DESIGN ANALYSIS AND ALGORITHMS

LAB ASSIGNMENT-5

NAME : PRATHAPANI SATWIKA

REG.NO. : 20BCD7160

Q) Write a program that uses dynamic programming algorithm to solve the optimal binary search tree problem.

CODE :

```
package Lab5;

public class OptimalBinarySearchTree {
    static int optimalSearchTree(int keys[], int freq[], int n) {
        int cost[][] = new int[n + 1][n + 1];
        for (int i = 0; i < n; i++)
            cost[i][i] = freq[i];
        for (int L = 2; L <= n; L++) {
            for (int i = 0; i <= n - L + 1; i++) {
                int j = i + L - 1;
                cost[i][j] = Integer.MAX_VALUE;
                  int off_set_sum = sum(freq, i, j);
                for (int r = i; r <= j; r++) {
                    int c = ((r > i) ? cost[i][r - 1] : 0)
                            + ((r < j) ? cost[r + 1][j] : 0) + off_set_sum;
                    if (c < cost[i][j])
                        cost[i][j] = c;
                }
            }
        }
        return cost[0][n - 1];
    }
    static int sum(int freq[], int i, int j) {
        int s = 0;
        for (int k = i; k <= j; k++) {
            if (k >= freq.length)
                continue;
            s += freq[k];
        }
        return s;
    }

    public static void main(String[] args) {

        int keys[] = { 18, 26, 30 };
        int freq[] = { 4, 88, 45 };
```

```
        int n = keys.length;
        System.out.println("Cost of Optimal BST is "
                + optimalSearchTree(keys, freq, n));
    }

}
```

```
1  package Lab5;
2
3  public class OptimalBinarySearchTree {
4      static int optimalSearchTree(int keys[], int freq[], int n) {
5          int cost[][] = new int[n + 1][n + 1];
6          for (int i = 0; i < n; i++)
7              cost[i][i] = freq[i];
8          for (int L = 2; L <= n; L++) {
9              for (int i = 0; i <= n - L + 1; i++) {
10                 int j = i + L - 1;
11                 cost[i][j] = Integer.MAX_VALUE;
12                   int off_set_sum = sum(freq, i, j);
13                 for (int r = i; r <= j; r++) {
14                     int c = ((r > i) ? cost[i][r - 1] : 0)
15                             + ((r < j) ? cost[r + 1][j] : 0) + off_set_sum;
16                     if (c < cost[i][j])
17                         cost[i][j] = c;
18                 }
19             }
20         }
21         return cost[0][n - 1];
22     }
23     static int sum(int freq[], int i, int j) {
24         int s = 0;
25         for (int k = i; k <= j; k++) {
26             if (k >= freq.length)
27                 continue;
28             s += freq[k];
29         }
30         return s;
31     }
32
33     public static void main(String[] args) {
34
35         int keys[] = { 18, 26, 30 };
36         int freq[] = { 4, 88, 45 };
37         int n = keys.length;
38         System.out.println("Cost of Optimal BST is "
39                 + optimalSearchTree(keys, freq, n));
40     }
41
42 }
```

OUTPUT :

```
<terminated> OptimalBinarySearchTree [Java Application]
Cost of Optimal BST is 186
```