

Practical 6

Name	Dipansh Tiwari
Section	A4
Roll no	B3-35

Task 1

Code-

```
def optimal_bst(keys, p, q, n):
```

```
    e = [[0 for _ in range(n + 2)] for _ in range(n + 2)]
```

```
    w = [[0 for _ in range(n + 2)] for _ in range(n + 2)]
```

```
    for i in range(1, n + 2):
```

```
        e[i][i - 1] = q[i - 1]
```

```
        w[i][i - 1] = q[i - 1]
```

```
    for l in range(1, n + 1):
```

```
        for i in range(1, n - l + 2):
```

```
            j = i + l - 1
```

```
            e[i][j] = float('inf')
```

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

```
            for r in range(i, j + 1):
```

```
                cost = e[i][r - 1] + e[r + 1][j] + w[i][j]
```

```
                if cost < e[i][j]:
```

```
                    e[i][j] = cost
```

```
    return e[1][n]
```

```
n = int(input("Enter number of book IDs: "))
```

```
keys = list(map(int, input("Enter the sorted book IDs: ").split()))
```

```
p = list(map(float, input("Enter probabilities of successful searches: ").split()))
```

```
q = list(map(float, input("Enter probabilities of unsuccessful searches: ").split()))
```

```
min_cost = optimal_bst(keys, p, q, n)

print(f"Minimum expected cost of OBST: {min_cost:.4f}")
```

Output-

```

~~~~~~
IndexError: list index out of range
PS D:\3rd Semester\DAA> & C:/Users/harsh/AppData/Local/Programs/Python/Python313/python.exe "d:/1st Semester/FOP/pract7.py"
Enter number of book IDs: 4
Enter the sorted book IDs: 10 20 30 40
Enter probabilities of successful searches: 0.1 0.2 0.4 0.3
Enter probabilities of unsuccessful searches: 0.05 0.1 0.055 0.05 0.1
Minimum expected cost of OBST: 2.9150
PS D:\3rd Semester\DAA>

```

Task 2-

Code-

class Solution

```

{
    static int optimalSearchTree(int keys[], int freq[], int n)
    {
        int cost[][] = new int[n][n];
        int sum[][] = new int[n][n];

        for (int i = 0; i < n; i++)
        {
            cost[i][i] = freq[i];
            sum[i][i] = freq[i];
        }

        for (int len = 2; len <= n; len++)
        {
            for (int i = 0; i <= n - len; i++)

```

```

{
    int j = i + len - 1;

    cost[i][j] = Integer.MAX_VALUE;

    sum[i][j] = sum[i][j - 1] + freq[j];

    for (int r = i; r <= j; r++)
    {
        int left = (r > i) ? cost[i][r - 1] : 0;

        int right = (r < j) ? cost[r + 1][j] : 0;

        int temp = left + right + sum[i][j];

        if (temp < cost[i][j])
        {
            cost[i][j] = temp;
        }
    }
}

return cost[0][n - 1];
}
}

```

Output-

Problem

Editorial

Submissions

Comments

Output Window

Compilation Results

Custom Input

Compilation Completed

Case 1

Input:

2
10 12
34 50

Your Output:
118

Expected Output:
118

Java (21)

Start Timer

```
2 {
3     static int optimalSearchTree(int keys[], int freq[], int n)
4     {
5         int cost[][] = new int[n][n];
6         int sum[][] = new int[n][n];
7
8         for (int i = 0; i < n; i++)
9         {
10             cost[i][i] = freq[i];
11             sum[i][i] = freq[i];
12         }
13
14         for (int len = 2; len <= n; len++)
15         {
16             for (int i = 0; i <= n - len; i++)
17             {
18                 int j = i + len - 1;
19                 cost[i][j] = Integer.MAX_VALUE;
20                 sum[i][j] = sum[i][j - 1] + freq[j];
21
22                 for (int r = i; r <= j; r++)
23                 {
24                     int left = (r > i) ? cost[i][r - 1] : 0;
25                     int right = (r < j) ? cost[r + 1][j] : 0;
26                     int temp = left + right + sum[i][j];
27
28                     if (temp < cost[i][j])
29                     {
30                         cost[i][j] = temp;
31                     }
32                 }
33             }
34         }
35
36         return cost[0][n - 1];
37     }
38 }
```