Lecture 32-33

- Heap Tree (Heap)

IT205: Data Structures (AY 2023/24 Sem II Sec B) — Dr. Arpit Rana

Heap (Tree)

A heap tree (H) is an **almost complete binary tree** if it satisfies the following properties:

- For each node ***nd*** in H, the value at ***nd*** is ***greater than or equal to*** the value of each of the children of ***nd***.

  Or in other words,

- ***nd*** has the value which is ***greater than or equal to*** the value of every successor of ***nd***.

Such a heap tree is called <u>max-heap</u>.

Heap (Tree)

A heap tree (H) is an **almost complete binary tree** if it satisfies the following properties:

- For each node ***nd*** in H, the value at ***nd*** is ***less than or equal to*** the value of each of the children of ***nd***.

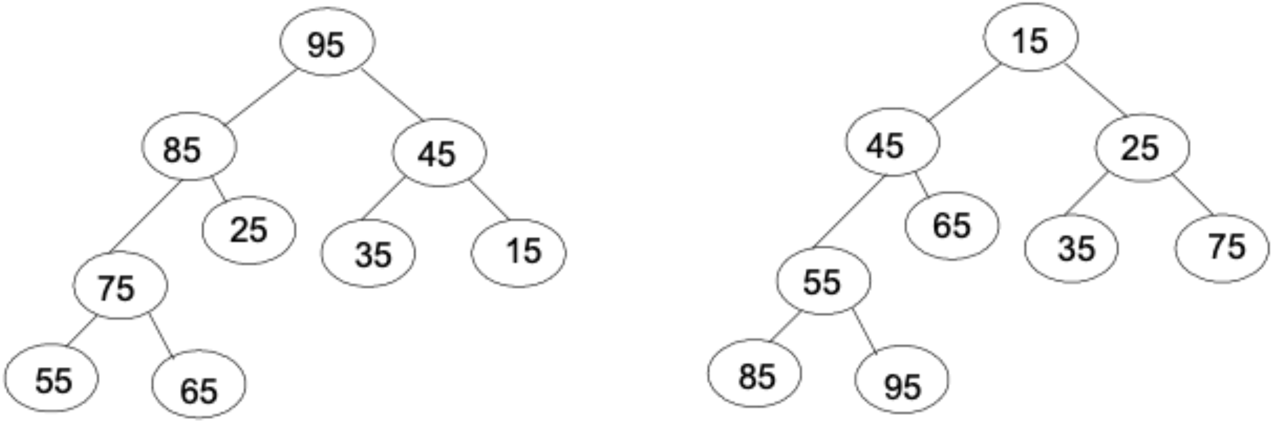
  Or in other words,

- ***nd*** has the value which is ***less than or equal to*** the value of every successor of ***nd***.

Such a heap tree is called <u>min-heap</u>.

Heap (Tree)

Examples of a Max- and a Min- Heap



(a) Max heap (b) Min heap

Representation of a Heap (Tree)

Array representation of a Heap has certain advantages over its linked representation:

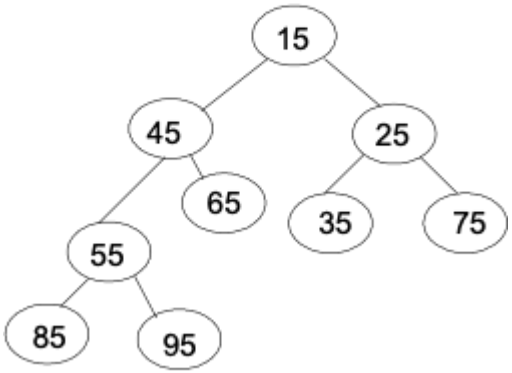
- As it is an almost complete binary tree, null entries will only be at the tail of the array, so there is no wastage of memory

- No need to maintain links of descendants (children). It can be automatically implied by performing simple arithmetic operations

  - recall that p: root, left-son: 2p +1, right-son: 2p + 2 (when the base address is 0)

Representation of a Heap (Tree)

Array representation of a Heap has certain advantages over its linked representation:

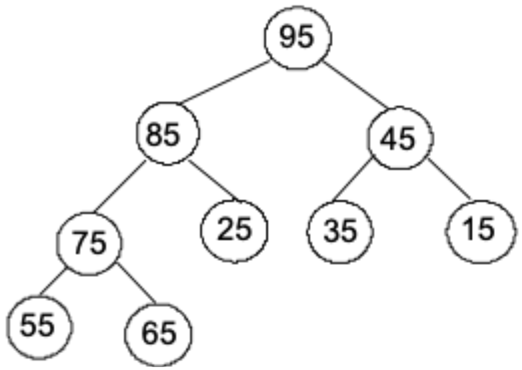


(b) Min heap

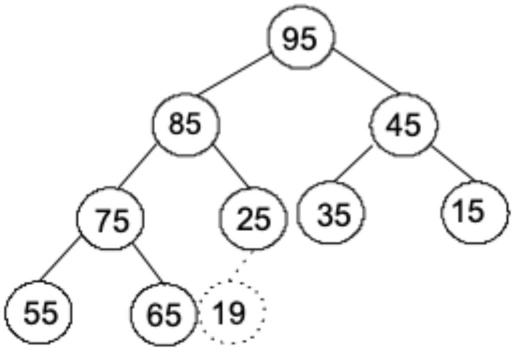
1	2	3	4	5	6	7	8	9	10	11	12	13	14	
15	45	25	55	65	35	75	85	95	.	.	.	.	.	.

Insertion in a Heap

Case 1: A trivial case



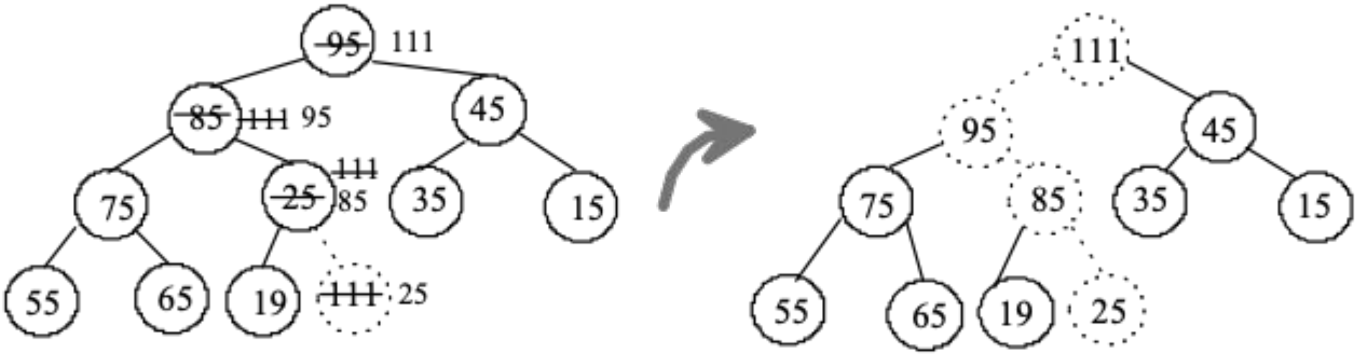
Max heap



Inclusion of **19** in the fashion of almost complete binary tree and it satisfies the Max heap property

Insertion in a Heap

Case 2: A non-trivial case



When 111 is inserted into the heap tree

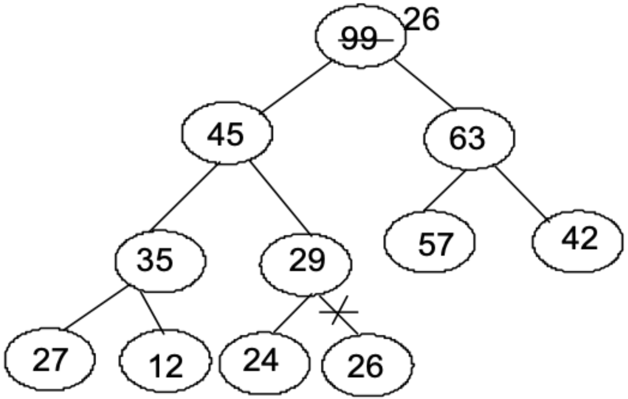
Inclusion of **111** in the fashion of almost complete binary tree but it does not satisfy the Max heap property and needs to move up unless it reaches to its appropriate position

Deletion in a Heap

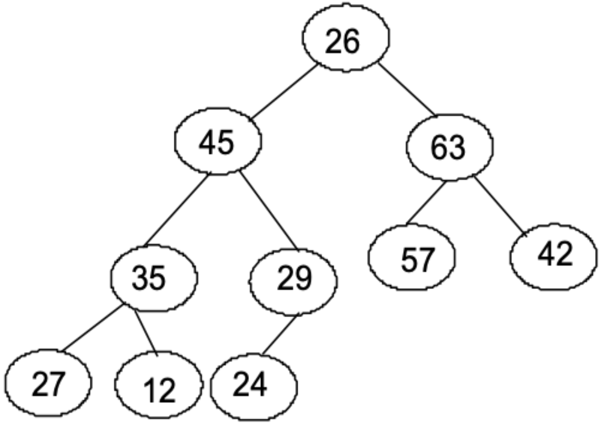
Any node can be deleted from a heap tree. But from the application point of view, deleting the root node has some special importance.

- Read the root node into a temporary storage, say ITEM.
    - Replace the root node by the last node in the heap tree. Then reheap the tree as stated below:
        - Let newly modified root node be the current node. Compare its value with the value of its two children. Let X be the child whose value is the largest. Interchange the value of X with the value of the current node.
        - Make X as the current node.
    - Continue reheap, until the current node is not an empty node.

Deletion in a Heap

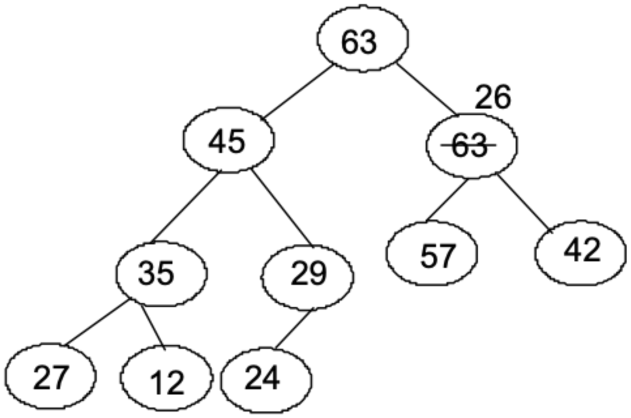


Deleting the node with data 99

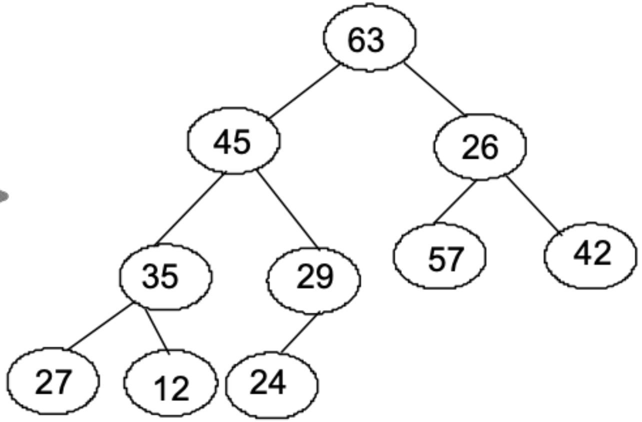


Heap tree after repalcing 99 by 26

Deletion in a Heap

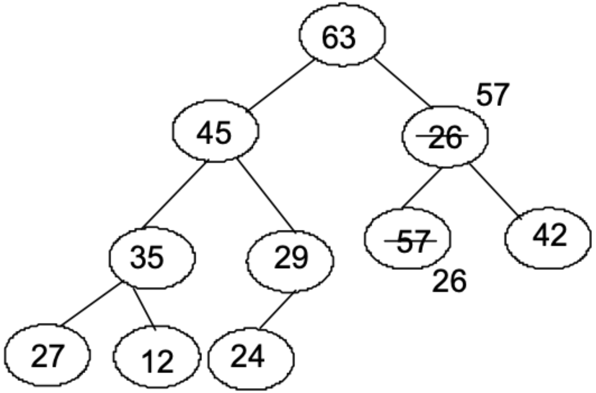


Rebuilding after adjusting 63

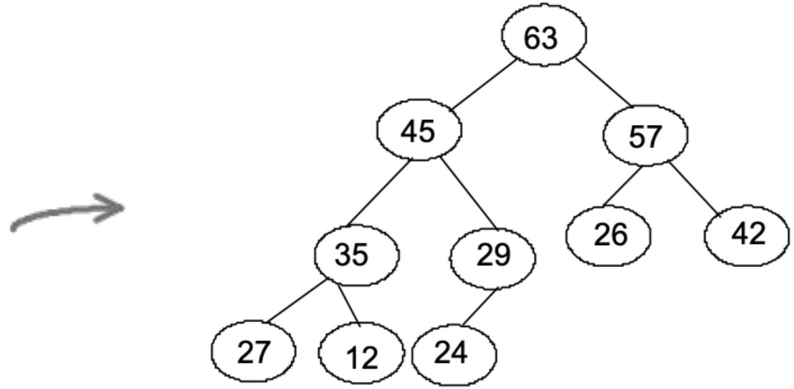


Heap tree after rebuild

Deletion in a Heap

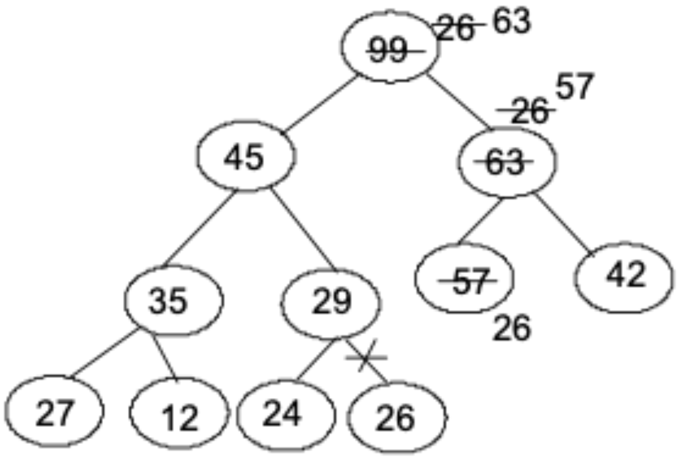


Rebuild the tree at 26

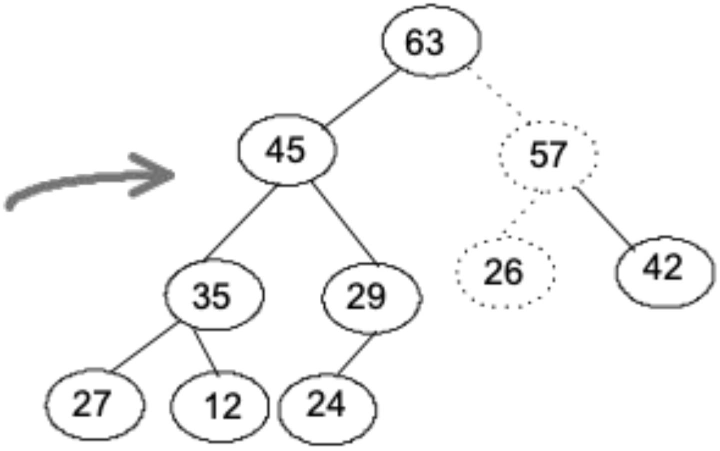


Heap tree after rebuild

Deletion in a Heap



Deleting the node with data 99

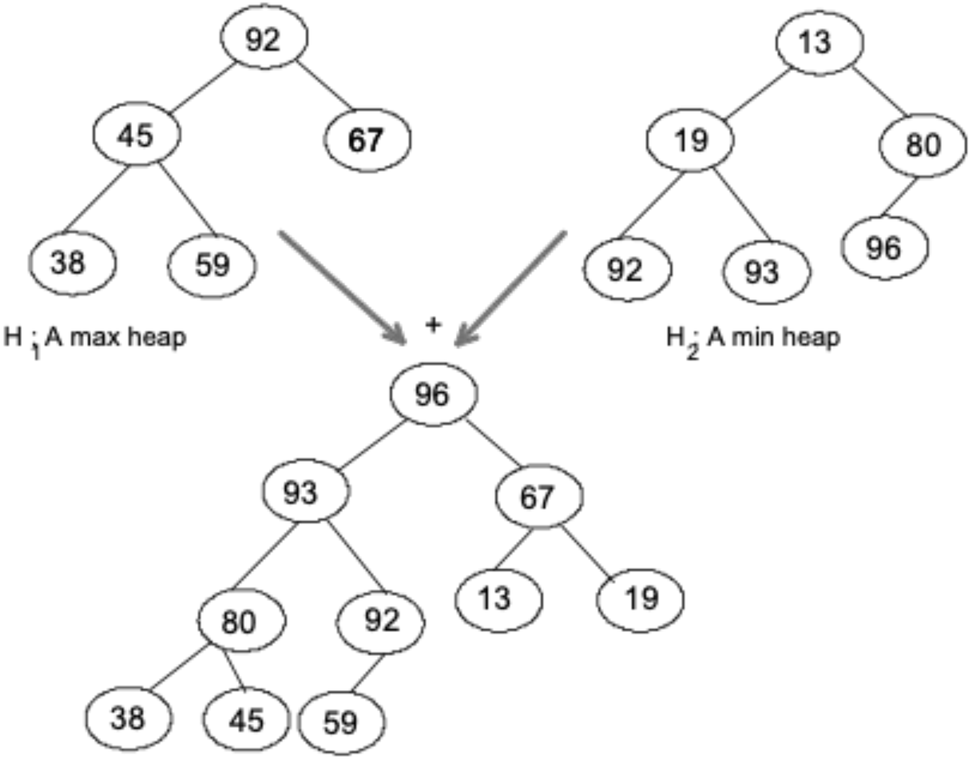


After deletion of 99

Merging Two Heaps

- Consider, two heap trees H_1 and H_2.
- Merging the tree H_2 with H_1 means to include all the nodes from H_2 to H_1.
- H_2 may be min heap or max heap and the resultant tree will be min heap if H_1 is min heap else it will be max heap.
- Merging operation consists of two steps:
  - Continue steps 1 and 2 while H_2 is not empty:
    - Delete the root node, say x, from H_2.
    - Insert the node x into H_1 satisfying the property of H_1.

Merging Two Heaps



Resultant max heap after merging H1 and H2

Applications of Heap Tree: Sorting (Heapsort)

Step 1:

- Build a heap tree with the given set of data.

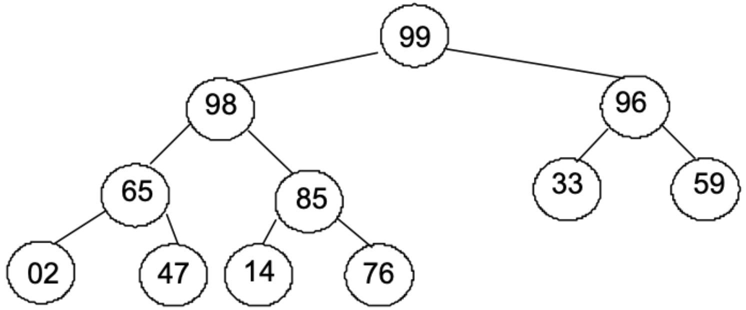
Step 2:

- Delete the root node from the heap.
- Rebuild the heap after the deletion.
- Place the deleted node in the output.

Step 3:

- Continue Step 2 until the heap tree is empty.

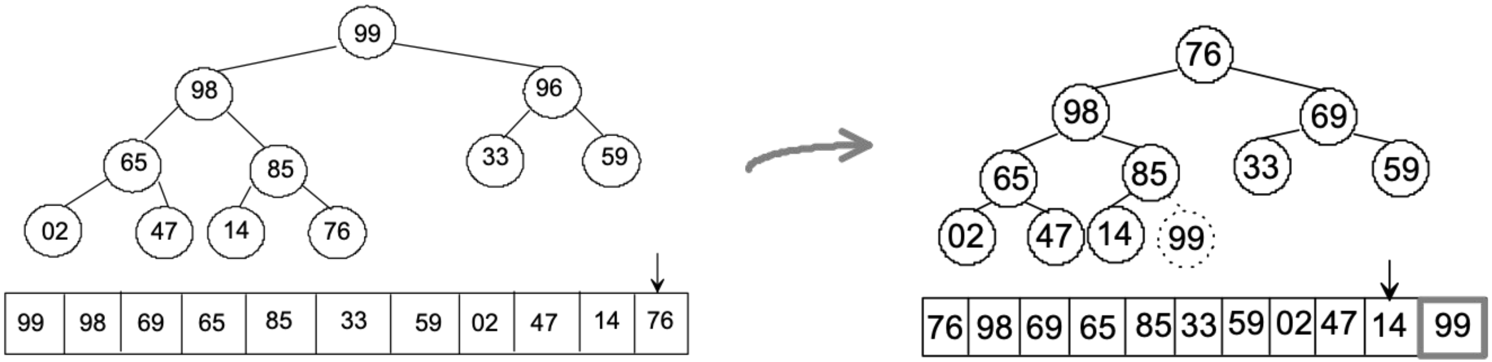
33, 14, 65, 02, 76, 69, 59, 85, 47, 99, 98



99	98	69	65	85	33	59	02	47	14	76

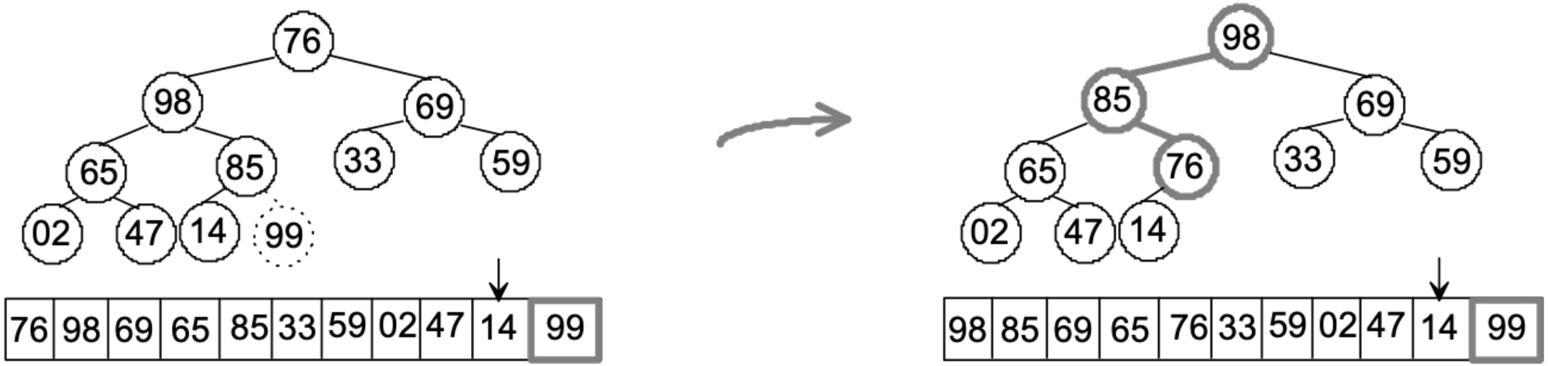
Building (max) heap tree from the given set of data

Applications of Heap Tree: Sorting (Heapsort)



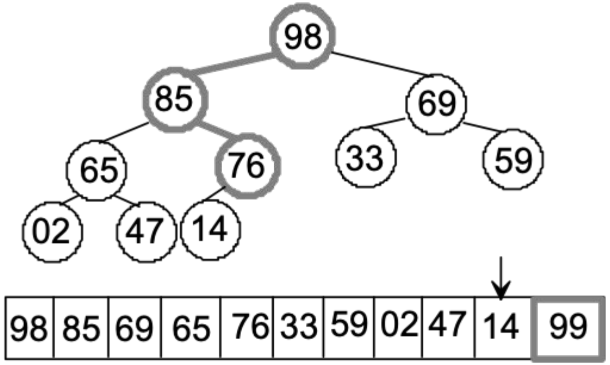
Swapping the root and the last node

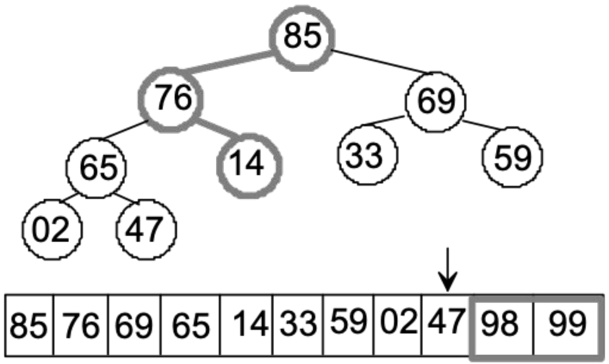
Applications of Heap Tree: Sorting (Heapsort)



Rebuild the heap tree

Applications of Heap Tree: Sorting (Heapsort)





Repeat deleting root and rebuilding heap

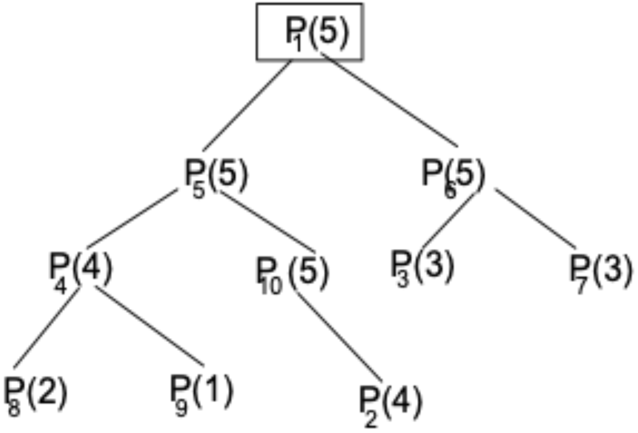
Applications of Heap Tree: Priority Queue

Consider the following processes, their arrival with their priorities:

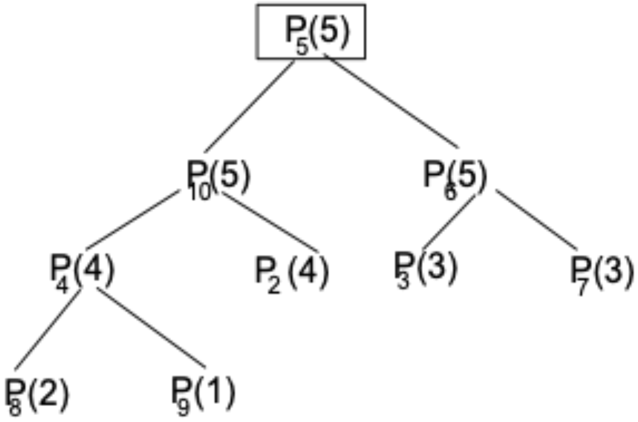
Process	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10
Priority	5	4	3	4	5	5	3	2	1	5

Ordering of processing should be: P1 ← P5 ← P6 ← P10 ← P2 ← P4 ← P3 ← P7 ← P8 ← P9

Applications of Heap Tree: Priority Queue



(a) Priority queue heap (subscript indicates order of process whereas number in parantheses means its priority)



(b) After the removal of P_1

Next Lecture

- Introduction to Graphs