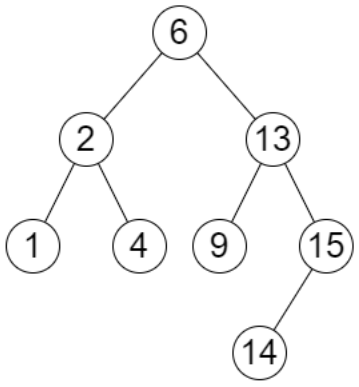
Algorithms Laboratory (CS29203)
Lab Test - II
Department of CSE, IIT Kharagpur

7th November 2024

Question-1

(25 points)
Consider a Binary Search Tree (BST) having n number of nodes. The tree nodes contain only integer values. You are also given an integer array `S[ ]` of size m. Your task is to find a 2D array `K` of size m where `K[i] = [Xi, Yi]` such that,

- `Xi` is the largest value in the tree which is less than or equal to `S[i]`. If there is no such value, then the answer is -1.
- `Yi` is the smallest value in the tree which is greater than or equal to `S[i]`. If there is no such value, then the answer is -1.



For example, consider the BST in the above figure. The input of the tree is taken as follows:

$$[6, 2, 13, 1, 4, 9, 15, -1, -1, -1, -1, -1, -1, 14]$$

In this format, the nodes are defined in level order from left to right and a missing child is denoted as -1. Let `S = [2,5,16]`. Then the 2D array `K` will be `K = [[2,2], [4,6], [15,-1]]`. Following is the detailed explanation:

- Largest number in the tree which is ≤ 2 is 2, and the smallest number ≥ 2 is also 2. Then the first entry of the array `K` will be $[2, 2]$.
- Largest number in the tree which is ≤ 5 is 4, and the smallest number ≥ 5 is 6. Then the second entry of the array `K` will be $[4, 6]$.
- Largest number in the tree which is ≤ 16 is 15, and the smallest number ≥ 16 does not exist. Then the third entry of the array `K` will be $[15, -1]$.

Important: You are **not** allowed to implement **array** based representation of trees (you have to use standard linked representation of trees).

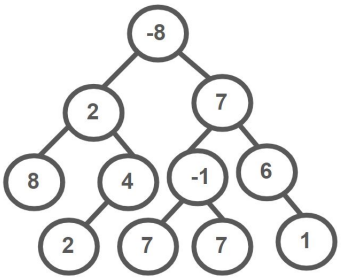
Example:

```
(Input) Enter the number of nodes in the tree: 3
        Enter the tree nodes [ ]: 4  -1  9
        Enter the array S[ ]: 3

(Output) K = [[-1, 4]]
```

Question-2

(35 points)
Consider a binary tree with n nodes. You are given an integer x, and you have to count the total number of paths in the tree whose sum is equal to x. The path can be any path that is on the root-to-leaf path in the binary tree, or it can be a direct path from the root to a leaf. Alternatively put, a path from node i to node j is valid if i is an ancestor of j. For example, consider the following binary tree and $x = 6$:



Then there are 7 paths in the tree with a path sum of 6 as follows: $-8 \to 7 \to 6 \to 1$, $2 \to 4$, $4 \to 2$, $7 \to -1$, $-1 \to 7$, $-1 \to 7$, 6. An idea to solve the problem is to traverse the binary tree and for every node, check if there is a path starting with it having the sum of all its nodes equal to x recursively. The time complexity of the solution will be $O(n^2)$. *Actually this problem can also be solved in $O(n)$, but this is not mandatory for this question.*

Example:

```
(Input) x = 6
        Building the tree:
                Node *root = newNode(-8);
                root->left = newNode(2);
                root->right = newNode(7);
                root->left->left = newNode(8);
                root->left->right = newNode(4);
                root->right->left = newNode(-1);
                root->right->right = newNode(6);
                root->left->right->left = newNode(2);
                root->right->left->left = newNode(7);
                root->right->left->right = newNode(7);
                root->right->right->right = newNode(1);

(Output) Number of paths = 7
```

Question-3

(40 points)
Suppose that you are given an unlimited supply of coins of given denominations. The goal is to find the **minimum** number of coins required to get the desired change. For example, consider a set of coin denominations `S = [1, 3, 5, 7]`. Given a desired change of 15, the minimum number of required coins is 3 because of the following possibilities:

- $7 + 7 + 1$
- $5 + 5 + 5$
- $3 + 5 + 7$

Similarly if the desired change is 18, then the answer is 4 because of the following possibilities:

- $7 + 7 + 3 + 1$
- $5 + 5 + 5 + 3$
- $7 + 5 + 5 + 1$

Your task is to solve the problem in two ways:

(a) (20 points) Solve and implement the problem using **recursion**. Of course the complexity will be exponential. *Recur to see if the total can be reached by including the coin or not for each coin of given denominations. If choosing the current coin resulted in the solution, update the minimum number of coins needed. Finally, return the minimum value we get after exhausting all combinations.*

(b) (20 points) Improve the solution by implementing **dynamic programming** approach. *Compute T[i] for each* $1 <= i <= target$*, which stores the minimum number of coins needed to get a total of i.* The time complexity should not increase O(n*target), where n is the total number of coins and target is the total change required.

Example:

```
(Input)  Enter the number of coin denominations n: 4
         Enter the target value: 15
         Enter the array S[ ]: 1  2  3  4

(Output) The minimum number of coins required to get the desired change is: 4
```