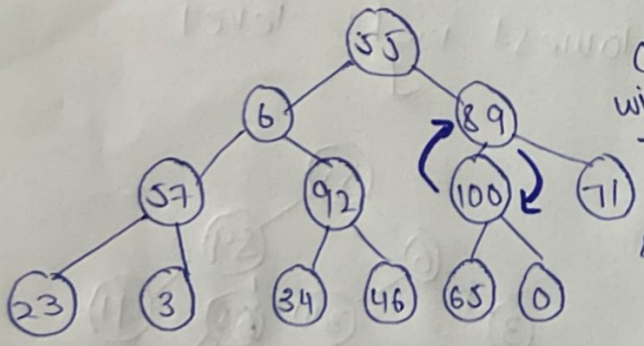


Compare "3" with 23 and 57.

swap "3" with "57".

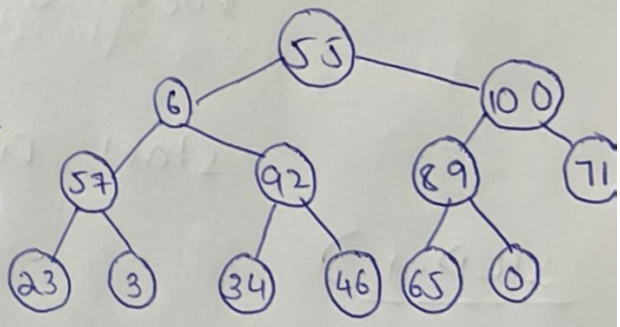


STEP - 4

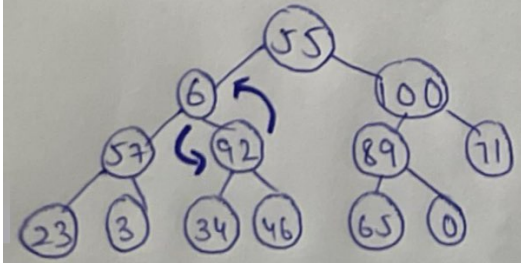


Compare "89" with 100 and 71

swap 89 with 100

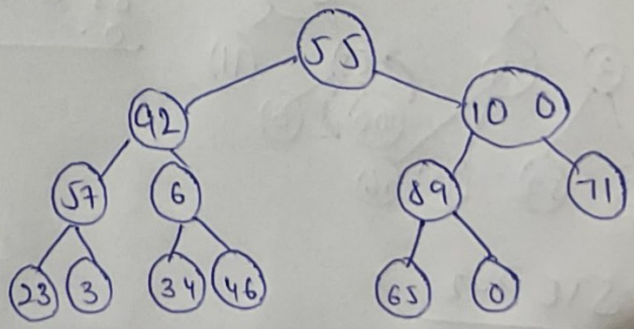


STEP - 5

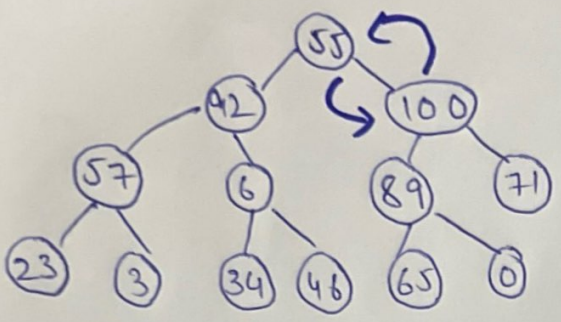


Compare "6" with 57 and 92

swap 6 with 92.

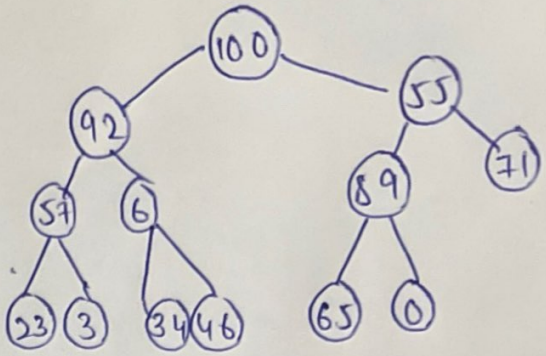


STEP – 6)

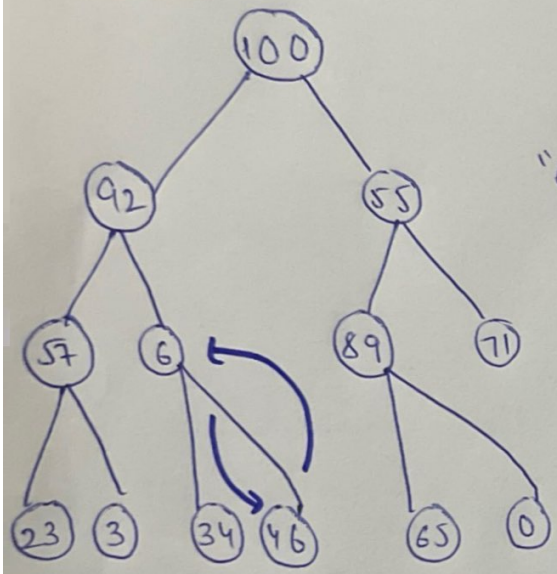


compare "55" with 92 & 100.

swap 55 and 100.

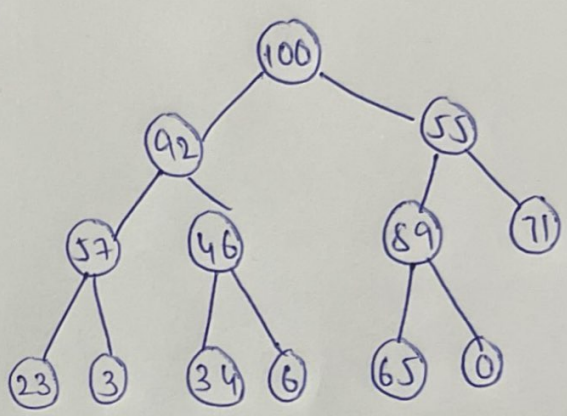


STEP – 7)

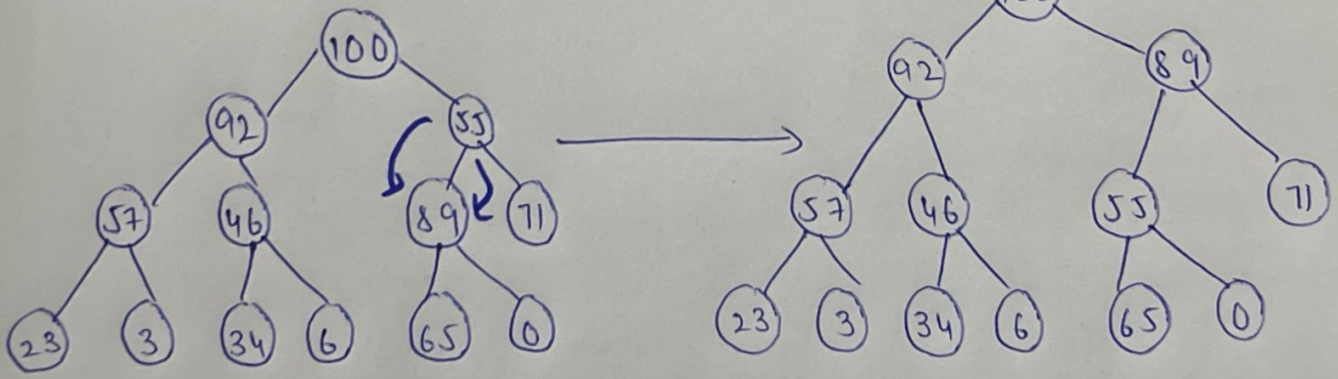


compare "6" with 34 and 46.

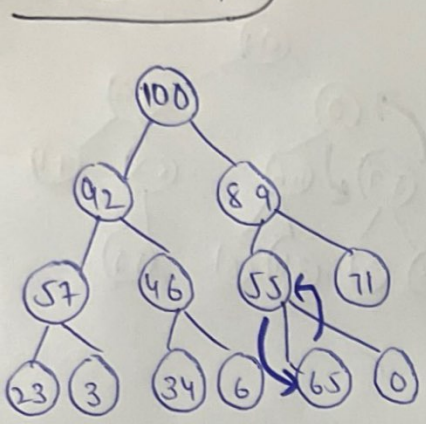
swap 6 with 46.



STEP – 8)

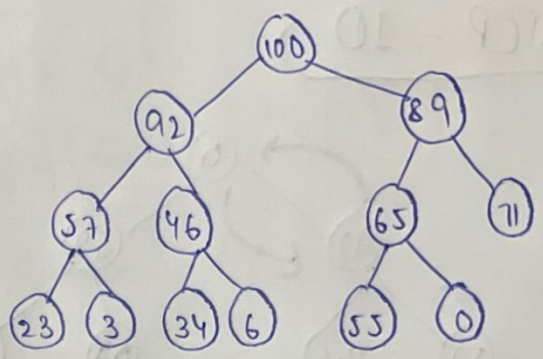


STEP - 9)



compare 55 with 65 and 0

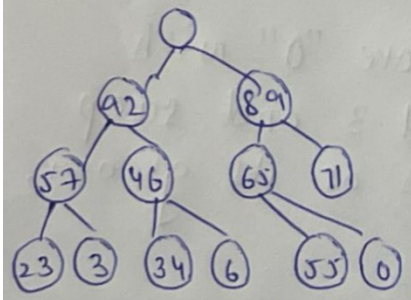
swap 55 with 65



So we have build max-heapify now we'll sort the array using max heapify. We'll swap the first (also the largest) with the last leaf node and work top to bottom.

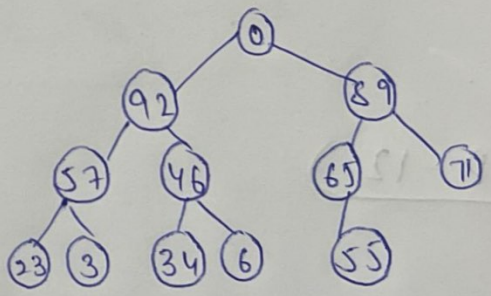
Sorting step-1)

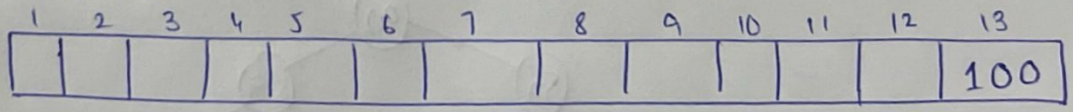
swap "100" with "0"



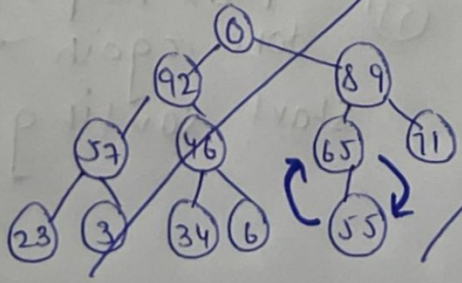
lowest right element will replace 100.

which is "0"



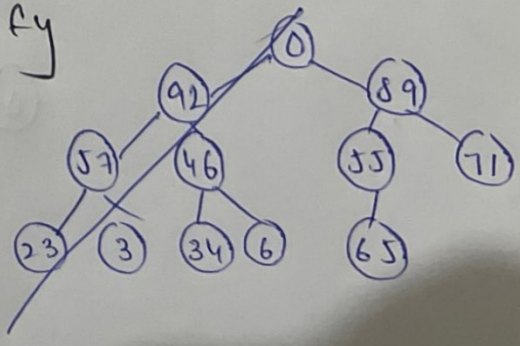


STEP - 10) Again make Max-heapify

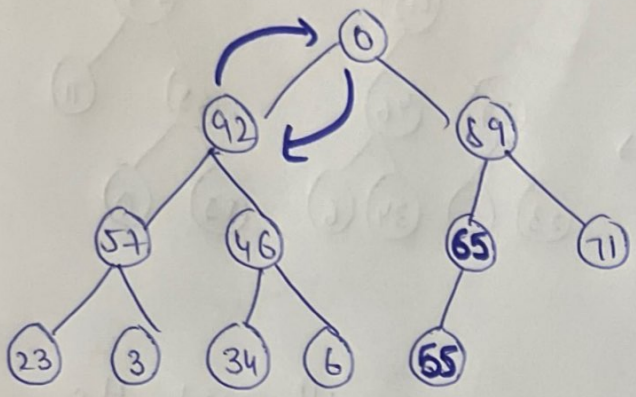


compare "55" with 65

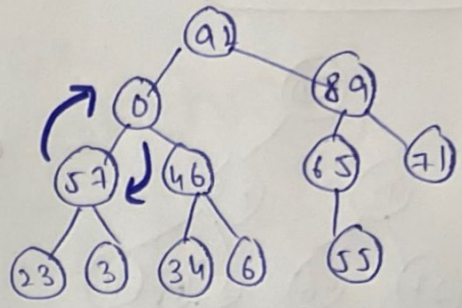
As 55 < 65 swap 55 with 65.



STEP - 10)

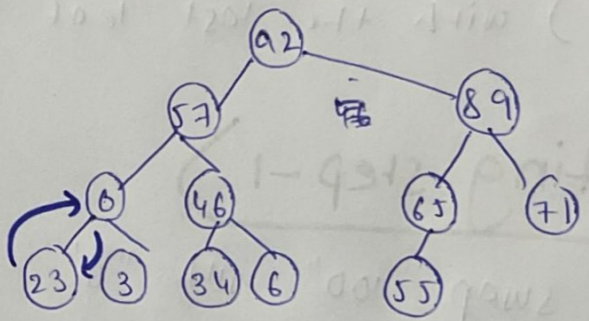


as 0 is less than 92 swap both.



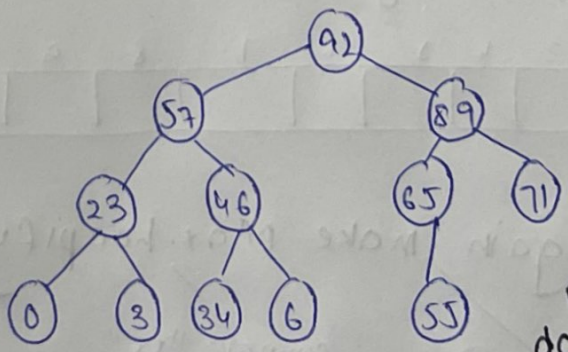
compare "0" with 57 and 46. as 57 is greater swap 0 with 57.

STEP - 11)



STEP - 12)

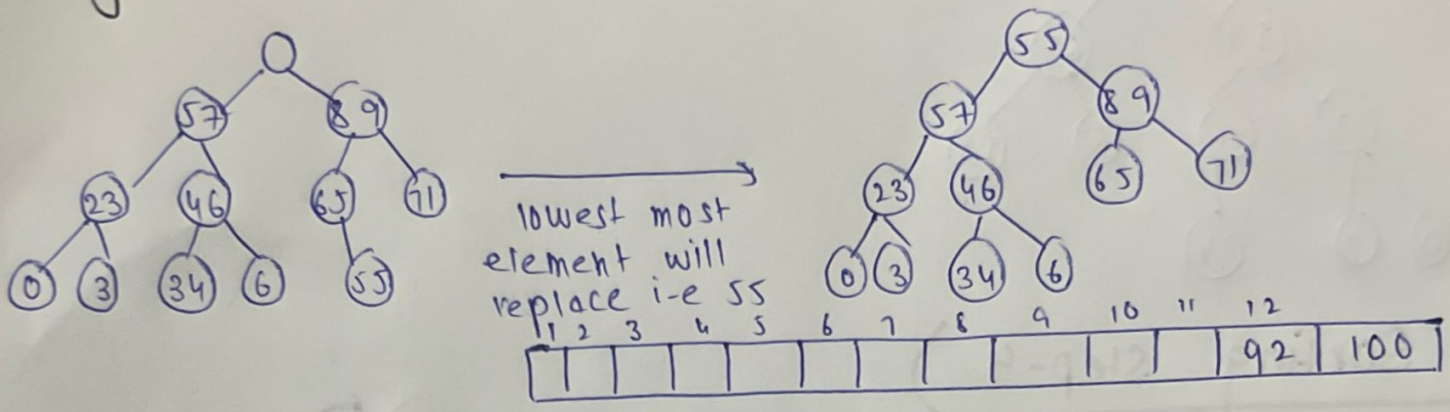
compare "0" with 23 and 3 and swap with 23 as its greater.



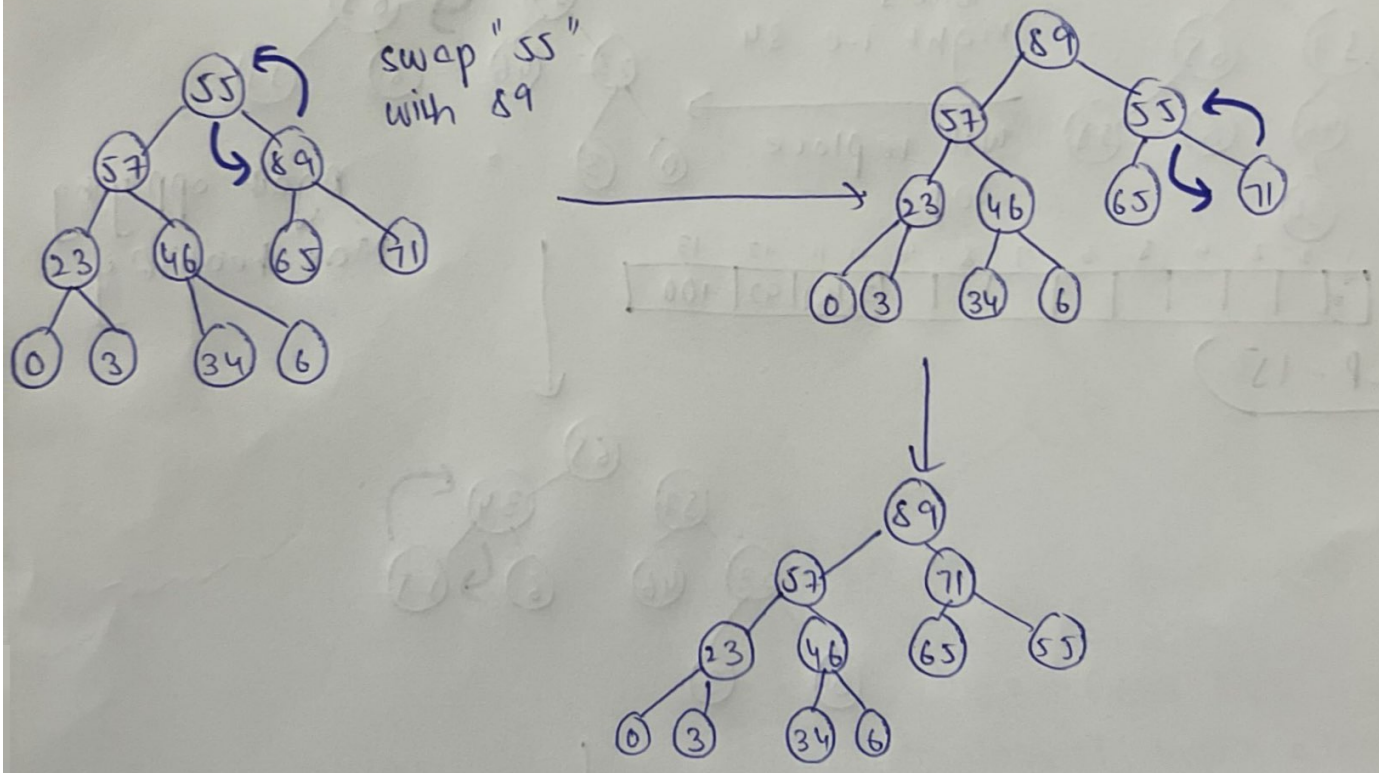
Max-heapify done, again start sorting.

Sorting step - 2

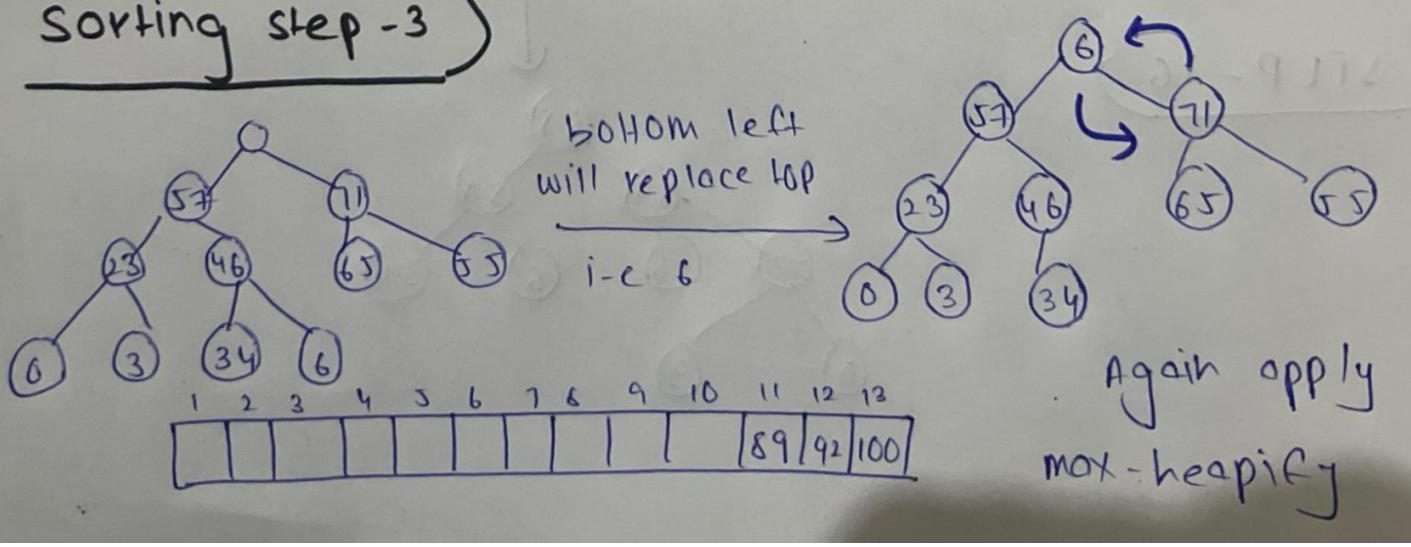
Again delete from top i.e 92



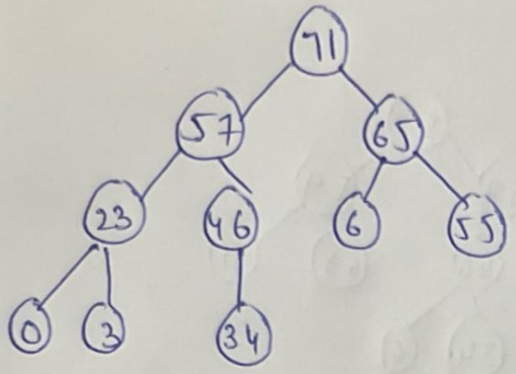
STEP-13 Making Max-heapify again



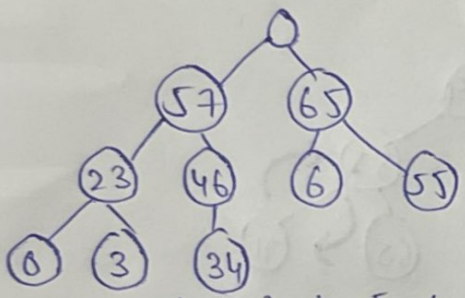
Sorting step - 3



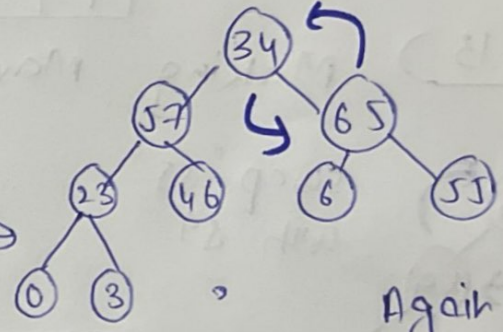
STEP - 14



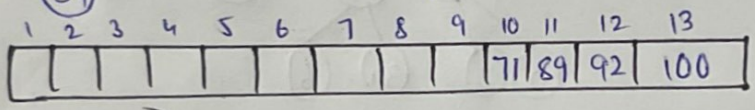
Sorting step-4



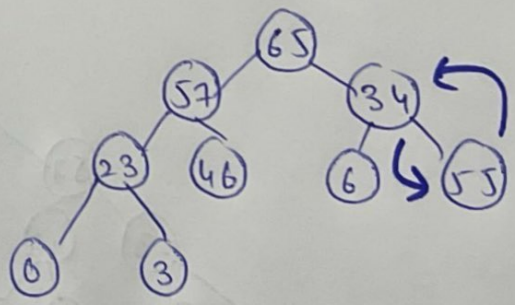
lowest right i-e 34 will replace top



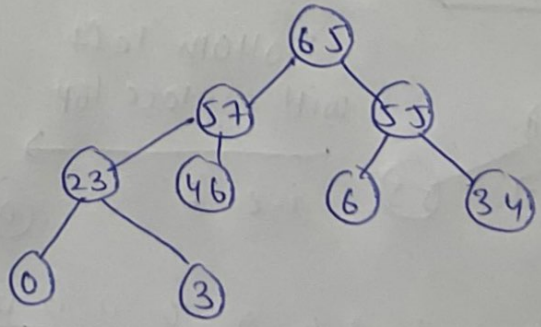
Again applying Max-heapify



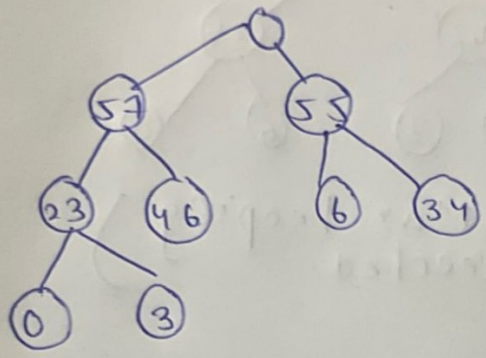
STEP - 15



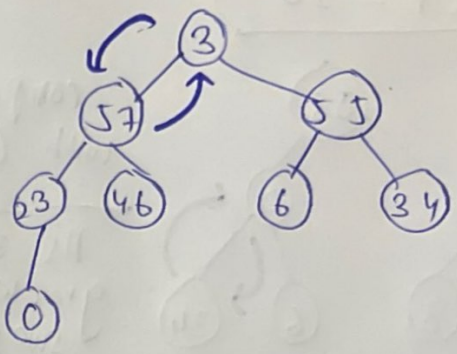
STEP - 16



SORTING STEP - 5



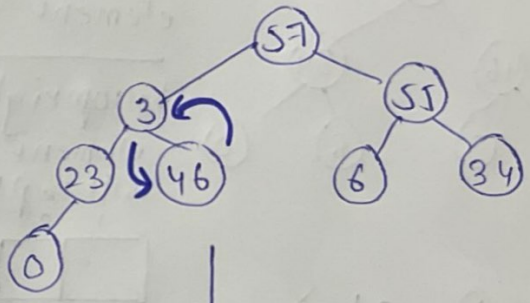
lowest right i-e 3 will replace the top element.



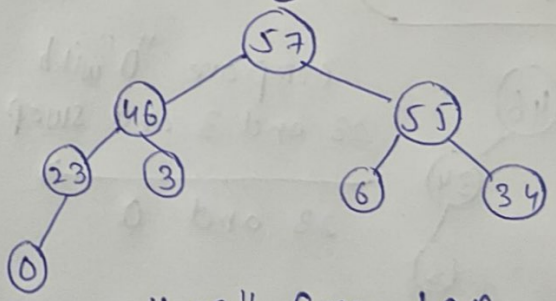
Again making max heapify:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
									65	71	89	92	100

STEP - 17



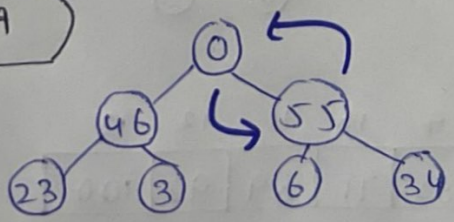
STEP - 18



Remove "57" from top and the lowest right element i-e 0 will replace it.

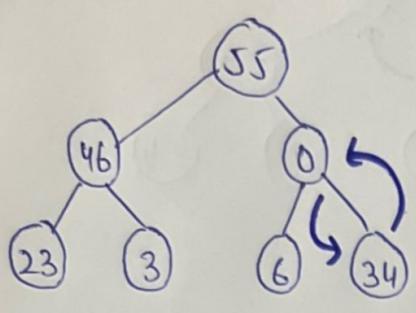
1	2	3	4	5	6	7	8	9	10	11	12	13
							57	65	71	89	92	100

STEP - 19



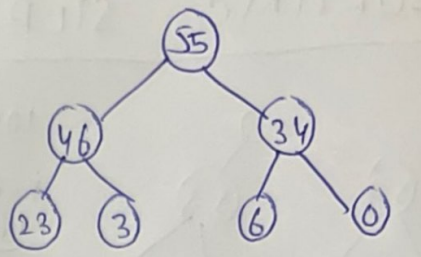
swap "0" with 55. as 55 is greater than "0" and "46".

STEP – 20



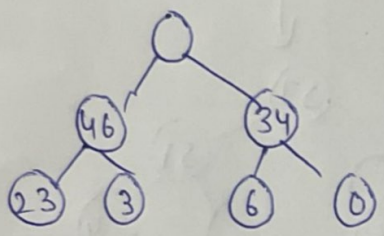
compare "0" with 6 and 34.

As 34 is greater then swap 0 and 34.



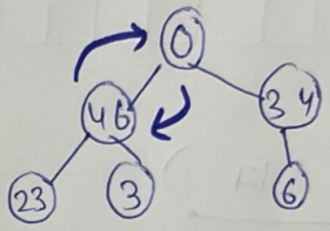
(Max-heapify reached)

Sorting step – 6



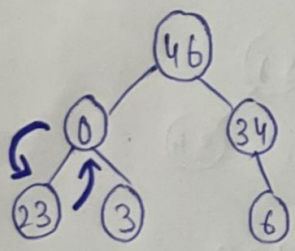
remore top-most element i-e 55

bottom-right element i-e 0 will replace it



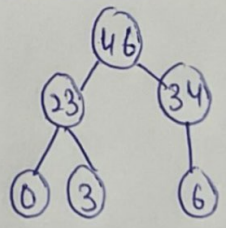
1	2	3	4	5	6	7	8	9	10	11	12	13
							57	65	71	89	92	100

STEP – 21

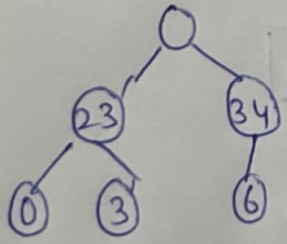


compare "0" with 23 and 3 and swap

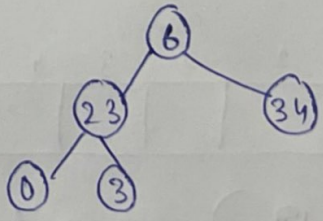
23 and 0



Sorting step – 7

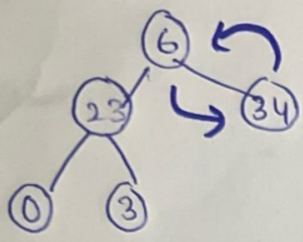


remore top-most element i-e 46 and "6" will replace it.



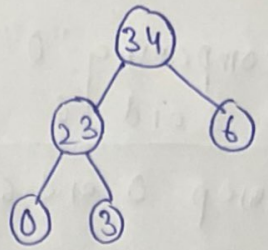
1	2	3	4	5	6	7	8	9	10	11	12	13
						46	57	65	71	89	92	100

STEP – 22)



compare "6" with 23 & 34.

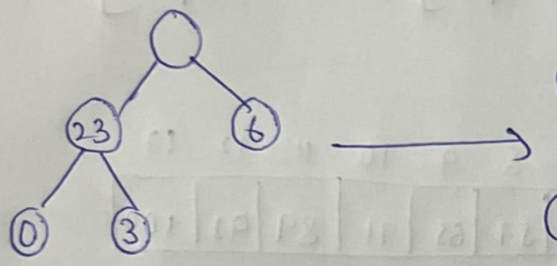
swap "6" with 34 as it's larger.

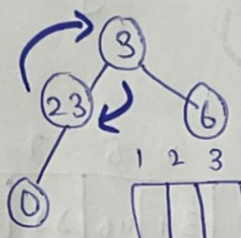


MAX-HEAPIFY REACHED

Sorting - step 8)

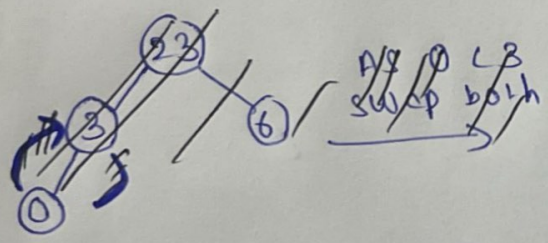
Remove top element i-e 34 and lowest right element i-e 3 will replace it.

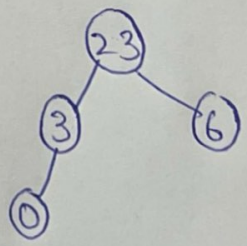




1	2	3	4	5	6	7	8	9	10	11	12	13
				34	46	55	57	65	71	89	92	100

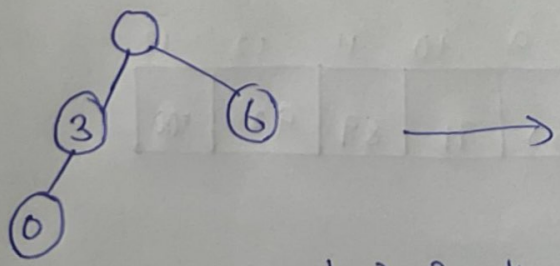
STEP – 23)

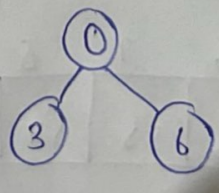




Sorting step-9)

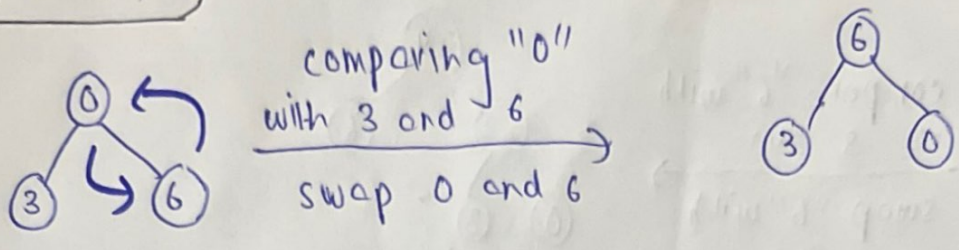
Remove top-most i-e 23, and left-right "0" will replace it





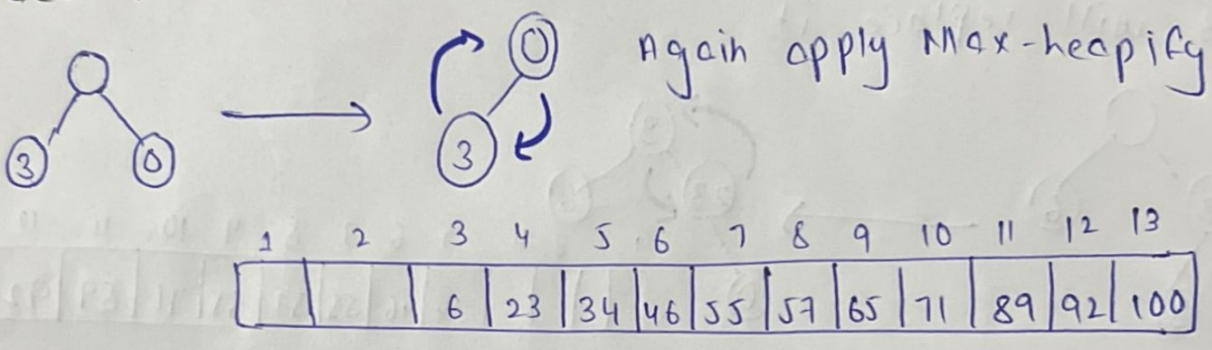
1	2	3	4	5	6	7	8	9	10	11	12	13
			23	34	46	55	57	65	71	89	92	100

STEP – 24)



Sorting step – 10)

Remove topmost element i-e 6 and bottom-right "0" will replace it.



1	2	3	4	5	6	7	8	9	10	11	12	13
		6	23	34	46	55	57	65	71	89	92	100

STEP – 25)

Sorting step – 11)

Remove top i-e 3 and as 0 is the last element. place it in the array too.

1	2	3	4	5	6	7	8	9	10	11	12	13
0	3	6	23	34	46	55	57	65	71	89	92	100

Array is sorted.