```cpp
#include<iostream>
#include<algorithm>
#include<limits.h>
#include<string.h>
#include<iomanip>
#include<math.h>
#include<cmath>
#include<random>
#include<queue>
//#include<bits/stdc++.h>
using namespace std;
const double INF=1000000.0;
queue <int> q;
bool visited[401];
int T, m, E, n, s, t, u, v;
double t0, a1, a2;
int Q[8001][3], k[401],path[401][2];//, Q_temp[8001][3]
//int path[8001][401][2];
double dist[401];
void SPFA(int,int);
//void SPFA(int, int, int);
void printAndUpdate(int, int);//更新road的te并打印该路径结果
void Print(int, int,int);
void Update(int, int,int);
struct node
{
    int end;//目的地
    double te;//行驶时间（会变化，需更新）
    double t0;//空载时（即没有车辆通过）的行驶时间
    double a;//拥堵参数
    //int fe;//流量fe为所有通过路段e的车辆总和
    int next;
}road[10001 * 2];
int main()
{
    cin >> T;
    for (int i = 0; i < T; i++)
    {
        memset(k, -1, sizeof(k));
        cin >> m >> E >> n;//分别表示路口结点总数、双向路段总数、导航请求总数n
        for (int j = 0; j < n; j++)
        {
            cin >> s >> t;//(0<=s<n)、(0<=t<n)、s≠t，请求的出发点和目的地点
            //Q_temp[j][0] =
                Q[j][0] = s;
            //Q_temp[j][1] =
                Q[j][1] = t;
            //Q_temp[j][2] =
                Q[j][2] = j;
        }
        for (int j = 0; j < 2 * E-1; j += 2)
        {
            cin >> u >> v >> t0 >> a1 >> a2;
            road[j].end = v;
            road[j].te = t0;//最初行驶时间定义
            road[j].t0 = t0;
```

```
57              road[j].a = a1;
58              road[j].next = k[u];
59              //road[j].fe = 0;
60              k[u] = j;
61              road[j + 1].end = u;
62              road[j + 1].te = t0;
63              road[j + 1].t0 = t0;
64              road[j + 1].a = a2;
65              road[j + 1].next = k[v];
66              //road[j + 1].fe = 0;
67              k[v] = j + 1;
68          }
69
70          /*
71          for (int j = 0; j < n; j++)
72          {
73              int k = rand() % (j+ 1) / (j + 1) * (j + 1);
74              swap(Q_temp[j][0], Q_temp[k][0]);
75              swap(Q_temp[j][1], Q_temp[k][1]);
76              swap(Q_temp[j][2], Q_temp[k][2]);
77          }
78          */
79
80          for (int j = 0; j < n; j++)
81          {
82              SPFA(Q[j][0], Q[j][1]);//最短路径：dist(Q[j][0], Q[j][1])
83              printAndUpdate(Q[j][0], Q[j][1]);//更新road的te并打印该路径结果
84              //SPFA(Q_temp[j][0], Q_temp[j][1], Q_temp[j][2]);
85              //Update(Q_temp[j][0], Q_temp[j][1], Q_temp[j][2]);
86          }
87          //for(int j = 0; j < n; j++)
88          //  Print(Q[j][0], Q[j][1], Q[j][2]);
89          /*
90          for (int j = 0; j < ceil(double(n) / 500.0); j += 500)
91          {
92              for (int k = 0; k < min(500, n - 500 * j); k++)
93              {
94                  SPFA(Q[j+k][0], Q[j+k][1]);//最短路径：dist(Q[j][0], Q[j][1])
95                  Print(Q[j+k][0], Q[j+k][1]);
96              }
97              for (int k = 0; k < min(500, n - 500 * j); k++)
98                  Update(Q[j+k][0], Q[j+k][1]);//更新road的te并打印该路径结果
99          }*/
100     }
101     return 0;
102 }
103 void SPFA(int s, int t)
104 //void SPFA(int s, int t,int index)
105 {
106     for (int i = 0; i < m; i++)
107         dist[i] = DBL_MAX;
108     memset(visited, 0, sizeof(visited));
109 //  memset(path, -1, sizeof(path));
110     for (int i = 0; i < 401; i++)
111         for (int j = 0; j < 2; j++)
112             //path[index][i][j] = -1;
```

```
113             path[i][j] = -1;
114     for (int j = 0; j < 2 * E - 1; j += 2)
115     {
116         v = road[j].end;
117         u = road[j + 1].end;
118         path[v][0] = u;
119         path[v][1] = j;
120         path[u][0] = v;
121         path[u][1] = j + 1;
122         //path[index][v][0] = u;////////////////
123         //path[index][v][1] = j;//////////////
124         //path[index][u][0] = v;///////////
125         //path[index][u][1] = j + 1;//////////////
126     }
127     int start = s;
128     while (!q.empty())q.pop();
129     dist[s] = 0.0;
130     visited[s] = 1;
131     q.push(s);
132     while (!q.empty())
133     {
134         start = q.front();
135         q.pop();
136         visited[start] = 0;
137         for (int x = k[start]; x != -1; x = road[x].next)//所有以start为起点的路在
              road中的位置
138         {
139             int y= road[x].end;//y表示这个路的终点
140             //cout << road[x].te<<"******"<< dist[start] << endl;
141             ///cout << "sssss" << start <<"dist"<< dist[start] <<"yyyyy"<<y<<
                "dist_y" << dist[y] << endl;
142             if (dist[y] > dist[start] + road[x].te)
143             {
144                 //cout << "******" << setprecision(20)<<dist[y] << endl;
145                 path[y][0] = start;
146                 path[y][1] = x;//用于打印路径
147                 //path[index][y][0] = start;////////
148                 //path[index][y][1] = x;    //////////////
149                 dist[y] = dist[start]+ road[x].te;
150                 if (!visited[y])
151                 {
152                     visited[y] = 1;//从s到y的路程发生了改变，则以y为起点的路的长度可
                        能也改变
153                     q.push(y);
154                 }
155             }
156         }
157     }
158 }
159
160 void printAndUpdate(int start, int end)
161 {
162     int result[10001];
163     int num=1,num_e=0;
164     for (int i = end; i !=start; i = path[i][0])
165     {
```

```
166         //cout << "****"<<i << endl;
167         result[num_e++] = i; num += 2;
168         road[path[i][1]].te += road[path[i][1]].a * road[path[i][1]].t0;
169     }
170     cout << num << endl;
171     cout << start << " ";
172     for (int i = num_e - 1; i >= 0; i--)
173     {
174         cout << path[result[i]][1] << " " << result[i];
175         if (i != 0)cout << " ";
176         else cout << endl;
177     }
178 }
179 
180 /*
181 void swap(int a, int b)
182 {
183     int temp = a;
184     a = b;
185     b = temp;
186 }
187 */
188 
189 /*
190 void printAndUpdate(int end,int num)
191 {
192     if (path[end][0]!=-1)
193         printAndUpdate(path[end][0],num+2);
194     if (path[end][1] != -1)
195     {
196         cout << " " << path[end][1] << " " << end;
197         road[path[end][1]].te += road[path[end][1]].a* road[path[end][1]].t0;
198         //cout << road[path[end][1]].te << endl;
199     }
200     else
201     {
202         cout << num + 1 << endl;
203         cout << end;
204     }
205 }
206 */
207 
208 /*
209 void Print(int start, int end,int index)
210 {
211     int result[10001];
212     int num = 1, num_e = 0;
213     for (int i = end; i != start; i = path[index][i][0])
214     {
215         result[num_e++] = i; num += 2;
216         //road[path[end][1]].fe++;
217     }
218     cout << num << endl;
219     cout << start << " ";
220     for (int i = num_e - 1; i >= 0; i--)
221     {
```

```
222         cout << path[index][result[i]][1] << " " << result[i];
223         if (i != 0)cout << " ";
224         else cout << endl;
225     }
226 }
227
228 void Update(int start, int end,int index)
229 {
230     for (int i = end; i != start; i = path[index][i][0])
231         road[path[index][end][1]].te += road[path[index][end][1]].a * road[path
              [index][end][1]].t0;
232 }*/
```