Railway planning

Problem ID: 6railwayplanning

Introduction

The railway system was heavily oversized during the end of the Cold War and the party secretary from Minsk has been tasked with liquidate money from the budget by removing some routes. Still the minister of transport demands that it should be possible to transport C people a day from Minsk to Lund, since that is the number of students that want to take part in the incredible course in Algorithms.

Luckily, as help, you have a map over all routes. Furthermore you have constructed a list over the routes that would be most profitable to remove.

Aims

The goals of the lab are:

- Implementing a Network-Flow-algorithm.
- Debugging your code.
- Structuring your code in a logical fashion.
- Reason about correctness of your algorithm.
- Reason about upper bounds for time complexity.

Problem formulation

You are given the structure of the railway system, in the form of nodes (cities) and edges (routes connecting the cities). For each edge you will be given the capacity (the number of students it can carry). Then you will be given a plan for which routes to remove in a specific order – in which you need to remove the routes if possible. If you cannot (which you cannot if the total maximal flow from the Minsk to Lund is less than C) remove a route you stop your plan and instead you start implementing the plan. Your task is to answer how many of the routes on your list you can remove and what the maximal flow of students from Minsk to Lund will be after removing these routes.

Input

The first line consists of four integers N, M, C, P, the number of nodes, the number of edges, the number of students to transfer from Minsk (node 0) to Lund (node $N-1$) and the number of routes in your plan. It is guaranteed that $2 \leq N \leq 1000, 1 \leq M \leq \frac{N(N-1)}{2}, 0 \leq P \leq M$ and that the capacity of the network before any routes are removed is at least C. Then follows M lines where the i-th line (0-indexed) consists of three integers u_i, v_i, c_i, where $0 \leq u_i, v_i < N$ are the nodes and c_i is the capacity of the route, $1 \leq c_i \leq 100$. Note that students can travel both from u_i to v_i and from v_i to u_i (the graph is undirected) but in total at most c_i soldiers. It is guaranteed that the each pair of nodes occurs at most once in the input and that $u_i \neq v_i$. Then follows P lines with one integer each, where the i-th line contains the index of the i-th route you want to remove. Note that you have to remove them in exactly this order as they are given and it is guaranteed that all routes you want to remove are unique in the input.

Output

The output should contain one line with two space-separated integers x, f, where x is the number of routes you can remove (while still having a maximal flow of at least C) and f is the maximal flow from node 0 to node $N-1$ after removing these routes.

Examination and Points of Discussion

To pass the lab make sure you have:

- Have successfully implemented the algorithm with the correct time complexity.

- Have neat and understandable code.
- Have descriptive variable names.
- Have filled in the blanks in the report.
- Have run the check_solution script to validate your solution.

During the oral presentation you will discuss the follwoing questions with the lab assistant:

- What is the time complexity, and more importantly why?
- Which other (well-known) algorithmic problems can be solved using Network-Flow?
- If the capacities of the edges are very large, how can one get a different (better) time complexity?

Sample Input 1	**Sample Output 1**
3 3 10 3 0 1 10 0 2 10 1 2 10 0 2 1	2 10