South African Computer Olympiad
Final Round
Day 2: Future Stars

Overview

Author	**Keegan Carruthers-Smith**	**Graham Poulter**	**Max Rabkin**
Problem	**lights**	**seen**	**ptotoot**
Source	lights.java lights.py lights.c lights.cpp lights.pas	seen.java seen.py seen.c seen.cpp seen.pas	ptotoot.java ptotoot.py ptotoot.c ptotoot.cpp ptotoot.pas
Input file	lights.in	seen.in	ptotoot.in
Output file	lights.out	seen.out	ptotoot.out
Time limit	1 second	2 seconds	2 seconds
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: Future Stars

Lights

Author

Keegan Carruthers-Smith

Introduction

Being a Monty Python fan means one thing, you love traffic lights, but only when they are green. Unfortunately you are a Monty Python fan in Africa, where the traffic lights don't work as they are supposed too.

Task

The traffic lights are in a row. Each light has its own individual toggle. When a light's switch is toggled, that light and every light to the right of it gets toggled. You are given the initial states of the lights (either green or red; there is no amber). Your task is to make all of the traffic lights green with the minimum number of toggles.

Example

Suppose that the traffic lights are like this:

Red Red Green Green Red Green

The minimum number of toggles required to make all of the lights green is 4. (Toggle lights 1, 3, 5 and 6 where 1 is the left-most light).

Input (lights.in)

The first line contains the number of traffic lights, N. The next N lines each contain an integer, which is either 0 or 1, with 0 indicating a green light and 1 a red light. The lights are listed from left to right.

Sample input

```
6
1
1
0
0
1
0
```

Output (lights.out)

The output file contains N lines, with line each indicating how many times the corresponding light was toggled. The order of the lights is the same as in the input file. You must output the unique solution for which the number of toggles is minimised.

Sample output

```
1
0
1
0
1
1
```

Constraints

- $1 \leq N \leq 100000$

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

Time limit

1 second. Python: 10 seconds.

South African Computer Olympiad
Final Round
Day 2: Future Stars

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.