```cpp
#include <bits/stdc++.h>

using namespace std;

#define F(i,L,U) for((i)=(L);(i)<(U);(i)++)
#define FE(i,L,U) for((i)=(L);(i)<=(U);(i)++)
typedef vector<int> vi;
typedef pair<int, int> ii;
typedef pair<int, double> id;
typedef vector<ii> vii;
typedef vector<id> vid;

#define INF 1000000000

// class UF{
// public:
//   vi p, rank, setSize, m;
//   int numSets;
//
//   UF(int N){
//     m.assign(N, 0);
//     setSize.assign(N, 1); numSets = N; rank.assign(N,0);
//     p.assign(N,0);
//     for(int i = 0; i < N; i++) {
//       p[i] = i;
//       m[i] = i;
//     }
//
//   }
//
//   int findSet(int i){
//
//     if (p[i] == i){
//       return i;
//
//     }else{
//      int x =  findSet(p[i]);
//      m[i] = max(m[i], m[x]);
//      return p[i] = x;
//
//     }
//
// //     return (p[i] == i) ? i : (p[i] = findSet(p[i]));
//
//   }
//
//   bool isSameSet(int i, int j){
//     return findSet(i) == findSet(j);
```

```
49: //   }
50: //
51: //   void unionSet(int i, int j){
52: //     if (!isSameSet(i,j)){
53: //       numSets--;
54: //       int x = findSet(i), y = findSet(j);
55: //
56: //       if (rank[x] > rank[y]){
57: //      m[x] = max(m[x], m[y]);
58: //      p[y] = x; setSize[x] += setSize[y];
59: //       }else{
60: //      m[y] = max(m[x], m[y]);
61: //      p[x] = y; setSize[y] += setSize[x];
62: //
63: //      if (rank[x] == rank[y]) rank[y]++;
64: //       }
65: //
66: //
67: //     }
68: //
69: //   }
70: //
71: // };
72:
73:
74: #define LINE(line) fgets(line, sizeof(line), stdin)
75:
76: int T, F, F_u, F_n;
77:
78: vector<vid> AdjList;
79:
80:
81: int main(){
82:   int i, j, k, u, v;
83:   double w;
84:
85:   while(scanf("%d %d %d %d", &T, &F, &F_u, &F_n), (T||F||F_u||F_n)){
86:
87:     F_u--;
88:     F_n--;
89:
90:     AdjList.assign(F, vid());
91:
92:     F(i,0,T){
93:       scanf("%d %d %lf", &u, &v, &w);
94:       u--;
95:       v--;
96:       w *= -1;
```

```
 97:       AdjList[u].push_back(id(v, w));
 98:     }
 99:
100:     vector<double> dist(F, INF); dist[F_u] = 0.0;
101:     F(i,0,F-1){
102:       F(u,0,F){
103:          F(j,0,(int)AdjList[u].size()){
104:             id v = AdjList[u][j];
105:             dist[v.first] = fmin(dist[v.first], dist[u] + v.second);
106:          }
107:       }
108:     }
109:
110:     if (dist[F_n] == (double) INF){
111:       printf("FALSE\n");
112:       continue;
113:     }
114:
115:     bool has = false;
116:
117:      F(u,0,F){
118:          F(j,0,(int)AdjList[u].size()){
119:             id v = AdjList[u][j];
120:             if (dist[v.first] > dist[u] + v.second ){
121:               has = true;
122:             }
123:          }
124:       }
125:
126:     if (has){
127:       printf("TRUE\n");
128:     }else{
129:       printf("FALSE\n");
130:     }
131:   }
132:
133:
134:   return 0;
135: }
```