It is 23:55 on December 31. Squark is playing Twelve Months.

Twelve Months is a poker divination game well-known in some regions of China. It is believed to be able to predict one's luck in each month in the coming year on New Year's Eve. 48 cards, a standard 52-card deck excluding 4 Kings, are used in the game. First the 48 cards are shuffled randomly and evenly distributed into 12 stacks facedown. Each stack stands for a month, i.e. the first stack stands for January, the second stack stands for February, etc. Then, starting from the first stack, the player repeats the following process until all the cards are opened in the current stack.

1. Open the topmost unopened card from the current stack;
2. Move to the stack pointed by the number of the card just opened (Ace is the first, Jack is the eleventh and Queen is the twelfth), i.e. the new stack becomes the current stack.

According to the prediction, the player will be lucky in a month if the stack standing for the month is cleared, i.e. all the cards in that stack are opened. Of course, each player can only do the divination once in one year. Repeating the divination will have no effect. Maybe you have realized that the first stack will always be cleared, so the player will always be satisfied because he/she will be at least lucky in the coming month.

Squark has distributed his cards and opened a few of them according to the rule. Suddenly, an urgent phone call comes and interrupts his game. It is from Sevenkplus asking him about some difficult algebraic geometry problems. These problems are so difficult that Squark spends a lot of time but solves none of them. When Squark comes back to his cards, they has been gathered and put into one stack. Squark cannot restore the game to the situation when he left. Furthermore, Squark cannot play the game again as the clock has passed 12 o'clock and it is January 1 now. Fortunately, Squark can still remember the cards he opened and the exact open sequence. So he wants to know the probability of each possible result if he could have continued the game.

As January is always a lucky month, there are 2^{11} possible results regarding the luckiness in the other months. We define a comparison method for these results to sort them, which compares the luckiness in December first, and then in November, and so on. To be unlucky is considered to be smaller than to be lucky.

To simplify the work to test your program, Squark considers the number of months in each year to be n instead of 12. In this way, the number of possible results is 2^{n-1} instead of 2^{11}.

Input

The first line contains an integer T ($T \leq 40$) denoting the number of the test cases.

For each test case, the first line contains 2 integers n ($2 \leq n \leq 12$) and m ($0 \leq m \leq 4n$), where n is the number of months in each year and m is the number of cards Squark opened. The second line contains m integers, standing for the cards opened by Squark (1 for Ace, 11 for Jack and 12 for Queen) in the order how Squark opened them.

Note that the second line is a blank line if $m = 0$.

Output

For each test case, output 2^{n-1} lines, each containing a real number representing the probability of each possible result in the order as described above. Your answer is considered to be correct if and only if it satisfies at least one of the two conditions below.

1. The absolute error is at most 10^{-12};
2. The relative error is at most 10^{-9}.

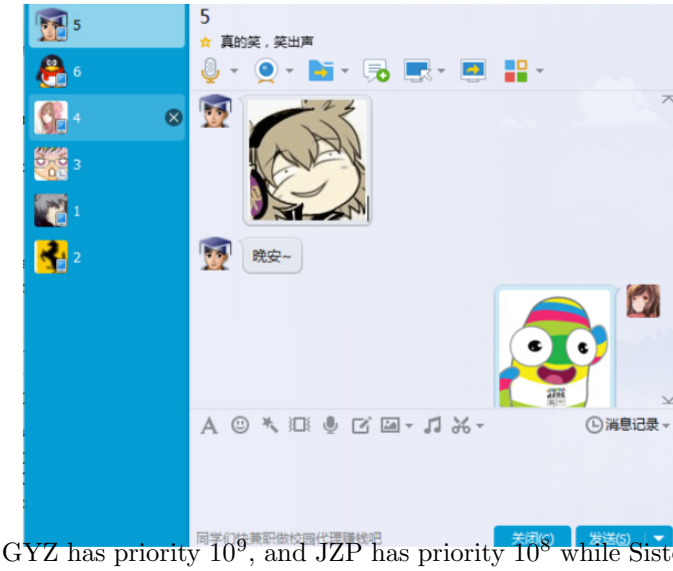
Sample Input

```
2
2 4
2 2 2 2
2 0
```

Sample Output

```
0.0000000000000
1.0000000000000
0.5000000000000
0.5000000000000
```

As everyone knows, DRD has no girlfriends. But as everyone also knows, DRD's friend ATM's friend CLJ has many potential girlfriends. One evidence is CLJ's chatting record.



CLJ chats with many girls all the time. Sometimes he begins a new conversation and sometimes he ends a conversation. Sometimes he chats with the girl whose window is on the top.

You can imagine CLJ's windows as a queue. The first girl in the queue is the top girl if no one is "always on top".

Since CLJ is so popular, he begins to assign a unique positive integer as priority for every girl. The higher priority a girl has, the more CLJ likes her. For example, GYZ has priority 10^9, and JZP has priority 10^8 while Sister Soup has priority 1, and Face Face has priority 2.

As a famous programmer, CLJ leads a group to implement his own WM (window manager). The WM will log CLJ's operations. Now you are supposed to implement the log system. The general logging format is '`Operation #`X`:` $LOGMSG$`.`', where X is the number of the operation and $LOGMSG$ is the logging message.

There are several kinds of operations CLJ may use:

1. `Add` u`:` CLJ opens a new window whose priority is u, and the new window will be the last window in the window queue. This operation will always be successful except the only case in which there is already a window with priority u. If it is successful, $LOGMSG$ will be '`success`'. Otherwise $LOGMSG$ will be '`same priority`'.

2. `Close` u`:` CLJ closes a window whose priority is u. If there exists such a window, the operation will be successful and $LOGMSG$ will be '`close` u `with` c', where u is the priority and c is the number of words CLJ has spoken to this window. Otherwise, $LOGMSG$ will be '`invalid priority`'. Note that ANY window can be closed.

3. `Chat` w`:` CLJ chats with the top window, and he speaks w words. The top window is the first window in the queue, or the "always on top" window (as described below) instead if there exists. If no window is in the queue, $LOGMSG$ will be '`empty`', otherwise the operation can be successful and $LOGMSG$ will be '`success`'.

4. `Rotate` x`:` CLJ performs one or more Alt-Tabs to move the x-th window to the first one in the queue. For example, if there are 4 windows in the queue, whose priorities are 1, 3, 5, 7 respectively and CLJ performs '`Rotate 3`', then the window's priorities in the queue will become 5, 1, 3, 7. Note that if CLJ wants to move the first window to the head, this operation is still considered "successful". If x is out of range (smaller than 1 or larger than the size of the queue), $LOGMSG$ will be '`out of range`'. Otherwise LOGMSG should be '`success`'.

5. `Prior:` CLJ finds out the girl with the maximum priority and then moves the window to the head of the queue. Note that if the girl with the maximum priority is already the first window, this operation is considered successful as well. If the window queue is empty, this operation will fail and $LOGMSG$ must be '`empty`'. If it is successful, $LOGMSG$ must be '`success`'.

6. `Choose` u`:` CLJ chooses the girl with priority u and moves the window to the head of the queue.This operation is considered successful if and only if the window with priority u exists. $LOGMSG$ for the successful cases should be '`success`' and for the other cases should be '`invalid priority`'.

7. `Top` u`:` CLJ makes the window of the girl with priority u always on top. Always on top is a special state, which means whoever the first girl in the queue is, the top one must be u if u is always on top. As you can see, two girls cannot be always on top at the same time, so if one girl is always on top while CLJ wants another always on top, the first will be not always on top any more, except the two girls are the same one. Anyone can be always on top. $LOGMSG$ is the same as that of the Choose operation.

8. `Untop:` CLJ cancels the "always on top" state of the girl who is always on top. That is, the girl who is always on top now is not in this special state any more. This operation will fail unless there is one girl always on top. If it fails, $LOGMSG$ should be '`no such person`', otherwise

should be `success`.

As a gentleman, CLJ will say goodbye to every active window he has ever spoken to at last, "active" here means the window has not been closed so far. The logging format is '`Bye` u`:` c' where u is the priority and c is the number of words he has ever spoken to this window. He will always say good bye to the current top girl if he has spoken to her before he closes it.

Input

The first line contains an integer T ($T \leq 10$), denoting the number of the test cases.

For each test case, the first line contains an integer n ($0 < n \leq 5000$), representing the number of operations. Then follow n operations, one in a line. All the parameters are positive integers below 10^9.

Output

Output all the logging contents.

Hint: This problem description does not relate to any real person in THU.

Sample Input

```
1
18
Prior
Add 1
Chat 1
Add 2
Chat 2
Top 2
Chat 3
Untop
Chat 4
Choose 2
Chat 5
Rotate 2
Chat 4
Close 2
Add 3
Prior
Chat 2
Close 1
```

Sample Output

```
Operation #1: empty.
Operation #2: success.
Operation #3: success.
Operation #4: success.
Operation #5: success.
Operation #6: success.
Operation #7: success.
Operation #8: success.
Operation #9: success.
Operation #10: success.
Operation #11: success.
Operation #12: success.
Operation #13: success.
Operation #14: close 2 with 8.
Operation #15: success.
Operation #16: success.
Operation #17: success.
Operation #18: close 1 with 11.
Bye 3: 2
```

There are n people standing in a line. Each of them has a unique **id** number.

Now the Ragnarok is coming. We should choose 3 people to defend the evil. As a group, the 3 people should be able to communicate. They are able to communicate if and only if their id numbers are pairwise coprime or pairwise not coprime. In other words, if their **id** numbers are a, b, c, then they can communicate if and only if $[(a, b) = (b, c) = (a, c) = 1]$ or $[(a, b) \neq 1$ and $(a, c) \neq 1$ and $(b, c) \neq 1]$, where (x, y) denotes the greatest common divisor of x and y.

We want to know how many 3-people-groups can be chosen from the n people.

Input

The first line contains an integer T ($T \leq 5$), denoting the number of the test cases.

For each test case, the first line contains an integer n ($3 \leq n \leq 10^5$), denoting the number of people. The next line contains n distinct integers a_1, a_2, ..., a_n ($1 \leq a_i \leq 10^5$) separated by a single space, where a_i stands for the id number of the i-th person.

Output

For each test case, output the answer in a line.

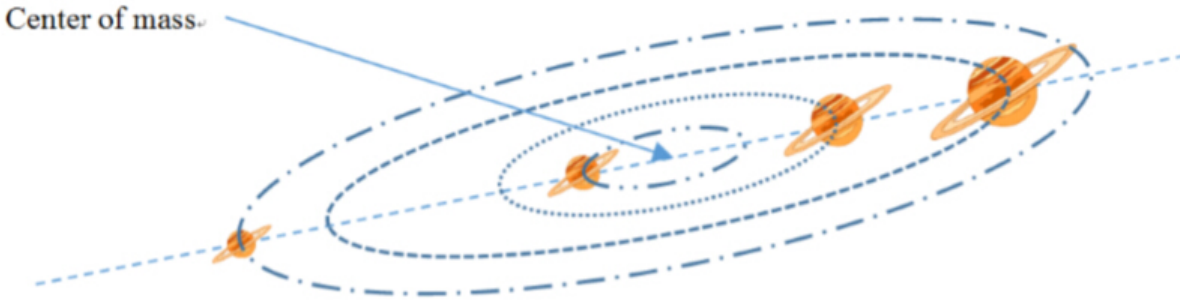
Sample Input

```
1
5
1 3 9 10 2
```

Sample Output

```
4
```

Good news for us: to release the financial pressure, the government started selling galaxies and we can buy them from now on! The first one who bought a galaxy was Tianming Yun and he gave it to Xin Cheng as a present.



To be fashionable, DRD also bought himself a galaxy. He named it Rho Galaxy. There are n stars in Rho Galaxy, and they have the same weight, namely one unit weight, and a negligible volume. They initially lie in a line rotating around their center of mass.

Everything runs well except one thing. DRD thinks that the galaxy rotates too slow. As we know, to increase the angular speed with the same angular momentum, we have to decrease the moment of inertia.

The moment of inertia I of a set of n stars can be calculated with the formula

$$I = \sum_{i=1}^{n} w_i \cdot d_i^2,$$

where w_i is the weight of star i, d_i is the distance form star i to the mass of center.

As DRD's friend, ATM, who bought M78 Galaxy, wants to help him. ATM creates some black holes and white holes so that he can transport stars in a negligible time. After transportation, the n stars will also rotate around their new center of mass. Due to financial pressure, ATM can only transport at most k stars. Since volumes of the stars are negligible, two or more stars can be transported to the same position.

Now, you are supposed to calculate the minimum moment of inertia after transportation.

Input

The first line contains an integer T ($T \leq 10$), denoting the number of the test cases.

For each test case, the first line contains two integers, n ($1 \leq n \leq 50000$) and k ($0 \leq k \leq n$), as mentioned above. The next line contains n integers representing the positions of the stars. The absolute values of positions will be no more than 50000.

Output

For each test case, output one real number in one line representing the minimum moment of inertia. Your answer will be considered correct if and only if its absolute or relative error is less than 1e-9.

Sample Input

```
2
3 2
-1 0 1
4 2
-2 -1 1 2
```

Sample Output

```
0
0.5
```

Hatsune Miku is a popular virtual singer. It is very popular in both Japan and China. Basically it is a computer software that allows you to compose a song on your own using the vocal package.

Today you want to compose a song, which is just a sequence of notes. There are only m different notes provided in the package. And you want to make a song with n notes.

Also, you know that there is a system to evaluate the beautifulness of a song. For each two consecutive notes a and b, if b comes after a, then the beautifulness for these two notes is evaluated as $score(a, b)$.

So the total beautifulness for a song consisting of notes a_1, a_2, ..., a_n, is simply the sum of $score(a_i, a_{i+1})$ for $1 \leq i \leq n-1$.

Now, you find that at some positions, the notes have to be some specific ones, but at other positions you can decide what notes to use. You want to maximize your song's beautifulness. What is the maximum beautifulness you can achieve?

Input

The first line contains an integer T ($T \leq 10$), denoting the number of the test cases.

For each test case, the first line contains two integers n ($1 \leq n \leq 100$) and m ($1 \leq m \leq 50$) as mentioned above. Then m lines follow, each of them consisting of m space-separated integers, the j-th integer in the i-th line for $score(i, j)$ ($0 \leq score(i, j) \leq 100$). The next line contains n integers, a_1, a_2, ..., a_n ($-1 \leq a_i \leq m$, $a_i \neq 0$), where positive integers stand for the notes you cannot change, while negative integers are what you can replace with arbitrary notes. The notes are named from 1 to m.

Output

For each test case, output the answer in one line.

Sample Input

```
2
5 3
83 86 77
15 93 35
86 92 49
3 3 3 1 2
10 5
36 11 68 67 29
82 30 62 23 67
35 29 2 22 58
69 67 93 56 11
42 29 73 21 19
-1 -1 5 -1 4 -1 -1 -1 4 -1
```

Sample Output

```
270
625
```

Squark has a game machine, which has $2n$ slots in a row, numbered from 1 to $2n$, inclusively. He plays a game with the machine for several rounds. In each round, he divides that row into k segments with marks at boundaries between adjacent segments. Each segment must contain an even number of slots. Then, the machine generates a random permutation of $\{1, 2, \ldots, 2n\}$ and displays the permutation on the slots. Finally, the machine calculates the inversion number of each segment and multiplies them together to get the score of the round. The inversion number of a sequence is the number of inversions (also called inversion pairs) in the sequence.

Squark can play the game for as many rounds as he wants, but he must use different partitions in different rounds. Two partitions are considered to be different if there is a mark somewhere in one partition but not in the other. The total game score is simply the sum of the score of each round. However, the machine has been intruded by a malware, which changes the permutation before the machine calculates the score of each round. It picks each segment and sorts the numbers in the slots numbered with even numbers.

For example, if $n = 2$, $k = 1$ and the permutation generated is (2, 4, 1, 3). The malware will pick numbers 4 and 3 (because they are in slots numbered with 2 and 4) and sort them, changing the permutation into (2, 3, 1, 4). So Squark will get a score of 2 (pairs (2, 1) and (3, 1)) in this round. Squark wants to know the maximum expected game score he can get.

Input

The first line contains an integer T ($T \leq 10^4$) denoting the number of the test cases. For each test case, there is one line containing two integers, n ($1 \leq n \leq 2000$) and k ($1 \leq k \leq n$) as mentioned above.

Output

As the answer for the problem can be very large, please calculate it as an irreducible fraction A/B and output $(A \cdot B^{-1}) \bmod (10^9 + 7)$ for each test case in a separate line. Here, B^{-1} is the multiplicative inverse of B *modulo* $10^9 + 7$. The input constraints guarantee that B and $10^9 + 7$ are coprime, so this expression is properly defined.

Hint: The maximum expected game scores in the example are 1/2, 1/4, 13/6 respectively.

Sample Input

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

Sample Output

```
500000004
250000002
166666670
```

Little Bob's computer has 2^n bytes of memory. For convenience, n-bit integers 0 to $2^n - 1$ are used to index these bytes.

Now he wants to assign a value to each byte of the memory. In this problem, a byte is composed of m bits. Therefore he can only assign 0 to $2^m - 1$ (inclusive) to a single byte.

Bob has some preferences on which value to be assigned to each byte. For the byte indexed by i, if it is assigned with value j ($0 \leq j < 2^m$), the preference score for it is $w_{i,j}$.

In addition, each byte has a threshold value. For two different bytes indexed by a and b, if the following two conditions are satisfied, there will be an additional score (u_a xor u_b):

1. a and b only have one bit of difference in their binary forms;
2. The assigned value of byte a is not less than its threshold value, or the assigned value of byte b is not less than its threshold value.

The total score of an assignment solution is the sum of the preference scores of all bytes plus the sum of all additional scores.

Bob wants to find an assignment solution with the maximum total score. If there are multiple solutions, you can output any of them.

Input

The first line contains an integer T ($T \leq 3$), denoting the number of the test cases.

For each test case, the first line contains two integers, n and m ($1 \leq n, m \leq 8$), as mentioned above. The second line contains 2^n integers, and the i-th integer is the threshold value for byte i. The threshold values are between 0 and $2^m - 1$, inclusively. The third line contains 2^n integers, and the i-th integer is u_i ($0 \leq u_i < 1024$). The next 2^n lines give all preference scores. Each line contains 2^m integers, and the j-th integer of the i-th line is $w_{i,j}$ ($-1024 \leq w_{i_j} < 1024$).

Output

For each test case, output one line consisting of 2^n integers between 0 and $2^m - 1$, and the i-th integer is the value assigned to byte i in the assignment solution with the maximum total score.

Sample Input

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

Sample Output

```
2 2 3 0 3 1 0 3
```

Xiaoqiang entered the "shortest code" challenge organized by some self-claimed astrologists. He was given a boolean function taking n inputs (in C++):

```
bool f(bool x1, bool x2, bool x3){
//your code goes here
//return something
}
```

x_1	x_2	x_3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

All possible inputs and expected outputs of this function have been revealed (see table on the right):

Xiaoqiang's code must be like:

`bool` a `= NAND(`b`,`c`);`

where "a" is a newly defined variable, "b" and "c" can be a constant (0/1) or a function parameter (x1/x2/x3) or a previously defined variable. `NAND` is the "not-and" function:

`NAND(`b`,`c`)=!(`b`&&`c`)`

Because `NAND` is universal, Xiaoqiang knew that he could implement any boolean function he liked. Also, at the end of the code there should be a return statement:

`return` y`;`

where y can be a constant or a function parameter or a previously defined variable. After staring at the function for a while, Xiaoqiang came up with the answer:

```
bool a = NAND(x1, x2);
bool b = NAND(x2, x3);
bool y = NAND(a,b); return y;
```

Xiaoqiang wants to make sure that his solution is the shortest possible. Can you help him?

Input

The first line contains an integer T ($T \leq 20$) denoting the number of the test cases.

For each test case, there is one line containing 8 characters encoding the truth table of the function.

Output

For each test case, output a single line containing the minimum number of lines Xiaoqiang has to write.

Sample Input

```
1
00010011
```

Sample Output

```
4
```

Osu! is a very popular music game. Basically, it is a game about clicking. Some points will appear on the screen at some time, and you have to click them at a correct time.

Now, you want to write an algorithm to estimate how diffecult a game is.

To simplify the things, in a game consisting of N points, point i will occur at time t_i at place (x_i, y_i), and you should click it exactly at t_i at (x_i, y_i). That means you should move your cursor from point i to point $i+1$. This movement is called a jump, and the difficulty of a jump is just the distance between point i and point $i+1$ divided by the time between t_i and t_{i+1}. And the difficulty of a game is simply the difficulty of the most difficult jump in the game.

Now, given a description of a game, please calculate its difficulty.

Input

The first line contains an integer T ($T \leq 10$), denoting the number of the test cases.

For each test case, the first line contains an integer N ($2 \leq N \leq 1000$) denoting the number of the points in the game. Then N lines follow, the i-th line consisting of 3 space-separated integers, t_i ($0 \leq t_i < t_{i+1} \leq 10^6$), x_i, and y_i ($0 \leq x_i, y_i \leq 10^6$) as mentioned above.

Output

For each test case, output the answer in one line.

Your answer will be considered correct if and only if its absolute or relative error is less than 1e-9.

Hint: In memory of the best osu! player ever Cookiezi.

Sample Input

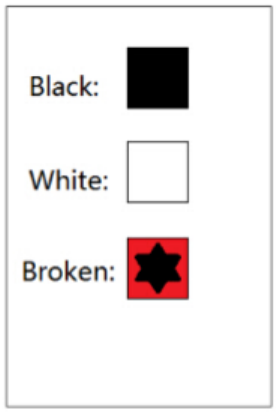
```
2
5
2 1 9
3 7 2
5 9 0
6 6 3
7 6 0
10
11 35 67
23 2 29
29 58 22
30 67 69
36 56 93
62 42 11
67 73 29
68 19 21
72 37 84
82 24 98
```

Sample Output

```
9.2195444573
54.5893762558
```

Nothing is more beautiful than square! So, given a grid of cells, each cell being black or white, it is reasonable to evaluate this grid's beautifulness by the side length of its maximum continuous subsquare which fully consists of white cells.

Now you're given an $N \times N$ grid, and the cells are all black. You can paint some cells white. But other cells are broken in the sense that they cannot be paint white. For each integer i between 0 and N inclusive, you want to find the number of different painting schemes such that the beautifulness is exactly i. Two painting schemes are considered different if and only if some cells have different colors. Painting nothing is considered to be a scheme.



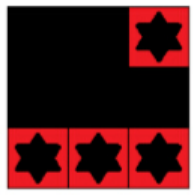


Fig. J(a)

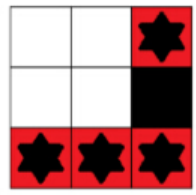


Fig. J(b)

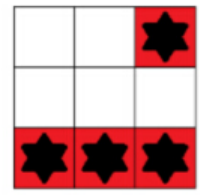


Fig. J(c)

For example, $N = 3$ and there are 4 broken cells as shouwn in Fig. J(a). There are 2 painting schemes for $i = 2$ as shown in Fig. J(b) and J(c).

You just need to output the answer *modulo* $10^9 + 7$.

Input

The first line contains an integer T ($T \leq 10$) denoting the number of the test cases.

For each test case, the first line contains an integer N ($1 \leq N \leq 8$), denoting the size of the grid is $N \times N$. Then N lines follow, each line containing an N-character string of '`o`' and '`*`', where '`o`' stands for a paintable cell and '`*`' for a broken cell.

Output

For each test case, for each integer i between 0 and N inclusive, output the answer in a single line.

Sample Input

```
2
3
oo*
ooo
***
8
oooooooo
oooooooo
oooooooo
oooooooo
oooooooo
oooooooo
oooooooo
oooooooo
```

Sample Output

```
1
29
2
0
1
401415247
525424814
78647876
661184312
550223786
365317939
130046
1
```

A toy is made up of N vertices and M undirected edges in the 2D plane. As usual, you want to know how many ways there are to color the vertices of the toy. You have totally C colors. And of course, to make things fun, you think that if one color configuration can be rotated to get another, these two configurations should be considered the same. Rotation means 2D in-plane rotation and reflection is not considered as rotation.

For instance, consider coloring the toy on the right with 2 colors. The coordinates of the vertices are:

1. (0,0)
2. (1,0)
3. (0,1)
4. (-1,0)
5. (0,-1)

The toy has 6 edges: (1,2), (1,3), (2,3), (3,4), (4,5), (5,2).

As a 2D being, this toy has no symmetry. So there are 32 ways to color it. Had the first two edges been removed, there would be only 12 different ways.

You should output the answer *modulo* $10^9 + 7$.

Input

The first line contains an integer T ($T \leq 20$) denoting the number of the test cases.

Each test case begins with three positive integers N ($1 \leq N \leq 50$), M ($0 \leq M \leq N(N-1)/2$) and C ($1 \leq C \leq 100$).

Then follow N lines. Each line contains 2 integers in range $[-10000, 10000]$ describing a vertex.

Then follow M lines. Each line contains 2 integers in range $[1, N]$ representing an edge. There are neither duplicate edges nor self-loops.

Output

For each test case, output one line containing the answer.

Sample Input

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

Sample Output

```
32
12
```

Little Bob studied Trie in Data Structure class. He learned that Trie is a rooted tree to store some strings. There is a character on each edge of a Trie. (In this problem, the valid characters are lowercase Latin letters.)

Bob draws a Trie with n nodes, and the nodes are numbered with integers from 1 to n. The number of the root node is always 1.

He uses S_i to denote the string concatenated with characters on the path from the root to node i. It can be seen that S_1 is the empty string. Furthermore, for a set $P \subseteq \{1, 2, \ldots, n\}$, $S_P = \{S_i | i \in P\}$.

Bob defines a kind of trait on this Trie. The trait can be described with a set $P \subseteq \{1, 2, \ldots, n\}$. We say a string a has this trait, if and only if there exists a string $b \in S_P$ such that a ends with b. Here string a ends with string b means the last $|b|$ characters of a is exactly b. (Note that every string ends with the empty string).

Bob also defines a mapping f on $P \subseteq \{1, 2, \ldots, n\}$, and $f(P)$ is also a subset of $\{1, 2, \ldots, n\}$. $i \in f(P)$ if and only if there exists $j \in P$ such that S_j ends with S_i.

Initially, there is no traits been defined. Bob will add traits whenever he wants. Let D_i denote the number of traits that S_i has until now. Note that D_i may change when a new trait is added.

Besides adding new traits, Bob may also ask you something about his Trie. In each query, Bob will give you a set $P \subseteq \{1, 2, \ldots, n\}$, and you need to compute

$$\sum_{i \in f(P)} D_i.$$

Are you willing to do this for Bob?

Input

The first line contains an integer T ($T \leq 5$), denoting the number of the test cases.

For each test case, the first line contains an integer n ($1 \leq n \leq 10^5$).

From the second to the n-th line, this $n - 1$ lines describe Bob' s Trie. The i-th line contains an integer u ($u < i$), and a lowercase letter c, which means that the father of node i is node u and the character on that edge is c. We ensure that for each node i, letters on edges connecting i and its children are all different.

The next line contains an integer m ($m \leq 10^5$), which means there are m operations.

The next m lines describe all operations. In each line, the first integer indicates the type of this operation. 1 means Bob will add a new trait and 2 means Bob is asking you a question. The second integer k is the size of set P, and the next k integers are the elements of P. We ensure that these k integers are different, and they are all between 1 and n. The total size of the m sets will not be larger than $5 \cdot 10^5$.

Output

For each test case, output the answer for each query operation, one answer in a line.

Sample Input

```
1
6
1 a
1 b
1 c
3 a
3 b
5
1 2 3 4
2 2 5 6
1 2 2 3
2 2 4 5
2 1 6
```

Sample Output

```
2
3
4
```