− Submissions

Dynamic Grid

6pt	Not attempted **1392/1881 users** correct (74%)
8pt	Not attempted **1288/1368 users** correct (94%)

gBalloon

9pt	Not attempted **353/666 users** correct (53%)
17pt	Not attempted **266/338 users** correct (79%)

IP Address Summarization

10pt	Not attempted **123/236 users** correct (52%)
19pt	Not attempted **73/118 users** correct (62%)

Virtual Rabbit

11pt	Not attempted **18/166 users** correct (11%)
20pt	Not attempted **3/8 users** correct (38%)

− Top Scores

nhho	100
sundar95	80
Shaon	80
ajkrish95	80
ojas.deshpande	80
NAFIS	69
JunoYu	69
wcwswswws	69
karanaggarwal	69
VotBear	69

Problem A. Dynamic Grid

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 6 points	Solve A-small
Large input 8 points	Solve A-large

Problem

We have a grid with **R** rows and **C** columns in which every entry is either 0 or 1. We are going to perform **N** operations on the grid, each of which is one of the following:

- Operation M: Change a number in one cell of the grid to 0 or 1
- Operation Q: Determine the number of different *connected regions* of 1s. A connected region of 1s is a subset of cells that are all 1, in which any cell in the region can be reached from any other cell in the region by traveling between cells along edges (**not** corners).

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case starts with one line with two integers, **R** and **C**, which represent the number of rows and columns in the grid. Then, there are **R** lines of **C** characters each, in which every character is either `0` or `1`. These lines represent the initial state of the grid.

The next line has one integer, **N**, the number of operations to perform on the grid. **N** more lines follow; each has one operation. All operation Ms will be of the form `M x y z`, meaning that the cell at row x and column y should be changed to the value z. All operation Qs will be of the form `Q`.

Output

For each test case, output one line containing "Case #x:", where x is the test case number (starting from 1). Then, for every operation Q in the test case, in order, output one line containing the number of connected regions of 1s.

Limits

1 ≤ **T** ≤ 10.
1 ≤ **R, C** ≤ 100.
0 ≤ **x** < **R**.
0 ≤ **y** < **C**.
0 ≤ **z** ≤ 1.

Small dataset

1 ≤ **N** ≤ 10.

Large dataset

1 ≤ **N** ≤ 1000.

Sample

```
Input          Output

1              Case #1:
4 4            4
0101           2
0010           2
0100           2
1111
7
Q
M 0 2 1
Q
M 2 2 0
Q
M 2 1 0
Q
```

All problem statements, input data and contest analyses are licensed under the Creative Commons Attribution License.

© 2008-2017 Google   Google Home - Terms and Conditions - Privacy Policies and Principles

Powered by

Google Cloud Platform

Round D APAC Test 2016

A. Dynamic Grid
B. gBalloon
C. IP Address Summarization
D. Virtual Rabbit

Questions asked 1

Submissions

Dynamic Grid

6pt Not attempted **1392/1881 users** correct (74%)

8pt Not attempted **1288/1368 users** correct (94%)

gBalloon

9pt Not attempted **353/666 users** correct (53%)

17pt Not attempted **266/338 users** correct (79%)

IP Address Summarization

10pt Not attempted **123/236 users** correct (52%)

19pt Not attempted **73/118 users** correct (62%)

Virtual Rabbit

11pt Not attempted **18/166 users** correct (11%)

20pt Not attempted **3/8 users** correct (38%)

Top Scores

nhho	100
sundar95	80
Shaon	80
ajkrish95	80
ojas.deshpande	80
NAFIS	69
JunoYu	69
wcwswswws	69
karanaggarwal	69
VotBear	69

Problem B. gBalloon

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 9 points	Solve B-small
Large input 17 points	Solve B-large

Problem

The G tech company has deployed many balloons. Sometimes, they need to be collected for maintenance at the company's tower, which is located at horizontal position 0. Each balloon is currently at horizontal position $\mathbf{P_i}$ and height $\mathbf{H_i}$.

G engineers can move a balloon up and down by sending radio signals to tell it to drop ballast or let out air. But they can't move the balloon horizontally; they have to rely on existing winds to do that.

There are **M** different heights where the balloons could be. The winds at different heights may blow in different directions and at different velocities. Specifically, at height j, the wind has velocity $\mathbf{V_j}$, with positive velocities meaning that the wind blows left to right, and negative velocities meaning that the wind blows right to left. A balloon at position P at a height with wind velocity V will be at position P+V after one time unit, P+2V after two time units, etc. If a balloon touches the tower, it is immediately collected.

It costs $| H_{original} - H_{new} |$ points of energy to move one balloon between two different heights. (This transfer takes no time.) You have **Q** points of energy to spend, although you do not need to spend all of it. What is the least amount of time it will take to collect all the balloons, if you spend energy optimally?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follows. The first line of each case has three integers **N**, **M**, and **Q**, representing the number of balloons, the number of height levels, and the amount of energy available.
The second line has **M** integers; the jth value on this line (counting starting from 0) is the wind velocity at height j.
Then, **N** more lines follow. The ith of these lines consists of two integers, $\mathbf{P_i}$ and $\mathbf{H_i}$, representing the position and height of the ith balloon.

Output

For each test case, output one line containing "Case #x: y", where x is the test case number (starting from 1) and y is the minimum number of time units needed to collect all of the balloons, returns IMPOSSIBLE if it's impossible to collect all the balloons using the energy given.

Limits

Small dataset

$1 \le \mathbf{T} \le 100$.
$1 \le \mathbf{N} \le 10$.
$1 \le \mathbf{M} \le 10$.
$-10 \le \mathbf{V_j} \le 10$.
$1 \le \mathbf{Q} \le 10$.
$0 \le \mathbf{H_i} < \mathbf{M}$.
$-10 \le \mathbf{P_i} \le 10$.

Large dataset

$1 \le \mathbf{T} \le 25$.
$1 \le \mathbf{N} \le 100$.
$1 \le \mathbf{M} \le 1000$.
$-100 \le \mathbf{V_j} \le 100$.
$1 \le \mathbf{Q} \le 10000$
$0 \le \mathbf{H_i} < \mathbf{M}$.
$-10000 \le \mathbf{P_i} \le 10000$.

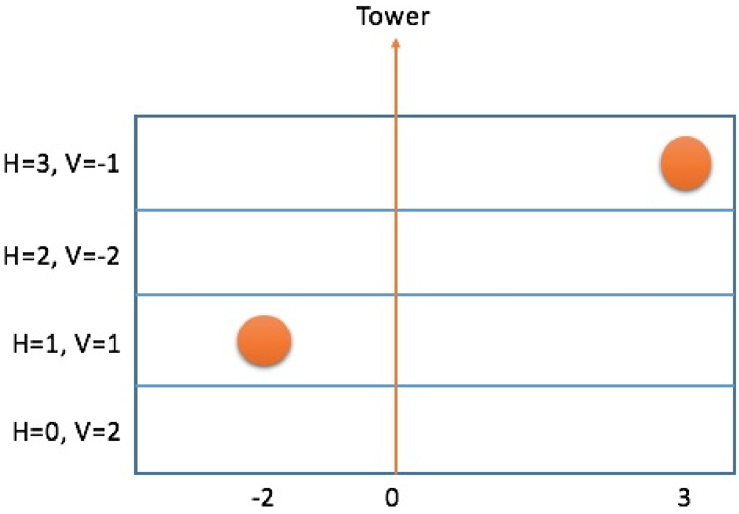
Sample

```
Input          Output

2              Case #1: 2
```

```
2 4 1        Case #2: IMPOSSIBLE
2 1 -2 -1
3 3
-2 1
1 3 1
1 -1 -2
-2 2
```

Here is an example:



In the sample case, there are two balloons in the sky, and you have 1 energy point to use. The best solution is to immediately spend 1 energy point to move the balloon at position 3, height 3 down to height 2. Once you've done that, it will take 2 time units for both balloons to reach the tower.

All problem statements, input data and contest analyses are licensed under the Creative Commons Attribution License.

© 2008-2017 Google   Google Home - Terms and Conditions - Privacy Policies and Principles

Powered by

Google Cloud Platform

apac test
powered by Google Code Jam

Practice Mode

Contest scoreboard | Sign in

Round D APAC Test 2016

A. Dynamic Grid

B. gBalloon

C. IP Address Summarization

D. Virtual Rabbit

Questions asked 1

Submissions

Dynamic Grid

6pt	Not attempted **1392/1881 users** correct (74%)
8pt	Not attempted **1288/1368 users** correct (94%)

gBalloon

9pt	Not attempted **353/666 users** correct (53%)
17pt	Not attempted **266/338 users** correct (79%)

IP Address Summarization

10pt	Not attempted **123/236 users** correct (52%)
19pt	Not attempted **73/118 users** correct (62%)

Virtual Rabbit

11pt	Not attempted **18/166 users** correct (11%)
20pt	Not attempted **3/8 users** correct (38%)

Top Scores

nhho	100
sundar95	80
Shaon	80
ajkrish95	80
ojas.deshpande	80
NAFIS	69
JunoYu	69
wcwswswws	69
karanaggarwal	69
VotBear	69

Problem C. IP Address Summarization

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 10 points	Solve C-small
Large input 19 points	Solve C-large

Problem

An IP (Internet Protocol) address is a number that is assigned to each device on the Internet. At the time being, most devices use version four of this protocol (IPv4). An IPv4 address is a 32-bit string. IPv4 addresses are normally represented in dot-decimal notation, which consists of four decimal numbers called octets, each ranging from 0 to 255 (inclusive), separated by dots, e.g., 172.16.254.1. Each octet represents a group of 8 bits (one byte) of the address. The first 8 bits of the string (when interpreted as an unsigned integer, with the most significant bit first) form the first octet, the next 8 bits form the second octet, and so on.

An IP subnet addresses is used to represent a group of devices that belong to the same network. IP subnet addresses are expressed in the format of an IP address, followed by a slash and then a prefix length ranging from 0 to 32. A subnet address stands for all IP addresses that have the same first P bits of the given address, where P is the prefix length. For example 10.8.0.0/9 represents 2^{23} addresses that all have `000010100` (the first nine bits of 10.8.0.0) as their first 9 bits, that is, 10.0.0.0 through 10.127.255.255. Note that 10.8.0.0/9 and, for example, 10.0.0.0/9 (or any other address within the subnet) would be equivalent ways to refer to the same subnet, because those addresses start with the same nine bits.

A subnet is *normalized* if the bits of the address other than the prefix are all zeroes. For example, 10.8.1.0/24 and 10.8.1.2/24 represent the same subnet, but 10.8.1.0/24 is normalized. The normalization of 255.255.255.255/13 is 255.248.0.0/13.

You will be given a list of subnet addresses, and you must output the shortest ordered list of subnets such that all the addresses are normalized and an address belongs to some subnet in the input if and only if it belongs to some subnet in the output.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each begins with one line with an integer **N**, the number of subnets, and is followed by **N** more lines, each of which has a subnet addresses. Each subnet address is of the form `A.B.C.D/P`, where A, B, C, and D are integers from 0 to 255, inclusive, and P is an integer from 0 to 32, inclusive. No integer (apart from 0) has leading zeroes.

Output

For each test case, output one line containing "Case #x:", where x is the test case number (starting from 1). Then output a list of subnet addresses, one per line, meeting the conditions described above. These addresses must be normalized and must be ordered. An address X comes before another address Y if X's first integer is smaller than Y's first integer, or if X and Y have the same first integer but X's second integer is smaller than Y's second integer, and so on.

Note that the requirements of the problem guarantee that there is a single unique answer for each test case.

Limits

1 ≤ **T** ≤ 50.

Small dataset

1 ≤ **N** ≤ 10.

Large dataset

1 ≤ **N** ≤ 10000.

Sample

```
Input                 Output

3                     Case #1:
```

```
2                    10.0.0.0/8
10.1.2.3/8           Case #2:
10.2.3.4/17          10.0.0.0/8
2                    Case #3:
10.2.3.4/9           52.22.138.0/23
10.128.2.3/9         52.56.134.128/26
10                   52.227.80.0/22
224.147.224.186/18   58.32.0.0/12
58.45.85.53/14       83.248.0.0/13
52.56.134.139/26     224.147.192.0/18
52.227.82.227/22     238.223.58.128/27
83.250.251.44/13
83.250.12.64/16
58.40.52.11/14
52.22.138.56/23
238.223.58.151/27
58.32.52.11/13
```

All problem statements, input data and contest analyses are licensed under the Creative Commons Attribution License.

© 2008-2017 Google   Google Home - Terms and Conditions - Privacy Policies and Principles

Powered by

Google Cloud Platform

Round D APAC Test 2016

A. Dynamic Grid

B. gBalloon

C. IP Address Summarization

D. Virtual Rabbit

Questions asked 1

− Submissions

Dynamic Grid

6pt	Not attempted **1392/1881 users** correct (74%)
8pt	Not attempted **1288/1368 users** correct (94%)

gBalloon

9pt	Not attempted **353/666 users** correct (53%)
17pt	Not attempted **266/338 users** correct (79%)

IP Address Summarization

10pt	Not attempted **123/236 users** correct (52%)
19pt	Not attempted **73/118 users** correct (62%)

Virtual Rabbit

11pt	Not attempted **18/166 users** correct (11%)
20pt	Not attempted **3/8 users** correct (38%)

− Top Scores

nhho	100
sundar95	80
Shaon	80
ajkrish95	80
ojas.deshpande	80
NAFIS	69
JunoYu	69
wcwswswws	69
karanaggarwal	69
VotBear	69

Problem D. Virtual Rabbit

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 11 points	Solve D-small
Large input 20 points	Solve D-large

Problem

Alice just bought a virtual pet rabbit. The rabbit hops around on a screen and can be "fed" by pressing a button. Alice is fond of the rabbit, but she is also busy, and doesn't want to spend too much time taking care of it. However, if the rabbit goes without "food" for too long, it "dies" and Alice loses the game.

Every day, Alice gets up at time **G**, goes to work at time **W**, returns home at time **H**, and goes to bed at time **B**. Alice cannot feed the rabbit while she is at work or asleep -- that is, in the intervals [**W**, **H**) and [**B**, **G**). Note that times **W** and **B** themselves are not valid feeding times, whereas times **H** and **G** are. In any other second, Alice can either push a button to instantly feed the rabbit, or not push the button. Between every two seconds, the rabbit determines the number of consecutive seconds in which it has not been fed, and "dies" if that duration is equal to **X**.

It is currently 00:00:00 on Day 0, and the rabbit has just been delivered to Alice's house by the mail service. (The mail carrier pushes the button at 00:00:00, even if Alice is asleep, and then leaves.) Alice wants to make sure the rabbit is still "alive" at 00:00:00 on day **D**. What is the minimal number of times that she needs to feed the rabbit, if she can keep the rabbit "alive" at all?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow; each consists of 6 lines. The first 5 lines represent the times **G**, **W**, **H**, **B**, and **X** in "hh:mm:ss" format. The last line consists of one integer **D**.

Output

For each test case, output one line containing "Case #x: y", where x is the test case number (starting from 1) and y is the minimal number of times Alice needs to feed the rabbit. If it's impossible for the rabbit to be alive at 00:00:00 on day **D**, output -1.

Limits

1 ≤ **T** ≤ 100.
It's guaranteed that Alice always goes to bed before midnight and gets up at or after midnight. **G**, **W**, **H**, and **B** are in strictly increasing order within the same day.
00:00:00 ≤ **G** < **W** < **H** < **B** ≤ 23:59:59.
00:00:00 < **X** ≤ 23:59:59.

Small dataset

1 ≤ **D** ≤ 1000.

Large dataset

1 ≤ **D** ≤ 10^{14}.

Sample

```
Input        Output

3            Case #1: 200
08:00:00     Case #2: -1
09:00:00     Case #3: 86401
18:00:00
22:00:00
12:00:00
100
08:00:00
09:00:00
18:00:00
22:00:00
01:00:00
1

00:00:00
12:00:00
```

```
12:00:01
23:59:59
00:00:02
2
```

In sample case #1, Alice could feed the rabbit at 08:00:00 and 20:00:00 every day.

In sample case #2, the poor rabbit will be "dead" before Alice even wakes up on Day 0.

All problem statements, input data and contest analyses are licensed under the Creative Commons Attribution License.

© 2008-2017 Google   Google Home - Terms and Conditions - Privacy Policies and Principles

Powered by

Google Cloud Platform