Experiment No.7

Optimal Binary Search Trees using Dynamic Programming

Reg.No.24141045

Program :

```
#include <stdio.h>

#define MAX 10

#define INF 9999

int main() {

        int n, i, j, k;

        float p[MAX], q[MAX], w[MAX][MAX], c[MAX][MAX];

        int r[MAX][MAX];

        printf("Enter number of keys: ");

        scanf("%d", &n);

        printf("Enter probabilities of keys (p1 to p%d):¥n", n);

        for (i = 1; i <= n; i++)

                scanf("%f", &p[i]);

        printf("Enter probabilities of dummy keys (q0 to q%d):¥n", n);

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

                scanf("%f", &q[i]);

        // Initialization

        for (i = 0; i <= n; i++) {
```

```
        w[i][i] = q[i];

        c[i][i] = 0;

        r[i][i] = 0;

}

  // OBST Calculation

  for (int m = 1; m <= n; m++) {                  // m = chain length

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

              j = i + m;

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

              float min = INF;

              int min_k = 0;

              for (k = i + 1; k <= j; k++) {

                    float cost = c[i][k - 1] + c[k][j];

                    if (cost < min) {

                          min = cost;

                          min_k = k;

                    }

              }

              c[i][j] = w[i][j] + min;

              r[i][j] = min_k;

        }

  }
```

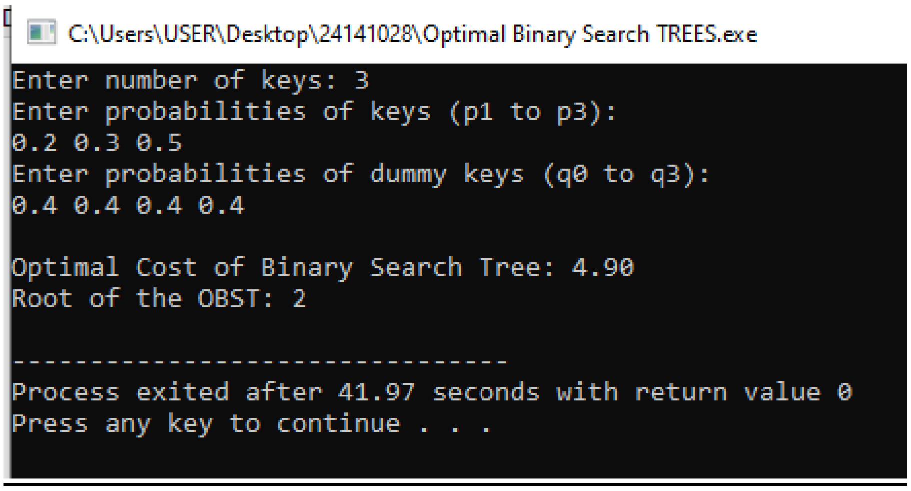
```
        printf("¥nOptimal Cost of Binary Search Tree: %.2f¥n", c[0][n]);

        printf("Root of the OBST: %d¥n", r[0][n]);

        return 0;

}
```

Output :



C:\Users\USER\Desktop\24141028\Optimal Binary Search TREES.exe

```
Enter number of keys: 3
Enter probabilities of keys (p1 to p3):
0.2 0.3 0.5
Enter probabilities of dummy keys (q0 to q3):
0.4 0.4 0.4 0.4

Optimal Cost of Binary Search Tree: 4.90
Root of the OBST: 2

--------------------------------
Process exited after 41.97 seconds with return value 0
Press any key to continue . . .
```

Time Complexity :

The outer loop (m) runs n times.

The middle loop (i) also runs roughly n times (depending on m).

The innermost loop (k) runs up to n times (for each range).

So total operations ≈ $O(n \times n \times n) = O(n^3)$

Space Complexity :

Because we use 2D matrices w[][], c[][], and r[][] of size n x n : $O(n^2)$

Real-Time Applications :

Compiler Design:

Used in symbol table organization to minimize average search time.

Database Indexing:

For faster access to records with different search probabilities.

Search Optimization:

Used when different items have different frequencies of search.

Speech and Pattern Recognition:

Helps in optimizing search structures in probabilistic models.

Decision Trees in AI:

OBST concept is used for probabilistic decision making.