Problem QA
Counterfeit Money

Time limit: 3 seconds

Memory limit: 1024 megabytes

Problem Description

The banknotes of the ICPC Kingdom have anti-counterfeiting measures. Each banknote has an exclusive serial number, and this serial number is divisible by 13. In other words, if the serial number is not divisible by 13, then the banknote is counterfeit. To verify whether a number is divisible by 13, we can directly divide the number by 13. Yet, there is another method:

Partition the digits of the given decimal number into groups starting from the right, where each group has three digits. Now, treat each group as a three-digit number. Then, from the rightmost group, apply subtraction and addition operations alternately to the three-digit number and obtain the result. If the result is divisible by 13, then the original number is divisible by 13. Otherwise, it is not.

For example, for the number 123,456,789, if we apply subtraction and addition operations alternately from the rightmost group of 3 digits, we get $789 - 456 + 123 = 456$. As 456 is not divisible by 13, the original number 123,456,789 is not divisible by 13.

For another example, for the number 593,825,856, if we apply subtraction and addition operations alternately from the rightmost group of 3 digits, we get $856 - 825 + 593 = 624$. As 624 is divisible by 13 ($624 = 13 \times 48$), the original number 593,825,856 is divisible by 13.

Based on the above method, write a program to verify whether a banknote is counterfeit or not.

Input Format

The input contains several test cases. The first line stands for the number of test cases t. The next t lines will each contain a positive number. The given number may contain up to 1000 digits.

Output Format

For each input number, output the absolute value of the result when we apply the above alternate-add-subtract method. Then, on the same line, output "`YES`" if the input number is divisible by 13, and "`NO`" otherwise. There is a space between the output value and `YES`/`NO`.

Technical Specification

- $1 \le t \le 1000$.
- Each input number may contain up to 1000 digits.

Sample Input 1

```
2
123456789
593825856
```

Sample Output 1

```
456 NO
624 YES
```

Hint

- string and simulation
- Partition the given number into sets starting from the right, each group has three digits. We have the following two methods:
  - From the rightmost group of 3 digits apply the subtraction and addition operations alternatively and find the result. If the result is either a 0 or it can be divisible by 13 completely without leaving a remainder, then the number is divisible by 13 (simulate the statement of problem).
  - Numbered the group from the right. Let S_{odd} is the sum of groups with numbered odd, and S_{even} is the sum of groups with numbered even. If $|S_{odd} - S_{even}|$ is either a 0 or it can be divisible by 13 completely without leaving a remainder, then the number is divisible by 13.

Problem QB
Recurring Decimal to Fractions

Time limit: 3 seconds

Memory limit: 1024 megabytes

Problem Description

Given two strings of numbers representing a fraction smaller than one in recurring decimal form. The first string s_1 indicates the non-repeating part after the decimal point of the recurring decimal and the second string s_2 indicates the repeating part of the recurring decimal such as

```
1
012
```
means$0.1\overline{012}$

Return two integers n, d represent the fraction in the form of numerator and denominator. The two integers should be relatively prime.

Input Format

The first line contains an integer $T (\leq 40)$, representing the number of test cases. Each test case below contains two lines. For each test case, the first line has two integers a and b separated by a space. The second line is a string s_1 with length a and the third line is a string s_2 with length b.

Output Format

Each test case outputs two integers n and d, separated by a space. The first integer n is the numerator and the second integer d is the denominator. It is necessary to simplify the fraction so that the numerator and denominator are relatively prime.

Technical Specification

- $1 \leq a$
- $1 \leq b$
- $1 \leq a + b \leq 10$

Sample Input 1

```
2
1 3
0
012
3 1
085
3
```

Sample Output 1

```
2 1665
32 375
```

Hint

- Euclidean algorithm

Problem QC
Where the Lantern Lights are Dimming

Time limit: 3 seconds

Memory limit: 1024 megabytes

Problem Description

The Lantern Festival features many lanterns on display. In the darkness of the night, they cast beautiful shadows and reflections, attracting numerous visitors to come and go. With so many lanterns on display simultaneously, it's impossible to showcase the unique features of each. As such, the organizers switch on the lights of some lanterns while turning off others in rotation. At any given time, some lanterns are illuminated while others rest. Additionally, some lanterns remain perpetually off due to malfunctions, missing their chance to dazzle.

To engage the visitors in the Lantern Festival, the organizers also hold a scoring event. If a visitor is **very satisfied** with the festival, they will receive a pack of stickers worth 3 points each, and they will affix one 3-point sticker to each lantern on display. If they feel **satisfied**, they will receive a pack of 1-point stickers and place one on each displaying lantern. If they are **disappointed** with the festival, they will get a pack of -2 point stickers and apply one to each displaying lantern. In other words, those lanterns that are resting and not illuminated won't have an opportunity to receive any stickers from this visitor. After the festival ends, please help write a program to calculate the total points from all the stickers on the lanterns.

Input Format

The input begins with a line containing two integers, n and m. The next n lines describe the initial state of n lanterns, numbered from 0 to $n-1$. Each of these lines contains two integers: s_i and p_i. s_i represents the state of the lantern:

- 1 if the i-th lantern is initially on,
- 0 if it is off,
- -1 if it is out of order (meaning it is perpetually off and cannot be turned on)

p_i indicates the total points from stickers that are already on the i-th lantern.

The following m lines represent m events given in chronological order, each corresponding to one of two even types: switching or scoring.

- Lines beginning with the letter `W` signify a switching event. They have two subsequent integers, l_j and r_j, which imply that the state of lanterns numbered in the range $[l_j, r_j]$ (inclusive) will be toggled.

- Lines beginning with the letter `C` denote a scoring event by a visitor. These lines have a single subsequent integer, $q_j \in \{-2, 1, 3\}$, indicating the sticker score assigned by the visitor. Every lantern currently on display receives a sticker with q_j points from the visitor.

Output Format

Output a single integer that is the total points from all the lanterns after the festival.

Technical Specification

- $1 \leq n \leq 1,000,000$
- $1 \leq m \leq 1,000,000$
- $s_i \in \{-1, 0, 1\}$
- $-10,000 \leq p_i \leq 10,000$
- $0 \leq l_j \leq r_j < n$
- $q_j \in \{-2, 1, 3\}$

Sample Input 1

```
3 3
0 0
0 0
0 0
W 0 2
W 1 1
C 3
```

Sample Output 1

```
6
```

Sample Input 2

```
5 5
1 5
0 0
-1 2
1 0
0 -2
C 1
W 0 4
C -2
W 1 3
C 3
```

Sample Output 2

```
9
```

Hint

- Maintaining an integer y that is the total points. Initially, $y = \sum p_i$.
- Skip all the out-of-order lanterns and build an array A for normal lanterns only.
- Maintaining an integer x that represents the number of displaying lanterns.
- In each scoring event, y is increased by $q_j \times x$
- The states of the n lanterns are maintained in a segment tree with lazy propogation for state flipping.
- For a given input range $[l_j, r_j]$, find the exact $[l'_j, r'_j]$ indexes from A by using binary search to skip the out-of-order lanterns. Then, perform a range update within the new range.

The complexity of each scoring event is $O(1)$, and the complexity of each switching event is $O(\log n)$. The overall complexity is $O(m \log n)$.

Almost blank page

Problem QD
Quarantine Policy

Time limit: 3 seconds
Memory limit: 1024 megabytes

Problem Description

The 2019 novel coronavirus, COVID-19, can be transmitted between humans through water droplets and close contact. The transmission is especially easy and fast in relatively crowded or confined spaces, such as airplanes or trains. If someone is infected with COVID-19, then passengers occupying the adjacent seats will be infected easily.

To prevent the spread of the virus, we can take precautions, such as washing hands regularly and avoiding touching our eyes, nose, or mouth, to avoid infection. In addition, governments have also implemented special measures such as isolation and quarantine for this purpose. For instance, when someone on an airplane caught the coronavirus, the person will need to be isolated. Moreover, persons occupying the seats adjacent to the infected person will need to be quarantined. Precisely, there are two types of adjacent seats. One is directly adjacent, that is the seat is in the front, rear, left, or right of the virus seat. The other one is diagonally adjacent, that is the seat is in the front-left, front-right, rear-left, or rear-right of the virus seat. In the quarantine policy, someone whose seat is directly adjacent will be quarantined for d_1 days, and someone whose seat is diagonally adjacent will be quarantined for d_2 days. If there is more than one infected person adjacent to some seat, the number of days of quarantine will not be accumulated.

Please write a program to output which seats whose occupying persons need to be quarantined, and the number of days of quarantine. If a seat whose occupying person needs to be quarantined for different days, output the maximum of such days.

Input Format

The input contains several test cases. The first line stands for the number of test cases t. For each test case, the first line contains four integers n, m, d_1, d_2 ($0 < n, m \leq 100$, $1 \leq d_2 \leq d_1 < 10$), which stands for that there are n lines and m columns of the airplane, and a seat will be quarantined d_1 days if the seat is adjacent to the virus seat directly (i.e., front, rear, left, right), and a seat will be quarantined d_2 days if it is adjacent to the virus seat in the diagonal directions (front-left, front-right, rear-left, rear-right). The next n lines contain exactly m characters and represent the seats on the airplane.

Each healthy seat is represented by a '.' character and each virus seat is represented by a 'V' character.

Output Format

For each airplane, first print the following message in a line alone:

Airplane #z:

where z stands for the label of the airplane (starting with 1). The next n lines replace each '.' character in the input seats by the corresponding number of days to quarantine for that seat.

Technical Specification

- $1 \leq t \leq 1000$.
- $0 < n, m \leq 100$.
- $1 \leq d_2 \leq d_1 < 10$.

Sample Input 1

```
2
4 4 7 3
.V..
....
..V.
....
2 2 1 1
V.
..
```

Sample Output 1

```
Airplane #1:
7V70
3773
07V7
0373
Airplane #2:
V1
11
```

Hint

- simulation
- Similar as UVA10189 (Minesweeper): counts the total mines adjacent to a square.
- The differences are (1)There are two types of adjacent seats: adjacent directly and diagonal direction. (2) If there is more than one confirmed case adjacent to someone on the same flight, the number of days of quarantine will not be accumulated.

Problem QE
Slabstones Rearrangement

Time limit: 3 seconds

Memory limit: 1024 megabytes

Problem Description

Babara has a garden. She has bought some rectangular slabstones and worked out an initial placement of all slabstones in the garden. The shape of the garden is rectangular. An edge of a slabstone should be either parallel or orthogonal to an edge of the garden. There exists a slabstone touching the left, right, bottom, and top edges of the garden, respectively. All the slabstones are contained in the garden. Meanwhile, none of the slabstones overlap in the initial placement. Babara enjoys stepping slabstones from one to another every day. However, Babara would like to redesign her garden to make room for some other purposes. She is wondering how tight the slabstones can be packed together if they can only be shifted horizontally (i.e., left or right) without changing their vertical coordinates. Furthermore, if the vertical dimensions of any two slabstones overlap (not including touching of their ends), their relative locations in the horizontal direction should be maintained. That is, if the vertical dimensions of slabstones R and Q overlap and, before shifting, slabstone Q is on the right of slabstone R, Q should be still on the right of R after shifting or vice versa. Besides, there is a minimal horizontal spacing between two slabstones during slabstone rearrangement if their vertical dimensions overlap. The slabstones should remain non-overlapping after shifting. Nevertheless, their horizontal edges may touch. Now you are asked to help Babara calculate the largest area that can be spared for other purposes.

Input Format

The first line holds an integer specifying the number of test cases. It is then followed by the input data of the test cases. The first line of the input for each test case gives two integers. The first one specifies the number of rectangular slabstones whereas the second one gives the minimal horizontal spacing between two slabstones. Then, each of the following lines contains four integers. The first two integers specify the initial x and y coordinates of the bottom-left corner of a slabstone in the garden. The remaining two integers specify the initial x and y coordinates of the top-right corner of a slabstone. Two adjacent numbers are separated by a whitespace.

Output Format

The output of a test case takes a line. It contains the largest area saved by shifting the slabstones. If no area can be saved, just output zero.

Technical Specification

- The number of test cases is not more than 32.
- All the coordinates are 32-bit unsigned integers.
- A garden's area is not larger than the maximal 32-bit unsigned integer.
- The width and length of a slabstone are 32-bit unsigned integers. They should be larger than zero.
- The minimal horizontal spacing between any two slabstones is a 32-bit unsigned integer and should be greater than zero.
- The number of slabstones is from 4 to 100.

Sample Input 1

```
2
4 2
2 6 4 12
8 4 16 8
7 10 11 16
18 4 20 18
6 4
2 5 4 11
2 14 6 17
7 10 10 16
9 4 16 7
11 11 16 14
18 4 20 18
```

Sample Output 1

```
28
0
```

Hint

- Longest path on DAC for geometric objects
- The relative horizontal positions of rectangles (i.e., slabstones) need to be converted into a directed acyclic graph where a rectangle is treated as a vertex and the overlapping of two rectangle's vertical dimensions has to be modeled as an edge. Hence, a directed edge from rectangle X to rectangle Y means that rectangle X is positioned relaively to the left of rectangle Y and their vertical dimensions overlap. Associated with each vertex is the width of the underlying rectangle whereas associated with each edge is the minimal spacing between two rectangles. Once an drected acyclic graph is ready, a longest path algorithm can be aplied to find out the required X dimension of a garden. As a result, we can obtain the largest area that can be saved.

Problem QF
Baker's Dilemma

Time limit: 3 seconds

Memory limit: 1024 megabytes

Problem Description

A baker has N bakery orders from customers that he must fulfill, but he can only handle one order a day. For the i^{th} order, the baker needs to spend D_i $(1 \leq D_i \leq 1000)$ consecutive days to complete it; however, for every day of delay, the baker must be fined S_i $(1 \leq S_i \leq 10000)$. For example, if the baker receives four orders to make biscuits, the number of days required for each order is 3, 1, 2, 5, and the penalty for each day of delay is 4, 1000, 2, 5. If the baker's work order is 1 2 3 4, the penalty will be $4 \times 0 + 1000 \times 3 + 2 \times 4 + 5 \times 6 = 3038$, but if the work order is 2 1 3 4, the penalty will be $1000 \times 0 + 4 \times 1 + 2 \times 4 + 5 \times 6$. $0 + 4 \times 1 + 2 \times 4 + 5 \times 6 = 42$, so the latter penalty is less. Please write a program to help the baker to find out the sequence of work which has the least penalty.

Input Format

The first line of the input has a positive integer T representing the number of groups of data. Then, there are T groups of data. For each group, the first line has an integer N between 1 and 1000 representing the number of orders, followed by N lines, each with two integers separated by a space character, representing the number of days required for each order, D, and the penalty, S, for each day of delay, in that order.

Output Format

For each set of data, output the sequence of jobs with the smallest penalty on one line. Each job is represented by its number, separated by a blank character. If there is more than one set of answers, print the one with the smallest dictionary order. Note that each group of jobs is numbered starting with 1.

Technical Specification

- $1 \leq T \leq 1000$.
- $1 \leq N \leq 1000$.
- $1 \leq D_i \leq 1000, \forall 1 \leq i \leq N$.
- $1 \leq S_i \leq 10000, \forall 1 \leq i \leq N$.

Sample Input 1

```
2
4
3 4
1 1000
2 2
5 5
5
3 4
1 1000
8 8
2 2
5 6
```

Sample Output 1

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

Hint

First, we assume that, in a group, all orders are received at the same time (say time 0). The different permutation will have different penalty. We can sort these orders according to the product of finished time and penalty, that is, employing greedy policy.

Problem QG
Obtuse Triangle

Time limit: 3 seconds
Memory limit: 1024 megabytes

Problem Description

Given three integers $a \geq b \geq c$ representing the three edges length of a triangle. If the square of one edge length is larger than the sum of the other two squares of edge length, $a^2 > b^2 + c^2$ then return 1. Otherwise, return 0.

Input Format

The first line contains an integer $T(\leq 40)$, representing the number of test cases. Each test case below contains contains three lines. For each test case, there are three integers a b c separated by a space.

Output Format

Each test case outputs one integer s. $s = 1$ means the test case is an Obtuse Triangle. $s = 0$ means the test case is not an Obtuse Triangle.

Technical Specification

- $1000 \geq a \geq b \geq c \geq 1$.
- $b + c > a$

Sample Input 1

```
3
1 1 1
3 2 2
298 234 157
```

Sample Output 1

```
0
1
1
```

Sample Input 2

```
2
5 4 3
30 23 20
```

Sample Output 2

```
0
0
```

Hint

- basic calculation

Almost blank page

Problem QH
Mysterious Triangles

Time limit: 8 seconds
Memory limit: 1024 megabytes

Problem Description

Dr. H is the chief technology officer of the International Collaboration Probing Company. He is also a well-mathematician and archaeologist. He followed the expedition deep into central Iran and found suspected Persian cultural relics in caves. We know that ancient Persian mathematics was very well developed. In the cave, Dr. H discovered some mysterious triangles, which were inscribed on the walls of the cave in order from small to large as shown in Fig. 1. Based on Dr. H's mathematical background, he reasoned and found that the composition of these mysterious triangles has a special regularity. If we call the top row the zeroth row, followed by the first row, the second row, and so on, then the kth row of a n-dimensional mysterious triangle contains the following $k+1$ numbers: $C(k,0)$, $C(k,1)$, $C(k,2)$,...,$C(k,k)$, where $C(k,r) = \frac{k!}{r!(k-r)!}$ for $0 \leq r \leq k$. Since $C(k,0) = C(k,k) = 1$, the left and right ends of each row must be 1. Moreover, if we define that $C(k,r) = 0$ when $r < 0$ or $r > k$, then every number in this mysterious triangle except the 1 in the zeroth column satisfies the following recursive relationship:

$$C(k,r) = C(k-1,r-1) + C(k-1,r)$$

In order to discover more secrets in this cave, given a non-negative integer k, Dr. H wants to know how many even numbers are contained from the 0th row to the kth row of an n-dimensional mysterious triangle. For example, given $k = 4$ in the 5-dimensional mysterious triangle, the number of all even integers from the 0th row to the 4th row equals 4, because there is an even integer 2 in the second row, and even integers $4, 6, 4$ in the fourth row. Given a non-negative integer n and k, your task is to write a computer program to calculate the number of even integers contained from the 0th row to the kth row of an n-dimensional mysterious triangle. Note that if $k > n$, the output is the number of even integers contained from the 0th row to the nth row of an n-dimensional mysterious triangle. On the other hand, if the result is larger than or equal to $10^9 + 7$, you should output the value modulo $10^9 + 7$, that is, the remainder obtained using the actual value divided by $10^9 + 7$.

Input Format

The first line of the input file contains an integer L ($L \leq 20$) that indicates the number of test cases as follows. For each test case, the first line contains two integers (separated by whitespaces) representing n and k, respectively.

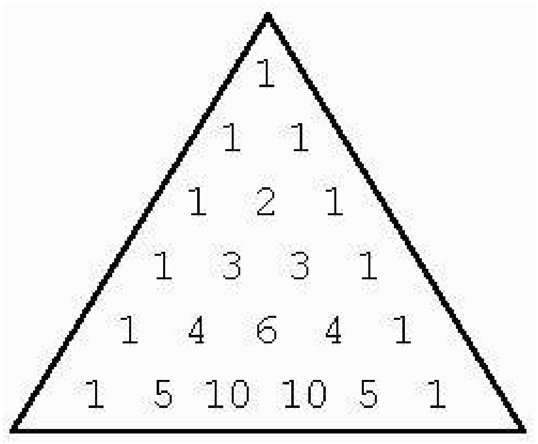

Figure 1: 5-Dimensional mysterious triangle

Output Format

The output contains one line for each test case. Each line contains one non-negative integer representing the sum of all the odd numbers contained in the kth row of an n-dimensional mysterious triangle. Note that if the result is larger than or equal to 10^9+7, you should output the value modulo 10^9+7, that is, the remainder obtained using the actual value divided by 10^9+7.

Technical Specification

- $L \leq 20$.
- $1 \leq n \leq 10^8$.
- $0 \leq k \leq 10^8$ for each test case.

Sample Input 1

```
1
5 4
```

Sample Output 1

```
4
```

Hint

The zeroth column of Pascal's triangle has an odd number, the first column and the second column each have two odd numbers, and the third column has four odd numbers; if we look down from the top of the triangle row by row, the odd numbers are 1, 2, 2, 4, 2, 4, 4, 8, 2, 4, 4, ..., they are all integer powers of 2; and the even numbers are $0, 0, 1, 0, 3, 2, 3, 0, 7, 6, 7, \ldots$. The kth row of Pascal's triangle contains the following $k+1$ numbers:$C(k,0), C(k,1), C(k,2), \ldots, C(k,r)$, where $C(k,r) = \frac{k!}{r!(k-r)!}$, where $0 \leq r \leq k$. Since $C(k,0) = C(k,k) = 1$, the left and right ends of each row must be 1. The following result is a key point: When k is even and r is odd, $C(k,r)$ must be even. When k is odd or r is even, $C(k,r)$ is even if and only if $C(\lfloor\frac{k}{2}\rfloor, \lfloor\frac{r}{2}\rfloor)$ is even. Therefore, one can use the above result to come out a recursive algorithm for solving the problem.

Almost blank page

Problem QI
Statistics

Time limit: 3 seconds

Memory limit: 1024 megabytes

Problem Description

The result of an objective questionnaire is made up of characters 'Y' and 'N', e.g. "YYNNYNNYYY" represents the result of 10 questions, with the result of each question being represented by one character. The result of each question is represented by a character. A 'Y' indicates that the answer to the question is agree, while an 'N' indicates that the answer is disagree. Since the questions in the questionnaire are specially designed, the score for each correct answer is the sum of the number of questions with which it agrees and the previous ones in a row, e.g., the score for the 10th question is 3 because it agrees with the previous two questions in a row (totaling 3 questions). Therefore, the score for "YYNNYNNYYY" is 10 (= 1+2+0+0+1+0+0+1+2+3). Write a program to calculate the score of the questionnaire result.

Input Format

The first line of the input is an integer T, which means there are T questionnaire results. This is followed by T strings of 'Y' and 'N', each of which has a length greater than 0 and less than 80, indicating the result of each questionnaire. The input guarantees that there are no blank characters between 'Y' and 'N'.

Output Format

For each questionnaire result, print its score on one line.

Technical Specification

- $0 \leq T \leq 100$.

Sample Input 1

```
5
YYNNYNNYYY
YYNNYYNNYY
YNYNYNYNYNYNYN
YYYYYYYYYY
YYYYNYYYYNYYYYN
```

Sample Output 1

```
10
9
7
55
30
```

Hint

Read a line of string at a time, each character for a problem. For each problem, by definition, count the score. Finally, output the total of all the problem scores.

Problem QJ
Lead Time Estimation

Time limit: 1 second

Memory limit: 1024 megabytes

Problem Description

The lead time is critical for earning orders, and several issues would result in various lead times when preparing the products. For example, the processes switched between different working areas and the workload of each process. The production manager should accurately estimate the lead time for the target products when the order inquiry is received. Please develop a program to help the product manager estimate the lead time.

The lead time of an inquiry stands for the production time, including several jobs such as material preparation, manufacturing, quality checking, shipping, etc. The processing time of each position could be determined by an execution time while transferring the process from one job to the next requires a transmission time. For an inquiry, the production manager has three pieces of information about the target products.

- The number of jobs and transmissions, e.g. $|T^j|$ and $|T^t|$,
- the processing time of each job, where $T^j = \{t^j_0, t^j_1, t^j_2, \ldots, t^j_{|T^j|-1}\}$, and
- the transmission time between jobs, where $T^t = \{t^t_0, t^t_1, t^t_2, \ldots, t^t_{|T^t|-1}\}$.

The process may begin or end with several starting jobs. This situation could easily insert virtual starting and finishing jobs to simplify the problem. We can assume that the inquiry processes include one starting job and one finishing job.

Input Format

The input includes three parts: (1) the number of jobs and transmissions in the first row, (2) the job processing time in the second row, and (3) the transmission time in the remaining rows. The first input row consists of $|T^j|$ and $|T^t|$ with a space for the separation. The second input row should be T^j, and the comma separates each element in T^j. The transmission time information is revealed from row number three to $(2 + |T^t|)$. Each row of transmission time information involves three data: the source job, the destination job, and the transmission time.

Moreover, the test cases have some restrictions.

- There may be one transmission time for any pair of jobs at most.
- There may be multiple cases in a file.

Output Format

The total time of delivering the products that is denoted by z should be provided, and that means all jobs should be done in z. Also, the production manager is interested in the manufacturing process. If there is exactly one processing path that dominates the lead time, please output the job sequence of the processing path and the letter "M" in upper case otherwise. A comma separates each element in the output sequence. In other words, the output sequence should be like either "z,v_1,v_2,..." or "z,M", where v_1 and v_2 represent the jobs.

Technical Specification

Each inquiry includes precisely one entry and one exit. In each inquiry, at least one path will dominate the lead time, indicating no cyclical manufacturing processes. The boundaries of each variable are listed as follows.

- $2 \leq |T^j| \leq 50$.
- $1 \leq |T^t| \leq 100$.
- $1 \leq t^j_x, t^t_y \leq 50$.

Sample Input 1

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

Sample Output 1

```
41,0,1,3,7
```

Sample Input 2

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

Sample Output 2

```
53,0,1,4,5
53,M
```

Hint

- The lead time of first case is 53. There is only one processing path that dominates the lead time, e.g $0 \rightarrow 1 \rightarrow 4 \rightarrow 5$, so the program outputs the sequence "53,0,1,4,5".

- The lead time of second case is also 53, but there are two dominated processing paths: $0 \rightarrow 1 \rightarrow 3 \rightarrow 5$ and $0 \rightarrow 1 \rightarrow 4 \rightarrow 5$. Therefore, the output sequence is “53,M”.

Problem QK
Chemical Storage

Time limit: 1 second

Memory limit: 1024 megabytes

Problem Description

International Chemical Producing Company (ICPC) is an international company that manufactures various chemicals. The company built several chemical rooms for storing chemicals. They made a short railroad to connect two chemical rooms for convenience to move the chemicals. The network of the chemical rooms and the railroads form a special tree graph in which all the nodes are within distance 2 of a central path. We label each node a number sequentially from 1. Fig. 2 gives an example of networks.

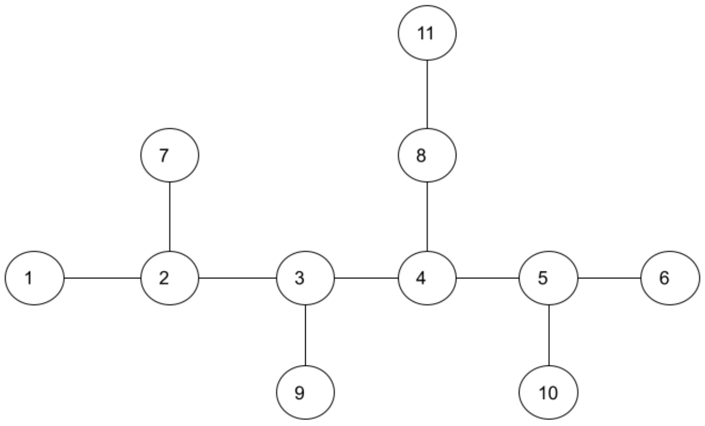

Figure 2: An example of networks with chemical rooms and railroads.

Each chemical product will be stored in a tank placed in a chemical room. Since the chemicals may leak into the air, the safety rule is that the chemicals cannot be placed in two adjacent rooms to avoid adverse chemical reactions between the chemicals. Fig. 3 gives two chemical placement network examples: (a) is safety, and (b) is unsafety.

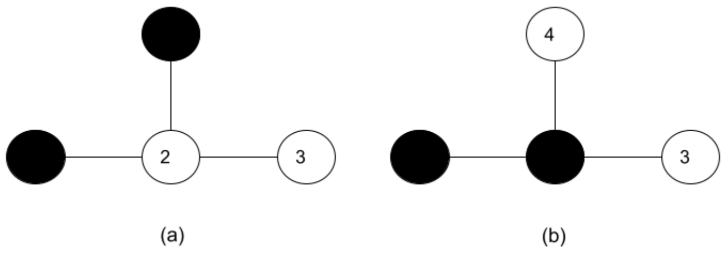

Figure 3: Two examples of chemical placement networks: (a) safety and (b) unsafety.

Sometimes, the workers must clean the tanks and move some chemicals to the other chemical rooms. Peter is the worker, and his manager will assign him a task with two safety placement networks: the source network T_s and the destination network T_d. A task is called *feasible* if a possible strategy exists to move the chemicals from T_s to T_d following the safety rules; otherwise, it is called *infeasible*. Notice that we do not restrict chemicals to be placed in a specific room. If T_s and T_d are the same, the task is also treated as feasible. Fig. 4 and Fig. 5 show examples of feasible and infeasible tasks, respectively.

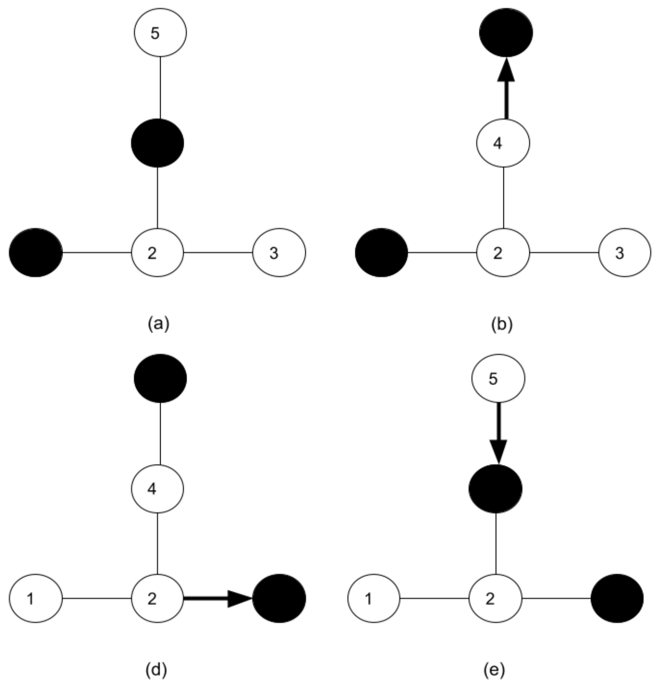

Figure 4: A feasible task: (a) the source network, (b) move the chemical from 4 to 5, (c) move the chemical from 1 to 2, (d) move the chemical from 2 to 3, (e) move the chemical from 5 to 4 to the destination network.

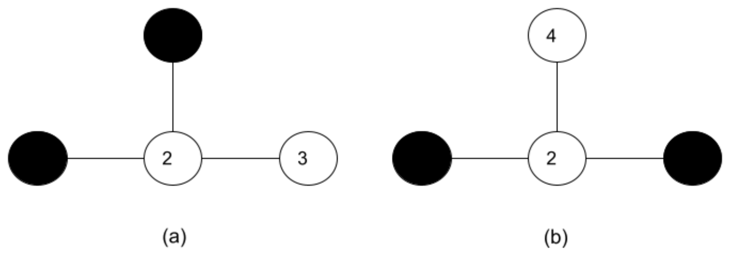

Figure 5: An infeasible task: (a) the source network, (b) the destination network.

Please write a program to help Peter judge whether a task is feasible or not.

Input Format

The first line contains exactly one integer t, which represents the number of test cases. Each test case below contains four lines. For each test case, the first line contains two integers n and m, where n represents the number of chemical rooms and m represents the number of chemicals; the second line contains $n-1$ integers $r_1, r_2, \cdots r_{n-1}$, which represents that room $i+1$ has a railroad connecting to room r_i for $1 \leq i \leq n-1$; the third line contains m integers s_j for $1 \leq j \leq m$, representing the room numbers in which chemicals are placed at the source network; and the fourth line contains m integers d_k for $1 \leq k \leq m$, representing the room numbers in which chemicals are placed at the destination network.

Output Format

Each test case outputs 1 if the task is feasible, otherwise outputs 0 in a line.

Technical Specification

- $5 \leq t \leq 10$.
- $1 \leq m \leq n \leq 10,000$.
- $1 \leq r_i < i+1$, $1 \leq i \leq n-1$.
- $1 \leq s_j \leq n$ and $s_p \neq s_q$ if $p \neq q$, $1 \leq d_j \leq n$ and $d_p \neq d_q$ if $p \neq q$.

Sample Input 1

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

Sample Output 1

```
0
1
1
0
1
1
```

Hint

This problem was inspired by the *sliding token problem* for trees introduced in [1], in which a linear time algorithm proposed.

A node with chemical is called *chemical node*. Let I_s and I_d be the (independent) sets of chemical nodes of T_s (source network) and T_d (destination network), respectively. For a set I of chemical nodes in a network, T, a chemical node in I is called *rigid* if it cannot move at all. We can apply the following rules to identify the rigid nodes.

1. If $|T| = 1$ with one chemical node u, then u is rigid.
2. Suppose $|T| \geq 2$. A chemical node u in the chemical node set I of T is rigid if and only if for every neighbor v of u in T, there exists a chemical node w in $I \cap T^v_w$ is rigid, where

T^v_w is the subtree containing v and w.

The algorithm is based on the following two key points.

1. If I_s and I_d have different placements of rigid nodes, then the task is unfeasible.
2. Otherwise, we obtain a forest by deleting the rigid nodes together with their neighbors. The answer is feasible as long as each tree in the forest contains the same number of chemical nodes in I_s and I_d.

Then, the following algorithm efficiently finds the rigid nodes iteratively.

1. Define and compute $degI(w) = |N(T, w) \cap I|$ for all vertices $w \in V(T)$, where $N(T, w)$ denotes the neighbors of w in T.
2. Define and compute $M = \{\ u \in I|$ there exists $w \in N(T, u)$ such that $degI(w) = 1\ \}$, that is, M is the set of chemical nodes that can be immediately slid.
3. Repeat the following steps (i)–(iii) until $M = \emptyset$.
   (i) Select an arbitrary node $u \in M$, and remove it from M and I.
   (ii) Update $degI(w) = degI(w) - 1$ for each neighbor $w \in N(T, u)$.
   (iii) If $degI(w)$ becomes one by the update (ii) above, then add the node $u \in N(T, w) \cap I$ into M.
4. Output I. Note that, since $M = \emptyset$, all chemical nodes in I are now (T, I)-rigid.

Reference

[1] E. D. Demaine, M. L. Demaine, E. Fox-Epstein, D. A. Hoang, T. Ito, H. Ono, Y. Otachi, R. Uehara, and T.akeshi Yamada, 'Linear-time algorithm for sliding tokens on trees', Theoretical Computer Science, Volume 600, Pages 132-142, 2015.