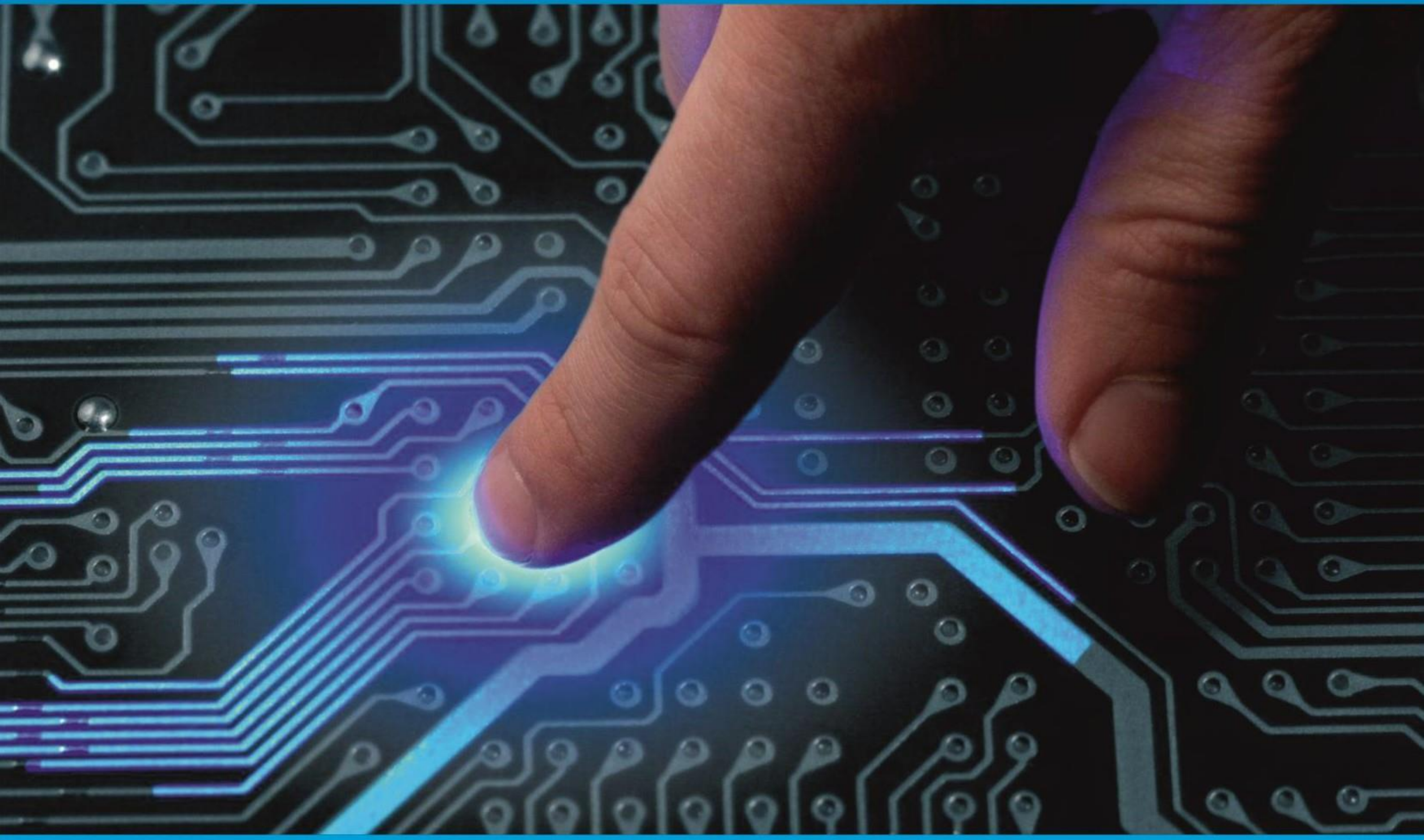
e-ISSN: 2320-9801 | p-ISSN: 2320-9798



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 8, Issue 12, December 2020

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 7.488

Design an Optimal Shape Binary Search Tree using C-language

Sanskar Gupta[1],Somya Seth[2], Divyanshi Sahu[3]

B. Tech Student, Department of Computer Science and Engineering, SR Group Of Institutions, Jhansi, UP, India[1]

B. Tech Student, Department of Computer Science and Engineering, SR Group Of Institutions, Jhansi, UP, India[2]

B. Tech Student, Department of Computer Science and Engineering, SR Group Of Institutions, Jhansi, UP, India[3]

ABSTRACT: Binary Search Tree (BST*)* is a node-based binary tree data structure. It is one of the most widely used techniques for searching in non-linear data structure. If the BST is not designed in Optimal shape then the searching and insertion may need extra number of comparisons. In present literature, a number of BST algorithms have been proposed to design the BST in optimal shape. In this study, we are going to give an optimal shape to a BST using a Code/Program in C-language *i.e.*Input random numbers. User will give n number of inputs and their output will be processed through the code and will maintain a BST. With this output, user will check whether an optimal shape BST is designed or not.

KEYWORDS: C-Program, Height of Tree, Non-linear Data Structure.

I. INTRODUCTION

The logical or mathematical model of a particular organization of data is known as *Data Structure*. It has been classified into two types: Primitive Data Structures and Non-primitive Data Structures. Non-primitive data structures are further classified into linear data structures and non-linear data structures.

Trees are important segment of non-linear data structures. There are different types of trees like binary search trees (BST), AVL (Height Balancing Tree), Threaded Binary tree and so on.

Binary Search Tree (BST) is one of the most important data structures in computer science. It enables one to search for and find an element with an average running time $f(n) = O(log_2 n)$. It also enables one to easily insert and delete an element.

BST is generally constructed by comparing the data with its root node. If this node having greater value then it is placed on the right side of the root node, while lesser value is placed on the left side of the root. We have used simple C-program having two inputs and one output. The user will randomly insert the value in the input variables and with respect to output values the BST will be maintained.

In Present literature, several BST algorithms have been proposed to maintain the BST in Optimal shape [3,5-11].
In this paper, we will design the new BST using a C-program/Code.

II. LITERATURE REVIEW

Not much work has been done in designing and maintaining an optimal shape BST using C-language. But because in this research paper, we are epically focusing on the binary search tree. There have been many questions regarding with the optimal shape of BST using C-language which have been left unanswered like at what values the shape can be optimal? How they can be formed? Etc. In this paper, we describe the results of the study of BST and its shape as follows: Given a task to perform with the Code/Program (Input n numbers of input and With its output, a BST will be maintained or designed).

The result of this study yields a specific research contribution to the formation of BST i.e. an optimal shape or structure. With the different input values, the output will varies accordingly and with that output, BST will be maintained.

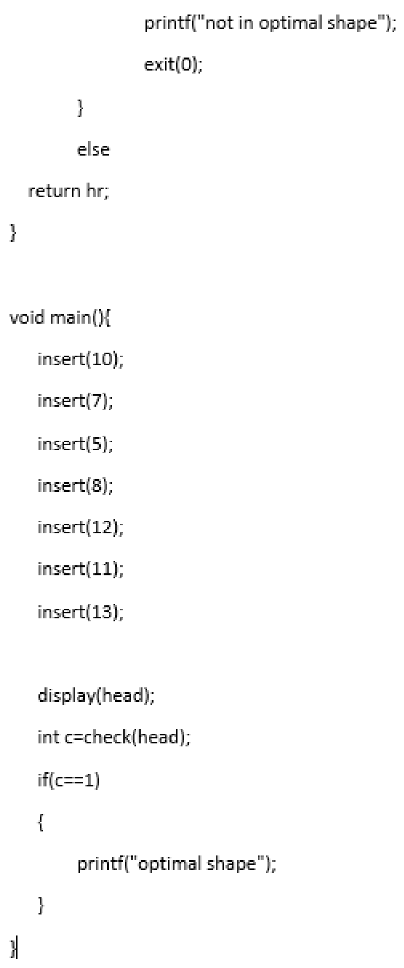
III. CODE AND OUTPUT DIAGRAMS

```
#include<stdio.h>

#include<conio.h>

#include<stdlib.h>


struct node {

    struct node *left;

    int data;

    struct node *right;

   } *head = NULL;


void insert(int n) {

    struct node *t=head;

    if (head == NULL){

         struct node *p = (struct node*)malloc(sizeof(struct node));

          p->data = n;

          p->right = NULL;

          p->left = NULL;

          head = p;

    }

    else{

         if(t->data > n){

                insert_left(n,t);

         }

         else{

                insert_right(n,t);

         }

   }

}


void insert_left(int n,struct node *p){

    if(p->left == NULL && n < p->data){

        struct node *t = (struct node*)malloc(sizeof(struct node));

         p->left = t;

         t->left = NULL;

         t->data = n;

         t->right = NULL;

    }

    else if(n < p->data && p->left!=NULL){

         p = p->left;

         insert_left(n,p);

    }

    else{

        insert_right(n,p);

    }

}


void insert_right(int n,struct node *p){


    if(p->right == NULL && n > p->data){

        struct node *t = (struct node*)malloc(sizeof(struct node));

         p->right = t;

         t->left = NULL;

         t->data = n;

         t->right = NULL;

    }

    else if(n > p->data && p->right!=NULL){

         p = p->right;

         insert_right(n,p);

    }

    else{

        insert_left(n,p);
```

Continued…

```
    }
}

void display(struct node *p){
        if(p == NULL){
            return ;
        }
        printf("%d\n",p->data);
        display(p->left);
        display(p->right);
}

int check(struct node *p)
{
        int hl=0,hr=0;
        if(p==NULL)
        {
                return 1;
        }
        hl=1+heightleft(p->left);
        hr=1+heightright(p->right);
        if((hl-hr)!=0)
        {
                printf("not in optimal shape");
                exit(0);
        }
        else
   {
        return 1;
   }
}

int heightleft(struct node *t)
{
        int hl=0,hr=0;
        if(t==NULL)
        {
                return -1;
        }
        hl=1+heightleft(t->left);
        hr=1+heightright(t->right);
        if((hl-hr)!=0)
        {
                printf("not in optimal shape");
                exit(0);
        }
        else
   return hl;
}

int heightright(struct node *t)
{
        int hl=0,hr=0;
        if(t==NULL)
        {
                return -1;
        }
        hl=1+heightleft(t->left);
        hr=1+heightright(t->right);
        if((hl-hr)!=0)
        {
```

Continued…



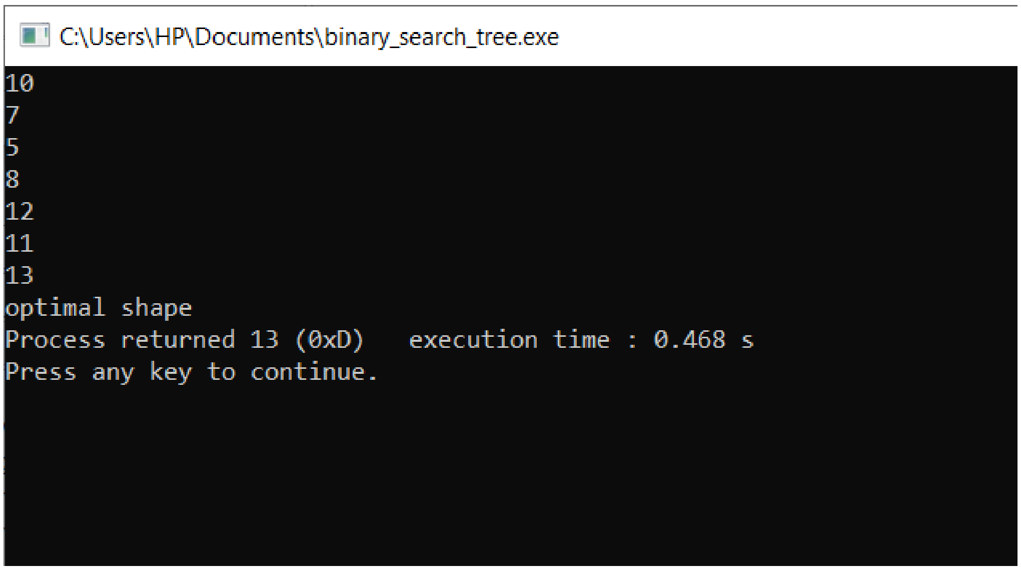
```
                printf("not in optimal shape");
                exit(0);
            }
            else
  return hr;
}

void main(){
    insert(10);
    insert(7);
    insert(5);
    insert(8);
    insert(12);
    insert(11);
    insert(13);

    display(head);
    int c=check(head);
    if(c==1)
    {
        printf("optimal shape");
    }
}
```

Figure 3.1C Program which yields and Check an Optimal structure of BST

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 7.488 |

|| Volume 8, Issue 12, December 2020 ||

| DOI: 10.15680/IJIRCCE.2020.0812030 |



```
C:\Users\HP\Documents\binary_search_tree.exe
10
7
5
8
12
11
13
optimal shape
Process returned 13 (0xD)   execution time : 0.468 s
Press any key to continue.
```

Figure 3.2 Output of the above **Code**

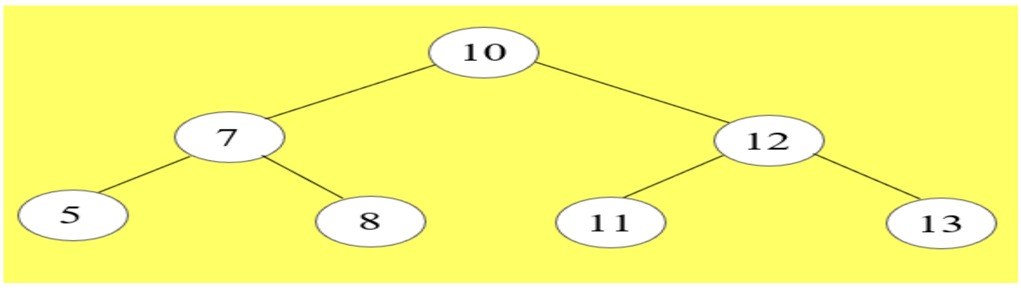


Figure** 3.3 Implementation of above **Output

IV. CONCLUSION

This is a small effort made in the field of BST and one of its most important features *i.e. Optimal Shape*. But now maintaining an optimal shape BST using C-language, is more easy and convenient. This paper helps in reduction of time complexity which generally causes to design a BST.

REFERENCES

1. **Seymour Lipschutz ;** A textbook on "Data Structures with C".
2. Patel, Z., Senjaliya, N., & Tejani, A. (2019). AI-enhanced optimization of heat pump sizing and design for specific applications. International Journal of Mechanical Engineering and Technology (IJMET), 10(11), 447-460.
3. Inayat-ur-Rehman, Saif-ur-Rehman Khan, M.Sikandar Hayat Khayal "A Survey on Maintaining Binary Tree in Optimal Shape", 2009 International conference on Information Management and Engineering.
4. R.Sugumar, C.Jayakumar, A.Rengarajan, "An Efficient Blocking Algorithm for Privacy Preserving Data Mining", International Journal of Computing, USA, Vol. 3, Issue. 8, pp. 73-77, August 2011
5. Senjaliya, N., & Tejani, A. (2020). Artificial intelligence-powered autonomous energy management system for hybrid heat pump and solar thermal integration in residential buildings. International Journal of Advanced Research in Engineering and Technology (IJARET), 11(7), 1025-1037.
6. Geeks_for_Geeks, "Online Platform".