

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
1 package Lab7;
2 import java.util.*;
3
4 public class BranchandBound
5 {
6
7     static int N = 4;
8     static int final_path[] = new int[N + 1];
9     static boolean visited[] = new boolean[N];
10    static int final_res = Integer.MAX_VALUE;
11    static void copyToFinal(int curr_path[])
12    {
13        for (int i = 0; i < N; i++)
14            final_path[i] = curr_path[i];
15        final_path[N] = curr_path[0];
16    }
17    static int firstMin(int adj[][], int i)
18    {
19        int min = Integer.MAX_VALUE;
20        for (int k = 0; k < N; k++)
21            if (adj[i][k] < min && i != k)
22                min = adj[i][k];
23        return min;
24    }
25    static int secondMin(int adj[][], int i)
26    {
27        int first = Integer.MAX_VALUE, second = Integer.MAX_VALUE;
28        for (int j=0; j<N; j++)
29        {
30            if (i == j)
31                continue;
32
33            if (adj[i][j] <= first)
34            {
35                second = first;
36                first = adj[i][j];
37            }
38            else if (adj[i][j] <= second &&
39                    adj[i][j] != first)
40                second = adj[i][j];
41        }
42        return second;
43    }

```

package-info.java

BranchandBound.java

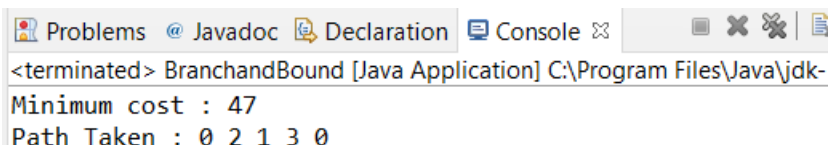
```
44  static void TSPRec(int adj[][], int curr_bound, int curr_weight,
45                    int level, int curr_path[])
46  {
47      if (level == N)
48      {
49          if (adj[curr_path[level - 1]][curr_path[0]] != 0)
50          {
51              int curr_res = curr_weight +
52                          adj[curr_path[level-1]][curr_path[0]];
53              if (curr_res < final_res)
54              {
55                  copyToFinal(curr_path);
56                  final_res = curr_res;
57              }
58          }
59          return;
60      }
61      for (int i = 0; i < N; i++)
62      {
63          if (adj[curr_path[level-1]][i] != 0 &&
64              visited[i] == false)
65          {
66              int temp = curr_bound;
67              curr_weight += adj[curr_path[level - 1]][i];
68              if (level==1)
69                  curr_bound -= ((firstMin(adj, curr_path[level - 1]) +
70                                firstMin(adj, i))/2);
71              else
72                  curr_bound -= ((secondMin(adj, curr_path[level - 1]) +
73                                firstMin(adj, i))/2);
74              if (curr_bound + curr_weight < final_res)
75              {
76                  curr_path[level] = i;
77                  visited[i] = true;
78                  TSPRec(adj, curr_bound, curr_weight, level + 1,
79                        curr_path);
80              }
81              curr_weight -= adj[curr_path[level-1]][i];
82              curr_bound = temp;
83              Arrays.fill(visited, false);
84              for (int j = 0; j <= level - 1; j++)
85                  visited[curr_path[j]] = true;
86          }
87      }
```

```

87     }
88 }
89 static void TSP(int adj[][])
90 {
91     int curr_path[] = new int[N + 1];
92     int curr_bound = 0;
93     Arrays.fill(curr_path, -1);
94     Arrays.fill(visited, false);
95     for (int i = 0; i < N; i++)
96         curr_bound += (firstMin(adj, i) +
97                     secondMin(adj, i));
98     curr_bound = (curr_bound==1)? curr_bound/2 + 1 :
99                 curr_bound/2;
100    visited[0] = true;
101    curr_path[0] = 0;
102    TSPRec(adj, curr_bound, 0, 1, curr_path);
103 }
104 public static void main(String[] args)
105 {
106     int adj[][] = {{0, 26, 6, 28},
107                  {31, 0, 43, 15},
108                  {22, 18, 0, 13},
109                  {8, 29, 33, 0}   };
110
111    TSP(adj);
112
113    System.out.printf("Minimum cost : %d\n", final_res);
114    System.out.printf("Path Taken : ");
115    for (int i = 0; i <= N; i++)
116    {
117        System.out.printf("%d ", final_path[i]);
118    }
119 }
120 }
121

```

OUTPUT :



The screenshot shows an IDE window with tabs for 'Problems', 'Javadoc', 'Declaration', and 'Console'. The 'Console' tab is active, displaying the output of the Java application. The output consists of two lines: 'Minimum cost : 47' and 'Path Taken : 0 2 1 3 0'. The window title bar indicates the application is 'BranchandBound [Java Application]' running in 'C:\Program Files\Java\jdk-'.

```

<terminated> BranchandBound [Java Application] C:\Program Files\Java\jdk-
Minimum cost : 47
Path Taken : 0 2 1 3 0

```