Date

⇒ APPLICATION OF TREE

→ Heap Sort

It is a complete binary tree such that value of every node as large as values of it's children nodes.

It is very efficient sorting algorithm for large data, the time complexity in worst case is $O(n \log_2 n)$, the average case is not good (highest).

It requires order of $O(n)$ extra space.

It has 2 phases :-

i. Creating heap
ii. Selection or Deleting heap

Values :-

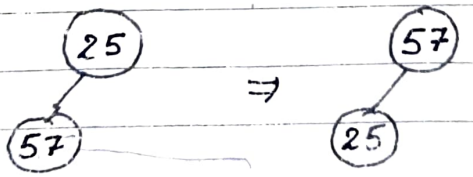
25	57	48	37	12	92	86	33

⇒ CREATING HEAP (MAX Heap)

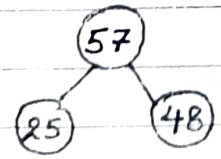
add 25

In this parent node must be greater than its children & Insertion must be from left to right until level not filled completely

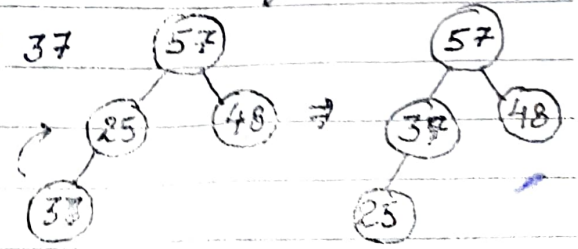
add 57



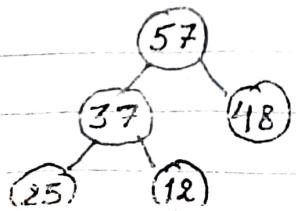
add 48



add 37

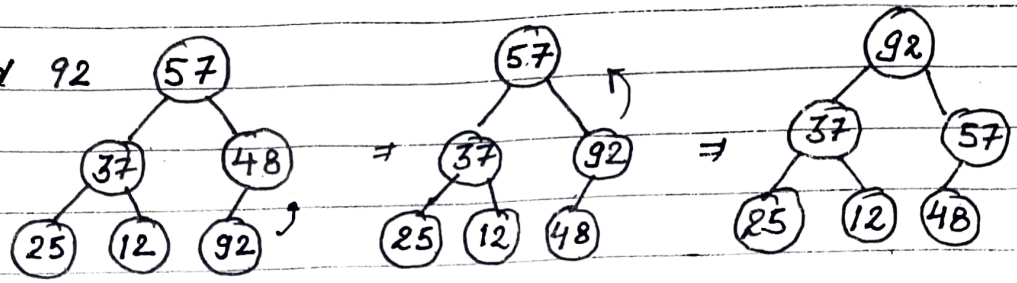


add 12

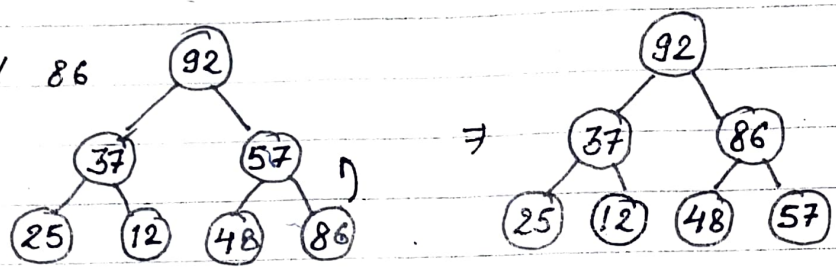


Date

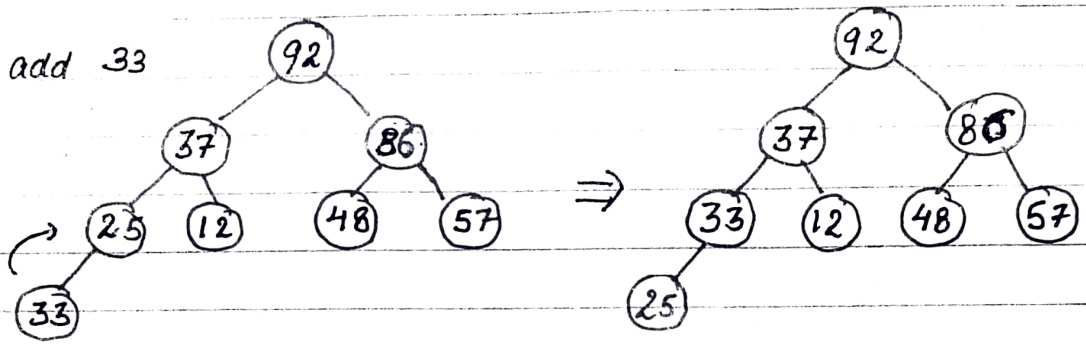
add 92



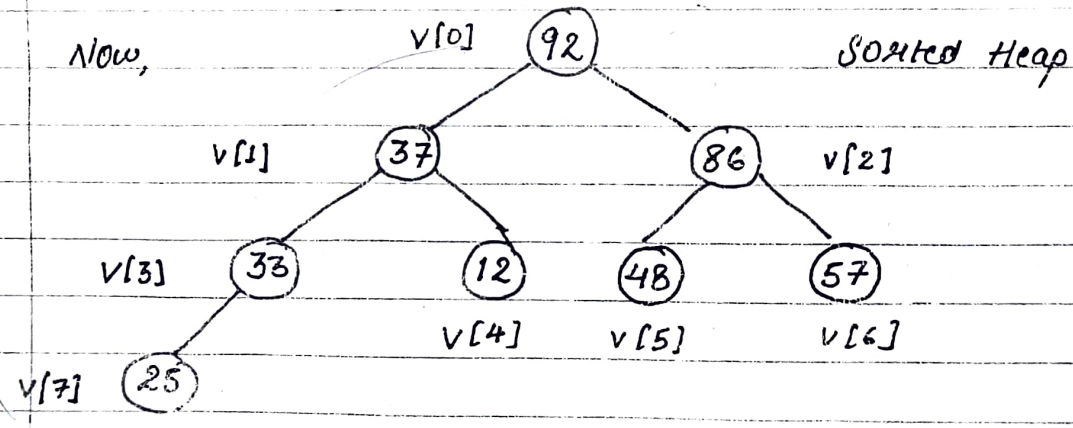
add 86



add 33



Now,



For ith node

⇒ Left child of ith = (2*i+1)th.

Right child of ith = (2*i+2) th

i = node.

Parent node of ith = $(i-1)/2$

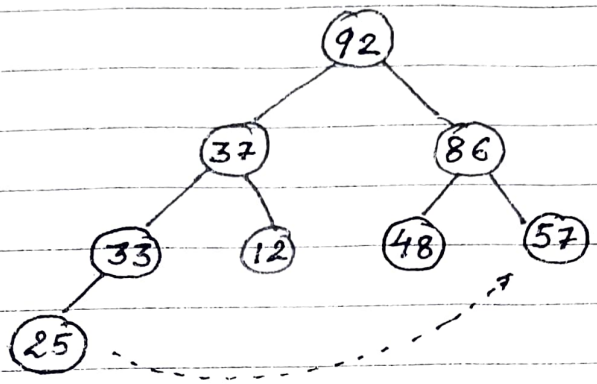
	0	1	2	3	4	5	6	7
v	92	37	86	33	12	48	57	25

Date

⇒ Deleting of Heap :

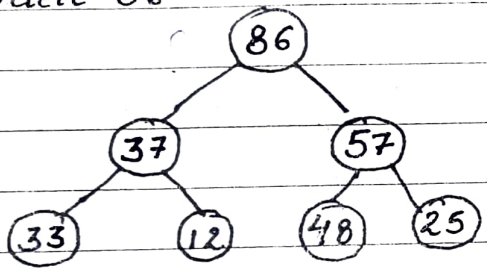
Delete the root in every pass and reheap the tree. In this make greatest node as root node.

Delete 92

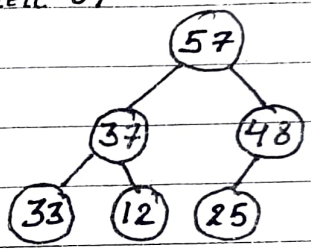


92

Delete 86

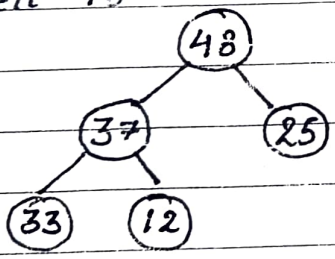


Delete 57

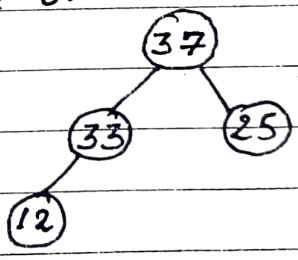


57, 86, 92

Delete 48

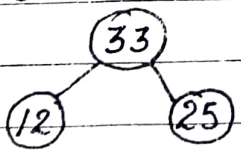


Delete 37

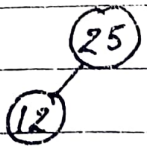


37, 48, 57, 86, 92

Delete 33



Delete 25



33, 37, 48, 57, 86, 92

Delete 12

25, 33, 37, 48, 57, 86, 92

12, 25, 33, 37, 48, 57, 86, 92

Date

⇒ ALGORITHM

INSERT_HEAP (TREE, N, ITEM)

A heap H with N elements is stored in the array TREE, and an ITEM of information is given. This procedure inserts ITEM as a new element of H. PTR gives the location of ITEM as it rises in the tree, and PAR denotes the location of the parent of ITEM

1. [Add new node to H and initialise PTR]
   set N = N+1 and PTR = N
2. [Find location to insert ITEM]
   repeats steps 3 to 6 while PTR < 1
3. set PAR = [PTR/2] [Location of parent node]
4. If ITEM ≤ TREE[PAR], then
   set TREE[PTR] = ITEM and return
   [End of if]
5. set TREE[PTR] = TREE[PAR] move nodes down
6. set PTR = PAR (update PTR)
   [End of step 2 loop]
7. [Assign ITEM as root of H]
   set TREE[1] = ITEM
8. RETURN

DELETE_HEAP (TREE, N, ITEM)

A heap H with N elements is stored in the array TREE. This procedure assign the root TREE[1] of H to the variable ITEM and then reheaps the remaining elements. The variable LAST saves the value of the original last node of H. The pointers PTR, LEFT & RIGHT gives the location LAST and its left and right children as last sinks in TREE

Date

1. Set ITEM = TREE[1] [removes root of H]
2. Set LAST = TREE[N] and N = N-1
   [removes last node of H]
3. Set PTR = 1, LEFT = 2 and RIGHT = 3
   [initialise pointers]
4. Repeat steps 5 to 7 while RIGHT ≤ N
5. If LAST ≥ TREE[LEFT] and
   LAST ≥ TREE[RIGHT], then
   set TREE[PTR] = LAST and Return
   [End of if]
6. If TREE[RIGHT] ≤ TREE[LEFT], then
   set TREE[PTR] = TREE[LEFT] and PTR = LEFT
   Else
   set TREE[PTR] = TREE[RIGHT] and PTR = RIGHT
   [End of if]
7. set LEFT = PTR and RIGHT = LEFT + 1
   [End of step 4 loop]
8. If LEFT = N and if LAST < TREE[LEFT], then
   set PTR = LEFT
9. set TREE[PTR] = LAST
10. Return