```c
// Recursive optimized Cprogram to find the diameter of a Bi
#include <stdio.h>


// A binary tree node has data, pointer to left child
// and a pointer to right child
struct node {
    int data;
    struct node *left, *right;
};

// function to create a new node of tree and returns pointer
struct node* newNode(int data);

int max(int a, int b) { return (a > b) ? a : b; }

int diameterOpt(struct node* root, int* height)
{
    // lh --> Height of left subtree
    // rh --> Height of right subtree
    int lh = 0, rh = 0;

    // ldiameter --> diameter of left subtree
    // rdiameter --> Diameter of right subtree
    int ldiameter = 0, rdiameter = 0;

    if (root == NULL) {
        *height = 0;
        return 0; // diameter is also 0
    }

    // Get the heights of left and right subtrees in lh and
    // rh And store the returned values in ldiameter and
    // ldiameter
    ldiameter = diameterOpt(root->left, &lh);
    rdiameter = diameterOpt(root->right, &rh);

    // Height of current node is max of heights of left and
    // right subtrees plus 1
    *height = max(lh, rh) + 1;

    return max(lh + rh + 1, max(ldiameter, rdiameter));
}

// Helper function that allocates a new node with the
// given data and NULL left and right pointers.
struct node* newNode(int data)
{
    struct node* node
        = (struct node*)malloc(sizeof(struct node));
    node->data = data;
    node->left = NULL;
    node->right = NULL;

    return (node);
}

// Driver Code
int main()
{

    /* Constructed binary tree is
            1
            / \
        2    3
        / \
    4    5
    */
    struct node* root = newNode(1);
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->left->right = newNode(5);

    int height = 0;

    // Function Call
    printf("Diameter of the given binary tree is %d",diameterOpt(ro


    return 0;
}

// This code is contributed by probinsah.

```