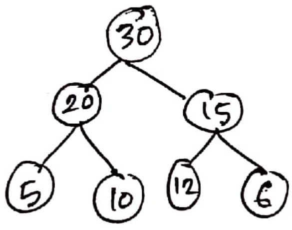
Binary Heap

Lecture 1

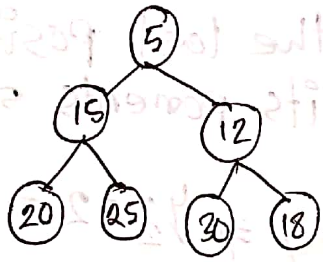
1. what is a Heap
2. Insert in a Heap
3. Deleting from Heap
4. Heap sort
5. Heapify
6. priority Queues

Heap is a complete Binary tree

Max Heap



min Heap



max heap: All the decindent of root is small.

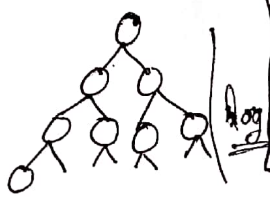
min heap: All the dicindent of root is gretter

T

30	20	15	5	10	12	6
1	2	3	4	5	6	7

complete Binary tree mean no gap in Array

Node at index = i
left child at = 2*i
Right child at = 2*i+1



in complete Binary tree height will be log n only

Scanned with CamScanner

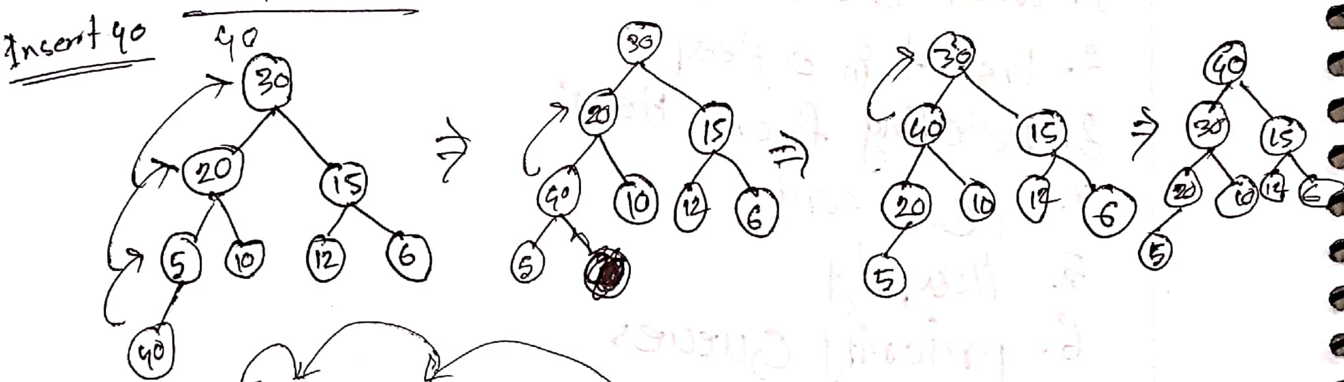
Insert Binary Heap

Lecture 2

when we insert we insert in the last in array and check and Replace in the Right position

Max Heap

Insert 40



	40	40		40				8		
T	30	20	15	5	10	12	6			
	1	2	3	4	5	6	7	40		

40 insert in the last position and check its parents small or gatter

$8/2 = 4 \Rightarrow 4/2 = 2 \Rightarrow \frac{2}{2} = 1$

Lecture 3 O(logn)

call part 1

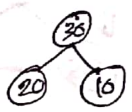
```
void Insert(int A[], int n)
{
    int temp, i=n;
    temp = A[n];
    while(i>1 && temp >A[i/2])
    {
        A[i] = A[i/2];
        i = i/2;
    }
    A[i]=temp;
}
```

```
void createHeap()
{   int A[]={0,10,20,30
             25,5,40,35
    int i;
    for(i=2, i<=7; i++)
    {   insert(A,i);
    }
```

O(nlogn)

Lecture 34

30	20	10	40	15	12	25



complete Binary tree

insert Right to left and create heap

This is the Idea

Compair with parent newly Insert element

create Heap

10	20	30	25	5

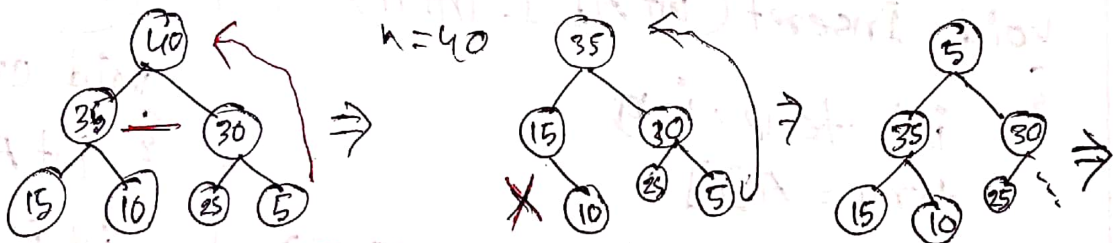
1. 10

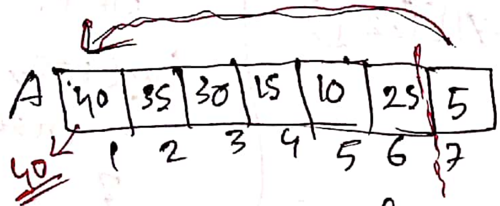
2. Insert 20

This is Ideda create Heap.

Delete Binary Heap

Lecture: 5





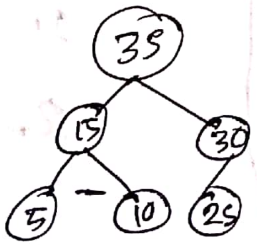
problem

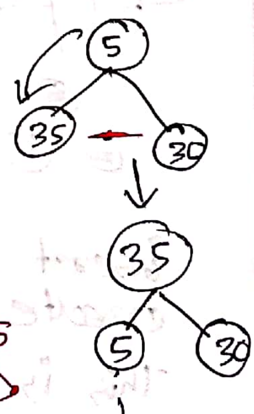
When 40 is gone the Right most elemet copy in Root

*** From heap : Delete only Root

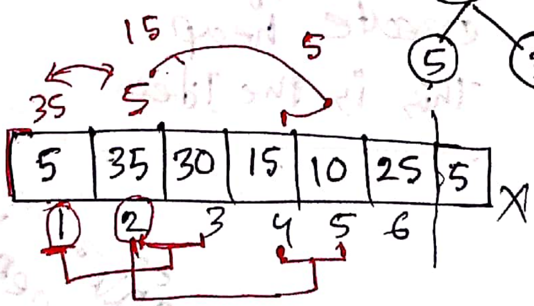
You can delete Height proity element

1. Root delete
2. copy Last element (5) to Root new
3. compiir Root Both child who is getter (35,-30) change with (5)
4. again follow (3) step again and again





In Array ⇒ A :



Scanned with CamScanner

```
void Delete(int A[], int n)
{  int x, i, j;
   x = A[n];
   A[i] = A[n];

   i=1, j=2*i;
 while( j <= n-1)

   if(A[j+1] > A[j])
     j = j+1;   // position chang
                // to chind elemet
                      4,5 like
   if (A[i] < A[j])
    { swap (A[i], A[j]);
      i = j;
      j = 2*j;
    }
   else
      break;
 } A[n] = x;  ——————>  store in last Delete
                         element
```

Heap

1. Create Heap — $n\log n$
2. Delete elem 1 by 1 — $n\log n$

$2n\log n$

$O(n\log n)$

↓

sort Array

** When we delete any element from heap.

1. we store in that last
* again and again delete then we get ~~fast~~ sort array

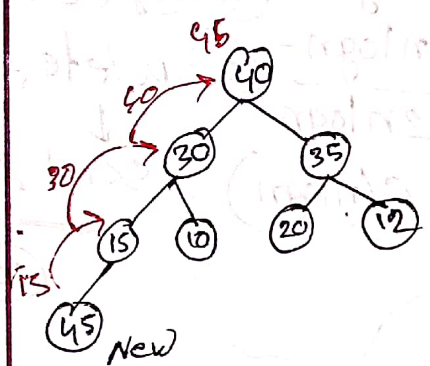
*** | 35 | 10 | 20 | 25 | 15 | | ⇒ | 10 | 15 | 20 | 25 | 35 |

Binary Heap

Heapify

Lecture 7

Insert

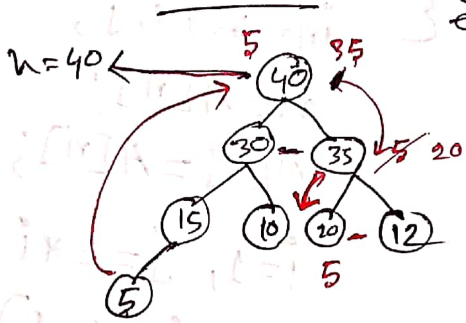


upword

Leaf to Root sending element.

Delete

only Root element Deleted



① send ~~las~~ Last elemet to Root
② compair Root ~~elem~~ to children whom is biger

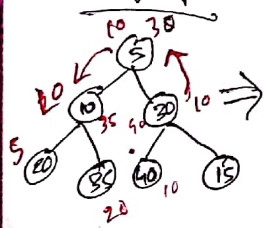
* Root to leaf sending element

$e^{5n} \cdot 5$

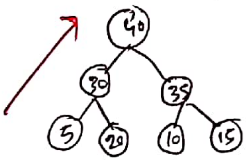
$\boxed{5 * e^{5n}}$

creat Heap

binary Represent



max heap

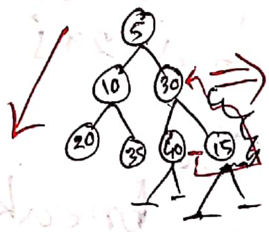


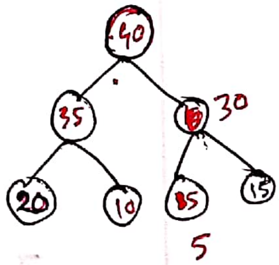
A	5	10	30	20	35	40	15

$O(n\log n)$

Heapify

binary tree





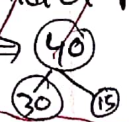
Downword

A	5	10	30	20	35	40	15

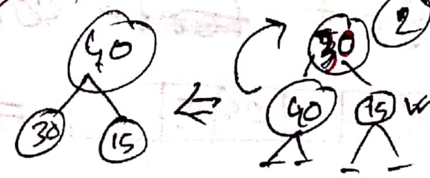
$O(n)$

① compair leaf biger send to emidiet Root

30 → (40-15) ⇒



② Check in down word



Binary Heap as Priority queue

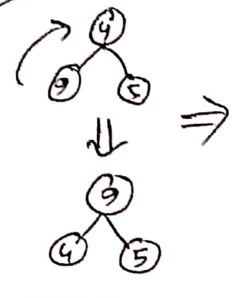
element ⇒ 4, 9, 5, 10, 6, 20, 8, 15, 2, 18

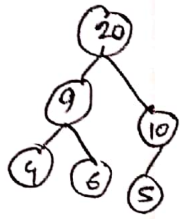
larger the element height the priority

max heap

A

4	9	5	10	6	20	8		----
1	2	3	4	5	6	7	8	





Insert $O(1)$

Delete $O(n) \Rightarrow \underbrace{(n}_{\text{search}}+\underbrace{n)}_{\text{move}} = 2n \Rightarrow O(n)$

Delete ⇒ search the height element and Delete and move elemem

shift

Insert $O(\lg n)$

Delete $O(\log n)$

*

Time Reduce