10th October 2022 (Monday)
Scribed Notes 23

Student ID :-
202212111
202212112
202212113
202212114
202212115 (Absent)

From previous lectures we recall :-
Closed form/Open form
Rule of sum
Rule of product
Principle of Inclusion & Exclusion
Counting by cases
Binomial theorem
Counting using recursion

Computational Problem: It is a mathematical function from a domain of legal inputs to a co-domain of corresponding expected outputs.
Remark : There is a difference between computational problems and computation. A problem is a function that has a domain and a co-domain
Merge sort, quick sort etc. are different sorting algorithms to one problem and algorithms interact with problems.

Binary Heap

It has 2 Aspects:-

1. Structural - It is complete upto some level
2. Key value - The elements are arranged in a specific manner and that is true recursively for all the nodes .

Two types:-

1. MinHeap-The key at root is always less than it's children. This is true recursively for all nodes

2. MaxHeap-The key at root is always greater than it's children. This is true recursively for all nodes

Number of ways to accommodate elements in Min heap

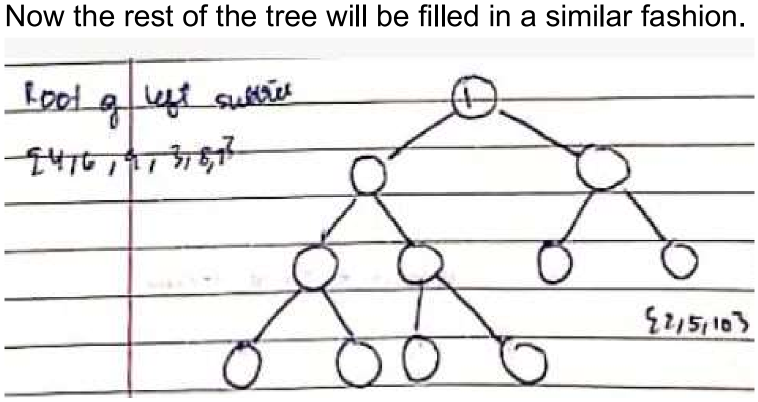
If we have n elements to accommodate in a Min binary heap Because of the key value property the value of root is fixed and is $^{n}C_{1}$ but we have full flexibility to divide the remaining ones into whichever subtree we want keeping the key value .

- There is only one element as the root, it must be the smallest number. Now we have n-1 remaining elements.
- We can divide the remaining n-1 elements to the left sub-tree and right subtree in any way we want . Now if there are L elements in the left sub-tree and R elements in the right sub-tree.
- Then, L + R = n-1 and The number of ways to assign elements to left-subtree is $^{n-1}C_{L}$ and in left subtree we again recursively have the same procedure
- Using Rule of product
    - Recursive Function : $H(n) = {}^{n-1}C_{L(n)} \times H(L(n)) \times H(R(n))$

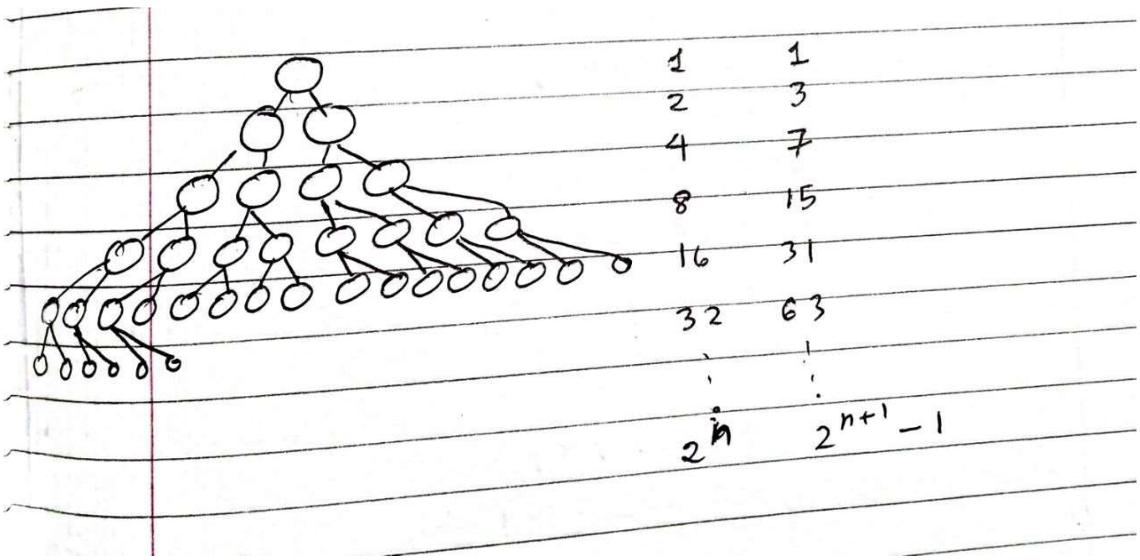
H(n) is not closed because it is recursive

One possible arrangement in Heap with n=10:
Consider integers as : 1,4,6,9,3,8,7,2,5,10

Here 1 will go to the root because of key property
And now the rest 9 can be divided randomly in to the two sub trees ,we will choose these 6 for left side {4,6,9,3,8,7} among them 3 which is the smallest will be the root for that level node.
{2,5,10} is for right subtree and 2 is the root for that level.
Now the rest of the tree will be filled in a similar fashion.



While dividing the elements we have to keep in mind the structural property if, say we have to accommodate 37 elements then we can accommodate 31 elements upto level 4 and then for the remaining 37-31 = 6 elements we have to see if 6 is greater or lesser than the half of the no. of elements that can be accommodate by level 5 or not if greater , then we will fill the elements in the left subtree and accommodate the rest in right subtree and if not then only fill the elements in left subtree so as to keep one subtree complete . In this case 6 is less than 32/2 = 16 and so we will put the rest in the left subtree.



For example , This is how we will divide 101 elements ……