

Heap & its Identification

Identification

$\rightarrow k$
 $\rightarrow \text{smallest / Largest}$
} Heap

Types of Heap

Min Heap

$k + \text{Largest}$

\downarrow

Means min value
- h to de

Max Heap

$k + \text{Smallest}$

\downarrow

Means maximum
value h to de

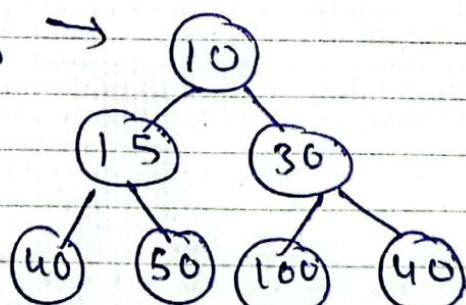
• Binary Heap is Complete Binary Tree

$$\begin{aligned}
 \text{left}(i) &= 2i + 1 \\
 \text{right}(i) &= 2i + 2 \\
 \text{parent}(i) &= \left\lfloor \frac{i-1}{2} \right\rfloor
 \end{aligned}$$

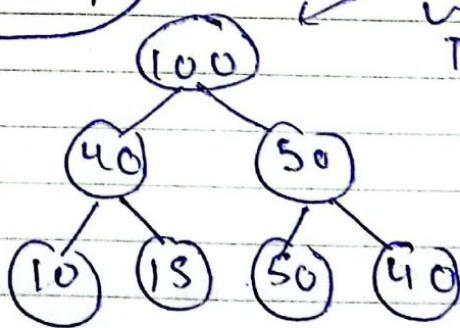
0 based
Indexing

maximum
value at
Top.

minimum
value
at top



MIN
Heap



MAX
Heap

33	14	12	10	11	10	12	14
34	21	22	23	24	25	26	27
35	28	29	30	31	30	25	26

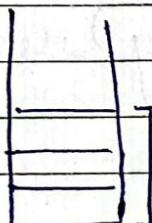
sunday |

UZ

08.00 Heap ke Q \rightarrow sare sorting Q hote hain



(n log n) TC

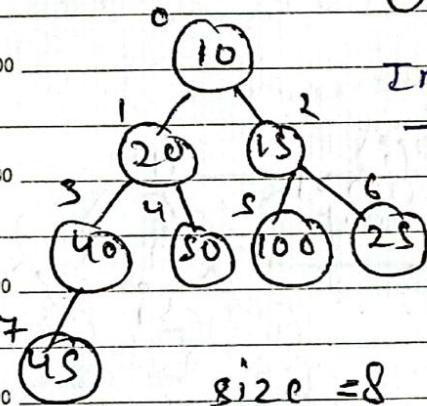


$n \log k$

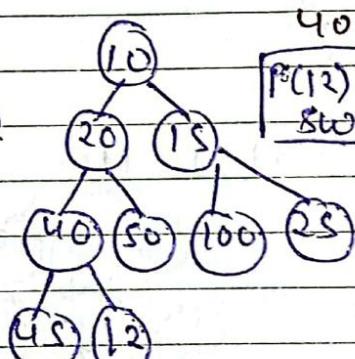
Heap ke mostly Q hame sorting hase hote hain

12.00 01.00 1) Basic solution of Heap Problem is
Sorting.

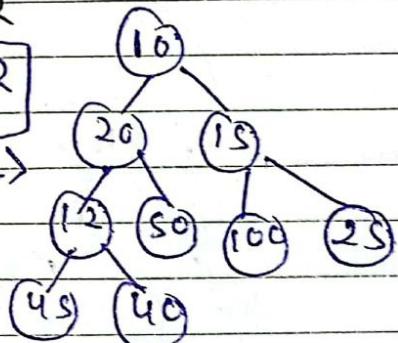
Binary Heap Insert



Insert $\rightarrow 12$



$P(12) > 12$
swap



size = 8

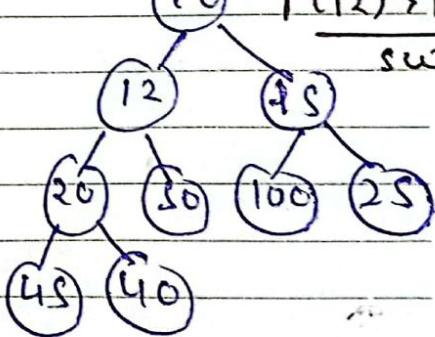
size ++
 $currentSize - 1 = 7$

$20 > 12$
 $P(12) > 12$
swap

Stop Condition

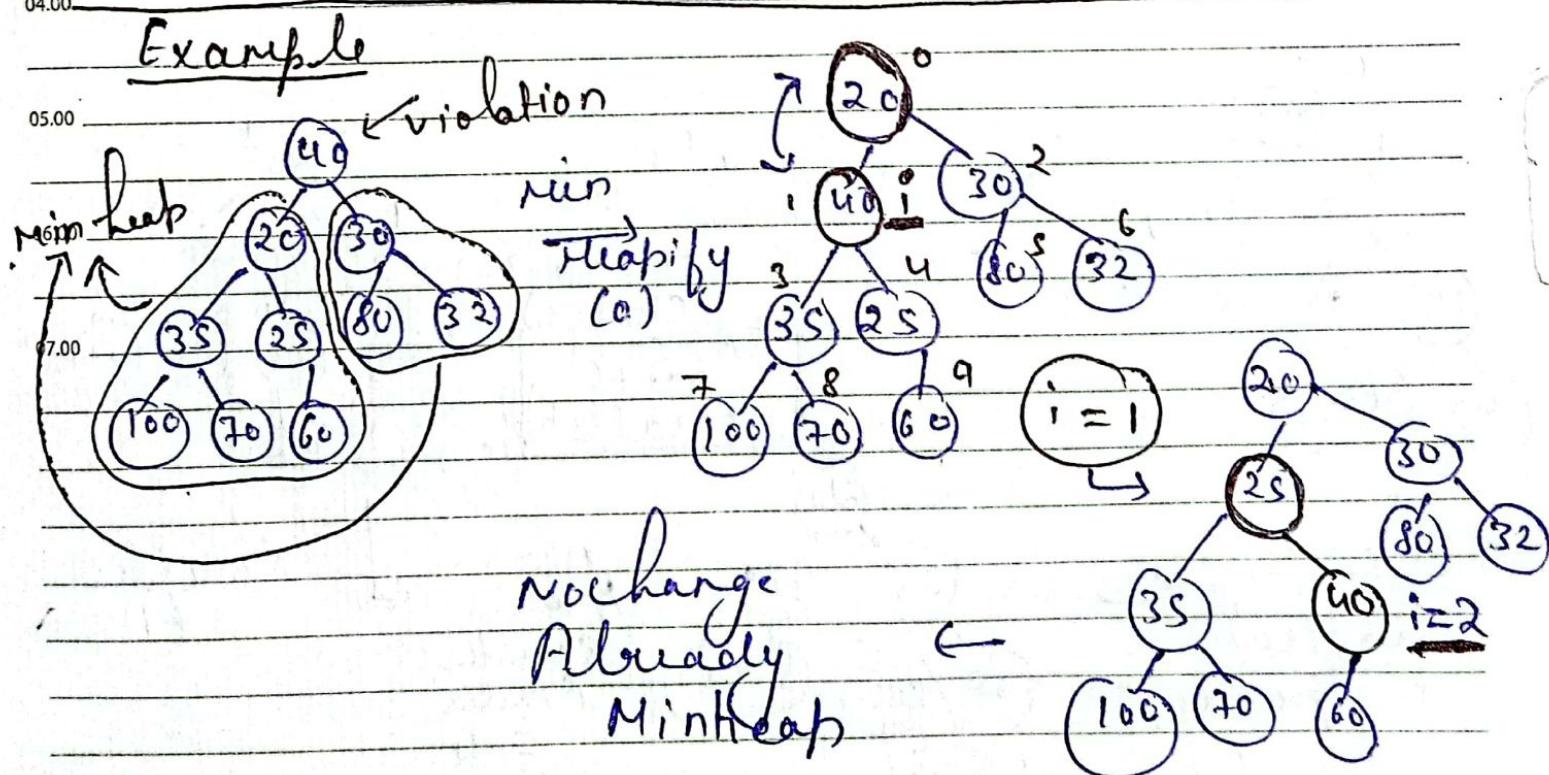
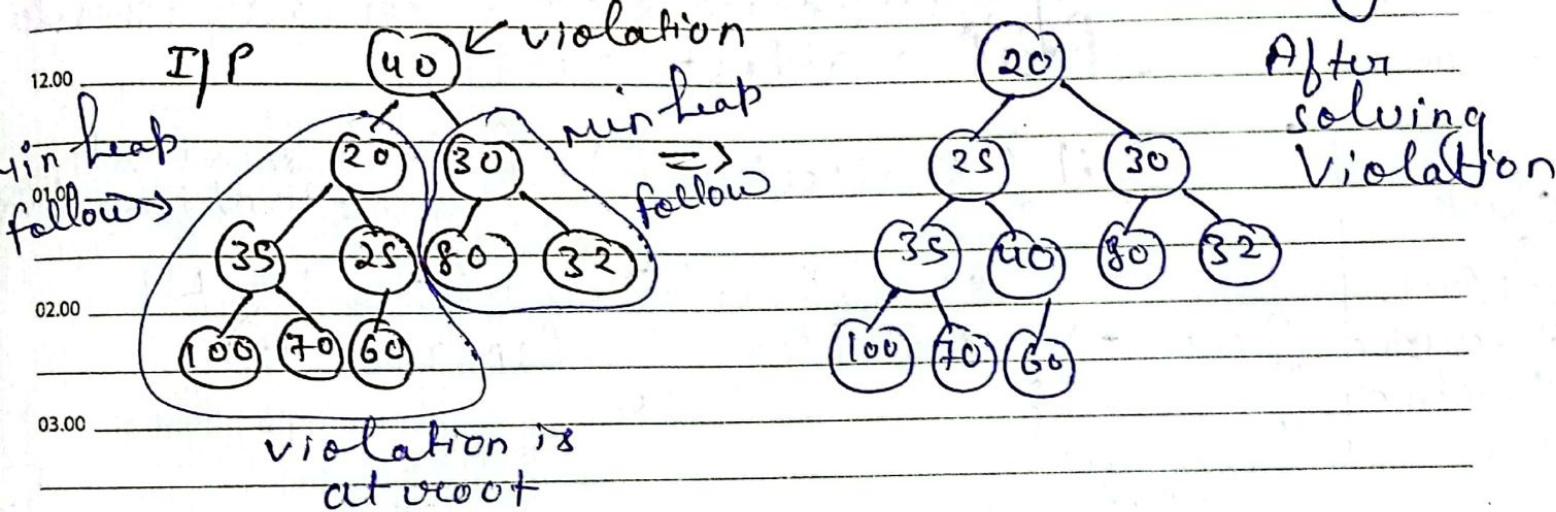
\rightarrow if Parent < current node

\rightarrow current node now at root.



Binary Heap (Heapify)

- Mainly used as subroutine in many standard operation.
- Given min/max heap with one possible violation. Fix the min/max Binary Heap.



05

2017
july
wednesday

June 2017		July 2017											
SUN	MON	TUE	WED	THU	FRI	SAT	SUN	MON	TUE	WED	THU	FRI	SAT
22	23	24	25	26	27	28	29	30	31	1	2	3	4
5	6	7	8	9	10	11	12	13	14	15	16	17	18
12	13	14	15	16	17	18	19	20	21	22	23	24	25
19	20	21	22	23	24	25	26	27	28	29	30	24	25
26	27	28	29	30	27	28	29	30	24	25	26	27	28

Algo void ~~MinHeapify~~ MinHeapify (int i)

08.00

} { int li = left(i), ri = right(i)
int smallest = i

find
the
minimum
value

→

if (li < size && arr[li] < arr[smallest])
smallest = li

11.00

if (ri < size && arr[ri] < arr[smallest])
smallest = ri

12.00

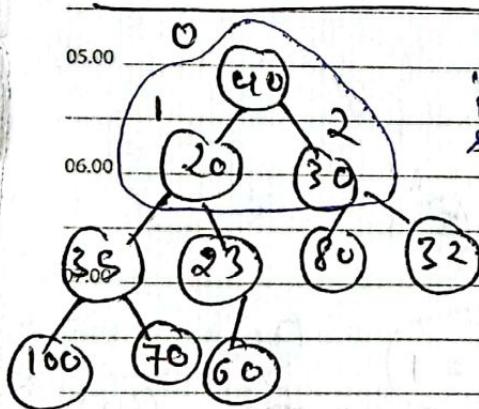
} if (smallest != i) (check smallest
score to non leaf index)

01.00

check
call
Recursive
Function
Again → MinHeapify (smallest)
Call
with
smallest
value

02.00

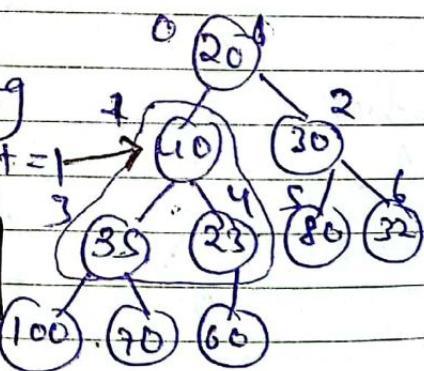
04.00



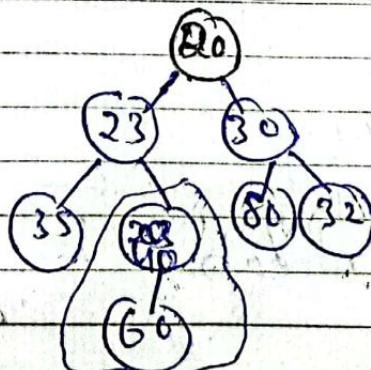
i=0
smallest=0.

After
processing
smallest=1
(index)

(Lab call
with
smallest)



All satisfy
min Heap
Property.



After
process
smallest
(index)
= 4

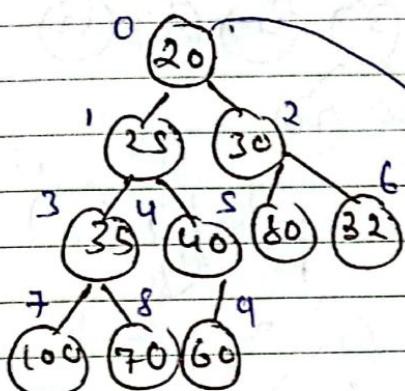
31	1	2	3	4	5	6	35	1	4	3
32	7	8	9	10	11	12	36	4	5	6
33	14	15	16	17	18	19	37	11	12	13
34	21	22	23	24	25	26	38	18	19	20
35	28	29	30	31			39	25	26	27

july
thursday

06

$T C \rightarrow (\log N) \rightarrow \text{Height of Tree}$

Binary Heap (Get Min
& Extract Min)



getMin - It simply return the minimum element from Min Heap

[20 | 25 | 30]

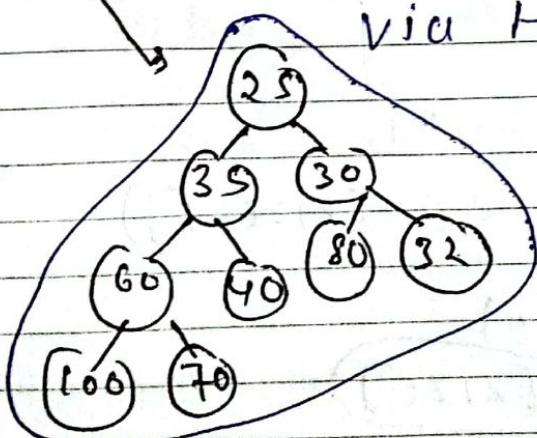
return arr[0]

$T C \rightarrow O(1)$ (It does not change the Binary Heap)

Extract Min → remove minimum element from Heap & Return it

2 things Always remember after remove
→ size --

→ Binary Heap change → so u have to make MinHeap / MaxHeap via Heapify.



this result show after performing

ExtractMin() function
(It change the Binary Heap)

07

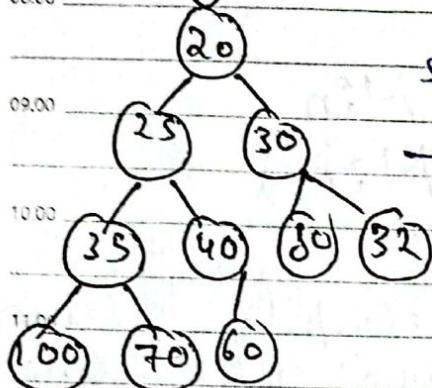
2017
july
friday

June 2017							July 2017						
Wk	M	T	W	T	F	S	S	Wk	M	T	W	F	S
22			1	2	3	4		26	1	2	3	4	
23	5	6	7	8	9	10	11	27	5	6	7	8	9
24	12	13	14	15	16	17	18	28	10	11	12	13	14
25	19	20	21	22	23	24	25	29	17	18	19	20	21
26	26	27	28	29	30			30	24	25	26	27	28

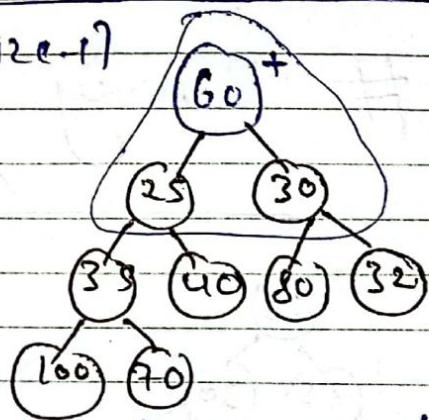
first element
last element

Algo -

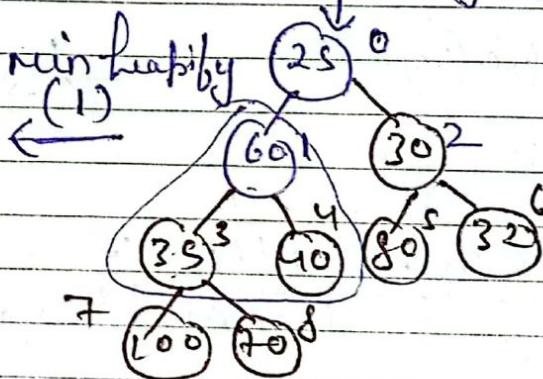
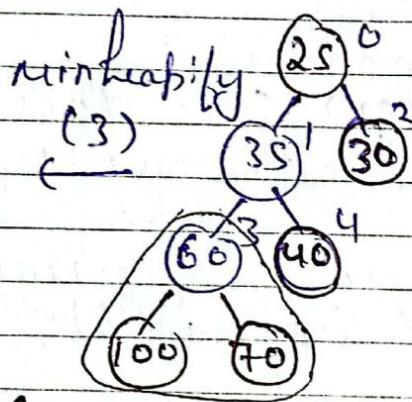
swap (arr[0], arr[size-1])
size--



swap (arr[0], arr[size-1])
size--

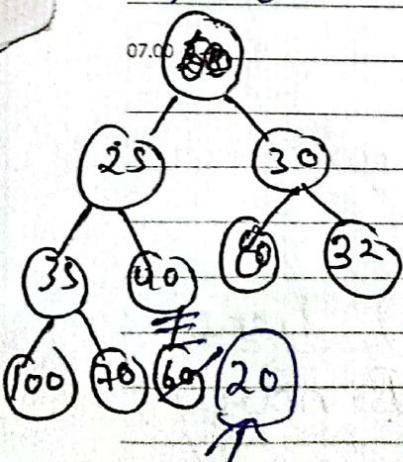


minheapy(0) index



Code int extractMin()

```
26 60
} if (size == 0) return INT-MAX
    if (size == 1) 20.
    { size--;
    } return arr[0]
```



swap (arr[0], arr[size-1])

size--

minheapy(0)

{ return arr[size]}

return arr[size]

August 2017						
Wk	M	T	W	T	F	S
31	1	2	3	4	5	6
32	7	8	9	10	11	12
33	14	15	16	17	18	19
34	21	22	23	24	25	26
35	28	29	30	31		

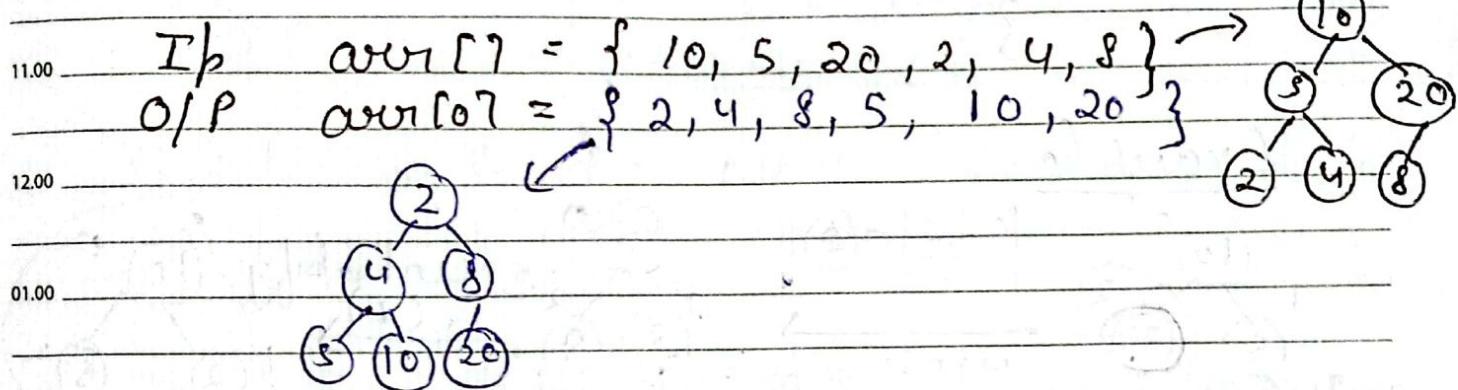
September 2017						
Wk	M	T	W	T	F	S
35					1	2
36	4	5	6	7	8	9
37	11	12	13	14	15	16
38	18	19	20	21	22	23
39	25	26	27	28	29	30

2017
july
monday

10

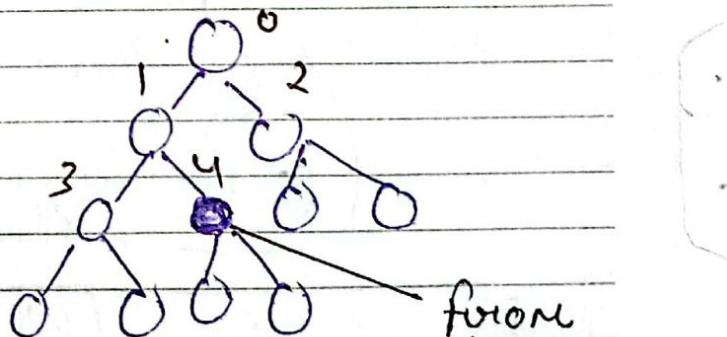
Binary Heap (Build Heap).

Given a random Array, rearrange its elements to form a min / Max Heap



Build Heap → Pick (Bottom most, Right most internal node and do minHeapify(i))

Find index (i)
then
minHeapify(i)
minHeapify(i-1)
minHeapify(i-2)
...
minHeapify(0)



from
here we
start heapify
then four

3, 2, 1, 0

last node → (size-1)	
last node Parent	
(size-1)-1	
2	

11

2017
july
tuesday

June 2017							July 2017							
Wk	M	T	W	T	F	S	S	Wk	M	T	W	T	F	S
22				1	2	3	4	26/31	31				1	2
23	5	6	7	8	9	10	11	27	3	4	5	6	7	8
24	12	13	14	15	16	17	18	28	10	11	12	13	14	15
25	19	20	21	22	23	24	25	29	17	18	19	20	21	22
26	26	27	28	29	30			30	24	25	26	27	28	29

Start Parent last

No child parent

Algo

void buildHeap()

{

for (int i = $\lceil \frac{\text{size}-2}{2} \rceil$; i >= 0; i--)

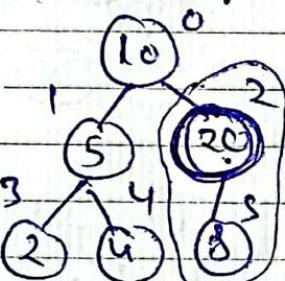
{

min heapify(i)

{

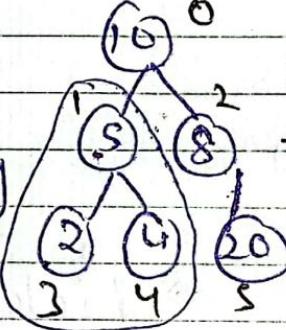
{}

{}

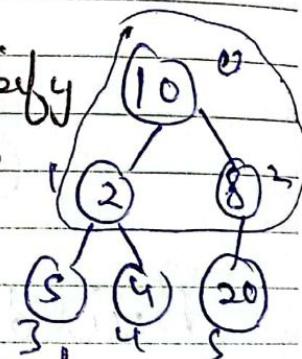
Example

$$\lceil \frac{6-2}{2} \rceil = 2$$

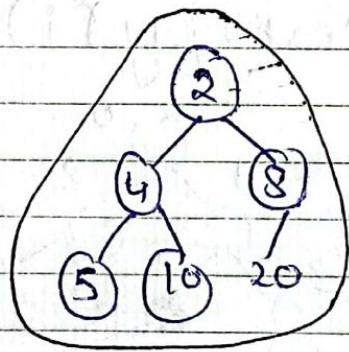
min heapify(2)



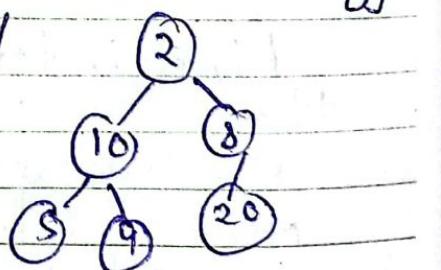
min heapify(1)



size =



min heapify(1)

Time Complexity $\rightarrow O(N)$

August 2017						
Wk	M	T	W	T	F	S
1	1	2	3	4	5	6
2	7	8	9	10	11	12
3	14	15	16	17	18	19
4	21	22	23	24	25	26
5	28	29	30	31		

September 2017						
Wk	M	T	W	T	F	S
35				1	2	3
36	4	5	6	7	8	9
37	11	12	13	14	15	16
38	18	19	20	21	22	23
39	25	26	27	28	29	30

2017

july

wednesday

12

H Heap Sort

08.00

09.00 • It is based on selection sort Algo

10.00

Find out maximum element in group & swap with last Element.

11.00

12.00 In selection Sort → we do linear search to find maximum element.
TC → $O(N^2)$

02.00

In Heap Sort → we reconstruct the given array and form max heap.

03.00

Means,
root ge maximum value uthay array ke last me daal dia.

04.00

TC → $O(N \log N)$ → maximum height of tree.

05.00

Codel

06.00

void buildHeap(@int arr[], int n)

{
 for (int i = $\lceil \frac{n-1}{2} \rceil$; i >= 0; i--)

{
 maxheapify(arr, i);

} }

↑ his
index
pertaining
hai;

CS CamScanner

13

2017

july

thursday

06

June 2017

Wk	M	T	W	T	F	S	S
22			1	2	3	4	
23	5	6	7	8	9	10	11
24	12	13	14	15	16	17	18
25	19	20	21	22	23	24	25
26	26	27	28	29	30		

07

Wk	M	T	W	T	F	S	S
26/31	31				1	2	
27	3	4	5	6	7	8	9
28	10	11	12	13	14	15	16
29	17	18	19	20	21	22	23
30	24	25	26	27	28	29	30

② void Maxheapsify (int arr[], int n, int ind)

08.00

{ int largest = ind(index) ;

 find
 largest

10.00

 int left = 2 * ind + 1 ;

 int right = 2 * ind + 2 ;

11.00

 if (left < n && arr[left] > arr[largest])
 largest = left ;

12.00

 if (right < n && arr[right] > arr[largest])

01.00

 largest = right ;

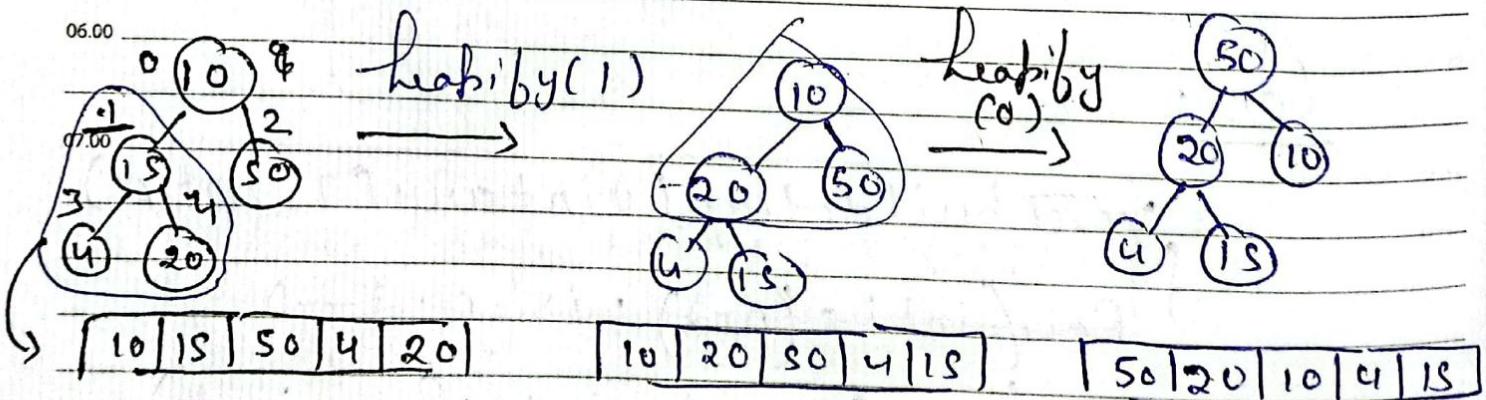
02.00

 if (largest != ind)

here
do
swap
call
heapsify
fun

 swap (arr[largest], arr[ind])
 maxheapsify (arr, n, target) ;

}



maxheapsify
then Apply
Heap sort

August 2017						
Mo	Tu	We	Th	Fri	Sa	Su
1	2	3	4	5	6	
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

September 2017						
Mo	Tu	We	Th	Fri	Sa	Su
35					1	2
36	4	5	6	7	8	9
37	11	12	13	14	15	16
38	18	19	20	21	22	23
39	25	26	27	28	29	30

2017
july
friday

14

void HeapSort(int arr[], int n)

f buildHeap (int arr, n)

for (int i = n - 1; i >= 1; i--)

{

swap (arr[0], arr[i]);

heapify (arr, i, 0)

size index

working
visualise
on
array

- Repeatedly swap root with last Element
- Reduce Heap size by 1 and do Heapify () .

heapify always from 0 index (Root)

i.e. size) always decrease by 1

Time Complexity - $O(n \log n)$

15

2017
july
saturday

June 2017							July 2017						
Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa	Su
22			1	2	3	4	27	3	4	5	6	7	1
23	5	6	7	8	9	10	11	28	10	11	12	13	14
24	12	13	14	15	16	17	18	29	17	18	19	20	21
25	19	20	21	22	23	24	25	30	24	25	26	27	28
26	26	27	28	29	30								

July 2017						
Mo	Tu	We	Th	Fr	Sa	Su
26/31	31					
27	3	4	5	6	7	1
28	10	11	12	13	14	8
29	17	18	19	20	21	15
30	24	25	26	27	28	23

