DESIGN ANALYSIS AND ALGORITHMS

LAB ASSIGNMENT-6

NAME: PRATHAPANI SATWIKA

REG.NO.: 20BCD7160

Q) Implementation of Travelling salesman problem using Branch and Bound

CODE :

```
package Lab7;
import java.util.*;

public class BranchandBound
{

    static int N = 4;
    static int final_path[] = new int[N + 1];
    static boolean visited[] = new boolean[N];
    static int final_res = Integer.MAX_VALUE;
    static void copyToFinal(int curr_path[])
    {
        for (int i = 0; i < N; i++)
            final_path[i] = curr_path[i];
        final_path[N] = curr_path[0];
    }
    static int firstMin(int adj[][], int i)
    {
        int min = Integer.MAX_VALUE;
        for (int k = 0; k < N; k++)
            if (adj[i][k] < min && i != k)
                min = adj[i][k];
        return min;
    }
    static int secondMin(int adj[][], int i)
    {
        int first = Integer.MAX_VALUE, second = Integer.MAX_VALUE;
        for (int j=0; j<N; j++)
        {
            if (i == j)
                continue;

            if (adj[i][j] <= first)
            {
                second = first;
                first = adj[i][j];
            }
            else if (adj[i][j] <= second &&
                    adj[i][j] != first)
                second = adj[i][j];
```

```
        }
        return second;
    }
    static void TSPRec(int adj[][], int curr_bound, int curr_weight,
                int level, int curr_path[])
    {
        if (level == N)
        {
            if (adj[curr_path[level - 1]][curr_path[0]] != 0)
            {
                int curr_res = curr_weight +
                        adj[curr_path[level-1]][curr_path[0]];
                if (curr_res < final_res)
                {
                    copyToFinal(curr_path);
                    final_res = curr_res;
                }
            }
            return;
        }
        for (int i = 0; i < N; i++)
        {
            if (adj[curr_path[level-1]][i] != 0 &&
                    visited[i] == false)
            {
                int temp = curr_bound;
                curr_weight += adj[curr_path[level - 1]][i];
                if (level==1)
                curr_bound -= ((firstMin(adj, curr_path[level - 1]) +
                                firstMin(adj, i))/2);
                else
                curr_bound -= ((secondMin(adj, curr_path[level - 1]) +
                                firstMin(adj, i))/2);
                if (curr_bound + curr_weight < final_res)
                {
                    curr_path[level] = i;
                    visited[i] = true;
                    TSPRec(adj, curr_bound, curr_weight, level + 1,
                        curr_path);
                }
                curr_weight -= adj[curr_path[level-1]][i];
                curr_bound = temp;
                Arrays.fill(visited,false);
                for (int j = 0; j <= level - 1; j++)
                    visited[curr_path[j]] = true;
            }
        }
    }
    static void TSP(int adj[][])
    {
        int curr_path[] = new int[N + 1];
        int curr_bound = 0;
        Arrays.fill(curr_path, -1);
        Arrays.fill(visited, false);
        for (int i = 0; i < N; i++)
```

```
            curr_bound += (firstMin(adj, i) +
                           secondMin(adj, i));
        curr_bound = (curr_bound==1)? curr_bound/2 + 1 :
                                      curr_bound/2;
        visited[0] = true;
        curr_path[0] = 0;
        TSPRec(adj, curr_bound, 0, 1, curr_path);
    }
    public static void main(String[] args)
    {
        int adj[][] = {{0, 26, 6, 28},
                       {31, 0, 43, 15},
                       {22, 18, 0, 13},
                       {8, 29, 33, 0}    };

        TSP(adj);

        System.out.printf("Minimum cost : %d\n", final_res);
        System.out.printf("Path Taken : ");
        for (int i = 0; i <= N; i++)
        {
            System.out.printf("%d ", final_path[i]);
        }
    }
}
```

package-info.java | BranchandBound.java

```java
package Lab7;
import java.util.*;

public class BranchandBound
{

    static int N = 4;
    static int final_path[] = new int[N + 1];
    static boolean visited[] = new boolean[N];
    static int final_res = Integer.MAX_VALUE;
    static void copyToFinal(int curr_path[])
    {
        for (int i = 0; i < N; i++)
            final_path[i] = curr_path[i];
        final_path[N] = curr_path[0];
    }
    static int firstMin(int adj[][], int i)
    {
        int min = Integer.MAX_VALUE;
        for (int k = 0; k < N; k++)
            if (adj[i][k] < min && i != k)
                min = adj[i][k];
        return min;
    }
    static int secondMin(int adj[][], int i)
    {
        int first = Integer.MAX_VALUE, second = Integer.MAX_VALUE;
        for (int j=0; j<N; j++)
        {
            if (i == j)
                continue;

            if (adj[i][j] <= first)
            {
                second = first;
                first = adj[i][j];
            }
            else if (adj[i][j] <= second &&
                    adj[i][j] != first)
                second = adj[i][j];
        }
        return second;
    }
```

package-info.java | BranchandBound.java

```java
    static void TSPRec(int adj[][], int curr_bound, int curr_weight,
                int level, int curr_path[])
    {
        if (level == N)
        {
            if (adj[curr_path[level - 1]][curr_path[0]] != 0)
            {
                int curr_res = curr_weight +
                        adj[curr_path[level-1]][curr_path[0]];
                if (curr_res < final_res)
                {
                    copyToFinal(curr_path);
                    final_res = curr_res;
                }
            }
            return;
        }
        for (int i = 0; i < N; i++)
        {
            if (adj[curr_path[level-1]][i] != 0 &&
                    visited[i] == false)
            {
                int temp = curr_bound;
                curr_weight += adj[curr_path[level - 1]][i];
                if (level==1)
                curr_bound -= ((firstMin(adj, curr_path[level - 1]) +
                                firstMin(adj, i))/2);
                else
                curr_bound -= ((secondMin(adj, curr_path[level - 1]) +
                                firstMin(adj, i))/2);
                if (curr_bound + curr_weight < final_res)
                {
                    curr_path[level] = i;
                    visited[i] = true;
                    TSPRec(adj, curr_bound, curr_weight, level + 1,
                        curr_path);
                }
                curr_weight -= adj[curr_path[level-1]][i];
                curr_bound = temp;
                Arrays.fill(visited,false);
                for (int j = 0; j <= level - 1; j++)
                    visited[curr_path[j]] = true;
            }
```

```java
        }
    }
    static void TSP(int adj[][])
    {
        int curr_path[] = new int[N + 1];
        int curr_bound = 0;
        Arrays.fill(curr_path, -1);
        Arrays.fill(visited, false);
        for (int i = 0; i < N; i++)
            curr_bound += (firstMin(adj, i) +
                           secondMin(adj, i));
        curr_bound = (curr_bound==1)? curr_bound/2 + 1 :
                                      curr_bound/2;
        visited[0] = true;
        curr_path[0] = 0;
        TSPRec(adj, curr_bound, 0, 1, curr_path);
    }
    public static void main(String[] args)
    {
        int adj[][] = {{0, 26, 6, 28},
                       {31, 0, 43, 15},
                       {22, 18, 0, 13},
                       {8, 29, 33, 0}    };

        TSP(adj);

        System.out.printf("Minimum cost : %d\n", final_res);
        System.out.printf("Path Taken : ");
        for (int i = 0; i <= N; i++)
        {
            System.out.printf("%d ", final_path[i]);
        }
    }
}
```

OUTPUT :

Problems @ Javadoc Declaration Console

<terminated> BranchandBound [Java Application] C:\Program Files\Java\jdk-

```
Minimum cost : 47
Path Taken : 0 2 1 3 0
```