6th October 2022 (Thursday)

Scribed Notes 22

Student ID :-

202212106

202212107 (Absent)

202212108 (Absent)

202212109 (Absent)

202212110

COUNTING

Binary search tree and Binary heap: difference between two in terms of memory is that binary heap has contiguous memory allocation and BST does not have contiguous memory.
It is contiguous because heaps are complete tree.

Geometric progression/sequence

a, ar, ar^2, ar^3, ….

GP is automatic after two terms.
For example, take any random two terms
7, -49
We can get next term by multiplying it by r
r = -49/7 = -7

7, -49, 343, -2401, 16807…….

Geometric Series

S = $a + ar + ar^2 + ar^3 + \ldots + ar^{n-1}$

$rS = ar + ar^2 + ar^3 + \ldots + ar^{n-1} + ar^n$

subtracting rS from S gives

$S(r-1) = ar^n - a$
$S = a(r^n - 1) / (r - 1)$

The geometric series formula for infinite terms is given as:

If $|r| < 1$, $S_\infty = a / 1-r$

Arithmetic Progression

$a, (a + d), (a + 2d), (a + 3d), \ldots\ldots, (a + d(n-1))$

Arithmetic Series

$\mathbf{S} = a + (a + d) + (a + 2d) + (a + 3d) + \ldots\ldots + (a + d(n-1))$

$$S_n = na + d \sum_{i=1}^{n-1} i$$

Arithmetico geometric series

$\mathbf{S_a} = a + (a + d) + (a + 2d) + \ldots\ldots + (a + d(n-1))$
$\mathbf{S_g} = g + gr + gr^2 \ldots\ldots + gr^{n-1}$

$\mathbf{S_{ag}} = ag + gr(a+d) + gr^2(a+2d) + gr^{n-1}(a+d(n-1))$

Computational Problem is a mathematical function from a domain of legal inputs to a codomain of corresponding expected output.

Algorithm is a systematic step by step method to solve a computational problem.

Binary Heap

Max heap: All the element below parent node is less than the parent node.
Min heap: All the element below parent node is greater than the parent node.

Heaps are complete binary tree so number of nodes till second last level will be in $(2^1 - 1), (2^2-1), (2^3-1), \ldots (2^k-1)$

At Every level number of nodes are 2^n.
For any heap sum till second last level is 2^h-1. Where h is height of tree.

Number of left node of heap of 101 nodes.

So the number of node till second last level is 63. Out of 63 left subtree has 31 nodes.
And at the last level leaf node are 38 out of that left side has 32 node
So 31 + 32 = 63 nodes are on the left of root.

In How many ways you can arrange 101 nodes in binary heap?

$H(101) = 100C_{63} . H(63) . H(37)$
$H(63) = 62C_{31} . H(31) . H(31)$
$H(31) = 30C_{15} . H(15) . H(15)$
$H(15) = 14C_{7}. H(7) . H(7)$
$H(7) = 6C_{3} . H(3) . H(3)$
$H(3) = 2C_{1}. H(1) . H(1)$

Same function call would be for H(37)

Recursively doing this would give the answer.

$H(nodes) = (nodes-1)C_{left} . H(left) . H (right)$

Where left = number of nodes in left subtree
Right = number of nodes in right subtree
Nodes is number of nodes