National Taiwan University

2013 Team Ranking Contest

A	The Thieves' Course	3 s
B	Monkey Island	3 s
C	Layered Maze	5 s
D	Starcraft	3 s
E	Ghost Trick	2 s
F	Lode Runner	3 s
G	Magicka	1 s
H	Portal	2 s
I	Super Hexagon	2 s
J	SpaceChem	2 s

August 25, 2012

(This page was intentionally left only almost blank.)

Prologue

Like Sam never left Frodo, like "behind every great man there's a great woman", we may also say, "behind every great programmer there are great games they play." Today, in order to become a great programmer, you will venture through the cruelest trial of ten great games. Either way, please have fun doing so.

- Q: The statements are lousy, will it be enough if I just read the I/O section?
  A: Patience, dear. Patience.
- Q: Is this some kind of Product placement?
  A: Yes, and we totally intend to recommend these excellent games.
- Q: How the he** is DotA not the list, then?
  A: It deservers a separate, individual problemset. We'll see.

A. The Thieves' Course

time limit: 3 seconds

You are making your own RPG games in an RPG maker. Intrigued by features in the game "The Thieves' Course", you decide to add a puzzle to your game.

我想要成爲很帥的盜賊！
爲此，我會很努力的學習的，
請大家多多指教。

You placed a button on the floor, and many treasure boxes lining beside it. This mysterious button will swap the boxes on some specified positions. The goal of this new puzzle is to sort the boxes according to their values, from the lowest to the highest. Of course, with this only button.

Now it's the time to check whether this button can truely sort these boxes. To estimate the difficulty to this puzzle, you also need to find the number of times pushing the button in order to properly sort them.

Input Format

The first line of the input file contains an integer T ($1 \leq T \leq 100$) indicating the number of test cases.

Each test case starts with a line containing an integer n ($1 \leq n \leq 100000$), indicating the number of boxes. Next line contains n *distinct* integers $v_1, v_2, \cdots, v_n$ ($1 \leq v_i \leq 10^9$) indicating the value of the i-th box. The third line contains a permutation $p_1, p_2, \cdots, p_n$. That means, after pressing the button, the boxes of position i will be moved to position p_i.

Output Format

For each test case, output `-1` if it is impossible to sort them, otherwise output the number modulo 1000000007.

Sample Input

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

Sample Output

```
0
1
-1
4
```

B. Monkey Island

time limit: 3 seconds

“Monkey Island” is a series of adventure game with intriguing plots, humorous dialogs. Usually, the main protagonist Guybrush Threepwood, Mighty Pirate$^{\text{TM}}$ is up against his biggest foe LeChuck to unravel a bunch of mysteries and save the world.

One of the crucial events in the “Monkey Island” series is a verbal combat known as the “Ultimate Insult”, where two challengers insult each other, and the one that fails to strike back (verbally) loses.

Similarly the programmers in the world of “Monkey Island” is also fond of the verbal combat, but they have developed their own version of it known as the “Binary Ultimate Insult”. As its name suggests, in “Binary Ultimate Insult”, the two challengers speak a binary string instead, and the one who is more insulting prevails.

However, not many (except you) knows that, the secret of “Binary Ultimate Insult” lies not in how obsolete or ugly the binary string looks, but in a secret measure: the number of “Perfect Insult Segment”.

A *Perfect Insult Segment* is a binary string w satisfying the following condition: For any non-empty partition $w = uv$, $u < v$ must hold, where the operator $<$ denotes the lexcographical order on both strings. If a binary string is a Perfect Insult Segment, then it is the most powerful in combat. If a binary string is not a Perfect Insult Segment, then its power is determined by how close it is to being a Perfect Insult Segment: if it could be partitioned into just a few Perfect Insult Segments, then it is still relatively strong.

Given a binary string, your job is to partition this string into minimum number of Perfect Insult Segment.

Input Format

The first line of the input file contains an integer T ($1 \leq T \leq 100$) indicating the number of test cases.

For each test case there is a binary string S, which length is no more than 10^5.

Output Format

For each test case output the desired number: the minimum number of partitions required such that each part is a Perfect Insult Segment.

Sample Input

```
4
0000000000000000
0000000000000001
0000100110101111
1011101001100010
```

Sample Output

```
16
1
1
6
```

C. Layered Maze

time limit: 5 seconds

Ha, it's time to play mini game during the typhoon vacation. This time we're going to play a traditional game called "Layered Maze." Imagine that you're in a 3D grid with N layers and each layer is a $H \times W$ grid. Each cell contians either a wall, a key, a door, an exit, or nothing at all. The keys have five colors: green, blue, orange, purple, and yellow. The key of each color can open certain doors with same color.

It is easy to solve a maze if you have already know the layout in each layer. But, in the beginning of this challenge, you can only know the layout of the maze where you stand at. Fortunately, you can get more information to the upper layer and to the lower layer. You will know all reachable cells to the upper layer, and will know the reachable cells to the lower layer if and only if in the relative position in this layer is an empty cell, a key, or an exit.

Once you are in an exit cell, you win this challenge and the game ends. In each step, you can move toward six directions: front, back, left, right, up and down. In each move you need 1 second. To open a door, you must stand one cell from one of the four directions to the door at same layer. But it needs no time to open a door, or pick up a key. Once the door is opened, the door disappears but you can still have that key.

But, I have no sufficient time playing this puzzle. Thus, before I start playing this game, can you tell me the minimum time that I need to solve this puzzle, if I am feeling extremely lucky?

Input Format

The first line of the input file contains an integer T $(1 \leq T \leq 100)$ indicating the number of test cases.

For each test case, first line contains three integers N, H, W. Then there is N blocks describing each layer of the maze, from the uppermost layer to the lowest layer. The keys is denoted by lowercase letters except letter '`s`' and '`e`' while the corresponding doors is denoted by uppercase letters. The starting position is denoted by an '`S`' character and the exit position is denoted by '`E`'. All the walls are filled with '`#`'. $(1 \leq N, H, W \leq 50)$

All mazes are legal and surrounded by walls. It is guaranteed that there are at most five types of key.

Output Format

For each test case output the minimum time in seconds. If there is no solution please outout `-1`.

Sample Input

```
2
2 3 5
#####
#S..#
#####

#####
#.#E#
#####

2 5 5
#####
#S.A#
#.#.#
#...#
#####

#####
###E#
#####
#a#b#
#####
```

Sample Output

```
3
9
```

D. StarCraft

time limit: 3 seconds

“StarCraft” is a real-time strategy game displaying the epic and everlasting battle between the three races: “Protoss”, “Terran”, and “Zerg”.

Compared to many other RTS games, the resource design in “StarCraft” is relatively simple. Mainly “mineral” and “vespene gas” are all that is needed to form a devastating army. Therefore, controlling (and denying) the resource could be very essential in winning games.

One of the most known unit in the game is probably the “Arclite Siege Tank”, which is a vehicle unit of the Terran race. By deploying itself into siege mode, the unit possesses a ridiculously large defense radious, and would destroy anything that comes near its range.

Another well-known unit is the “Dark Templar” of the Protoss race. It is a stealth unit with incredibly high damage, and is perfect for ambushing and harrasing a enemy site without certain preparation.

Today, DarkPi, as a Terran player, is practicing against DarkKelvin, who is a Protoss player. In the early stage of their game, since they are both rather unskilled, they are not going to do any early-harassing. Instead, the two players have both scouted that the map has N mineral sites, and they want to capture the mineral sites to steady their economy.

The N mineral sites could be described by a pair of integer (v_i, c_i), denoting the economy value it could provide if captured, and the cost required to maintain the defense of it after capturing it. The two players take turn taking actions. DarkPi makes the first move. During Darkpi's turn he would quickly seize P mineral sites to his control, and deploy Arclite Siege Tanks around it so that the sites are well-protected. During DarkKelvin's turn he will send small groups of Dark Templar to guard Q unoccupied mineral sites, so they can never be captured by DarkPi in the future.

$P+Q$ will always divide N, and note that the two players must capture exactly P (Q) sites during his turn (they will not give up their chance even if the action is not so cost-efficient).

Now given the economic value and defense costs for the N mineral sites, DarkPi wants to maximize the total cost-effeciency of his captured sites (i.e. $\sum_{i \in S} v_i / \sum_{i \in S} c_i$, where S is the set of sites occupied by Darkpi), while DarkKelvin wants to sabotage Darkpi's goal as much as possible. Please determine for DarkPi, what is the largest cost-effeciency obtainable if both players play optimally?

Input Format

The first line of the input file contains an integer T ($1 \leq T \leq 80$) indicating the number of test cases.

Each testcase starts with integers N, P, Q ($1 \leq N, P, Q \leq 700$, $(P+Q)|N$). Then follows N lines, each with two integeres v_i, c_i ($1 \leq v_i, c_i \leq 10^7$) describing the attributes of the mineral sites.

Output Format

For each test case, output the optimal cost-efficiency Darkpi can obtain in *simplified fraction.*

Sample Input

```
2
4 1 1
1 2
1 2
1 2
1 3
6 2 1
7 7
5 6
7 8
9 10
4 6
8 8
```

Sample Output

```
1/2
27/29
```

E. Ghost Trick

time limit: 2 seconds

“Ghost Trick” is a game where the main character dies at the very beginning and becomes a wandering soul. Through “the power of the dead”, as a soul you may jump around objects, observing the environment, and make huge influnce to the trajectory of real world by altering objects in subtle ways.

Well, that’s all you need to know. we don’t want to spoil the fun of the game, so we will say no further about the storyline.

Since probably not many of you have played the game before, we will formalize the game below. It is not exactly same as the original game, so please read carefully and stick to the rules that follow.

There are various objects in the screen, that as a soul you can warp to and “wander” on them. An object has the shape of a simple polygon, and while you’re on it you can move to any interior or boundary point of the shape.

You should consider yourself as a point. Also the screen is of width W, and its height is large enough (in terms of you can always see necessarily high or low if you want) such that you can consider it infinite. The screen is therefore basically an interval of width W always centered at you. You could warp to any object that (partially or completely) appears in the screen. Note that even if only a single point on the boundary appears on the edge of the screen, you can still warp to it.

Spiritual “warping” requires effort, though, and each different object you come in contact with consumes 1 unit of energy. (However revisiting a visited object does not require extra effort).

Now, given the objects (as simple polygons), and you can choose the object you begin on freely, the goal is to “observe” all objects using the minimum energy possible.

"Observing" could be done if and only if an object appears in the screen (even if just a single point in the closed area of the object appears counts). Please answer whether it is possible to observe all object, and if so, what is the minimum possible energy required?

Input Format

The first line of the input file contains an integer T ($1 \leq T \leq 60$) indicating the number of test cases.

Each testcase starts with integers N, W ($1 \leq N \leq 40000$, $1 \leq W \leq 10^9$), indicating the number of objects and the screen width, respectively. Then follows the description of the N objects, one by one, each on a single line. The description of each object begins with an integer K_i ($3 \leq K_i \leq 10$), being the number of vertices of the object (which is a simple polygon), then follows the coordinates (x_j, y_j) of the K_i vertices, given in clockwise or counter-clockwise order. The absolute value of the coordinates will not exceed 10^9.

Note that objects may possibly overlap.

Output Format

For each test case, if it is possible to start from some object and observe every objects, output the minimum energy required. Otherwise if it is not possible to observe all objects, please output "No trick." without quote.

Note that the starting object can be chosen arbitrarily, and does not consume energy upon beginning or revisiting.

Sample Input

```
2
7 2
3 1 0 1 1 5 2
3 2 3 3 3 4 1
4 3 0 3 1 8 1 8 0
4 4 2 6 2 6 3 4 3
3 7 7 6 0 9 2
3 10 4 8 3 9 5
4 11 1 12 3 12 5 11 4
2 2
4 1 1 2 1 2 2 1 2
4 7 7 7 8 8 8 8 7
```

Sample Output

```
2
No trick.
```

F. Lode Runner

time limit: 3 seconds

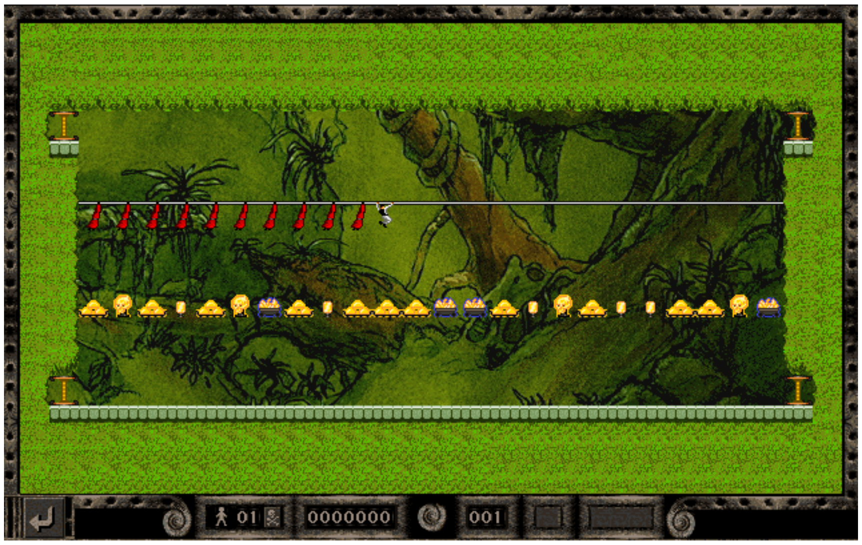
“Lode Runner” is a game where you, the lode runner, ventures through levels and levels to collect valuable treasure, while being chased down by vicious murderous monks. The only thing to stop you from being torn to pieces by the monks is various tools you may find in level, and taking the terrain to your advantage.

Let’s not get carried away into the specifics, though. Today, you’re going to solve a particular level of “Lode Runner”. Take notice that the rules are similar, but not the same with the original game, so be sure to follow the problem description carefully.

First some basics about the game “Lode Runner”. The map is simply a 2D platform world (that means gravity pulls you downward). There could be various “treasures” on the map, which you would want to collect. The player can collect the treasures simply by passing through them (for example, running or falling through them). However, the evil monks are not only vicious but greedy. Should they pass through a treasure, they will steal the treasure and drag it into their pockets. In the original game the treasure can be reclaimed upon the monk’s threat of death, however in this problem consider it stolen forever should the treasure be ever taken by a monk.

Some special devices that appear in the level of concern should also be introduced. A *“horizontal bar”* is – as it name suggests – simply a horizontal bar that the player can hang and move on the horizontal bar as they wish, and drop from it should they wish to. Note that you cannot move horizontally on the middle of a drop, that is, you drop completely vertically once you release yourself from the horizontal bar.

A *“teleporter”* is a pair of devices that would transport you to its counterpart once you enter. There could be multiple pairs of transporters on the map, and entering each of them would teleport you to the corresponding counterpart.



Now you are facing a particular troubling level. As seen in the picture above, the level consisting of a horizontal bar on the top, and a horizontal array of treasures some space down below. Since the treasures are in the air, the only way to acquire them is to drop from the horizontal bar, and collect the treasure during the drop. Also note that there are two corresponding pairs of teleporters, the two on the left and the two on the right are paired respectively. Therefore once you fall to the bottom, you will be able to swiftly run to the left (or right), then teleport back to the top left (or right) and repeat the drop-and-collect action again.

However the vicious monks are the complication. Starting from the very beginning they will follow right behind you, and closely copy your action. Initially you are at the bottom platfrom with a long array of monks piling right above you (which means they will fall and follow right after you, but you are free to run to left or right intially). So each time you choose to enter a (either bottom-left or bottom-right) teleporter, all of them will follow through and tags you closely as you cross the horizontal bar. When you decide to drop at some specific location, the whole array of monks also drops, thus taking treasures "behind" you while they are in the air. To be more specific, when you emerge from the left (right) teleporter and drop at position x, you take the treasure at position x, while all treasures with coordinate less (greater) than x are stolen.

You do not have to worry about begin caught by the monks though, as long as you only repeat the "drop-and-collect then teleport back" strategy, as you fall and runs significantly faster then them.

Obviously, to solve this level single-playerly is simple, just always enter the left teleporter and drop to collect the leftmost treasure availbale (so the monk will not steal anything) until you have collected every treasures. So Kelvin and Pipi decided to play the level in a competitive way: The two friends take turn deciding each turn of action. That is, Kelvin starts first and chooses to enter a teleporter, and drop (and possibly collect treasure) at a position of his choice, then Pipi does the similar, then Kelvin ... etc, until there are no more treasure to take. Each treasure has a specific value s_i, and the total score of a player is the sum of the treasures' value taken during by the player his turn.

So Kelvin wonders, if both players take the optimal strategy, how much at most can Kelvin score more than Pipi?

Input Format

The first line of the input file contains an integer T ($1 \leq T \leq 50$) indicating the number of test cases.

Each testcase starts with an integer N ($1 \leq N \leq 2000$), indicating the number of treasures. The second line is a list of N positive integers indicating the value of each treasure, from left to right. The value of the treasuers will not exceed 10^9.

Output Format

Output a single integer indicating the maximum amount of score Kelvin can win against Pipi.

Sample Input

```
2
5
3 3 3 3 3
8
3 1 5 2 10 1 1 5
```

Sample Output

```
3
7
```

(This page was intentionally left blank.)

G. Magicka

time limit: 1 seconds

Harry Potter? Nah, be a real magician in the game "Magicka". In "Magicka", your are a wizard apprentice venturing through a series of curious event, saving the world with your magic powers.

There are several basic element you can conjure: namely water, life, shield, cold, lightning, arcane, earth, fire, steam, and ice. If you conjure the elements in some specific sequence, they become a spell. For instance, casting "Earth-Steam-Water-Steam" then one creates a tornado; casting "Fire-Earth-Steam-Earth-Fire" summons a wave of meteor shower.

In one of the ordinary days, two wizards, Kael'Van and Daak'Phi are dueling – just for fun though, they are just very good friends who like to kill each other; May I remind you that casting "Life-Lightning" results in the revive skill that brings a very dead folk back to life.

Anyway, as Kael'Van yells *"!#RjE+%YH-D$"* and aims his wand forward, Daak'Phi also screams *"*Th_Dŝf'E-(W"* as a deadly ray shoots from his wand. To be honest you have no idea what they were yelling, but now it doesn't matter does it? That's why you are a programmer in NTU-CSIE, not a bloody wizard in Hogwarts! Miraculously, the two rays collides in the middle, and the elements that the spell is conducted of freeze in the air, aligned one after another between the two great wizards, and the duel went into a stalemate.

But Kael'Van and Daak'Phi are no ordinary wizard, they can meticulously maneuver their spell to out-duel other wizards, so their challenge begins.

Consider the ray of spell (now frozen between Kael'Van and Daak'Phi) as a string, where each element is represented by a case-sensitive english alphabet (either lowercase or uppercase). The left end of the string is the element closest to Kael'Van, while the rightmost element is the one closest to Daak'Phi. The two wizards take turn performing their move. Each time a wizard may either lessen their strength of output, so that the element nearest to him vanishes; or he may also force his will through the ray, so the element at the opposite end (farthest from him) vanishes. During a wizard's turn he can and must always take exactly one of the two actions stated above.

After several (or possibly none) maneuvers between the two wizard, if the elements remaining on the ray become symmetric, that is, the string representation of it become a palindrome, the ray becomes highly unstable and the wizard that was supposed to make the next move will be blown apart (and thus loses the duel). In that case, of course, the winner will kindly revive his defeated friend, but that is not of your concern.

Rather, being a programmer, you wonder that given the initial string representing the frozen ray of element, and if Kael'Van shall take the first move in the duel, who will win the duel, assuming they both make optimal decisions?

Note that the duel always end – as when only one element remains, it is always a palindrome.

Input Format

The first line of the input file contains an integer T ($1 \leq T \leq 5140$) indicating the number of test cases.

Each testcase consists of a single string S ($1 \leq |S| \leq 100000$), representing the ray of elements. S consists of only lowercase and uppercase english characters (treated case-sensitively). Note that S may or may not be a palindrome.

Note that the amount of testcase is tremendous, but it is guaranteed that *at most* 200 *of them will be maximal*, while the rest of them will be small cases.

Output Format

For each testcase, if Kael'Van shall win the fight, output "kael" in a single line; otherwise output "daak".

Sample Input

```
3
aaaa
abcde
aaab
```

Sample Output

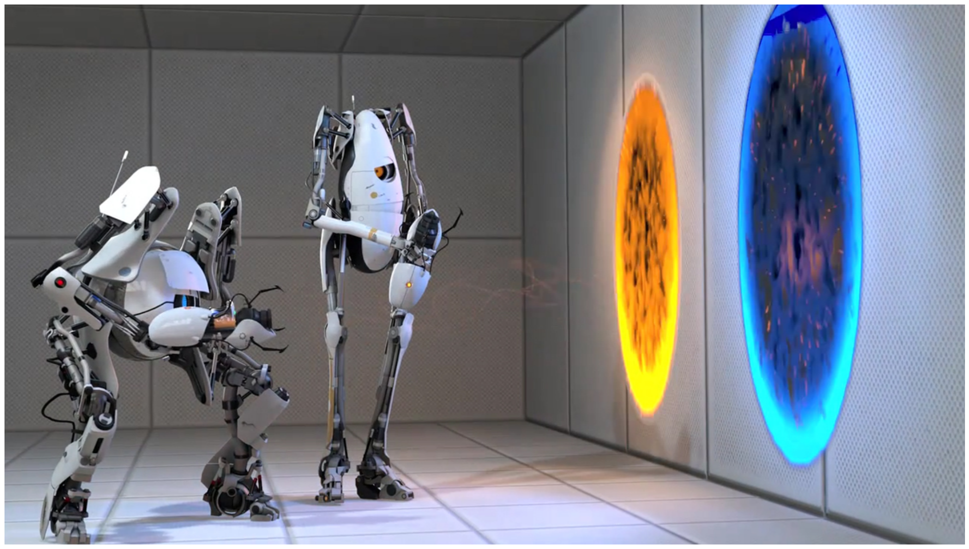
```
daak
daak
kael
```

H. Portal

time limit: 2 seconds

"Portal" is one of the most classic puzzle-game. The game primarily comprises a series of puzzles that must be solved by teleporting the player's character and simple objects using a shooting device known as "the Aperture Science Handheld Portal Device", a device that can create inter-spatial portals between two flat planes.

More specificially, the ASHPD could open (and maintain) two portal at the same time, and while two portals are opened, one could warp from one end to the other immediately. When a third portal is opened, the player who opens the portal must choose to destroy either of the two previous opened portals.



Now consider a two-player game on a given weighted undirected graph to describe the layout of a specific map (the weight on an edge being the walking distance required to travel through it). Player 1 wants to get from s_1 to t_1, player 2 wants to get from s_2 to t_2. Players can use their ASHPD guns to open a portal *at where he is currently standing*. Both players have their own ASHPD gun, so there could be at most two pairs of portal opened at once (but a portal can still only lead to the other portal created by the same ASHPD gun), and a player could also utilize the other player's portal.

Note that you should consider the time and players' movement continuous, as is the graph itself. Therefore, portals could be openend not only at graph nodes but also on edges.

Please answer what is the minimum total walking distance for both player to reach their destinations, that is, the sum of walking distance of both players.

Input Format

The first line of the input file contains an integer T ($1 \leq T \leq 50$) indicating the number of test cases.

Each testcase starts with a pair of integers N, M ($1 \leq N \leq 250, 0 \leq M \leq N(N-1)$), indicating the number of nodes and bidirectional edges the graph has. Then follows four integers s_1, t_1, s_2, t_2, describing the start and destination nodes of the two players. Finally M lines follow, each with three integers v_i, u_i, d_i indicating the presence of a bidirectional edge of length d_i pointing from node v_i to u_i. Each pair of (v_i, u_i) appears at most once in the list of edges. $1 \leq v_i, u_i, s_1, t_1, s_2, t_2 \leq n$, and $d_i \leq 10^6$.

Output Format

For each testcase, output a single integer: the time required for both players to (cooperatively) move to their destinations. In case it is impossible to do so (for either of the player), please output "impossible" without quotes.

Sample Input

```
2
5 5
1 3 2 4
1 2 1
2 3 2
3 4 4
2 5 3
5 4 2
4 0
1 2 3 4
```

Sample Output

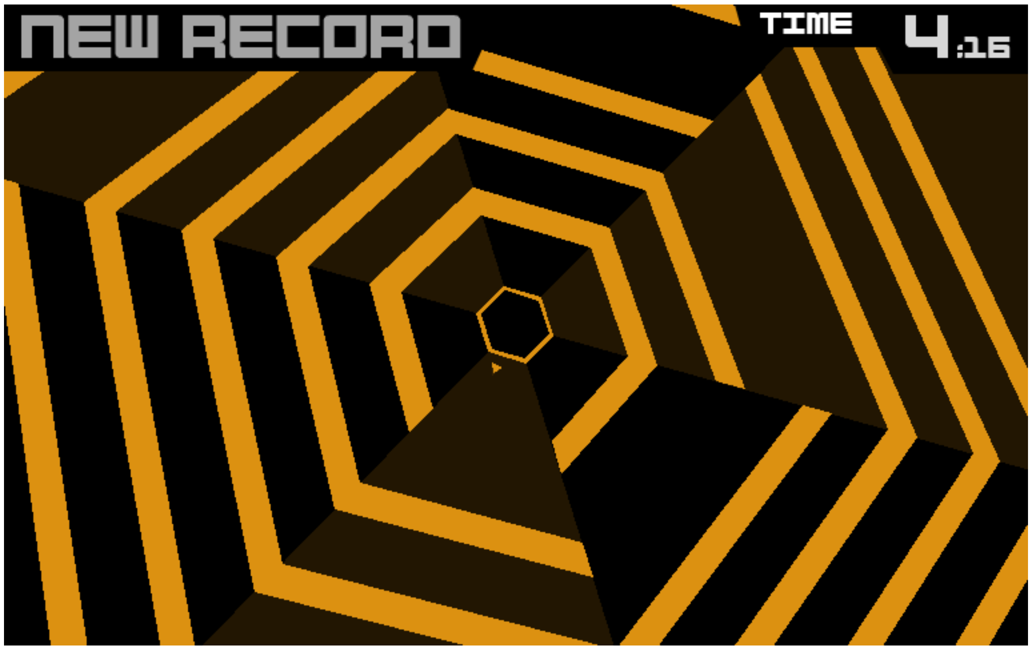
```
7
impossible
```

I. Super Hexagon

time limit: 2 seconds

"Super Hexagon" is a game that will defeat you, provoke you, humiliate you the first several times you play. I have made bets with numerous people: for the first ten times you play, survive for ten seconds then I owe you a drink.

None have yet to succeed.



As shown in the picture above, the tiny triangle just next to the smallest hexagon is the cursor you control. "Controlling" is very simple - you can either move in clockwise direction or counter clockwise direction, with the cursor's movement speed and distance to origin (center of the screen) fixed. "Walls" of different shapes will close in from outside until they eventually shrink into the smallest hexagon and disappear. Collapsing into a wall results in a game over. So basically, all you need to do is dodge all the wall as they close in on you, and survive as long as possible. Simple, right?

Since probably not many of you have played the game before, we will formalize the game below. It is not exactly same as the original game, so please read carefully and stick to the rules that follow.

In this problem, you should consider yourself as a point moving on unit circle centered at the origin. At time 0, you are at $(1, 0)$. Your movement speed is again fixed at a specified angular speed ω, at which you may move clockwise or counter-clockwise on the unit circle. Note that time is continuous and your control is infinitesimally precise, so you may move and stop freely and as rapidly as you want.

Walls in this problem are simply line segments – each individual wall i is described as a pair of points P_i, Q_i, which denotes the initial configuration of the wall; its movement is described by a shrinking ratio r, so that at time t, the wall shrink into a copy of itself with size e^{-rt}, i.e. the segment from $e^{-rt}P_i$ to $e^{-rt}Q_i$. Of course, this way we see that a wall never actually "disappear", but since it won't be affecting you once it completely shrinks into the unit circle, one can consider that the wall is non-existent after that. "Collapsing" is defined as touching the wall at anytime – you are not allowed to touch any points on the wall at any time.

The walls may be closed, or even possibly intersect.

The question of the problem is simple. For a given configuration, please answer if it is possible to escape? (By "escaping" it means dodging all the shrinking walls until they all shrink into the unit circle)

Input Format

The first line of the input file contains an integer T ($1 \le T \le 20$) indicating the number of test cases.

Each testcase starts with an integer N ($1 \le N \le 40$) and two real number r, ω ($1 \le r \le 10$, $1 \le \omega \le 10$), indicating the number of walls, shrinking ratio and angular speed. Then follows N lines, i-th line contains four real numbers Px, Py, Qx, Qy describing the wall, meaning that there's a wall from $P_i = (Px, Py)$ to $Q_i = (Qx, Qy)$. $(1.0 \le \sqrt{Px^2 + Py^2}, \sqrt{Qx^2 + Qy^2} \le 10000)$

All input are given to at most 2 digits. The distance from origin to the line connecting P_i, Q_i is at least 10^{-2}. Each wall has length at least 0.1. It's guarateed that the answer would not change even if we lengthen each segment by 10^{-2} on each side.

Output Format

For each testcase, if it is possible to escape, output "yes"; otherwise output "no".

Sample Input

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

Sample Output

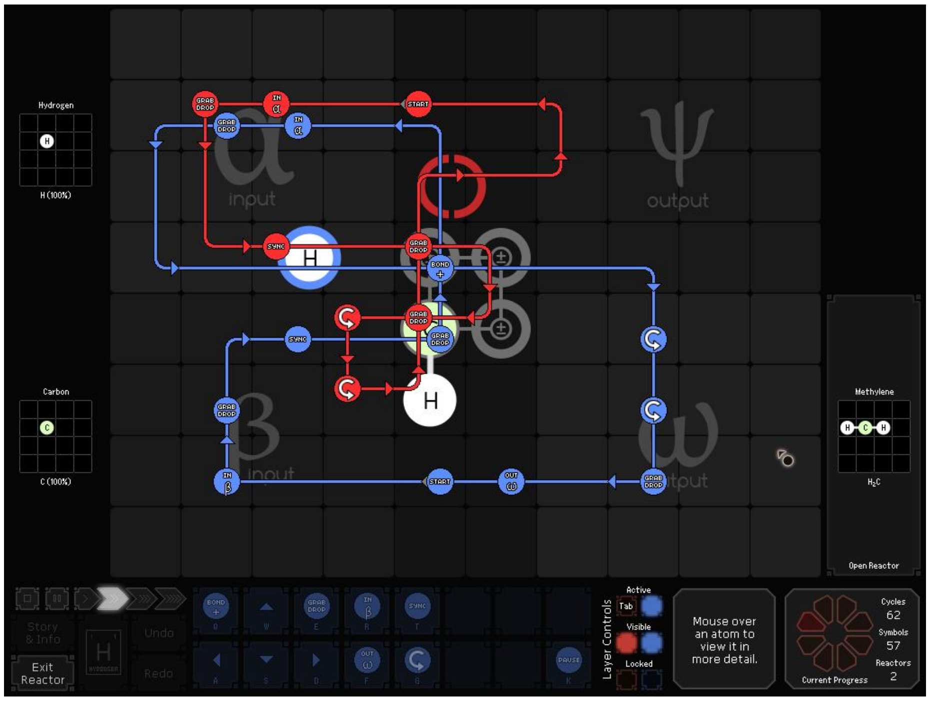
```
no
yes
```

(This page was intentionally left blank.)

J. SpaceChem

time limit: 2 seconds

"SpaceChem" is an extremely mind-twisting game. If you never understand how people like Sudoku so much as it is so easy and un-creative; if you possess the unbending will to solve stages after stages of interesting problem, where the difficulty ramps up so quickly; this is the right game to play.



In "SpaceChem", mostly you are asked to perform several actions such as forming chemical compounds, dissecting them, or even fusing them. But how the chemical reaction should be done is not your main concern, you mission is to design a layout of delivery/reaction network using a timed loop and control nodes.

"Well, that doesn't seem diffcult at all," you said, "any EE/CSIE student could design a circuit and those pipelining thingy, even my grandmother can!" Things are not always that simple, as you can see, the number of some reaction device is limited, and so is the grid to contain your design. As in any routing situation, space is always a concern and sometimes you cannot route too many things into a same cell.

The full view of the game is fairly sophisticated, so we'll try to tackle a simple version of a related problem here. It's quite different from solving an actual "SpaceChem" level, but please stick with the problem description.

Consider there is already a delivery network, seen as an undirected graph. The network is designed for delivering raw material for synthesizing alkyl compounds, so basically you are to deliver two kind of elements: "C" (carbon) and "H" (hydrogen).

Both elements come as a stream of atoms, entering the network at the specific source nodes s_c and s_h, and is destined for output nodes t_c and t_h, respectively. You can utilize an "atom carrier" to transport the elements. An atom carrier *must* start from one of the source node, carrying one atom to its destination along some path on the graph, then return to the same source node along some (possibly different) path.

It is required that during one phase exactly n_c carbon atoms and n_h hydrogen atoms should be brought from their source to their destination. Thus, exactly n_c atom carriers should make a round trip from s_c to t_c then back again, similarly with n_h.

The only problem is, as mentioned before, the routing of atom carriers cannot be arbitrary. Edges in the graph are of two kind: "free" and "restricted". "Free" edges can be used arbitray times by the atom carriers, while the "restricted" edges can only be used at most twice. For example, if the edge (u, v) is restricted, then you can have two carriers passing in the $u \to v$ direction, or one each in $u \to v$ and $v \to u$ direction; in both cases the edge's allowed usage is saturated and cannot be utilized by any other delivery path.

Given the adjacency matrix describing the graph, and the information of delivery requirements, please answer if it is possible to accomplish the mission.

Input Format

The first line of the input file contains an integer T ($1 \leq T \leq 100$) indicating the number of test cases.

Each testcase starts with an integer N ($1 \leq N \leq 50$), indicating the number of nodes. Then follows six integers being s_c, t_c, n_c, s_h, t_h, n_h ($0 \leq s_c, t_c, s_h, t_h \leq n - 1$, $1 \leq n_c, n_h \leq 50$). Finally there is an N-by-N matrix describing the adjacency matrix of the graph. The j-th character on the i-th line of the matrix describes the status of the edge (i, j), 'X' indicates no edge, 'O' indicates restriced edge, and 'N' inidicated an unrestricted (free) edge.

The adjacency matrix is always symmetric, and the characters on the diagonal is always "X". s_c, s_h, t_c, t_h will be pairwisely distinct.

Output Format

For each testcase, if it is possible to meet the requirement and deliver the necessary atoms, please output "Yes"; otherwise output "No".

Sample Input

```
2
4
0 2 1 1 3 2
XNXO
NXOX
XOXO
OXOX
4
0 2 1 1 3 1
XOXO
OXOX
XOXO
OXOX
```

Sample Output

```
No
Yes
```