IDI Open Programming Contest May 3rd, 2014

Problem Set

A Abandon Ship
B Battle Sheep
C Counting Digits (Easy)
D Dad's Card Trick
E Election
F Fired
G GG NO RE OMG CHEATZ
H Herding Sheep
I In The Shower (Easy)
J Join My Team
K Krusty's Burger
L London Underground

Jury and Problem Writers

Torbjørn Morland (Head Judge)
Alexander Georgiev
Magne Vikjord
Christian Neverdal Jonassen
Edvard Kristoffer Karlsen
Geir-Arne Fuglstad

Tips

- Tear the problem set apart and share the problems among you.
- Problems are not ordered by difficulty.
- Try solving the easy problems first. Two problems in this set are tagged with "(Easy)" to help point you in the right direction.
- If your solution fails on a problem, you can print your program and debug it on paper while you let someone else work on a different problem on the computer.
- If you need help, contact the judges.

Rules

- Each team consists of one to three contestants.
- One computer is used per team.
- You may not cooperate with persons not on your team.
- You may print your programs on paper to debug them.
- What you may bring to the contest:
  - Any written material (Books, manuals, handwritten notes, printed notes, etc).
  - Pens, pencils, blank paper, stapler and other useful non-electronic office equipment.
  - NO material in electronic form (CDs, USB pen and so on).
  - NO electronic devices (PDAs and so on).
- The only electronic content you may consult during the content is that specified by the organiser (see the web-page). You may not copy source code from web pages, etc.
- Your programs should read from standard in and write to standard out. Writing to standard error will result in a failed submission. C programs should return 0 from `main()`.
- Your program may use at most 100MB of memory.
- Your programs may not:
  - access the network,
  - read or write files on the system,
  - talk to other processes,
  - fork,
  - or similar stuff.
  - If you try, your program will hang or crash. If it hangs, it will take a couple of minutes before others will be able to run their programs. So please make an effort to not crack/break what we have spent our spare time preparing for you.
- Show common sense and good sportsmanship.

Problem A

Abandon Ship

As a Starph1337 Captain, you know that the list of tasks on a starship can be long. Even on the flagship of the International Planet of Federations, a lot of things need to be taken care of during battle: power needs to be diverted, matrices need to be re-inverted, and some day you might have to get to those well-decorated emergency escape pods. A brand new software program with the somewhat cumbersome and long yet completely informative name of CaptainsLogStardate41254.7TheseAreTheVoyagesOfTheStarship (Enterprise Edition) is under development. The first version should support giving orders during battle. There are three types of systems that should be supported:

- Category I: Systems will simply go out of whack once in a while. When they do, recalibrate them!
- Category II: Systems will suddenly (and for no apparent reason) no longer be usable in their current state. Invert them, or re-invert them, as needed. While you take care of these systems, the engineers are grepping the source code for `do_damage_to_self`.
- Category III: Life-critical. Divert power to them if you can, or abandon ship! **You have 20 units of extra power**.

Input specifications

The first line of input will be T, the number of cases. T cases follow.
The first line of each case will be four numbers A, B, C, D, separated by spaces. The first three numbers will be the number of systems in Category I, II and III, respectively. The fourth number is the number of damaged systems.
After this line follow A lines with the names of Category I systems, B lines with the names of Category II systems, C lines with Category III systems, and then D lines with names of damaged systems (in the order they were damaged).

Output specifications

For each damage, output the correct order that should be given to your crew. Category I systems will always need to be recalibrated when they are damaged, hence, output `recalibrate [system name]`. Category II systems will need to be inverted the first time they are damaged, then re-inverted the next time, then inverted again, and so on. Output `invert [system name]` or `re-invert [system name]` as needed. Category III systems start with 100 units of power and will lose 10 units for each damage taken.

The ship has 20 units of extra power in total that can be diverted to these damaged systems. Output `divert all power to [system name]`. As long as the system is still alive, give the order to divert extra power (even if the 20 extra units has been diverted). Whenever you divert power, your crew will use as much as they need to get it back to 100 units. When a life-critical (Category III) system's power is 10 units or less, give the order to abandon ship by outputting `ABANDON SHIP. REPEAT. ALL HANDS ABANDON SHIP.` to your crew and terminate the case (do not give further orders after this). All orders need to be in lower case except for the abandon ship order, which needs to be in upper case and with the punctuation exactly as written.

Notes and Constraints

- $0 \leq T \leq 100$
- $0 \leq A, B, C, D \leq 100$
- Each system name will start and end with a lowercase letter `a` - `z`.
- Each system name will consists of between 2 and 22 characters, and only lowercase letters `a`-`z` and spaces.

Given a SYSTEM, the orders will be one of the following:

- recalibrate SYSTEM
- invert SYSTEM
- re-invert SYSTEM
- divert all power to SYSTEM
- ABANDON SHIP. REPEAT. ALL HANDS ABANDON SHIP.

Sample input

```
2
1 1 2 5
plasma shifter
auxiliary matrix
life support
shields
plasma shifter
auxiliary matrix
auxiliary matrix
life support
shields
0 0 1 11
shields
shields
shields
shields
shields
shields
shields
shields
shields
shields
shields
shields
```

Output for sample input

```
recalibrate plasma shifter
invert auxiliary matrix
re-invert auxiliary matrix
divert all power to life support
divert all power to shields
divert all power to shields
divert all power to shields
divert all power to shields
divert all power to shields
divert all power to shields
divert all power to shields
divert all power to shields
divert all power to shields
divert all power to shields
divert all power to shields
ABANDON SHIP. REPEAT. ALL HANDS ABANDON SHIP.
```

Problem B

Battle Sheep

The game of Battle Sheep is played with two players, and follows these rules:

Each player has an $N \times N$ grid where they have placed four non-overlapping ships, each of the ships parallel to one of the axes. They place exactly one of each of the following ships:

Name	**Length**
Pram	1
Sail Boat	2
Battle Ship	3
Hangar Ship	4

Starting with the first player, they call out cells in the grid. If the cell called out by Player A was covered by a ship on Player B's grid, Player B has to tell Player A that it was a hit. If all cells covered by that ship have been hit by Player A, Player B has to announce that their ship sunk, and player A gets another turn. If no ship was sunk by Player A's move, it's Player B's turn. This continues until one player has sunk all the other player's ships. The player who managed to sink all the other player's ships is the winner.

Alice and Bob have played a few times, but they got tired of doing the actual moves and making the announcements, so they decided to automate the process a bit. Now they instead write down where they placed the ships and in what order they wanted to call out grid cells. They never call out the same grid cell twice.

Given the moves by each player, output what the players would have announced if they actually played the game. In other words, you have to find out which ships sunk (and in what order) and who the winner is. Being a gentleman, Bob always lets Alice start.

Input specifications

The first line of the input consists of a single integer, T, the number of test cases.
Each of the following T cases begin with a line consisting of a single integer, N, the size of the grids.
The following N lines represents Alice's grid, and each line consists of N characters. These characters are either '`.`', '`1`', '`2`', '`3`' or '`4`'. A '`.`' means that the cell is empty. '`1`', '`2`', '`3`' or '`4`' means that that cell is covered by one of Alice's ships. All cells with the same

number are covered by the same ship. The number does not necessarily correspond to the length of the ship.
The next N lines represent Bob's grid, and has the same format as Alice's grid.
The next $N \cdot N$ lines represent Alice's planned moves. Each line has two integers, R_i and C_i, the row and column she calls out for that move (if the game gets that far).
The next $N \cdot N$ lines represent Bob's planned moves, and has the same format as Alice's moves.

Output specifications

For each test case output one line for each ship that was sunk, on the format `PlayerX sank PlayerY's ShipName`, in the order they were sunk. The last line of the output for each test case should be the name of the winner.

Notes and Constraints

- $1 \leq T \leq 20$
- $4 \leq N \leq 10$
- $1 \leq R_i, C_i \leq N$
- Each ship will be represented as a contiguous line of the same character and be parallel to one of the axes.

Sample input

```
1
4
1..4
22.4
3334
...4
112.
..2.
.32.
4444
1 1
1 2
1 4
2 3
3 2
4 3
2 1
3 3
4 1
4 2
3 4
4 4
1 3
2 2
2 4
3 1
1 3
4 4
3 4
1 1
4 3
4 1
2 4
2 2
1 2
4 2
3 1
3 2
2 3
1 4
3 3
2 1
```

Output for sample input

```
Alice sank Bob's Sail Boat
Alice sank Bob's Pram
Bob sank Alice's Pram
Alice sank Bob's Hangar Ship
Alice sank Bob's Battle Ship
Alice
```

Problem C

Counting Digits (Easy)

You have been assigned the task of writing down all numbers starting at N up to and including M, and report back the number of times you had to write a zero. You realize that you don't really need to write down all the numbers to do this. Write a program that calculates the number of zeroes you need to write, so you can save yourself from getting writing cramps.

Input specifications

The first line of the input consists of a single integer, T, the number of test cases.
Each case is given on a separate line and consists of the two integers N and M separated by a space.

Output specifications

For each test case, output the number of zeroes you had to write to write down all the numbers.

Notes and Constraints

- $1 \leq T \leq 20$
- $0 \leq N \leq M \leq 1,000,000$

Sample input

```
3
0 10
33 1005
1 4
```

Output for sample input

```
2
199
0
```

Problem D

Dad's Card Trick

You are always doing impressive card tricks, and it's driving everyone mad. To put a stop to this, your dad decides to challenge you to do a trick he thinks is impossible.

He starts by blindfolding you, and gives you a deck of N cards. He tells you that K of the cards are face up. Your task is to find a sequence of operations to perform, with the following goal: The deck should be separated into two decks at the end of the sequence, such that each of the two decks have the same number of face up cards (not necessarily K), no matter which order the cards were in initially. The other deck is initially empty. You are allowed to do the following two operations in any order, any number of times:

1. Move a card from one deck to the other.
2. Flip over any one card in any of the two decks. If it was face up, it will now be face down and vice versa.

In order to impress your dad, you have decided not only to accept the challenge, but to find the shortest sequence of operations possible. Being a prominent programmer, you realize that you can cheat by writing a program that generates this for you.

Input specifications

The first line of the input consists of a single integer, T, the number of test cases.
Each of the following T test cases consists of a single line with two integers, N and K, separated by a space.

Output specifications

For each test case, output the minimum number of operations needed to achieve the goal. If it is not possible, output -1 instead.

Notes and Constraints

- $1 \leq T \leq 20$
- $1 \leq K \leq N \leq 1000$

Sample input

```
3
2 1
3 2
10 3
```

Output for sample input

```
2
3
6
```

Problem E

Election

In ancient Rome, they used to hold elections by playing a game called Throw the Coin. A quick note for those of you studying Roman history is in order: this is not accurate – in actuality, they held elections by having their cats fight in the arena. Historical accuracy aside, the game of Throw the coin was a game in which the goal was to get as many allies as possible. At the end of the game, the person with the most allies would win. Note that these games were fixed most of the time, and the winner would normally be the one with the fiercest looking cat. With this in mind, it is actually more historically accurate than we meant for this problem statement to be. Also please kindly note that, while this paragraph is very long, we can assure you beyond reasonable as well as unreasonable doubt that it could absolutely have been much, much longer, well beyond the point at which it would be considered ridiculous by anyone possessing any degree of sanity at all.

Each player would bring a coin to the game, any coin, as long as it was perfectly circular and had an integer radius. The players would then throw their coins so that the center of each coin landed on integer coordinates in a coordinate system. They would then mark the area that was covered by each coin, and after everyone had thrown their coin, they would compare the area each coin had covered. A player would become allies with another player if the areas of their coins overlapped. Strangely, the areas were somehow guaranteed to never touch in only one single point.

N players have lined up to play. Your task is to figure out who won, i.e. who got the most allies.

Input specifications

The first line of the input consists of a single integer, T, the number of test cases.
Each of the following T cases begins with a line with a single integer N.
Each of the next N lines consist of a player's name, and three integers X, Y and R, representing the x- and y-coordinates of his throw, and the radius of his coin, respectively.

Output specifications

For each test case output a single line with the name of the winning player or, in case of a tie, output `TIE`.

Notes and Constraints

- $1 \leq T \leq 20$
- $2 \leq N \leq 100$
- $-100 \leq X, Y \leq 100$
- $1 \leq R \leq 10$
- Each name consists of between 2 and 255 lowercase letters `a-z`.

Sample input

```
3
3
charlie 0 0 3
john 1 1 1
peter -1 -1 1
2
magne -100 -100 1
edvard 100 100 100
3
christian 0 0 10
christian 2 2 3
ruben -5 0 1
```

Output for sample input

```
charlie
TIE
christian
```

Problem F

Fired

Charlie is the head of HR of a major corporation, let's call it Major Corporation Inc., or MCI for short. All employees of MCI except the CEO have one or more people they report to. Nobody reports directly or indirectly to themselves.

The budget for this year just arrived, and the post for salaries has been cut by C dollars. Charlie doesn't really like firing people, so he has created a computer program that fires people automatically if they don't have anyone they report to until everyone reports to someone. The program will however not fire the CEO automatically, even though they don't report to anyone.

Charlie has decided that he is willing to fire one person manually, and this is where you come in. He needs you to write a program to figure out the best person to fire, and his program will take care of the rest of the firing. The total salaries of all people fired has to be at least C dollars, but it should be as close to C as possible. If there are multiple solutions with the same total salary, you should pick the person for Charlie to fire with the highest employee number. Note that, since it is theoretically possible for CEOs to be completely incompetent, Charlie may choose to fire the CEO.

Input specifications

The first line of the input consists of a single integer, T, the number of test cases.
Each of the following T test cases begins with a line containing two integers N and C, the number of employees at MCI, and the amount you need to save.
The next N lines, numbered 0 to $N-1$, have the following specification:
Two numbers S_i and R_i, the salary of employee number i and the number of people they report to. The rest of the line will have R_i numbers E_{ij}; the employee numbers for the people employee i reports to.

Output specifications

Output the employee number of the person that Charlie should fire.

Notes and Constraints

- $1 \leq T \leq 100$
- $1 \leq N \leq 200$
- $1 \leq C \leq \sum_i S_i$
- $1 \leq S_i \leq 100,000$
- $0 \leq R_i < N$, only one person will have $R_i = 0$.
- $0 \leq E_{ij} < N$

Sample input

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

Output for sample input

```
1
3
```

Problem G

GG NO RE OMG CHEATZ

Knowledge of the game Risk is not necessary to solve this problem. All dice in this problem are six sided dice.

The game of Rizk is a Bulgarian variant of Risk. It is a game played between two players, each with their own army. Every time it's your turn, you can choose to either attack or add more units to your army, but not both. An attack is played in the following way:

Let's say player A is attacking, and has U_a units, and player B is defending, and has U_b units. The attacker has access to D_a dice, and the defender has access to D_d dice. The attack consists of the following steps:

1. Player A rolls $\min(D_a, U_a)$ dice.

2. Player B rolls $\min(D_d, U_b)$ dice. Note that this is different from regular Risk.

3. After all dice have been rolled, the players sort their dice from highest to lowest and compare each pair of dice in turn. For example, if player A rolled 2, 5, 1, and 3, and player B rolled 4, 1, and 3, they first compare the 5 to the 4, then the 3 to the 3, and finally the 2 to the 1. Note that the lowest roll(s) of the player with more dice stay out of the comparisons.

4. For each of these comparisons, the losing player loses a unit. The defending player wins ties. In the example from above, A would lose 1 unit, and B would lose 2 units.

5. If both players have at least 1 unit left, go to step 1.

The winner of the battle is the player with units remaining at the end. Note that there will always be exactly one player with units remaining at the end.

Mike and his brother has been playing for hours. None of them dare to attack, because they fear to lose. It's Mike's turn, and he has decided to cheat a little bit in order to win. He has decided that he wants to attack if he has at least 75% chance of winning this turn. Mike has X units, and his brother has Y units. How many units does he have to sneak into the game in order to have at least 75% chance of winning this turn?

Input specifications

The first line of the input consists of a single integer, T, the number of test cases.
Each test case consists of 4 integers, D_a, D_d, X and Y, as explained in the problem statement.

Output specifications

For each test case, output the minimum number of additional units needed to have at least 75% chance of winning.

Notes and Constraints

- $1 \leq T \leq 30$
- $1 \leq D_a, D_d \leq 4$
- $1 \leq X, Y \leq 500$
- There will be no cases where adding or removing a unit will make the probability of winning closer to 0.75 than 10^{-6}

Sample input

```
4
4 3 10 10
1 1 10 10
4 1 100 1
1 4 2 3
```

Output for sample input

```
5
8
0
11
```

Problem H

Herding Sheep

Elly and her herd of sheep are in trouble again. After a long day of grazing it is time to bring them into the barns in order to be safe for the night. The size of the barns is limited - each barn can accommodate at most K sheep. Some of the barns might not be entirely filled; they can even remain completely empty. It is only important that each sheep is in a barn.

For simplicity the sheep can be represented as N points and the barns as M points with integer coordinates on the plane. It is possible that several sheep, several barns or several sheep and barns share the same coordinates.

Elly's sheep walk a unit of distance per second. Thus, if for example one of them is at position (0, 0) and wants to go to a barn at (1, 3) it will need approximately 3.16227766 seconds to arrive at its destination. If the barn was at coordinates (3, 4) instead, the animal would need exactly 5 seconds to get there.

Help Elly by calculating the minimal time needed for all sheep to get into barns. The sheep can move simultaneously and you can assume that they do not disturb each other's movement.

Input specifications

The first line of the input consists of a single integer, T, the number of test cases.
Each test case begins with three integers, N, M, and K – the number of sheep, the number of barns, and the maximum number of sheep in a barn, respectively.
After that follows N lines, each with two integers, X and Y, giving the x and y coordinates of a sheep.
Finally comes M lines, each with two integers, X and Y, giving the x and y coordinates of a barn.

Output specifications

For each test case, output the minimum time needed to get all sheep into barns, such that there are at most K sheep in any given barn. The answer should have a maximum relative or absolute error of at most 10^{-6}.

Notes and Constraints

- $1 \leq T \leq 20$
- $1 \leq N, M, K \leq 200$
- $-1000 \leq X, Y \leq 1000$
- $N \leq M \cdot K$

Sample input

```
2
5 3 2
2 13
9 6
4 8
13 7
11 3
2 11
10 6
4 12
7 3 3
-959 -542
-669 -513
160 717
473 344
-51 -548
703 -869
270 -181
957 -509
-6 937
-175 434
```

Output for sample input

```
7.810250
1251.891369
```

Problem I

In the Shower (Easy)

Your roommate recently gave you a somewhat passive-aggressive post-it note containing the question "how many empty shampoo bottles do we really need to keep in the shower?". Passive-aggressive post-it notes generally start with a single number P, the number of passive-aggressive statements and/or questions. P lines then follow, each line containing a passive-aggressive statement or question. Whenever P is equal to 1 the line containing it is typically omitted, and the entire note will just be one passive-aggressive statement or question. This is by far the most common.

Somewhat frustated over your roommate's inability to count, you decide to write a program that any of your roommates can run whenever they wonder how many empty shampoo bottles are required for a happy existence here in Sector 001. Of course, different roommates can (and probably will) have different definitions for "empty", and therefore it is only fair that they should input this definition before getting an answer.

Input specifications

The first line of input is T, the number of cases. T cases follow.
The first line of each case contains two numbers separated by a space: E, the number of attempts that must will be made at extracting visible amounts of shampoo before considering the bottle empty, and N, the number of candidate bottles.
Then follow N lines, each line with a single number describing how many attempts were needed for a particular bottle. If the number of attempts E is *exceeded*, we consider the bottle empty.

Output specifications

Output a single line containing the number of empty shampoo bottles.

Notes and Constraints

- $1 \leq T \leq 100$
- $1 \leq E \leq 1000$
- $1 \leq N \leq 10$

Sample input

```
1
5 3
1
5
6
```

Output for sample input

```
1
```

Problem J

Join My Team

You are putting together a competition similar to IDI Open where the major difference is team selection. The participants are asked to order the other participants according to whom they would like the most to be on their team. The first person on their list is the person they would like the most to be on their team, and the last person is the one they would like the least to be on their team.

Your task is to assign all competitors into teams of two, in such a way that they are happy with the assignment. They are happy with any assignment unless there exists four competitors, A, B, C and D, where A is on C's team, B is on D's team, A prefers B to C, and B prefers A to D.

Input specifications

The first line of the input consists of a single integer, T, the number of test cases.
Each of the following T cases begins with a line with a single integer, N, the number of participants.
The next N lines each have $N-1$ integers, P_{ij}, giving the preference list of participant number i, separated by spaces.

Output specifications

For each test case, output one possible solution that the participants are happy with, given by a list of teams on the form i:j (i is on a team with j), where each team is separated by a space. There may be many such lists, but any of them will be accepted. If there is no solution, output `NO SOLUTION`.

Notes and Constraints

- $1 \leq T \leq 20$
- $2 \leq N \leq 100$
- $1 \leq P_{ij} \leq N$
- $P_{ij} \neq i$

Sample input

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

Output for sample input

```
NO SOLUTION
1:4 2:3 5:6 7:8
1:6 2:3 4:5
```

Problem K

Krusty's Burger

Your friend Krusty is running a "build your own burger"-restaurant. You follow these steps to build a burger:

1. Pick a burger size.
2. Pick a bun from N_b different buns.
3. Pick no cheese or a cheese from N_c different cheeses.
4. Pick up to 3 toppings from N_t different toppings.
5. Pick no sauce or a sauce from N_s different sauces.

Burgers cost \$1 per 50 grams and you can choose any burger size, as long as you stick to a nonzero positive multiple of 50 grams. You can add *additional* cheeses, sauces or toppings for \$1 per extra item.

For example, a burger with 100 grams, on bun number 1, with 2 different cheeses, 5 toppings, and no sauce, would cost \$5: \$2 for the burger, \$1 for the extra cheese and \$2 for the extra toppings.

Realizing that you have a huge array of options, your friend would like you to find out how many different burgers you can build without exceeding a given budget, B. Two burgers are different if one contains a burger size, bun, cheese, topping or sauce that the other one doesn't contain.

Input specifications

The first line of the input consists of a single integer, T, the number of test cases.
Each of the following T cases consists of the 5 numbers B, N_b, N_c, N_t, N_s separated by spaces.

Output specifications

For each test case, output the number of different burgers that can be built within the budget. This number can be very large, so the answer should be given modulo 1,000,000,007

Notes and Constraints

- $1 \leq T \leq 20$
- $1 \leq B, N_b, N_c, N_s \leq 1000$
- $3 \leq N_t \leq 1000$

Sample input

```
3
5 2 1 4 1
1 1 1 3 1
5 5 5 5 5
```

Output for sample input

```
632
32
299700
```

Problem L

London Underground

London underground consists of many stations, numbered 1 to N, and many lines connecting them, numbered 1 to M. All stations are connected, but some times you have to switch between lines to travel between two stations. This takes S minutes per switch. You can switch between lines L_1 and L_2 on station S_i if both lines stop at station S_i. The lines go back and forth all day, with the same distance between stations in both directions. Trains leave in both directions from every station every minute.

Alice and Bob are visiting London. They use the underground a lot, but they keep feeling that they aren't always taking the fastest route. Help them find the fastest route between stations A and B.

Input specifications

The first line of the input consists of a single integer, T, the number of test cases.
Each of the following T cases begins with a line consisting of 5 integers S, N, M, A, B, separated by spaces. These numbers give the number of minutes it takes to switch lines, the number of stations, the number of lines, and the start and end stations, respectively. The next M lines of a test case describe each subway line, and has the following format. First an integer X, the number of stations on the subway line. Then follows X station specifications, separated by spaces, in the order they appear on the line. Each station specification consists of two integers, where the first number, Sn_i, is the station number, and the second number, St_i is the number of minutes from the start of the line to that station.

Output specifications

For each test case, output the minimum number of minutes it takes to travel from station A to station B.

Notes and Constraints

- $1 \leq T \leq 20$
- $1 \leq S \leq 100$
- $2 \leq N \leq 100$
- $1 \leq M \leq 10$
- $A \neq B$
- $1 \leq A, B \leq N$
- $1 \leq X \leq N$ for all lines.
- $St_0 = 0$ for all lines.
- $St_i > St_{i-1}$ for all lines.
- $0 \leq St_i \leq 1000$ for all lines.
- All stations numbered 1 to N will be a part of at least one line.
- All stations will be reachable from all other stations.

Sample input

```
3
1 5 1 1 5
5 2 0 1 2 3 5 5 10 4 15
3 4 2 1 4
3 1 0 2 2 3 5
3 2 0 3 10 4 11
1 4 2 1 4
3 1 0 2 2 4 15
3 2 0 3 1 4 2
```

Output for sample input

```
8
9
5
```