

---

## Assignment #4

### 1. Traveling Salesman Problem (TSP)

---

Date of Performance: 09/07/2024

Date of Submission: 17/09/2024

Student ID: 20220104108

Name: Mashrur Rahman

Group: C1

## Algorithm\TSP.cpp

```
1  #include <bits/stdc++.h>
2  #include <tuple>
3  using namespace std;
4  #define inf 99999
5  int reduce(vector<int> &row)
6  {
7      int min = inf + inf;
8      for (int i = 0; i < row.size(); i++)
9      {
10
11          if (row[i] < min && row[i] != inf)
12              min = row[i];
13      }
14      for (int i = 0; i < row.size(); i++)
15      {
16          if (row[i] != inf)
17              row[i] -= min;
18      }
19
20      return min;
21 }
22
23 int main()
24 {
25     int n, e;
26     cout<<"Node and Edges: ";
27     cin >> n >> e;
28     cout<<"From To Cost: "<<endl;
29
30     vector<vector<pair<int, int>>> adj(n + 1); // this is for 1 based graph
31     for (int i = 0; i < e; i++)
32     {
33         int u, v, w;
34         cin >> u >> v >> w;
35         adj[u].push_back({v, w});
36     }
```

```
37     vector<vector<int>> r, main;
38     for (int i = 0; i <= n; i++)
39     {
40         vector<int> t(n + 1, inf);
41         r.push_back(t);
42     }
43     for (int i = 1; i <= n; i++)
44     {
45         for (int j = 0; j < adj[i].size(); j++)
46         {
47             r[i][adj[i][j].first] = adj[i][j].second;
48         }
49     }
50     main = r; // this is for saving my cost
51     int bound = 0;
52     for (int i = 1; i <= n; i++)
53     {
54         vector<int> t;
55         for (int j = 1; j <= n; j++)
56         {
57             t.push_back(r[i][j]);
58         }
59         bound = bound + reduce(t);
60         for (int j = 1; j <= n; j++)
61         {
62             r[i][j] = t[j - 1];
63         }
64     }
65
66     for (int i = 1; i <= n; i++)
67     {
68         vector<int> t;
69         for (int j = 1; j <= n; j++)
70         {
71             t.push_back(r[j][i]);
72         }
73         bound = bound + reduce(t);
74         for (int j = 1; j <= n; j++)
```

```
75     {
76         r[j][i] = t[j - 1];
77     }
78 }
79
80 vector<int> vis(n + 1, 0), ans;
81 ans.push_back(1);
82 int itr = 1, node = 1;
83
84
85 while (itr <= n-1)
86 {
87     vis[node] = 1;
88     vector<tuple<int, int, vector<vector<int>>>> temp;
89
90     for (int i = 1; i <= n; i++)
91     {
92         int bound_temp = bound;
93         if (vis[i] == 1)
94             continue;
95         if (r[node][i] >= inf)
96             continue;
97         vector<vector<int>> t;
98         t = r;
99
100         bound_temp = bound_temp + t[node][i];
101
102         t[i][node] = inf;
103
104         for (int j = 1; j < t.size(); j++)
105         {
106             t[node][j] = inf;
107         }
108         for (int j = 1; j < t.size(); j++)
109         {
110             t[j][i] = inf;
111         }
112     }
```

```
113     for (int j = 1; j <= n; j++)
114     {
115         vector<int> t_rc;
116         for (int k = 1; k <= n; k++)
117         {
118             t_rc.push_back(t[j][k]);
119         }
120
121         int red = reduce(t_rc);
122         if (red >= inf)
123             bound_temp = bound_temp;
124         else
125             bound_temp = bound_temp + red;
126
127         for (int k = 1; k <= n; k++)
128         {
129             t[j][k] = t_rc[k - 1];
130         }
131     }
132
133     for (int j = 1; j <= n; j++)
134     {
135         vector<int> t_rc;
136         for (int k = 1; k <= n; k++)
137         {
138             t_rc.push_back(t[k][j]);
139         }
140
141         int red = reduce(t_rc);
142         if (red >= inf)
143             bound_temp = bound_temp;
144         else
145             bound_temp = bound_temp + red;
146
147         for (int k = 1; k <= n; k++)
148         {
149             t[k][j] = t_rc[k - 1];
150         }
```

```
151         }
152
153         temp.push_back({bound_temp, i, t});
154
155
156
157     }
158     sort(temp.begin(), temp.end());
159
160     auto it = temp[0];
161     r = get<2>(it);
162     ans.push_back(get<1>(it));
163     node = get<1>(it);
164
165
166
167     itr++;
168 }
169
170 int sum = main[ans[ans.size() - 1]][1];
171 cout<<"optimal path is: ";
172 for (int i = 0; i < ans.size(); i++)
173 {
174     if (i > 0)
175     {
176
177         sum = sum + main[ans[i - 1]][ans[i]];
178     }
179     cout << ans[i] << "-> ";
180 }
181
182 cout << 1 << endl;
183
184 cout<<"And the cost is: " << sum;
185 }
186
187 // 5 11
188 // 1 2 9
```

```
189 // 1 4 8
190 // 2 3 4
191 // 2 5 2
192 // 3 2 3
193 // 3 4 4
194 // 4 2 6
195 // 4 3 7
196 // 4 5 12
197 // 5 1 1
198 // 5 4 10
199
200 // another input
201
202 // 4 10
203 // 1 2 4
204 // 1 3 12
205 // 1 4 7
206 // 2 1 5
207 // 2 4 18
208 // 3 1 11
209 // 3 4 6
210 // 4 1 10
211 // 4 2 2
212 // 4 3 3
```