South African Computer Olympiad
Final Round
Day 2: Open

Overview

Author	**Graham Poulter**	**Max Rabkin**	**USAICO 2006**
Problem	**seen**	**ptotoot**	**cave**
Source	seen.java seen.py seen.c seen.cpp seen.pas	ptotoot.java ptotoot.py ptotoot.c ptotoot.cpp ptotoot.pas	cave.java cave.py cave.c cave.cpp cave.pas
Input file	seen.in	ptotoot.in	cave.in
Output file	seen.out	ptotoot.out	cave.out
Time limit	2 seconds	2 seconds	1 second
Number of tests	10	10	10
Points per test	10	10	10
Total points	**100**	**100**	**100**

The maximum total score is 300 points.

South African Computer Olympiad
Final Round
Day 2: Open

How Not To Be Seen

Author

Graham Poulter

Introduction

The How Not To Be Seen competition is going well, with several contestants not yet eliminated. They are presently hiding in the Queen's Garden, which has an unusual layout. The Queen's Garden is made up of C columns, each of which may be planted with trees, and R rows, each of which may be planted with shrubbery. We mark each row or column with a "1" if it planted, and mark it with a "0" otherwise. A square with neither tree nor shrubbery is too open to hide in, and a square with both is too crowded for hiding. A square with one or the other is a prime hiding spot. However, for the contest you can only rent a rectangle H rows high by W columns wide, which should be placed in an area with the greatest number of crowded or open spots (and hence the fewest hiding spots).

Task

Given the markings along the rows and columns, and two numbers H and W, find a rectangle that is H rows high and W columns wide which contains the most crowded or open squares possible, and output the number of such squares in that rectangle.

Example

Suppose the garden has $R = 4$ rows and $C = 5$ columns. The markings down the rows are 0011 and the markings across the columns are 10110. You are required to find the rectangle with height $H = 2$ and width $W = 3$ that contains the most crowded and open squares, and determine the number of such squares it contains.

If we represent the crowded or open squares with "1" and the hiding-spot squares "0", then the grid for this example looks as follows:

0	1	0	0	1
0	1	0	0	1
1	**0**	**1**	**1**	0
1	**0**	**1**	**1**	0

The most crowded/open squares that an $H = 2$ and $W = 3$ rectangle can contain in this example is 4, and an example of one such rectangle is shown in **bold** text.

Input (seen.in)

The first line of the input contains two space-separated integers, R and C, representing the number of rows and columns into which the garden is divided. The second line of the input contains two space-separated integers, H and W, representing the height and width of the search rectangle. The third and fourth lines respectively contain R and C characters, each of which is a "1" or "0", representing the planting state of that row/column

Sample input

```
4 5
2 3
0011
10110
```

Output (seen.out)

The first line of the output contains one integer, the greatest number of crowded or open squares that can be found in a rectangle H rows high by W columns wide.

Sample output

```
4
```

Constraints

- $1 \leq H \leq R \leq 100000$
- $1 \leq W \leq C \leq 100000$

In 50% of the test cases, $1 \leq R, C \leq 1000$.

Time limit

2 seconds. Python: 20 seconds.

Putting Things On Top Of Other Things

Author

Max Rabkin

Introduction

The Society For Putting Things On Top Of Other Things is having its annual faire, sponsored by International Paper. The main event, the stackathon, involves (what else?) putting things on top of other things; in this case, those things are squares of paper.

The Society has hired a square field, which they have divided into a grid. Some of the grid squares will play host to tents and marquees for the faire's other events, but the remainder will be used for the stackathon, where members of the Society will place as many large squares of paper on the field as they can, subject to these rules:

- No piece of paper can cover a square with a marquee.
- The edges of the paper must lie along the grid lines.
- No two pieces of the same size can be in the same position, but otherwise pieces can overlap.

Task

The sponsors need to know the maximum number of pieces of paper they need to provide.

Example

Consider the following field where "`M`" represents a marquee and "`.`" represents an empty square.

```
  1 2 3 4
1 . . . .
2 . M . .
3 M . . .
4 . . . .
```

Fourteen 1×1 and four 2×2 squares of paper can be placed on the field.

Input (ptotoot.in)

The first line consists of two space-separated integers W (the width and length of the field) and N, the number of marquees.

The next N lines consist of two space-separated integers, x_i and y_i, the coordinates of the ith marquee. Coordinates run from 1 to W along both axes.

Sample input

```
4 2
2 2
1 3
```

Output (ptotoot.out)

The output consists of a single line containing an integer P, the number of squares of paper that can be placed on the field.

Sample output

```
18
```

Constraints

- $1 \leq W \leq 1000$
- $1 \leq N \leq 10000$
- $1 \leq x_i, y_i \leq W$

In 50% of the test cases:

- $1 \leq W \leq 50$
- $1 \leq N \leq 300$

Time limit

2 seconds. Python: 20 seconds.

Cave of Caerbannog

Author

USAICO 2006

Introduction

King Arthur and his knights have learnt that there are many dangers in the Cave of Caerbannog. Some, such as the killer rabbit, can be fought off with holy hand-grenades, while others, such as the legendary Black Beast of Aaauugh, are more difficult to kill.

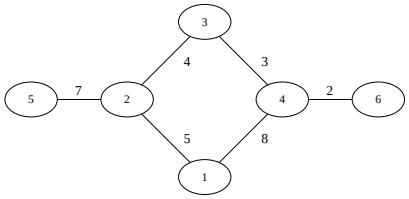
Task

Help King Arthur and his knights navigate safely through the cave by planning routes between various pairs of chambers in the cave. For each pair of chambers, they wish to know the maximum danger they will have to face.

You will be provided with a map of the cave. It consists of a number of tunnels and chambers. Each tunnel connected two different chambers, and for each tunnel you will be given a danger value for the tunnel (in number of soiled suits of armour). The chambers are considered to be safe.

Note that there may be more than one route between two points; if so, the knights will take the route where the maximum danger is as low as possible.

Example



In the figure above, circles indicate chambers and line segments indicate tunnels. The numbers on the line segments indicate the dangers of the tunnels. Going from 1 to 2 involves danger level 5. Going from 1 to 3 also involves danger level 5, by going via 2 (the knights will not go via 4, since that would involve danger level 8, which is higher).

Input (cave.in)

The first line of input contains three space-separated integers, N, T and Q. The cave consists of N chambers (numbered from 1 to N) connected by T tunnels. The next T lines describe the tunnels. Each line contains three integers A, B and D, separated by spaces, indicating that there is a tunnel between A and B that has danger level D.

The next Q lines describe the questions. Each question is a pair of chamber numbers, separated by a space.

There may be more than one tunnel between the same pair of chambers. The cave may consist of several separate components (i.e., there may not be a route from every chamber to every other chamber), but a route will exist between the two chambers in each question.

Sample input

```
6 6 8
1 2 5
2 3 4
3 4 3
1 4 8
2 5 7
4 6 2
1 2
1 3
1 4
2 3
2 4
5 1
6 2
6 1
```

Output (cave.out)

The output consists of Q lines, one for each question. Each line contains an integer, which is the maximum danger that the knights will face if they use the safest possible route between the specified chambers.

Sample output

```
5
5
5
4
4
7
4
5
```

Constraints

- $1 \leq N \leq 15000$

- $1 \leq T \leq 30000$
- $1 \leq Q \leq 20000$
- $1 \leq$ each danger value $\leq 10^9$

In 50% of the test cases:

- $1 \leq N \leq 1000$
- $1 \leq T \leq 1500$
- $1 \leq Q \leq 2000$

Time limit

1 second. Python: 10 seconds.