EXPERIMENT 9

Experiment No: 9 **Date:** 28/04/2021

Aim: Implementation of OBST using Dynamic Programming and estimate its step count

Theory:

OPTIMAL BINARY SEARCH TREE

- Optimal binary search tree (Optimal BST), sometimes called a weight-balanced binary tree, is a binary search tree which provides the smallest possible search time (or expected search time) for a given sequence of accesses (or access probabilities).
- When we know the frequency of searching each one of the keys, it is quite easy to compute the expected cost of accessing each node in the tree. An optimal binary search tree is a BST, which has minimal expected cost of locating each node
- Suppose we are given a list of keys k1 < k2 < . . . < kn, and a list of probabilities pi that each key will be looked up. An optimal binary search tree is a BST T that minimizes the expected search time.

$$\sum_{i=1}^{n} p_i(\mathrm{depth}_T(k_i) + 1).$$

- Where the depth of the root is 0. We will assume WLOG that the keys are the numbers 1, 2, . . . , n.

EXPERIMENT 9

ALGORITHM WRITING

- The optimal cost for freq[i..j] can be recursively calculated using following formula.

optcost\left (i, \right j) = \sum_{k=i}^{j} freq \begin{bmatrix}k\end{bmatrix} + min_{r=i}^{j}\begin{bmatrix} optcost(i, r-1) + optcost(r+1, j) \end{bmatrix}

- We need to calculate optCost(0, n-1) to find the result.
- The idea of above formula is simple, we one by one try all nodes as root (r varies from i to j in second term). When we make rth node as root, we recursively calculate optimal cost from i to r-1 and r+1 to j.
- We add sum of frequencies from i to j (see first term in the above formula), this is added because every search will go through root and one comparison will be done for every search.

ALGORITHM

Algorithm OBST(p,q,n)

//given n distinct identifiers a1<a2<.....an

//probabilities p[i], 1<=i<=n and q[i], 0<=i<=n this algorithm computes

//the cost[i][j] of optimal binary search trees tij for identifiers ai+1...aj

EXPERIMENT 9

```
//It also computes r[i][j] the root of tij

//w[i][j] is weight of tij

{

        for i:=0 to n-1 do

        {

          w[i,i] := q[i]

          c[i,i] := 0

          r[i,i] := 0

          w[i,i+1] := q[i] + q[i+1] +p[i+1]

          r[i,i+1] := i+1

          c[i,i+1] := q[i] + q[i+1] + p[i+1]

        }

        w[n,n] = q[n]

        r[n,n] = 0

        c[n,n] = 0

   for m:=2 to n do

   {

     for i:=0 to n-m do

     {
```

EXPERIMENT 9

```
            j := i+m
            w[i,j] := w[i,j-1]+p[j]+q[j]
            k := Find(c,r,i,j)
            //A value of l in range r[i,j-1]<=l<=r[i+1,j] that minimizes
c[i,l-1]+c[l,j]
            c[i,j] := w[i,j] + c[i,k-1] + c[k,j]
            r[i,j] := k
          }
        }
    }
```

Algorithm find(c,r,i,j)

```
{
    min : = inf
    for m := r[i,j-1] to r[i+1,j] do
    {
      if( c[i,m-1]+c[m,j]<min) then
      {
```

```
            min := c[i,m-1]+ c[m,j]

            l := m

          }

        }

        return l

    }
```

COMPLEXITY ANALYSIS

- The time efficiency is Θ (n^3) but can be reduced to Θ(n^2) by taking advantage of monotonic property of the entries.
- The monotonic property is that the entry R[i,j] is always in the range between R[i,j-1] and R[i+1,j].
- The space complexity is Θ(n^2) as the algorithm is reduced to filling the table.

EXPERIMENT 9

TRACING WITH EXAMPLE

	0	1	2	3	4
0	$w_{00} = 1/5$ $c_{00} = 0$ $r_{00} = 0$	$w_{11} = 1/10$ $c_{11} = 0$ $r_{11} = 0$	$w_{22} = 1/5$ $c_{22} = 0$ $r_{22} = 0$	$w_{33} = 1/20$ $c_{33} = 0$ $r_{33} = 0$	$w_{44} = 1/20$ $c_{44} = 0$ $r_{44} = 0$
1	$w_{01} = 7/20$ $c_{01} = 7/20$ $r_{01} = 1$	$w_{12} = 1/2$ $c_{12} = 1/2$ $r_{12} = 2$	$w_{23} = 7/20$ $c_{23} = 7/20$ $r_{23} = 3$	$w_{34} = 3/20$ $c_{34} = 3/20$ $r_{34} = 4$	
2	$w_{02} = 3/4$ $c_{02} = 11/10$ $r_{02} = 2$	$w_{13} = 13/20$ $c_{13} = 1$ $r_{13} = 2$	$w_{24} = 9/20$ $c_{24} = 3/5$ $r_{24} = 3$		
3	$w_{03} = 9/10$ $c_{03} = 8/5$ $r_{03} = 2$	$w_{14} = 3/4$ $c_{14} = 27/20$ $r_{14} = 2$			
4	$w_{04} = 1$ $c_{04} = 39/20$ $r_{04} = 2$				

1. $w_{00} = q(0) = 1/5$

$c_{00} = 0$

$r_{00} = 0$

EXPERIMENT 9

2. $w_{11} = q(1) = 1/10$
$c_{11} = 0$
$r_{11} = 0$

3. $w_{22} = q(2) = 1/5$
$c_{22} = 0$
$r_{22} = 0$

4. $w_{33} = q(3) = 1/20$
$c_{33} = 0$
$r_{33} = 0$

5. $w_{44} = q(4) = 1/20$
$c_{44} = 0$
$r_{44} = 0$

6. $w_{01} = p(1) + q(1) + w_{00} = 1/20 + 1/10 + 1/5 = 7/20$
$c_{01} = \min\limits_{\substack{0<k\leq 1 \\ k=1}} \{c_{00} + c_{11}\} + w_{01} = (0+0) + 7/20 = 7/20$
$r_{01} = 1$

7. $w_{12} = p(2) + q(2) + w_{11} = 1/5 + 1/5 + 1/10 = 1/2$
$c_{12} = \min\limits_{\substack{1<k\leq 2 \\ k=2}} \{c_{11} + c_{22}\} + w_{12} = (0+0) + 1/2 = 1/2$
$r_{12} = 2$

8. $w_{23} = p(3) + q(3) + w_{22} = 1/10 + 1/20 + 1/5 = 7/20$
$c_{23} = \min\limits_{\substack{2<k\leq 3 \\ k=3}} \{c_{22} + c_{33}\} + w_{23} = (0+0) + 7/20 = 7/20$
$r_{23} = 3$

VEDANT ALIAS SURYAJI SANJAY MALKARNEKAR
201205008

EXPERIMENT 9

9. $w_{34} = P(4) + q(4) + w_{33} = 1/20 + 1/20 + 1/20 = 3/20$

$C_{34} = \min_{\substack{3 < k \le 4 \\ k = 4}} \{C_{33} + C_{44}\} + w_{34} = (0+0) + 3/20 = 3/20$

$r_{34} = 4$

10. $w_{02} = P(2) + q(2) + w_{01} = 1/5 + 1/5 + 7/20 = 3/4$

$C_{02} = \min_{\substack{0 \le k \le 2 \\ k = 1,2}} \{C_{00} + C_{12}, C_{01} + C_{22}\} + w_{02} = \min(0 + 1/2, 7/20 + 0) + 3/4$

$= \min(1/2, 7/20) + 3/4 = 7/20 + 3/4 = 11/10$

$r_{02} = 2$

11. $w_{13} = P(3) + q(3) + w_{12} = 1/10 + 1/20 + 1/2 = 13/20$

$C_{13} = \min_{\substack{1 \le k \le 3 \\ k = 2,3}} \{C_{11} + C_{23}, C_{12} + C_{33}\} + w_{13}$

$= \min\{0 + 7/20, 1/2 + 0\} + 13/20$

$= \min(7/20, 1/2) + 13/20 = 7/20 + 13/20 = 1$

$r_{13} = 2$

12. $w_{24} = P(4) + q(4) + w_{23} = 1/20 + 1/20 + 7/20 = 9/20$

$C_{24} = \min_{\substack{2 < k \le 4 \\ k = 3,4}} \{C_{22} + C_{34}, C_{23} + C_{44}\} + w_{24}$

$= \min\{0 + 3/20, 7/20 + 0) + 9/20 = 3/20 + 9/20 = 3/5$

$r_{24} = 3$

13. $w_{03} = P(3) + q(3) + w_{02} = 1/10 + 1/20 + 3/4 = 9/10$

$C_{03} = \min_{\substack{0 < k \le 3 \\ k = 1,2,3}} \{C_{00} + C_{13}, C_{01} + C_{23}, C_{02} + C_{33}\} + w_{03}$

$= \min\{0 + 1, 7/20 + 7/20, 11/10 + 0\} + 9/10$

$= \min(1, 7/10, 11/10) + 9/10 = 7/10 + 9/10 = 8/5$

$r_{03} = 2$

EXPERIMENT 9

14. $w_{14} = p(4) + q(4) + w_{13} = 1/20 + 1/20 + 13/20 = 3/4$

$C_{14} = \min_{\substack{1 < k \le 4 \\ k = 2,3,4}} \{C_{11} + C_{24}, C_{12} + C_{34}, C_{13} + C_{44}\} + w_{14}$

$= \min(0 + 3/5, 1/2 + 3/20, 1 + 0) + 3/4 = \min(3/5, 13/20, 1) + 3/4$

$= \min(3/5 + 3/4) = 27/20$

$r_{14} = 2$

15. $w_{04} = p(4) + q(4) + w_{03} = 1/20 + 1/20 + 9/10 = 1$

$C_{04} = \min_{\substack{0 < k \le 4 \\ k = 1,2,3,4}} \{C_{00} + C_{14}, C_{01} + C_{24}, C_{02} + C_{34}, C_{03} + C_{44}\} + w_{04}$

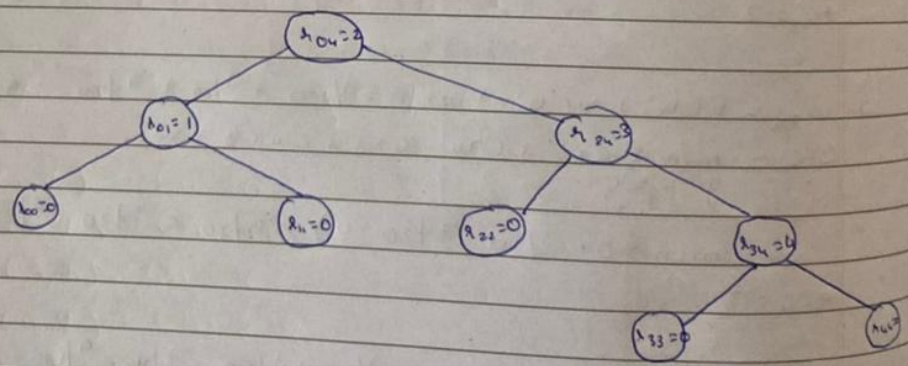
$= \min(0 + 27/20, 7/20 + 3/5, 11/10 + 3/20, 8/5 + 0) + 1$

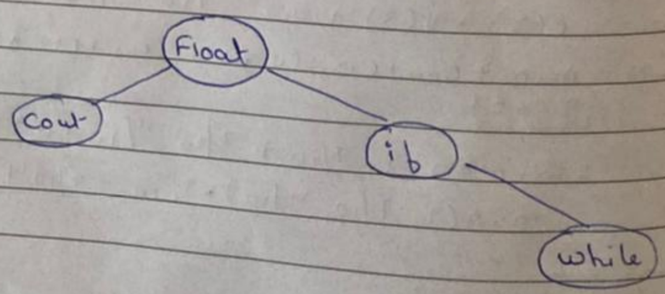
$= \min(27/20, 19/20, 5/4, 8/5) + 1$

$= 19/20 + 1 = 39/20$

$r_{04} = 2$

Root node = r[0][n-1]





EXPERIMENT 9

PROGRAM

```
#include<iostream>

#include<iomanip>

using namespace std;

int stepcount=0;

class OBST{


        float w[20][20];    //weight of OBST tij

        float c[20][20];    //cost of OBST tij

        float r[20][20];    //root of OBST tij

        public:

                int n;

                float p[20],q[20];

                void printpath(int i,int j);

                void OBST_algo(void);

                int Find(int i, int j);

};

int OBST::Find(int i, int j)

{
```

EXPERIMENT 9

```
        int min=INT_MAX;stepcount++;

        int l=0;stepcount++;


        for(int m=r[i][j-1];m<=r[i+1][j];m++)

        {

                stepcount++;

                stepcount++;

                if((c[i][m-1]+c[m][j])<min)

                {

                        min=c[i][m-1]+c[m][j];stepcount++;

                        l=m;stepcount++;

                }

        }

        stepcount++;

        stepcount++;return l;

}

void OBST::printpath(int i,int j)

{

        stepcount++;
```

EXPERIMENT 9

```
        if(i==j)
        {
                cout<<" null "<<endl;stepcount++;
                stepcount++;return;
        }
        cout<<r[i][j]<<endl;stepcount++;
        cout<<"Left Child of "<<r[i][j]<<" : ";stepcount++;
        printpath(i,r[i][j]-1);stepcount++;
        cout<<"Right Child of "<<r[i][j]<<" : ";stepcount++;
        printpath(r[i][j],j);stepcount++;
}
void OBST::OBST_algo(void)
{
        for(int i=0;i<n;i++)
        {
                stepcount++;
                w[i][i]=q[i];stepcount++;
                r[i][i]=0;stepcount++;
                c[i][i]=0.0;stepcount++;
```

EXPERIMENT 9

```
        w[i][i+1]=q[i]+q[i+1]+p[i+1];stepcount++;

        r[i][i+1]=i+1;stepcount++;

        c[i][i+1]=q[i]+q[i+1]+p[i+1];stepcount++;

}

stepcount++;

stepcount++;

w[n][n]=q[n];stepcount++;

r[n][n]=0;stepcount++;

c[n][n]=0.0;stepcount++;

for(int m=2;m<=n;m++)

{

        stepcount++;

        for(int i=0;i<=n-m;i++)

        {

                stepcount++;

                int j=i+m;stepcount++;

                w[i][j]=w[i][j-1]+p[j]+q[j];stepcount++;
```

EXPERIMENT 9

```
                int k=Find(i,j);stepcount++;

                c[i][j]=w[i][j]+c[i][k-1]+c[k][j];stepcount++;

                r[i][j]=k;stepcount++;


            }

            stepcount++;

        }


        cout<<"\n  | ";stepcount++;


    for(int i=0;i<=n;i++)

    {

        stepcount++;

        cout<<setw(10)<<i<<" | ";stepcount++;

        }

        stepcount++;

    cout<<endl<<"----------------------------------------------------------------
--------------------------------"<<endl;stepcount++;
```

EXPERIMENT 9

```
        for(int m=0;m<=n;m++)
    {
        stepcount++;
        cout<<m<<" | ";stepcount++;
        for(int i=0,j=m;i<=n&&j<=n;i++,j++)
        {
            stepcount++;
                cout<<"w("<<i<<","<<j<<") "<<w[i][j]<<"   |
";stepcount++;

            }
            stepcount++;
            cout<<"\n  | ";stepcount++;

            for(int i=0,j=m;i<=n&&j<=n;i++,j++)
        {
            stepcount++;
                cout<<"c("<<i<<","<<j<<") "<<c[i][j]<<"   |
";stepcount++;
```

EXPERIMENT 9

```
            }
            stepcount++;
            cout<<"\n  | ";stepcount++;

            for(int i=0,j=m;i<=n&&j<=n;i++,j++)
        {
            stepcount++;
                cout<<"r("<<i<<","<<j<<") "<<r[i][j]<<"   |
";stepcount++;
            }
            stepcount++;
            cout<<endl<<"--------------------------------------------------
--------------------------------"<<endl;stepcount++;

        }
    stepcount++;
    cout<<"Minimum Cost : "<<c[0][n]<<endl;stepcount++;
      cout<<"Root : ";stepcount++;
      stepcount++;printpath(0,n);
```

EXPERIMENT 9

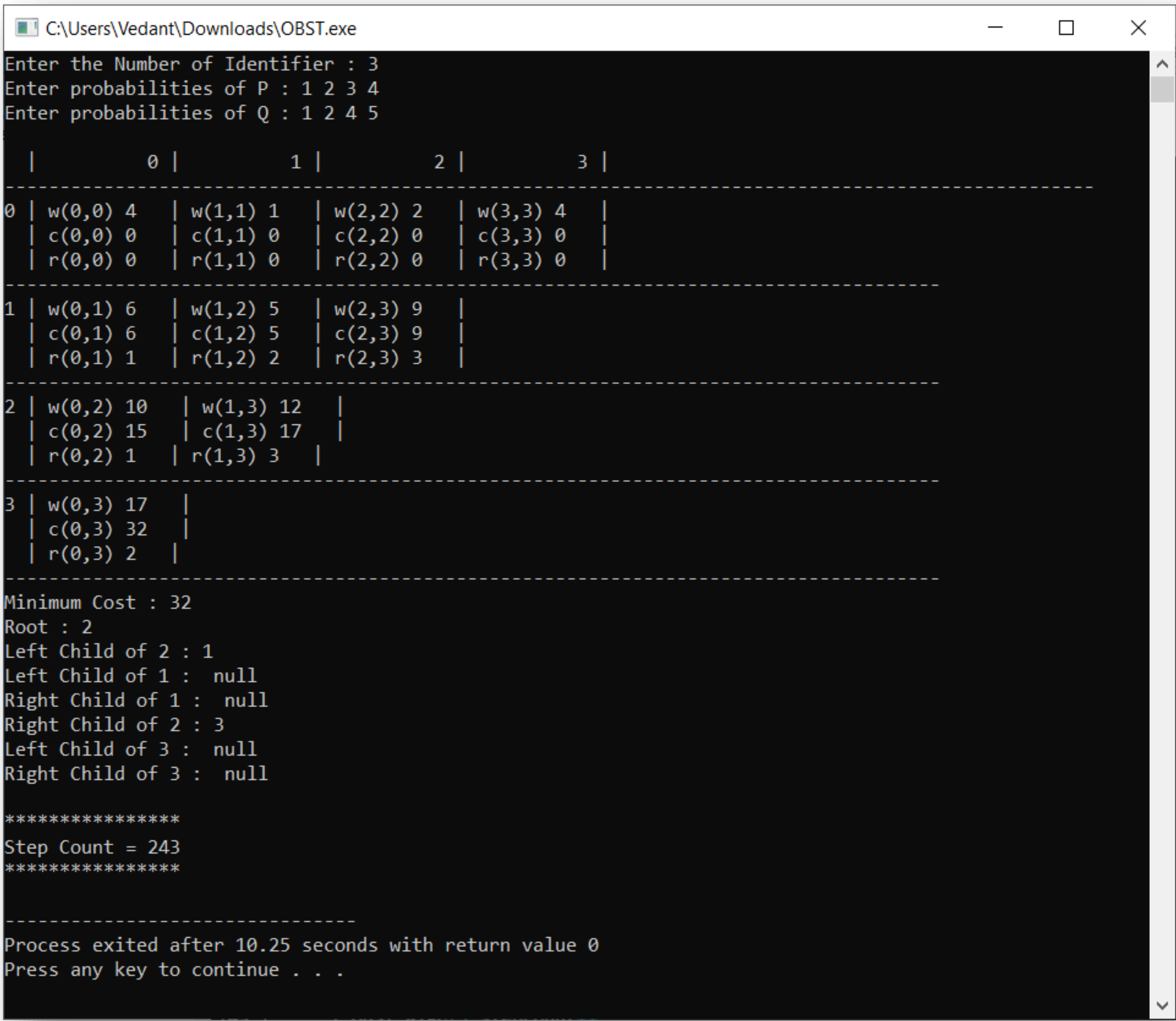
```
}
int main()
{
    class OBST T;
    cout<<"Enter the Number of Identifier : ";stepcount++;
    cin>>T.n;stepcount++;
    cout<<"Enter probabilities of P : ";stepcount++;
    for(int i=1;i<=T.n;i++)
    {
        stepcount++;
        cin>>T.p[i];stepcount++;
    }
    stepcount++;
    cout<<"Enter probabilities of Q : ";stepcount++;
    for(int i=0;i<=T.n;i++)
    {
        stepcount++;
        cin>>T.q[i];stepcount++;
```

EXPERIMENT 9

```
        }
        stepcount++;
        T.OBST_algo();stepcount++;
        stepcount++;
        cout<<"\n***************"<<endl;
        cout<<"Step Count = "<<stepcount<<endl;
        cout<<"***************"<<endl;
        return 0;
}
```

EXPERIMENT 9

OUTPUT



```
C:\Users\Vedant\Downloads\OBST.exe
Enter the Number of Identifier : 3
Enter probabilities of P : 1 2 3 4
Enter probabilities of Q : 1 2 4 5

  |          0 |          1 |          2 |          3 |
------------------------------------------------------------------------------------------------------
0 | w(0,0) 4    | w(1,1) 1    | w(2,2) 2    | w(3,3) 4    |
  | c(0,0) 0    | c(1,1) 0    | c(2,2) 0    | c(3,3) 0    |
  | r(0,0) 0    | r(1,1) 0    | r(2,2) 0    | r(3,3) 0    |
--------------------------------------------------------------------------------------
1 | w(0,1) 6    | w(1,2) 5    | w(2,3) 9    |
  | c(0,1) 6    | c(1,2) 5    | c(2,3) 9    |
  | r(0,1) 1    | r(1,2) 2    | r(2,3) 3    |
--------------------------------------------------------------------------------------
2 | w(0,2) 10    | w(1,3) 12    |
  | c(0,2) 15    | c(1,3) 17    |
  | r(0,2) 1    | r(1,3) 3    |
--------------------------------------------------------------------------------------
3 | w(0,3) 17    |
  | c(0,3) 32    |
  | r(0,3) 2    |
--------------------------------------------------------------------------------------
Minimum Cost : 32
Root : 2
Left Child of 2 : 1
Left Child of 1 :  null
Right Child of 1 :  null
Right Child of 2 : 3
Left Child of 3 :  null
Right Child of 3 :  null

****************
Step Count = 243
****************

--------------------------------
Process exited after 10.25 seconds with return value 0
Press any key to continue . . .
```

Conclusion

- Detailed concept of OBST using Dynamic Programming was studied successfully.
- Program using OBST Algorithm was executed successfully.
- The step count for the OBST Algorithm was obtained.