BAPC

Benelux Algorithm Programming Contest
Sponsored by Sogeti Delft 2005 ©

Problems

October 22, 2005

A ASM – The Abelian Sandpile Model

Problem

Modeling sandpiles is an interesting problem in statistical physics. When dropping a sand grain on an existing pile, most of the time the grain will stick to it or a couple of grains will slide down. Occasionally, however, adding one extra grain to a pile will lead to a huge avalanche of sand grains falling down.

A simple way to model sandpiles is the *Abelian Sandpile Model.* In this simple model the sandpile is described as a two-dimensional lattice with on each lattice site a height (the number of sand grains on that lattice site). When an additional grain is dropped on a lattice site, its height is increased by one. If the height becomes larger than a certain critical height, sand grains begin to topple. This is modeled by reducing the number of sand grains on the site that is too high by four, and increasing the heights of its four neighbors by one. If some of the neighbors exceed the critical height after this toppling, sand grains topple from those points too until the situation is stable again. If a sand grain falls off the lattice, the grain is gone.

Given an initial sandpile and the positions where the grains are dropped, determine the final sandpile.

Input

The first line of the input file contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with four integers, y, x, n and h with $1 \leq y, x \leq 100$, $0 \leq n \leq 100$ and $3 \leq h \leq 9$: the dimensions of the sandpile, the number of dropped sand grains and the critical height.
- y lines, each with x characters in the range '`0`'. . . '`9`': the heights of the initial sandpile. Each height is less than or equal to the critical height.
- n lines with two integers y_i and x_i with $1 \leq y_i \leq y$ and $1 \leq x_i \leq x$: the positions where the grains are dropped.

Output

For every test case in the input file, the output should contain y lines, each with x characters in the range '`0`'. . . '`9`': the heights of the final sandpile.

Example

Input

```
3
3 4 1 5
1023
2344
0000
2 3
3 4 1 4
1023
2344
0000
2 3
7 7 1 9
9999999
9999999
9999999
9999999
9999999
9999999
9999999
4 4
```

Output

```
1023
2354
0000
1034
2421
0011
7999997
9799979
9979799
9996999
9979799
9799979
7999997
```

B Cell Phones

Problem

Lone City is not a town, nor a village. It is not even a hamlet. It is just a few hundred houses scattered over the plain, like someone had fired a buckshot of buildings, with the highway cutting it in two parts. This lack of center has to do with the peoples attitude: they do not like living too close to each other. New houses are erected at decent distance from existing buildings.

Things changed when a new generation of Loners grew up. They wanted to enjoy the blessings of modern society. They wanted fast food, disco's and cell phones. And they found out that cell phones are useless outside the range of a transmitting antenna.

It took some time to convince the telephone provider that Lone City desperately needed a transmitting antenna. Finally the company agreed to erect an antenna, to be located somewhere along the highway, for easy maintenance. This type of antenna has a range of 1000 meters, so most likely not all Loners will be in range. Antoine Master, the head of the Antennae Department, tried to find a place for the transmitting antenna such that it would be profitable for as many people as possible. To that purpose he created a list of all houses, containing for each house its location and the number of inhabitants. (Initially only people over 18 were counted, until someone pointed out that the most dedicated cell phone users are under 18.)

And then Antoine started thinking...

Can you help Antoine to find the best place for the transmitting antenna?

Input

The first line of the input file contains a single number: the number of test cases to follow. Each test case has the following format:

- A line with a positive number h with $0 < h \leq 1000$: the number of houses in Lone City.
- h lines, each line containing three positive numbers, x, y, n, with $-10^4 \leq x, y \leq 10^4$ and $0 \leq n \leq 100$, representing the location and number of inhabitants of each house.

The highway is the line $y = 0$.

Output

For every test case in the input file, the output should contain a single number, on a single line: the number of people living in a house within reach of the antenna (that is: at a distance of at most 1000 meters), provided that the antenna is placed along the highway at the best place possible.

Example

Input

```
2
2
100 100 20
500 500 30
3
1000 -500 4
4000 500 2
5000 -400 3
```

Output

```
50
5
```

C Evacuation

Problem

The police of Delft has received a message that a bomb has been placed in the city's highest building. A crisis team is formed, and they decide to evacuate the building as fast as possible. Luckily it is past five, and most people have already left the building. Using the building's security cameras, the crisis team could easily make a list of the people left in the building, and the floors where they stay. The crisis team decides to use a single elevator for evacuation, one that happens to be at the top floor of the building.

To minimize the risk of the bomb being activated by the vibrations of the elevator, the elevator will only move down, and move down only once. An evacuation plan is made, and using the building's intercom, the people in the building are asked to use the stairs (upwards or downwards) to the floor where they will be picked up by the elevator. The elevator happens to be located next to the stairs. Some people on the lower floors may have to leave the building using only the stairs.

The elevator needs a constant time to move down a floor, and to close its doors (time needed to open the doors is ignored). People take a constant time to walk down or up a staircase as well. At the beginning, the doors of the elevator are closed.

Your task is to write a program to make the fastest evacuation plan possible, and calculate how fast everybody in the building can reach the ground floor.

Input

The first line of the input file contains a single number: the number of test cases to follow. Each test case has the following format:

- A line with three positive integers, m ($m \leq 100$): the time it takes the elevator to move down one floor, s ($s \leq 100$): the time needed for the elevator to close its doors after the last person on a floor has entered the elevator, and w ($w \leq 100$): the time needed for a person to walk to the next floor either up or down, using the stairway.
- A line with two integers n_f and n_w. The number n_f ($0 < n_f \leq 1000$) is the total number of floors in the building (excluding ground level), and n_w ($0 \leq n_w \leq n_f + 1$) is the number of floors on which people are waiting to be evacuated (these buildings really have huge elevators!).
- n_w lines, each with one integer, f_i, the floor on which persons are present ($0 \leq f_i \leq n_f$). Per test case, all f_i are unique. According to European convention the ground floor has number 0.

You may assume that all people involved will fit into the elevator.

Output

For every test case in the input file, the output should contain a single number, on a single line: the time needed to get everybody in the building on the ground floor.

Example

Input

```
3
1 1 4
5 3
5
1
0
1 1 4
5 6
0
1
2
3
4
5
10 10 20
1000 0
```

Output

```
6
8
0
```

D Giant Cover

Problem

A student at the Lutjebroek University of Technology wants to cover all buildings of the University with an enormous translucent plastic cover. This will make the use of umbrellas in this region unnecessary, significantly cutting costs.

The costs of the cover are proportional to its area. With the purpose of the cover in mind, the student wants to reduce the costs of the cover as much as possible. You are to write a program that will help him with this by calculating the minimal area of a cover.

The whole campus terrain of the University is flat and has a rectangular shape. All buildings on it have the shape of the union of a set of boxes, each of which stands on the ground. The cover must cover all buildings and will be attached to the four sides of the campus at ground level.

Input

The first line of the input file contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with the four integers x_1, y_1, x_2, y_2, separated by spaces, describing the campus terrain $[x_1, x_2] \times [y_1, y_2]$. The numbers satisfy $-10^4 \leq x_1 < x_2 \leq 10^4$ and $-10^4 \leq y_1 < y_2 \leq 10^4$.
- One line with the integer n, $0 \leq n \leq 400$, the number of boxes that form the buildings on the campus.
- n lines, with on the i^{th} line the five integers a_i, b_i, c_i, d_i, h_i, separated by spaces, describing a box with footprint $[a_i, c_i] \times [b_i, d_i]$ and height h_i above the ground. The numbers satisfy $x_1 \leq a_i < c_i \leq x_2$, $y_1 \leq b_i < d_i \leq y_2$ and $0 < h_i \leq 10^4$.

Note: $[a, c] \times [b, d]$ is a so called Cartesian product and denotes the rectangular area $\{(x, y) \in \mathbb{R}^2 : a \leq x \leq c, b \leq y \leq d\}$.

Output

For every test case in the input file, the output should contain a single number, on a single line: the area of the smallest cover, using a precision of four decimals behind the decimal point. The rounding should occur as usual; a digit is rounded up if the next digit is ≥ 5, otherwise it is rounded down.

Example

The last example testcase corresponds to figure 1.

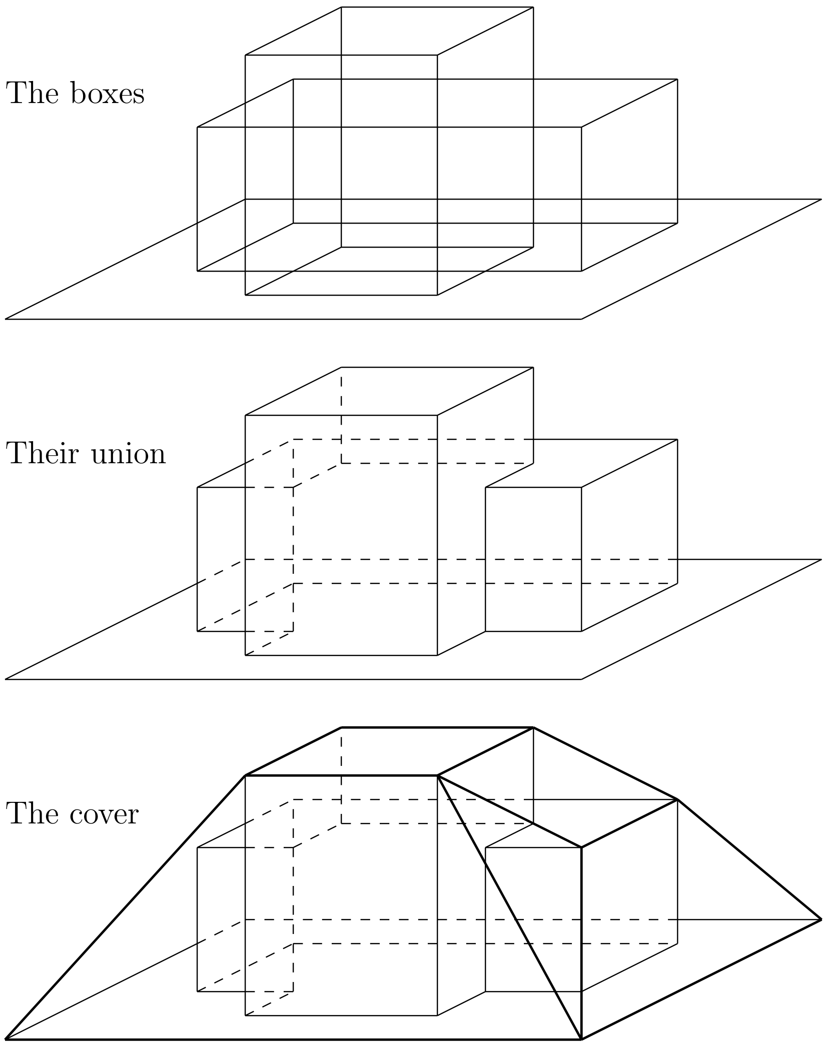


Figure 1: The last example testcase illustrated.

Input

```
3
0 0 12 10
0
0 0 12 10
1
2 2 8 8 3
0 0 12 10
2
2 4 10 8 3
4 2 8 6 5
```

Output

```
120.0000
169.7443
203.7598
```

E North-Western Winds

Problem

A strong North-Western wind is blowing. When sailing this means that you can sail to the East, to the South or to any direction between East and South. It is impossible to go either North- or Westwards.

In the ocean there are a large number of small islands. These islands are described by coordinate pairs (x, y) on a grid. The positive y-direction is Northwards and the positive x-direction is Eastwards. We’d like to sail from one island to another. For how many pairs of islands is this possible? (Note: a pair consists of two different islands.)

Input

The first line of the input file contains a single number: the number of test cases to follow. Each test case has the following format:

- One line with one number n with $1 \leq n \leq 75000$: the number of islands.
- n lines with two numbers x_i and y_i with $-10^9 \leq x_i, y_i \leq 10^9$: the coordinates of the islands. No two islands are located at the same coordinates.

Output

For every test case in the input file, the output should contain a single number, on a single line: the number of pairs of islands for which you can sail from one to the other.

Example

Input

```
2
4
-10 -10
-10 10
10 -10
10 10
3
1 3
2 2
3 1
```

Output

```
5
3
```

F Shepherds and Engineers

Problem

Westmoreland is a peaceful country, where quiet rivers slowly flow between rough moors, and where peaceful shepherds herd their flock. In the Northern part of the country the town is located, with its sheep market and its School of Engineering. Every now and then a shepherd will select part of his flock, and lead them to the market, to be sold. In former days the shepherds used to wade with their flock through the rivers, at a ford. After heavy rainfall, however, the rivers turned into violent streams, and crossing became impossible. Shepherds nevertheless trying to cross would lose some, or even all of their sheep.

A graduate from the School of Engineering recognized a societal problem to be solved. He asked the king some money, to build bridges. The king, who believed in private enterprise, did not make any money available, but allowed the engineer to build a bridge, and to impose a toll on every shepherd crossing that bridge. Moreover the king promulgated a law forbidding shepherds to wade through the river, when a bridge was available.

That first bridge became a great success – to the engineer, that is. Soon more bridges were built, and within a few years the country was full of bridges. The shepherds were less happy. Some engineers imposed quite excessive tolls, up to 100% of the sheep, which the shepherds considered as theft rather than toll.

Things changed when the king died, and his son (who happened to be married to a shepherd's daughter) accessed the throne. He promulgated a law against excessive toll, stating that:

> Whenever a shepherd with some sheep will cross a bridge and the engineer who built the bridge imposes a toll,
>
> 1. What the shepherd keeps shall be more than what the engineer takes,
> 2. What the shepherd keeps shall be an integral multiple of what the engineer takes.

The shepherds were quite happy with this law, and started inventing clever schemes to avoid toll.

> A shepherd wanting to sell 40 sheep in the town, and living four bridges away from the town, could enter the first bridge with 47 sheep. The maximum toll allowed would be 1 sheep, so he had 46 sheep left. At the next bridge he would pay no more than two sheep, and have 44 sheep left. Instead of paying a toll of 11 sheep at the next bridge, he would donate one sheep to a local shepherd, pay one sheep to cross the third bridge, and have 42 sheep left. Again donating one sheep to a local shepherd, he would pay 1 sheep at the last bridge, and enter the town with 40 sheep.

At their next annual meeting the shepherds raised money, to found a chair in Mathematical Engineering at the School of Engineering (to study the distribution of prime numbers). And they hired a Computer Engineer, to write a program to solve the following problem:

> Given the number s of sheep a shepherd wants to sell at the market, and the number b of bridges that shepherd has to cross to reach the market, what will be the minimum number of sheep he has to start his travel with?

The example given above shows that starting with 47 sheep, and crossing 4 bridges, one may enter the town with 40 sheep. It is not immediately evident, however, that the same result could not be obtained starting with less sheep.

Observe that occasionally the best solution may result in entering the town with more sheep than required.

Input

The first line of the input file contains a single number: the number of test cases to follow. Each test case has the following format:

- A line with two positive numbers: the number s ($0 < s \leq 10^6$) of sheep that should enter the town, and the number b ($0 \leq b \leq 1000$) of bridges to be crossed.

Output

For every test case in the input file, the output should contain a single number, on a single line: the number of sheep to start with.

Example

Input

```
3
40 4
13 1
10 10
```

Output

```
47
17
34
```

G Ski Lifts

Problem

Climate changes made the rolling mountains of Alaska a perfect place for all-season skiing. To transform this region into a successful area for skiing, however, ski lifts are needed. People skiing do not like walking upwards. They want to ski downhill; when needed they are willing to ski on the same level for some time.

Optimal use of the area requires that, starting from an arbitrary point, a skier should be able to reach any other point just by skiing downhill or staying at the same level, and occasionally taking a ski lift.

A sufficient amount of ski lifts must be planned and constructed such as to fulfill this condition. On the other hand, building more ski lifts than necessary is a waste of money.

What is the minimal number of ski lifts needed?

As ski lifts are built on high poles, we assume that a ski lift can be constructed from any place to any place, regardless of the terrain in between. A ski lift is unidirectional.

It is important to know that in Alaska one is not allowed to ski in any other direction than North, South, East or West.

Input

The first line of the input file contains a single number: the number of test cases to follow. Each test case has the following format:

- A line with two positive numbers w and l with $1 \leq w, l \leq 500$: the width and length of the area.
- w lines, each line containing l positive numbers h_{ij}, with $0 \leq h_{ij} \leq 10^9$, representing the height in each point of the area.

Output

For every test case in the input file, the output should contain a single number, on a single line: the minimum number of ski lifts to be built.

Example

Input

```
2
5 5
1 1 0 0 0
1 1 1 1 0
0 1 9 1 0
0 1 1 1 1
0 0 0 1 1
1 10
10 20 30 40 50 60 70 80 90 100
```

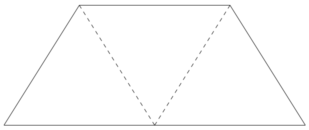
Output

```
2
1
```

H Tables

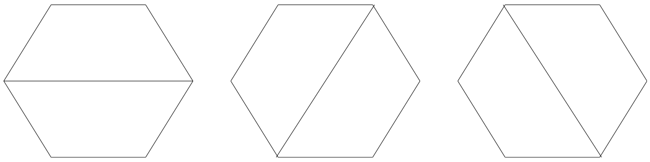
Problem

The cafeteria of the EWI Faculty of Delft University of Technology has a lot of tables. These tables are not square or rectangular, they have the form of a trapezoid, an isosceles trapezoid, with sides 1,1,1,2, as in the drawing below. It can be seen as half a hexagon or as composed of three equilateral triangles. Such a table offers place for five to sit, though at the acute angles there is few space for plates.



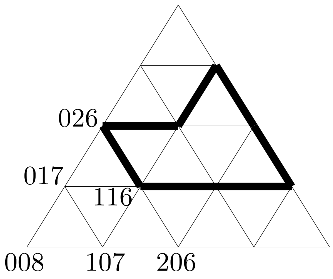
The shape of a table.

For larger companies, the tables can be put together to form a larger table. The more tables are used, the more shapes can be created. The other way round, one may ask whether a given shape can be built with these tables and whether that can be done in more than one way. A hexagon can be built from two tables in three different ways.



Three different ways to form a hexagon.

It could be argued that these three solutions may be reduced to a single solution, using rotation. For this problem we consider these three solutions to be different, however.



The triangle encoding.

The shapes to be formed are described using a pattern of triangles. The nodes in this pattern have a code formed from 3 digits, in the range $0 \ldots 8$. The sum of these three digits is 8. We describe a shape by enumerating the nodes of its contour: $[116, 314, 134, 125, 026]$. The contour is built connecting the nodes along the gridlines: connect the first node with the second node, the second node with the third one, etc., and finally connect the last node with the first. This implies that neighboring nodes are always on a common gridline.

The following rules apply:

1. When walking along a contour, the enclosed area is on the left-hand side;
2. The enclosing contour does not cross itself, nor are nodes used multiple times;

3. The enclosed shape has no holes (in fact this is a consequence of 1 and 2).

A shape is given by the nodes of the enclosing contour. Calculate the number of ways this shape can be made using the trapezoid tables.

Input

The first line of the input file contains a single number: the number of test cases to follow. Each test case has the following format:

- The first line contains a single number, n, the number of nodes of the contour.
- The second line contains n codes for the nodes of the contour, as described above. These codes are separated by a single space.

Output

For every test case in the input file, the output should contain a single number, on a single line: the number of different ways of creating the given shape with the trapezoid tables.

Example

Input

```
2
6
107 206 215 125 026 017
5
116 314 134 125 026
```

Output

```
3
0
```