Max Marks:

Question: 3

Given a LinkedList, where each node contains small case characters, you are asked to form a 'strong password' using characters from that LinkedList. A 'strong password' is the one in which no two characters are repeating. The output password must be a continuous subset of the given LinkedList.

Find the length of the strongest password that can be formed using the input LinkedList.

Example 1:

Input: s = a->b->c->a->b->c->b->b

Output: 3

Explanation: The answer is a->b->c, with the length of 3.

Example 2:

Input: s = p->w->w->k->e->w

Output: 3

Explanation: The answer is w->k->e, with the length of 3.

Notice that the answer must be a continuous subset, p->w->k->e is a subset and not a continuous subset

Expected Time Complexity: O(n)

Expected Space Complexity: O(1)

Answer:

Previous

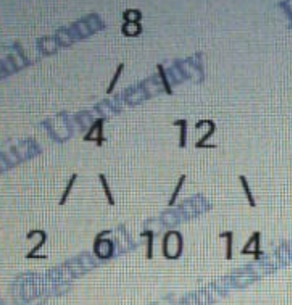
Clear Answer

Question: 2

Given a binary search tree and a key value (key may or may not be a node value), find the floor and ceil value for that particular key value.

Floor Value Node: Node with greatest data lesser than or equal to key value. If not found, return -1
Ceil Value Node: Node with smallest data larger than or equal to key value. If not found, return -1

For example, Let's consider the Binary Search Tree below -



Key: 11 Floor: 10 Ceil: 12
Key: 1 Floor: -1 Ceil: 2
Key: 6 Floor: 6 Ceil: 6
Key: 15 Floor: 14 Ceil: -1

Answer:

Language: C

Previous

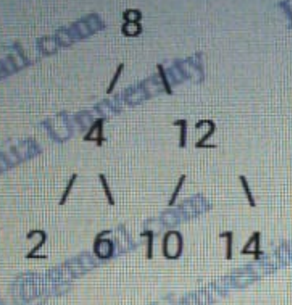
Clear Answer

Question: 2

Given a binary search tree and a key value (key may or may not be a node value), find the floor and ceil value for that particular key value.

Floor Value Node: Node with greatest data lesser than or equal to key value. If not found, return -1
Ceil Value Node: Node with smallest data larger than or equal to key value. If not found, return -1

For example, Let's consider the Binary Search Tree below -



Key: 11 Floor: 10 Ceil: 12
Key: 1 Floor: -1 Ceil: 2
Key: 6 Floor: 6 Ceil: 6
Key: 15 Floor: 14 Ceil: -1

Answer:

Language: C

Previous

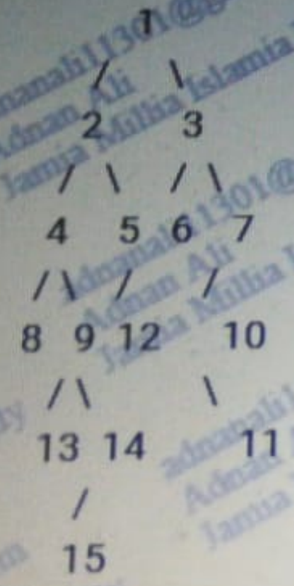
Clear Answer

Question: 1

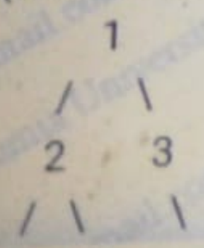
Given a binary tree, a complete path is defined as a path from root to a leaf. The sum of all nodes on that path is defined as the sum of that path.
Given a number K, you have to remove (prune the tree) all nodes which don't lie in any path with sum>=k.

Note: A node can be part of multiple paths. So we have to delete it only in case when all paths from it have sum less than K.

Consider the following Binary Tree



For input k = 20, the tree should be changed to following
(Nodes with values 6 and 8 are deleted)



< Previous

Clear Answer