

Southeastern European Regional Programming Contest
Bucharest, Romania
October 15, 2011

Problem H
Road Planner

Input File: H.IN
Output File: standard output
Program Source File: H.C, H.CPP, H.JAVA

An acyclic road network is composed of unidirectional road segments (edges), each connecting two intersections (vertices) out of a total of `N` intersections. The time necessary to travel on a road segment is always positive and it depends linearly on the number of cars `C` using the segment `s`:

$$T(s) = a_s C + b_s$$

a_s and b_s are two constants expressed as single precision floating point numbers, describing the properties of road segment `s`. The time necessary to cross an intersection is `0`.

A number of cars must travel from vertex `0` to vertex `N-1` in this road network. Each car chooses its route selfishly in order to reduce its own travel time; each car knows that all the other cars will also selfishly choose their route. You are asked to write a program that computes how many cars will travel on each of the possible paths from the source to the destination, and the time needed to travel on those paths.

The input file contains several tests and is organized as follows. The first line contains the number of tests. The following lines contain the specification of the tests. Each test contains on the first line the number of vertices, the number of edges and the number of cars, separated by whitespaces. After that, each edge is given on a separate line and contains the following fields separated by spaces: source vertex, destination vertex, a_s and b_s
The output will contain one line for each test input, specifying the minimum time for any car to travel in the network, rounded down to the nearest integer.

Example Input:	**Example output:**
2 4 4 4000 0 1 0.01 0 0 2 0 45.1 1 3 0 45.1 2 3 0.01 0 4 5 4000 0 1 0.01 0 0 2 0 45.1 1 3 0 45.1 1 2 0 0 2 3 0.01 0	65 80

The first input consists of `4` vertices and `4` edges; `4000` cars must travel from vertex `0` to vertex `3`. To achieve the smallest travel time, cars will choose one of the two possible paths `(0-1-3)` and `(0-2-3)` such that the number of cars on both is equal; hence minimum travel time is lower `(0.01 * 2000 + 45.1) = 65`.

The second input is similar to the first one, except for a zero cost edge that is added between vertices `1` and `2`. In this case, all cars selfishly choose to take the path `(0-1-2-3)` and the minimum travel time is `80`.