South African Computer Olympiad
Training Camp 1, 2007
Day 2

Overview

Author	**Carl Hultquist**	**Marco Gallotta**	**Richard Starfield**
Problem	**gridhunt**	**cheese**	**pizza**
Source	gridhunt.java gridhunt.c gridhunt.cpp gridhunt.pas	cheese.java cheese.c cheese.cpp cheese.pas	pizza.java pizza.c pizza.cpp pizza.pas
Input file	stdin	cheese.in	pizza.in
Output file	stdout	cheese.out	pizza.out
Time limit	1 second	2 seconds	1 second
Number of tests	20	10	10
Points per test	5	10	10
Total points	**100**	**100**	**100**

The maximum total score is 300 points.

South African Computer Olympiad
Training Camp 1, 2007
Day 2

Grid Hunt

Author

Carl Hultquist

Introduction

Task

The computer has a secret $N \times M$ grid of numbers, which increase strictly from left to right and from top to bottom. You can query what the value of a number is at any location in the grid. Given a target number to search for, find its position in the grid using as few queries as possible. You are guaranteed that the target number can be found. If the number can be found at more than one location in the grid, you only need to find one of the locations.

Example

Suppose the grid of numbers is as follows:

1	3	4
2	4	5
5	6	8

If we were looking for the number 5, for example, it's found in either row 3 column 1, or row 2 column 3.

Interaction

Interface with a reactive program via stdin and stdout (`cin` and `cout` in C++; `System.in` and `System.out` in Java). The first line read by your program from stdin will contain 3 space-separated integers, N, M and T, which are the number of rows, number of columns, and target number respectively. To query a cell in the grid, your program should write the string "`Q` r c" to stdout (without the quotes), where r and c are the row and column coordinates of the cell you wish to know the value of, then read a line from stdin which will contain the number. When you can identify which cell the target is in, write the string "`I` r c" to stdout (without the quotes), giving the coordinates of the cell.

For testing purposes, you may upload a data file. The first line contains N, M and T. The remaining lines contain the data in the grid.

Sample input

```
3 3 5
1 3 4
2 4 5
5 6 8
```

Sample interaction

Output	Input
	3 3 5
Q 1 1	1
Q 1 2	3
Q 1 3	4
Q 2 3	5
I 2 3	

Flushing

After writing each line to stdout, you need to flush it so that the evaluation will receive it. This can be done as follows:

C `fflush(stdout);`

C++ `cout.flush();`

Java `System.out.flush();`

Constraints

- $1 \leq N, M \leq 5000$
- $1 \leq V_{i,j} < 2^{31}$ for the value of any number $V_{i,j}$ in the grid

50% constraints

- $1 \leq N, M \leq 50$

Time limit

1 second.

Scoring

If your program does not identify a cell containing the target number for a test case, then you score 0% for that test case.

Assuming you correctly identify a cell containing the target number: if your program takes fewer than $M + N$ queries for a test case, then you score 100% for that test-case. Otherwise, you score $\frac{M+N}{Q}75\%$, where Q is the number of queries used by your program.

The Cheese Universe

Author

Marco Gallotta

Introduction

Wensleydale has to this day been baffled as to where all his cheese as been disappearing to. The program you wrote for him last year absolutely fascinated him. There was a small problem though, as all the cheese he had purchased disappeared before he could start selling it off. This happened every time he purchased a new batch.

Every night Wensleydale dreams about all his cheese. He dreams of one day finding a universe full of cheese. This cheese universe exists of almost nothing but cheese. If only this dream could come true!

Task

Wensleydale needs your help yet again. He gets so excited during his dream that he really does think that it may come true. However silly he may sound, you agree to help him out.

In his dream he often needs to get to a particular point in the universe. Since there is so much cheese, he wants to eat the minimum amount of cheese that is required to reach the required point.

Now usually this would be very easy — just travel in a straight line. However, most cheese have spherical holes filled with air. Wensleydale can enter a hole and travel through it without having to eat any cheese. This reduces the amount of cheese he has to eat. The holes in the cheese sometimes overlap one another, creating holes of various shapes.

Your task is to write a program that will tell Wensleydale the minimum amount of cheese he needs to eat to get from his starting position to the position he wants to end up at. The universe is of infinite size, so you can never leave the universe once you are inside.

Example

Suppose Wensleydale starts off at position $(0, 0, 0)$ and wants to get to the point $(6, 0, 0)$ and that there is only one hole with centre $(3, 0, 0)$ and radius 2.

The best route would be to go directly to the hole, exit the other side and then directly to the destination point. This would require him to eat 1 unit length of cheese to get into the hole and another 1 unit length to get from the other side of the hole to the final point, giving a total of 2.

Input (cheese.in)

The first line contains the three values, x_0, y_0, z_0, representing the starting point. The second line contains the three values, x_1, y_1, z_1, representing the destination point. The third line contains a single value, N, the number of holes in the universe.

The next N lines each contain four values x_i, y_i, z_i, r_i. These represent the centre (x_i, y_i, z_i) and radius r_i of the holes.

Sample input

```
0 0 0
6 0 0
1
3 0 0 2
```

Output (cheese.out)

The output must contain a single value, the minimum amount of cheese required to get from the starting point to the destination point. This value must be rounded off to two decimal places.

Sample output

```
2.00
```

Constraints

- $0 \leq N \leq 1000$
- $-2^{31} < x_i, y_i, z_i < 2^{31}$

50% constraints

- $0 \leq N \leq 100$

Time limit

2 seconds.

South African Computer Olympiad
Training Camp 1, 2007
Day 2

Scoring

An optimal answer will score 100%, while a sub-optimal or invalid answer will score 0%.

Grand Pizza Network

Author

Richard Starfield

Introduction

The Guji village is abuzz — a brand new set of pizza ovens has been installed in their huts, allowing them to provide their famous pizza feasts to twice the tourists at half the price! Although the cooking is done via a wood fire for that distinctive flavour, the fires can be started or doused using convenient electronic switches provided with the ovens.

There is a slight glitch, however. The wiring was done by Manic Fred, who got a bit carried away, and now switching an oven on or off in one hut automatically flips the states of all ovens (from on to off or vice versa) in neighbouring huts. Luckily these secondary changes don't affect any further ovens.

Task

The afternoon before a feast the Guji decide to enlist your aid. Given the hut layout and the current states of the ovens, the Guji would appreciate a list of the switches they need to flip so that all the ovens' fires are lit and ready.

Fred used the minimum amount of wiring to connect all the huts together, so there is exactly one route between any pair of huts.

Example

Suppose there are 3 huts in a line, numbered 1, 2 and 3, with the fires in 2 and 3 currently lit. The following sequence of switch flips will ignite all fires:

- flipping 3 will douse the fires in 2 and 3, leaving all fires off
- flipping 2 will light the fires in 1, 2, and 3.

Input (pizza.in)

The first line contains N, the number of huts. The huts are numbered from 1 to N. The next $N-1$ lines contain two space-separated integers, h_1 and h_2, specifying that those two huts are neighbours. The next N lines, one for each hut, contain the digit 0 or 1 to indicate whether the oven in that hut is initially on (1) or off (0).

Sample input

```
3
1 2
2 3
0
1
1
```

Output (pizza.out)

The output should consist of N lines, each containing a 0 or 1. A 1 on the ith line indicates that the switch belonging to the oven in the ith hut should be flipped. If it is not possible to light all the ovens, the output should consist of one line containing the integer -1. If there are multiple solutions, you may output any of them.

Sample output

```
0
1
1
```

Constraints

- $1 \leq N \leq 10000$

50% constraints

- $1 \leq N \leq 10$

Time limit

1 second.

Scoring

A correct solution (one that correctly lights all the fires or indicates if this is impossible) scores 100%. Anything else scores 0%.