Discrete Mathematics

Scribed Notes 16 (20th October)

- Trees are special case of graph.

Binary Trees:

1. Full Binary Trees
2. Complete Binary Trees
3. Perfect Binary Trees

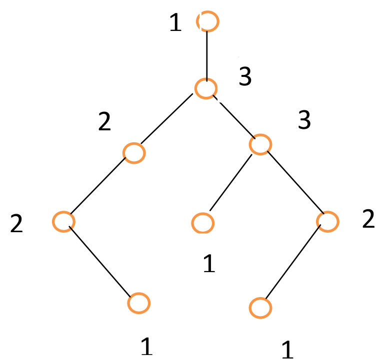
Properties of Binary Trees:

- Number of Edges in an **n** node tree is always **n-1**.
- Maximum possible degree in Binary tree = **3**.
- Maximum number of nodes at any level 'L' in a binary tree = $\mathbf{2^L}$.
- Maximum number of nodes in a binary tree of height h = $\mathbf{2^{h+1} - 1}$.
- Minimum possible degree in Binary tree = **1**.
- **Leaf Node:** No children Node **(degree 1).**
- **Internal Node:** At least one child **(degree 2 or 3)**.
- **Root:** No parent Node **(degree 1)**.
- **Total degree** of n node in a tree = 2n-2 = **2(n-1).**
- Possible degrees in NON-TRIVIAL binary trees: **1,2,3**

Degree Sequence in graph:

- **Degree Sequence** is a list of all degrees for every node in a graph.
- Degree sequence can be written in any order but sometimes we use increasing order for convenience.

Example 1:



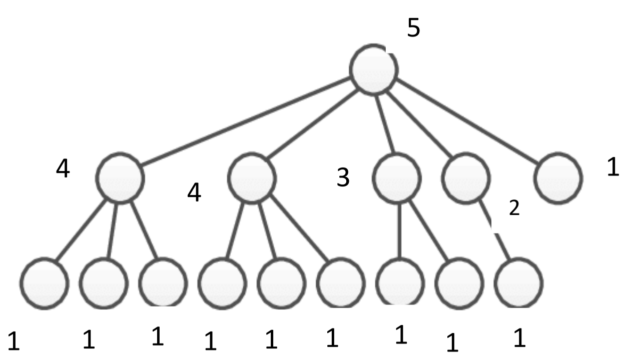
Degree sequence for this tree will be:
1,1,1,1,2,2,2,3,3

Total Nodes: 9

Example 2:

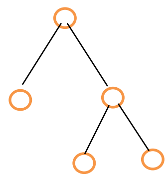
Degree sequence for this tree will be:
1,1,1,1,1,1,1,1,1,1,2,3,4,4,5

Total Nodes: 15



Full Binary Trees:

- Number of Internal Nodes = Number of Leaves – 1   (Structural Induction)



- In this diagram, number of leaves = 3
- Internal Nodes = 3 – 1 = 2

Proof :

- (By first theorem of Graph Theory)

  Internal Nodes + Root Node + Leaves Nodes = Total Nodes
  3(i-1) + 2 + L * 1 = 2(n-1)
  ∴ 3i – 3 + 2 + L +2 = 2n   - equation 1
  (L = Number of Leaf Nodes, i = Internal Nodes and n=Total Nodes)

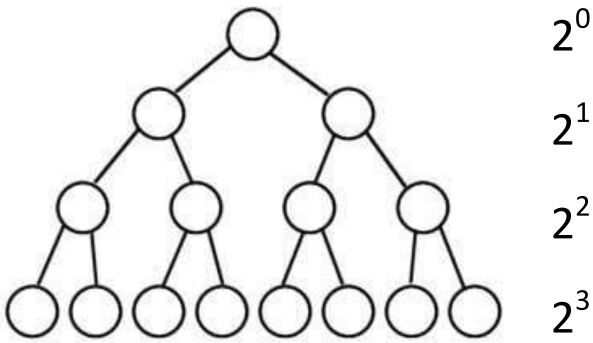
- **L + i = n** (Partition of nodes in leaves and internal nodes)
  ∴ 2L + 2i = 2n   - equation 2

- Now, we can compare equation 1 and equation 2.
  ∴ 3i – 3 + 2 + L +2 = 2L + 2i
  ∴ i + 1 = L   - equation 3

- Equation 3 is same equation as we have from structural induction.

Complete Binary Tree:

- In complete binary tree every level is at full capacity.



- **It is in G.P. (Geometric Progression).**
- $S = a + ar + ar^2 + ar^3 + \ldots + ar^{n-1}$ ___ (1)
- $rS = ar + ar^2 + ar^3 + \ldots + ar^{n-1} + ar^n$ ___ (2)

$\mathbf{S(r\text{-}1) = a(r^{n\text{-}1})}$

$$\therefore\ S = \left(\frac{r^n - 1}{r - 1} \right) * a$$

- For complete binary tree,

  $r = 2$, $a = 1$

  $\therefore$ **S = 2^n** - 1 → $\therefore \mathbf{S = 2^n} - 1$

  $\therefore \mathbf{S = 2^{L+1}} - 1$ (here, $n = L + 1$)

  (where, L=level of binary tree, S=Total Numbers of Nodes)

  Here, L is starting from 0,1,2,…..

Dynamic Sets

- Mathematical sets are unchanging, but the sets manipulated by algorithms can grow, shrink, or otherwise change over time. We call such sets **dynamic sets.**
- In Data Structure, we have dynamic sets because we can change it with time and we can **insert, delete and modify**.

Binary Heap

A Binary Heap is a Binary Tree with following properties.

(1) Structural Properties

- It's a complete binary tree
- All levels are completely filled **except possibly the last level**.
- The last level is **strictly filled from left to right**.

(2) Key value Properties

- Elements in the heap tree are arranged in specific order.
- The same property must be recursively true for all nodes in Binary Tree.

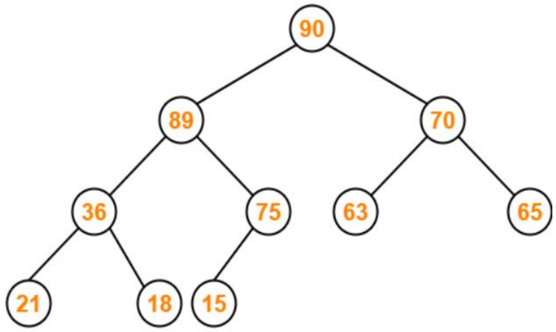
Types of Binary Heap

(1) Max Heap

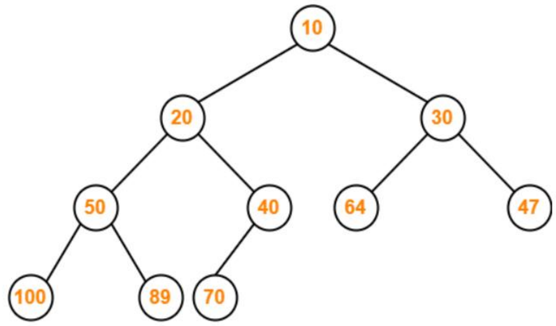
- In a Max-Heap the key present at the root node must be greatest among the keys present at all of it's children. The same property must be recursively true for all sub-trees in that Binary Tree.
- **$\forall$ nodes ; key [Node] > key [Children]**

(2) Min Heap

- In a Min-Heap the key present at the root node must be minimum among the keys present at all of it's children. The same property must be recursively true for all sub-trees in that Binary Tree.
- **$\forall$ nodes ; key [Node] < key [Children]**



Max Heap Example



Min Heap Example

In How many ways n distinct integers can be arranged in a Min Heap

General Approach :

- There is only one element as the **root**, it must be the smallest number. Now we have n-1 remaining elements.
- The structure of the heap nodes will remain the same in all instances, but only the values in the nodes will change.
- Assume there are **L** elements in the **left sub-tree** and **R** elements in the **right sub-tree**. Now for the root, L + R = n-1. From this we can see that we can **choose** any l of the remaining n-1 elements for the left sub-tree as they are all bigger than the root.
- We know that there are $_{n-1}C_L$ ways to do this. Next for each instance of these, we can have many heaps with **L elements** and for each of those we can have many heaps with **R elements**. Thus we can consider them as **recurrence**.
- Recursive Function : **H(n) = H(Ln) x H(Rn) x** $_{n-1}C_{Ln}$
- Base Cases: H(1) = 1, H(2) = 1, H(3) = 2

Example-1: In how many ways n=10 distinct integers can be arranged in a Heap.

Here, n = 10

H(10) = H(6) x H(3) x $_{9}C_{6}$

$= H(3) \times H(2) \times {}_5C_3 \times H(3) \times {}_9C_6$

$= {}_9C_6 \times {}_5C_3 \times H(2) \times (H(3))^2$

$= 84 \times 10 \times 1 \times (2)^2$

$= 840 \times 4$

$= 3360$

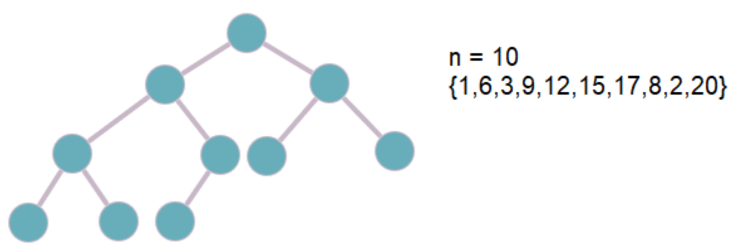
Explanation of Above calculation:

- For n=10, a Binary Tree will have level = 3 which means **total 15 nodes.**
- After inserting the **smallest** integer/key at **Root**, we are left with n=9, Left Side Binary Tree of Root with 6 nodes and Right side binary tree of Root with 3 nodes.
- From that 9 elements we can **choose** any 6 integers/keys to be inserted on left side binary tree which will be $\mathbf{{}_9C_6}$ and the remaining 3 will go on right side.
- But on left and right side binary tree we again need to choose the smallest integer/keys among remaining integers/keys for that level root node thus the recursion.
- Thus **H(6) for left** side binary tree, **H(3) for right** side binary tree and $\mathbf{{}_9C_6}$ for choosing the integers.
- And we do the same again for H(6) and H(3) **recursively**.

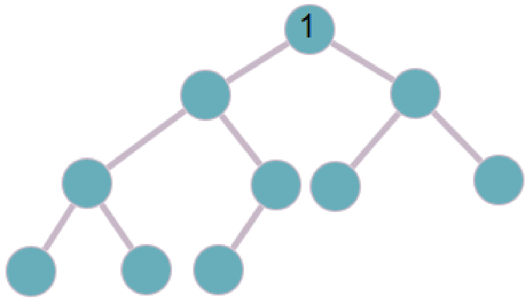
One possible arrangement in Heap with n=10:

Consider integers as : 1,6,3,9,12,15,17,8,2,20

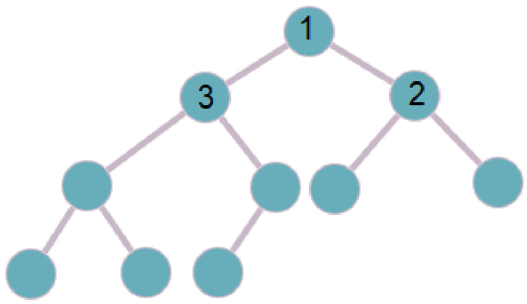
- Structure of a Heap (structure is fixed irrespective of integers):



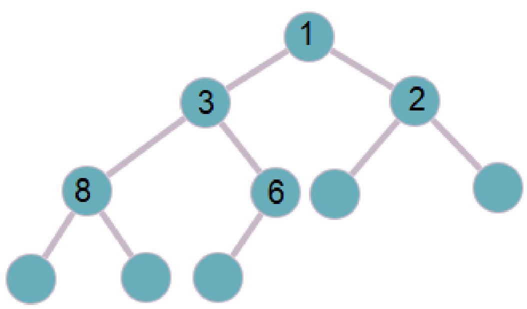
1. We choose the **smallest** element for the root.



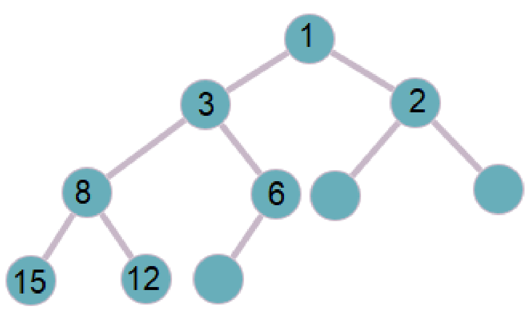
2. n = 9 from that 6 are chosen for left side binary tree: {6,3,9,12,15,8} and remaining 3 for right side binary tree: {17,2,20}, among them **smallest from each** will be the root for that level node.



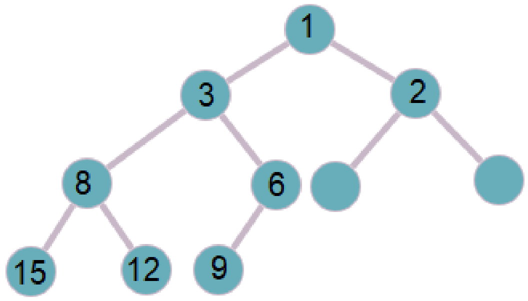
3. n = 5 from which **3 are chosen for left** side binary tree: {8,12,15} and remaining **2 for right** side binary tree: {6,9}, among them smallest from each will be the root for that level node.



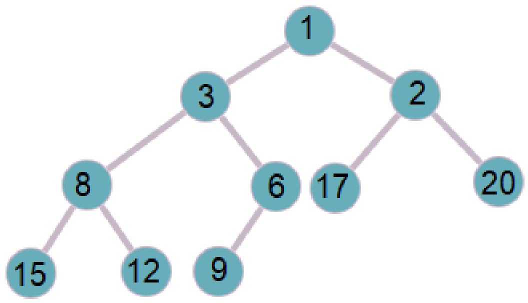
4. n = 2: {12,15} can be arranged in any way under its root on that level.



5. n = 1: {9} can be arranged in **only 1 way**.



6. n = 2: {17,20} can be arranged in any way under its root on that level.



Example 2: In how many ways n=17 distinct integers can be arranged in a Heap.

Here, n = 17

$H(17) = H(9) \times H(7) \times {}_{16}C_9$

$= H(5) \times H(3) \times {}_{8}C_5 \times H(3) \times H(3) \times {}_{6}C_3 \times {}_{16}C_9$

$= H(3) \times H(1) \times {}_{4}C_3 \times H(3) \times {}_{8}C_5 \times (H(3))^2 \times {}_{6}C_3 \times {}_{16}C_9$

$= (H(3))^4 \times H(1) \times {}_{4}C_3 \times {}_{8}C_5 \times {}_{6}C_3 \times {}_{16}C_9$

$= (2)^4 \times 1 \times 4 \times 56 \times 20 \times 11440$

$= 820019200$