26-02-2024 Prog-14.

BFS & DFS

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 100
Struct Node {
 int vertx;
 Struct Node * next;
};
Struct Node * CreateNode (int v){
 Struct Node * newNode = (struct Node *) malloc (size of (struct No
 newnode -> vertex = v;
 newnode -> next = NULL;
 return newnode;
}
void addEdge (Struct Node* adj[], int source, int dest) {
 Struct Node * newnode = CreateNode (dest);
 adj [source] = newNode;
}
void DFS (struct Node * adj[], int v, int checked []){
 Checkd [v] = 1;
 printf ("%d", v);
 Struct Node * temp = adj [v];
 while [temp != NULL){
 int adjvertex = temp -> vertex;
 if (! checkd [adjvertex]){
 DFS (adj, adjvertex, checkd);
 }
temp = temp -> next;
}
}
```

26.02.2024

```
void BFS(struct Node * adj[], int start vertex) {
  int checkd [MAX]= {0};
  int queue [MAX];
  int front=0, rear = 0;
  cheKd [start vertex] = 1;
  queue [rear++] = start vertex;
  while (front < read ) {
    int currentvertex = queue [front ++];
  printf ("%d ", currentvertex);
  struct Node * temp = adj [currerd vertex];
  while (temp! =NULL) {
    int adjverlex = temp -> vertex;
    if (! cheKd [adj vertex]) {
       checkd [adjvertex] = 1;
       queue [rear++] = adj vertex;
  }
  temp = temp->next;
   }
}
int main () {
  int V = 5;
  Struct Node *  adj [MAX] = {NULL};
  add Edge (adj, 0, 1);
  add Edge (adj, 0, 2);
  add Edge (adj, 1, 3);
  add Edge (adj, 1, 4);
  .
printf ("DFS Traversal: ");
DFS (adj, 0, visited);
printf ("\n");
```

26-02-2024

```
for (int i = 0 ; i < v ; i++) {
visited [i] = 0;
}
printf ("BFS Traversal :");
BFS (adj, 0, visited);
print ("\n");
return 0;
}
```

<u>o/p</u>

BFS traversal starting from vertex 0: 0 2 1 3

DFS traversal starting from vertex 0: 0 2 3 1

```
BFS
0 2 1 3
DFS
0 2 3 1
Process returned 0 (0x0)   execution time : 0.031 s
Press any key to continue.
```

26-02-2024

Leet code - Delete a node in BST

=>
```
Struct Tree Node * minValueNode (Struct Tree Node * node){
Struct Tree Node * current = node;

while (current && current -> left != NULL)
current = current -> left;
return current;
}
struct TreeNode * deleteNode (struct TreeNode * root, int key){
if (root == NULL) return root;
if (key < root -> val)
root -> left = delete Node (root -> left, key);

else if (key > root -> val)
root -> right = deleteNode (root -> right, key);

else {
if (root -> left == NULL){
Struct Tree Node * temp = root -> right;
free (root);
return temp;
} else if (root -> right == NULL){
Struct Tree Node * temp = root -> left;
free (root);
return temp;
}
Struct Tree Node * temp = minValueNode (root -> right);
root -> val = temp -> val;
root -> right = deleteNode (root -> right, temp -> val);
}
```

26-02-2024

return root;
}

2) Leet Code - Bottom left tree value

=> void findBottomLeft(struct TreeNode * node, int depth,
int * maxDepth, int * leftmostvalue){
if (node == NULL)
return;
if (depth > *maxDepth) {
*maxDepth = depth;
*leftmostvalue = node->val;
}
findBottomLeft(node->left, depth+1, maxDepth, leftmostvalue);
findBottomLeft(node->right, depth+1, maxDepth, leftmostvalue);
}
int findBottomLeftvalue(struct TreeNode* root){
int maxDepth=0;
int leftmostvalue = root->val;

findBottomLeft(root, 1, &maxDepth, &leftmostvalue);

return leftmostvalue;
}

Description Editorial Solutions **Submissions**

All Submissions

Accepted

user0435PL submitted at Feb 26, 2024 10:33

Editorial Solution

Runtime

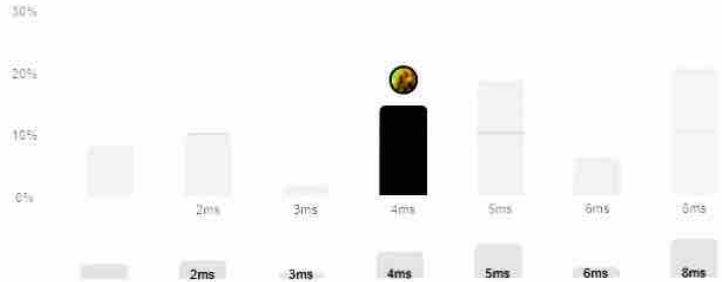
4 ms

Beats 79.17% of users with C

Memory

8.78 MB

Beats 82.50% of users with C



Code C

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     struct TreeNode *left;
 *     struct TreeNode *right;
 * };
 */
```

View more

More challenges

- 872. Leaf-Similar Trees
- 897. Increasing Order Search Tree
- 655. Print Binary Tree

Code

C Auto

```
19 findBottomLeft(node->right, depth + 1, maxDepth, leftmostValue);
20 }
21
22 int findBottomLeftValue(struct TreeNode* root) {
23 int maxDepth = 0;
24 int leftmostValue = root->val;
25
26 findBottomLeft(root, 1, &maxDepth, &leftmostValue);
27
28 return leftmostValue;
29 }
30
```

Saved to local

Ln 30, Col 1

Testcase Test Result

Accepted Runtime: 4 ms

- Case 1
- **Case 2**

Input

root =

[1,2,3,4,null,5,6,null,null,7]

Output

7

Expected

7

Contribute a testcase

Description Editorial Solutions **Submissions**

All Submissions

Accepted

user0435PL submitted at Feb 26, 2024 09:33

Editorial Solution

Runtime

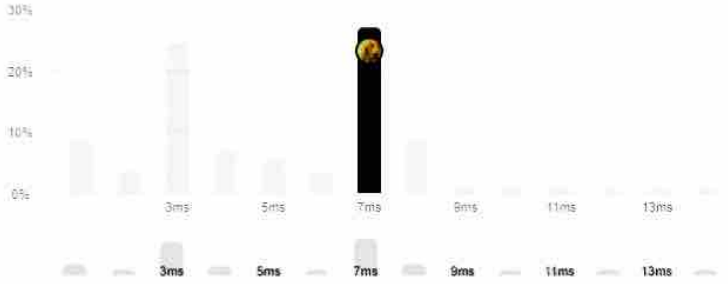
7 ms

Beats 45.53% of users with C

Memory

6.53 MB

Beats 93.81% of users with C



Code C

```c
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     struct ListNode *next;
 * };
 */
#include <stdio.h>
```

View more

More challenges

- **725. Split Linked List in Parts**

Code

C Auto

```c
}

odd->next = evenHead;
return head;
}
struct ListtNode* newNode(int val)
{
struct ListtNode* node = (struct ListtNode*)malloc(sizeof(struct ListtNode));
node->val = val;
node->next = NULL;
return node;
}

```

Saved to local

Ln 35, Col 1

Testcase Test Result

Case 1 Case 2

head =

[1,2,3,4,5]