CrossCountry Canada (WC '17 R1 S3)

https://wcipeg.com/problem/wc171s3

Problem

There is a given undirected weighted graph with N ($2 \le N \le 1{,}000$) vertices and M ($1 \le M \le 10{,}000$) edges. The shortest path from vertex 1 to N must be found - with a twist. No set of consecutive edges in the path can have a total length of more than L ($1 \le L \le 100$) without a 'stop' of length T ($1 \le T \le 100$) at any given marked city. If there is no possible path, then print -1.

Solution

Instead of thinking of the graph as N vertices, think of it as N * (L+1) vertices. Each vertex A in the initial graph is split into L+1 vertices (A,0), (A,2), … ,(A,L). If there is an undirected edge of length C from vertex A to vertex B, then in our new graph there are directed edges of length C from (A,0) -> (B,C), (A,1) -> (B,C+1), … , (A,L-C) -> (B,L), and directed edges of length C from (B,0) -> (A,C), (B,1) -> (A,C+1), … ,(B,L-C) -> (A,L). If a vertex D is designated as a 'rest stop', then there are edges of length T from (D,L) -> (D,0), (D,L-1) -> (D,0), … , (D,1) -> (D,0). All the 'gimmicks' with the problem using the initial graph are resolved here, and so with this new graph of N * (L+1) vertices a basic Dijkstra implementation can return the shortest path - which is the minimum of the shortest paths to (N,0), (N,1), … ,(N,L).

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

#define f first
#define s second

int dists[1010][110];
vector<pair<int, int>> G[1010];
int tims[1010];
int N,M,L,T,V1,V2,V3;

int main(){
    memset(dists, -1, sizeof dists);
    cin >> N >> M >> L >> T;
    for(int i = 0; i < N; i++) cin >> tims[i+1];

    priority_queue<pair<int, pair<int, int>>, vector<pair<int, pair<int, int>>>,
greater<pair<int, pair<int, int>>>> Q;

    for(int i = 0; i < M; i++){
        cin >> V1 >> V2 >> V3;

        G[V1].push_back({V3,V2});
        G[V2].push_back({V3,V1});
    }

    Q.push({0, {0,1}});
    while(!Q.empty()){
        pair<int, pair<int, int>> v = Q.top();
        Q.pop();

        if(dists[v.s.s][v.s.f] == -1){
            dists[v.s.s][v.s.f] = v.f;
            for(int i = 0; i < G[v.s.s].size(); i++){
                if(v.s.f + G[v.s.s][i].f <= L){
                    Q.push({v.f + G[v.s.s][i].f, {v.s.f + G[v.s.s][i].f, G[v.s.s][i].s}});
                }
            }
            if(tims[v.s.s] == 1){
                for(int i = 0; i < G[v.s.s].size(); i++){
                    if(G[v.s.s][i].f <= L){
                        Q.push({v.f + G[v.s.s][i].f + T, {G[v.s.s][i].f, G[v.s.s][i].s}});
                    }
                }
            }
        }
    }
    int tot = 1e+9;
    for(int i = 0; i < 101; i++){
        if(dists[N][i] != -1) tot = min(tot, dists[N][i]);
    }
    if(tot == 1e+9){
        cout << -1;
        return 0;
    }
    cout << tot << endl;
}
```