Southeastern European Regional Programming Contest

October 15, 2005

Bucharest, Romania

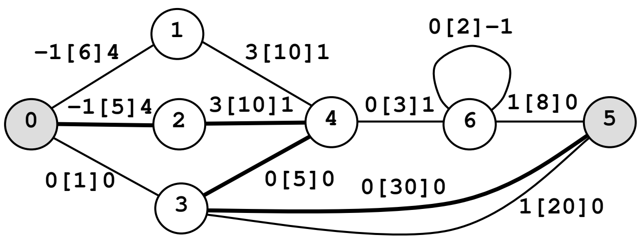
Problem F

Adventurous Driving

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

After a period of intensive development of the transportation infrastructure, the government of Ruritania decides to take firm steps to strengthen citizens' confidence in the national road network and sets up a compensation scheme for adventurous driving (CSAD). Those driving on a road with holes, bumps and other entertaining obstacles get compensation; those driving on a decent road pay tax. These compensations and taxes are obtained and paid in cash on entry on each road and depend on the entry point on the road. What you get and pay driving on a road from **A** to **B** may be different from what you get and pay driving on the same road from **B** to **A**. The Ruritarian authorities call *fee* the amount of money paid as tax or obtained as compensation on entry on a road. A positive fee is a tax; a negative fee stands for compensation.

John Doe plans to take advantage of CSAD for saving money he needs to repair his old car. When driving from **A** to **B**, John follows a path he calls *optimal*: a path that is *rewarding* and has the minimal length out of the paths with the minimal *weight* from **A** to **B**. In John's opinion, a path is rewarding if all the roads in the path are rewarding, and a road **(X,Y)** is rewarding if it has the minimal entry fee out of the roads leaving **X**. The weight of a path is the sum of the entry fees paid along the path. The length of a path cumulates the length of the roads in the path. The problem is helping John to compute the weight and the length of an optimal path from **A** to **B** on a given map.



For example, on the illustrated road map vertices designate cities and edges stand for roads. The label **fuv[L]fvu** of the road **(u,v)** shows the fee **fuv** for driving from **u** to **v**, the fee **fvu** for driving from **v** to **u**, and the length **L** of the road. The path **(0,2,4,3,5)** from **0** to **5** is optimal: it is rewarding, has weight **2** (**-1+3+0+0**) and length **50** (**5+10+5+30**). The path **(0,1,4,3,5)**, although rewarding and of weight **2**, has length **51**. The path **(0,3,5)** has weight **0** and length **20** but it is not rewarding.

Write a program that reads several data sets from a text file. Each data set encodes a road map and starts with four integers: the number **1≤n≤100** of towns on the map, the number **0≤m≤5000** of roads, the departure town **0≤A≤n-1**, and the destination town **0≤B≤n-1**. Follow **m** data quintuples **(u,v,fuv[L]fvu)**, where **u** and **v** are town identifiers (integers in the range **0..n-1**), **100≤fuv, fvu≤100** are integer fees for driving on the road **(u,v)**, and **1≤L≤100** is the integer length of the road. The quintuples may occur in any order. Except the quintuples, which do not contain white spaces, white spaces may occur freely in input. Input data terminate with an end of file and are correct. For each data set, the program prints – from the beginning of a line – the weight and the length of an optimal path, according to John's oppinion, from **A** to **B**. If there is no optimal path from **A** to **B** the text **VOID** is printed. If the weight of the optimal path from **A** to **B** has no lower bound the text **UNBOUND** is printed.

Input	Output
`3 3 0 2 (0,1,0[1]0) (0,2,1[1]0) (1,2,1[1]0)`	**VOID**
`3 3 0 2 (0,1,-1[1]1) (0,2,0[1]0) (1,2,0[1]1)`	**UNBOUND**

`7 11 0 5 (0,1,-1[6]4) (0,2,-1[5]4) (0,3,0[1]0) (1,4,3[10]1)`
`(2,4,3[10]1) (3,4,0[5]0) (3,5,0[30]0) (3,5,1[20]0)`
`(4,6,0[3]1) (6,5,1[8]0) (6,6,0[2]-1)`
	`2 50`

An input/output sample is in the table above. The first data set encodes a road map with no optimal path from `0` to `2`. The second data set corresponds to a map whose optimal path from `0` to `2` has an unbound weight. The third data set encodes the road map shown in the above figure.