Conquista

Autor: Huize Mao

Utilizamos el algoritmo de Dijkstra para calcular las distancias a todos los vértices desde los vértices iniciales de cada jugador. Aprovechamos que el algoritmo de Dijkstra permite determinar todas las ciudades visitadas hasta un cierto instante de tiempo para gestionar los códigos secretos mientras hacemos el Dijkstra. Hay que tener en cuenta ciertos detalles en la implementación.

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef vector<int> vi;
typedef long long ll;
#define sz(x) (int) (x).size()
#define rep(i, a, b) for(int i = a; i < (b); ++i)
#define pb push_back
#define all(x) begin(x), end(x)
#define pii pair<ll, ll>
#define dpii pair<ll, pii>
#define fir first
#define sec second

const int N = 2e5 + 6;

struct P {
    int u, v;
    ll w;
};
int n, m, h, j;
vector<P> act[N], e[N];
ll dist[3][N]; int ocup[N];

struct CustomCompare {
    bool operator()(const pair<ll, pair<ll, ll>>& a, const pair<ll, pair<ll, ll>>& b) const {
        // Compare based on x.first, and if equal, use x.second.first
        if (a.first != b.first) {
            return a.first < b.first;  // Larger x.first is preferred
        } else {
            return a.second.first < b.second.first;  // Larger x.second.first is preferred in case of a tie
        }
    }
};

priority_queue<pair<ll, pii>, vector<pair<ll, pii>>, CustomCompare> pq;

void re_act() {
    for(int i = 1; i <= n; i ++) {
        act[i].clear(), e[i].clear();
        ocup[i] = 0;
        dist[1][i] = dist[2][i] = 1e17;
    }
}

int main() {
    int T;
    cin >> T;
    while(T--) {
        cin >> n >> m;
        re_act();
        rep(i, 0, m) {
            int u, v, w, c; cin >> u >> v >> w >> c;
            if(c) act[c].pb({u, v, w});
            else {
                e[u].pb({0, v, w});
                e[v].pb({0, u, w});
            }
        }
        cin >> h;
        rep(i, 0, h) {
```

```
            int st; cin >> st;
            dist[1][st] = 0;
            pq.push({0, {1, st}});
        }
        cin >> j;
        rep(i, 0, j) {
            int st; cin >> st;
            dist[2][st] = 0;
            pq.push({0, {2, st}});
        }

        while(pq.size()) {
            auto p = pq.top().sec; pq.pop();
            int pers = p.fir, ciud = p.sec;
            if(ocup[ciud]) continue ;
            ocup[ciud] = pers;
            for(auto x : act[ciud]) {
                int num_ocup = 0;
                if(ocup[x.u]) ++num_ocup;
                if(ocup[x.v]) ++num_ocup;
                if(num_ocup == 2) {
                    continue ;
                } else if(num_ocup == 0) {
                    e[x.u].pb({pers, x.v, x.w});
                    e[x.v].pb({pers, x.u, x.w});
                } else {
                    if(ocup[x.u]) {
                        if(ocup[x.u] != pers) continue ;
                        else {
                            if(dist[pers][ciud] + x.w < dist[pers][x.v]) {
                                dist[pers][x.v] = dist[pers][ciud] + x.w;
                                pq.push({-dist[pers][x.v], {pers, x.v}});
                            }
                        }
                    } else {
                        if(ocup[x.v] != pers) continue ;
                        else {
                            if(dist[pers][ciud] + x.w < dist[pers][x.u]) {
                                dist[pers][x.u] = dist[pers][ciud] + x.w;
                                pq.push({-dist[pers][x.u], {pers, x.u}});
                            }
                        }
                    }
                }
            }
            for(auto nxt : e[ciud]) {
                if(nxt.u && nxt.u != pers) continue ;
                int nxt_ciu = nxt.v; ll w = nxt.w;
                if(!ocup[nxt_ciu] && dist[pers][ciud] + w < dist[pers][nxt_ciu]) {
                    dist[pers][nxt_ciu] = dist[pers][ciud] + w;
                    pq.push({-dist[pers][nxt_ciu], {pers, nxt_ciu}});
                }
            }
        }

        int jvanilla = 0, hmao = 0;
        for(int i = 1; i <= n; i ++) {
            if(ocup[i] == 1) ++hmao;
            else if(ocup[i] == 2) ++jvanilla;
        }
        int dif = jvanilla - hmao;
        if(dif > 0) {
            cout << "La hora de juego\n";
            cout << dif << endl;
        } else cout << "Hasta luego Huize, es la hora de Olivia\n\n";
    }
}
```