DESIGN ANALYSIS AND ALGORITHMS

LAB ASSIGNMENT-4

NAME : PRATHAPANI SATWIKA

REG.NO. : 20BCD7160

1Q) Implementation of travelling sales person problem using dynamic programming

CODE :

```
package Lab4a;
import java.util.Scanner;
public class TsproblemUsingDP
{
    public static void main(String[] args)
    {
        Scanner in = new Scanner(System.in);
        int c[][]=new int[10][10], tour[]=new int[10];
        int i, j,cost;
        System.out.print("Enter No. of Cities: ");
        int n = in.nextInt();
        if(n==1)
        {
            System.out.println("Path is not possible!");
            System.exit(0);
        }
        System.out.println("Enter the Cost Matrix:");
        for(i=1;i<=n;i++)
            for(j=1;j<=n;j++)
                c[i][j] = in.nextInt();
        for(i=1;i<=n;i++)
            tour[i]=i;
        cost = tspdp(c, tour, 1, n);
        System.out.print("The Optimal Tour is: ");
        for(i=1;i<=n;i++)
            System.out.print(tour[i]+"->");
        System.out.println("1");
        System.out.println("Minimum Cost: "+cost);
    }
    static int tspdp(int c[][], int tour[], int start, int n)
    {
        int mintour[]=new int[10], temp[]=new int[10], mincost=999,ccost, i, j, k;
        if(start == n-1)
        {
            return (c[tour[n-1]][tour[n]] + c[tour[n]][1]);
        }
        for(i=start+1; i<=n; i++)
        {
```

```
            for(j=1; j<=n; j++)
                temp[j] = tour[j];
            temp[start+1] = tour[i];
            temp[i] = tour[start+1];
            if((c[tour[start]][tour[i]]+(ccost=tspdp(c,temp,start+1,n)))<mincost)
            {
                mincost = c[tour[start]][tour[i]] + ccost;
                for(k=1; k<=n; k++)
                    mintour[k] = temp[k];
            }
        }
        for(i=1; i<=n; i++)
            tour[i] = mintour[i];
        return mincost;
    }
}
```

```
1  package Lab4a;
2  import java.util.Scanner;
3  public class TsproblemUsingDP
4  {
5      public static void main(String[] args)
6      {
7          Scanner in = new Scanner(System.in);
8          int c[][]=new int[10][10], tour[]=new int[10];
9          int i, j,cost;
10         System.out.print("Enter No. of Cities: ");
11         int n = in.nextInt();
12         if(n==1)
13         {
14             System.out.println("Path is not possible!");
15             System.exit(0);
16         }
17         System.out.println("Enter the Cost Matrix:");
18         for(i=1;i<=n;i++)
19             for(j=1;j<=n;j++)
20                 c[i][j] = in.nextInt();
21         for(i=1;i<=n;i++)
22             tour[i]=i;
23         cost = tspdp(c, tour, 1, n);
24         System.out.print("The Optimal Tour is: ");
25         for(i=1;i<=n;i++)
26             System.out.print(tour[i]+"->");
27         System.out.println("1");
28         System.out.println("Minimum Cost: "+cost);
29     }
30     static int tspdp(int c[][], int tour[], int start, int n)
31     {
32         int mintour[]=new int[10], temp[]=new int[10], mincost=999,ccost, i, j, k;
33         if(start == n-1)
34         {
35             return (c[tour[n-1]][tour[n]] + c[tour[n]][1]);
36         }
37         for(i=start+1; i<=n; i++)
38         {
39             for(j=1; j<=n; j++)
40                 temp[j] = tour[j];
41             temp[start+1] = tour[i];
42             temp[i] = tour[start+1];
43             if((c[tour[start]][tour[i]]+(ccost=tspdp(c,temp,start+1,n)))<mincost)
44             {
```

```
            if((c[tour[start]][tour[i]]+(ccost=tspdp(c,temp,start+1,n)))<mincost)
            {
                mincost = c[tour[start]][tour[i]] + ccost;
                for(k=1; k<=n; k++)
                    mintour[k] = temp[k];
            }
        }
        for(i=1; i<=n; i++)
            tour[i] = mintour[i];
        return mincost;
    }
}
```

OUTPUT :

```
Enter No. of Cities: 4
Enter the Cost Matrix:
5 9 4 7
1 5 6 3
4 2 3 8
1 6 4 8
The Optimal Tour is: 1->3->2->4->1
Minimum Cost: 10
```

2Q)Implementation of matrix chain multiplication problem using dynamic programming

CODE :

```
package Lab4b;
class MatrixChainMultiplication
{
    static int MatrixChainOrder(int p[], int n)
    {
        int m[][] = new int[n][n];
        int i, j, k, L, q;
        for (i = 1; i < n; i++)
            m[i][i] = 0;
        for (L = 2; L < n; L++)
        {
            for (i = 1; i < n - L + 1; i++)
            {
                j = i + L - 1;
                if (j == n)
                    continue;
                m[i][j] = Integer.MAX_VALUE;
                for (k = i; k <= j - 1; k++)
                {
                    q = m[i][k] + m[k + 1][j]
                        + p[i - 1] * p[k] * p[j];
                    if (q < m[i][j])
                        m[i][j] = q;
                }
            }
        }

        return m[1][n - 1];
    }
    public static void main(String args[])
    {
        int arr[] = new int[] { 5,8,9,6};
        int size = arr.length;

        System.out.println(
            "Minimum number of multiplications is : "
            + MatrixChainOrder(arr, size));
    }
}
```

```java
package Lab4b;
class MatrixChainMultiplication
{
    static int MatrixChainOrder(int p[], int n)
    {
        int m[][] = new int[n][n];
        int i, j, k, L, q;
        for (i = 1; i < n; i++)
            m[i][i] = 0;
        for (L = 2; L < n; L++)
        {
            for (i = 1; i < n - L + 1; i++)
            {
                j = i + L - 1;
                if (j == n)
                    continue;
                m[i][j] = Integer.MAX_VALUE;
                for (k = i; k <= j - 1; k++)
                {
                    q = m[i][k] + m[k + 1][j]
                        + p[i - 1] * p[k] * p[j];
                    if (q < m[i][j])
                        m[i][j] = q;
                }
            }
        }

        return m[1][n - 1];
    }
    public static void main(String args[])
    {
        int arr[] = new int[] { 5,8,9,6};
        int size = arr.length;

        System.out.println(
            "Minimum number of multiplications is : "
            + MatrixChainOrder(arr, size));
    }
}
```

OUTPUT :

```
<terminated> MatrixChainMultiplication [Java Application] C:\Prog
Minimum number of multiplications is : 630
```