```
/*                          North American */
/*    Invitational Programming Contest */
/* Hosted by the University of Chicago */
/*                       28-30 March, 2014 */
```

PROBLEMS

A: Banjo

Having mastered the classic games Banjo-Kazooie and Banjo-Tooie, Bob has moved on to playing their secretly-released sequel, Banjo-Threeie. The game can be modelled with a 2-dimensional coordinate system. The player's objective is simply to walk from one point to another as the titular bear Banjo (carrying his bird pal, Kazooie, in his backpack), who can walk at a speed of 1 unit per second. Sounds pretty simple!

However, there is a twist. There is a lava pit in the way, which is perfectly circular. Though the pit is hot, Banjo can still walk across its surface at the same speed. However, if he spends more than ***t*** consecutive seconds in the pit, he'll lose all of his health points. If Banjo steps out of the pit for even a moment, however, his health points will reset, and he'll be able to step back into the pit if he wants to.

To make sure that he's playing the game as optimally as possible, Bob wants to know the minimum possible amount of time required to walk from the start to the end, without spending more than ***t*** consecutive seconds in the lava pit.

Input

There will be several test cases in the input. Each test case will consist of a single line with eight integers:

x1 y1 x2 y2 xc yc r t

where (***x1***,***y1***) is where Banjo starts, (***x2***,***y2***) is where Banjo must go, (***xc***,***yc***) is the center of the lava pit, ***r*** (1≤***r***≤10,000) is the radius of the pit, and ***t*** (0≤***t***≤10,000) is the number of consecutive seconds Banjo can spend in the pit before he loses all of his health points.

All coordinates will be in the range -10,000≤**x**,**y**≤10,000, and all three points in a test case will be unique. The starting and ending points will not be in the lava, nor will they be on the edge of the lava. The input will end with a line with eight 0s.

Output

For each test case, output a single number equal to the minimum time for Banjo to go from the start to the finish, given to exactly two decimal places, rounded. Output each number on its own line, with no spaces. Do not print any blank lines between outputs.

Sample Input	**Sample Output**
0 0 10 0 5 0 3 5 0 0 0 0 0 0 0 0	11.00

B: Cheats

Cosmo is busy playing the little-known latest installment in the Legend of Zelda series of video games, Skyward Wind Mask of Twilight Time. In this game, the player must complete all ***n*** objectives as the young adventurer Link. However, some objectives must be done before others! Each objective **i**, **i** = 2..***n***, has a prerequisite, P_i, which must be completed before **i**. Of course, the first objective (always number 1) has no prerequisite. There are no cycles of dependencies which would cause an objective to indirectly depend on itself.

Like all games, however, this game has hidden cheats. There is one cheat for each objective **i** = 2..***n*** which allows it to be completed before its prerequisite! However, things still can't be done too much out of order. If objective **i**'s cheat is used, then instead of objective **i** being completed after it's prerequisite P_i, it just has to be completed after it's prerequisite's prerequisite, P_{P_i} (unless $P_i = 1$, in which case it can be completed at any point, since objective 1 has no prerequisite). Furthermore, using multiple cheats too close to each other can lead to unpredictable effects, so each objective can be involved in at most one use of a cheat. In other words, if you use a cheat on objective **i** so that you can complete it before its prerequisite P_i, then you can't use a cheat on P_i, nor on any other objective that has **i** as a prerequisite.

Cosmo would like to complete the game while exploiting at most ***k*** cheats. In how many different orders can he complete all ***n*** objectives, subject to these constraints? As this value can be very large, output it modulo 10^9+7.

Input

There will be multiple test cases in the input. Each test case will begin with two integers ***n*** ($1 \le n \le 200$) and ***k*** ($0 \le k < n$), indicating the number of objectives Cosmo must complete (***n***) and the maximum number of cheats he's willing to use (***k***). On the next line will be ***n***-1 space-separated integers ***p*** ($1 \le p \le n$), indicating the prerequisite objective for objectives 2, 3, ..., ***n*** (skipping 1, since objective 1 has no prerequisite). The input will end with a line with two 0s.

Output

For each test case, output a single integer, which indicates the number of ways Cosmo can achieve all ***n*** objectives while using ***k*** or fewer cheats. Output this number modulo 10^9+7. Do not output any spaces, and do not print any blank lines between answers.

Sample Input	Sample Output
5 1 1 1 5 1 0 0	38

Explanation of Sample Input/Output

This table lists all of the orders in which Cosmo can achieve all of the objectives for the Sample Input/Output using at most one cheat.

No Cheats	2's Cheat	3's Cheat	4's Cheat	5's Cheat
1 2 3 5 4	2 1 3 5 4	3 1 2 5 4	1 2 3 4 5	5 1 2 3 4
1 2 5 3 4	2 1 5 3 4	3 1 5 2 4	1 2 4 3 5	5 1 2 4 3
1 2 5 4 3	2 1 3 4 5	3 1 5 4 2	1 2 4 5 3	5 1 3 2 4
1 3 2 5 4			1 3 2 4 5	5 1 3 4 2
1 3 5 2 4			1 3 4 2 5	5 1 4 2 3
1 3 5 4 2			1 3 4 5 2	5 1 4 3 2
1 5 2 3 4			1 4 2 3 5	5 4 1 2 3
1 5 2 4 3			1 4 2 5 3	5 4 1 3 2
1 5 3 2 4			1 4 3 2 5	
1 5 3 4 2			1 4 3 5 2	
1 5 4 2 3			1 4 5 2 3	
1 5 4 3 2			1 4 5 3 2	

C: Diplomacy

You are a member of the Senate of an ancient empire governed by a mighty dictator. You have joined a bipartisan secret committee of the Senate that is plotting to overthrow the dictator.

In order for the plot to succeed, it is crucial that all the states in the empire ultimately support the plan--and to accomplish this, the governors of all the states need to be members of the same Party.

Right now, each state governor is a member of either the Orange Party or the Purple Party. Since you are confident that you can get either party to back the plot, it does not matter which party ultimately prevails.

The secret committee has studied the political situation and determined that two governors will influence each other if they are friends with each other and members of the same Party. To get all the state governors on board, each month a lobbyist will do whatever it takes to get one governor to switch parties. When this happens, all the friends of the governor who are members of the same Party will also switch affiliation, as will the friends of the friends within the party, and so on. To avoid suspicion, the secret committee will alternate Orange/Purple lobbyists each month. They may start the ball rolling with either party in the first month.

The secret committee also knows which governors are friends with each other, that each governor is friends with at least one other governor, and that there are no isolated groups that are only friends with each other.

Your task is to determine the minimum number of months required for all the state governors to be members of the same party. Once this occurs, the next steps in the plot can take place.

Input

There will be several test cases in the input. Each test case will begin with a line with two integers, n ($1 \le n \le 100$) and m ($n-1 \le m \le n(n-1)/2$), where n is the number of governors, and m is the number of known friendships. On the next line will be n integers, either 0 or 1, indicating the current party affiliation of the governors, in order (0=**ORANGE**, 1=**PURPLE**). On each of the following m lines will be two integers, a and b ($1 \le a < b \le n$) indicating that governor a and governor b are friends. As in life, friendships go both ways: if a is a friend of b, then b is also a friend of a. All m (a,b) pairs will be unique. The input will end with a line with two 0s.

Output

For each test case, output a single integer, indicating the minimum number of months necessary for every governor to belong to the same party. Output each integer on its own line, with no spaces. Do not print any blank lines between outputs.

Sample Input	Sample Output
4 3 0 1 0 0 1 2 2 3 2 4 5 4 0 1 1 0 1 1 2 2 3 3 4 4 5 0 0	1 2

D: Fantastic Problem

Andrew, the world-renowned problem writer, has finally decided to retire. However, he first wishes to put out one final masterpiece to seal his legacy and amaze competitors the world over. As you're his favorite pupil, he's asked you to help him proofread it before he submits it to the ICFP (International Committee for Fantastic Problems).

The problem proves to be quite the involved number theory affair, but you get together a solution eventually…only to find that it doesn't pass his data. After wasting hours debugging your code, you realize that it's correct – the data is invalid! Andrew's advanced age seems to have really caught up with him.

In his problem, you're given a sequence of ***n*** integers, $V_{1..n}$, with the important guarantee that within every interval of **k** consecutive integers ($V_i..V_{i+k-1}$, for some starting index **i**), any two integers will be coprime (two integers are coprime if they share no common factors besides 1). Note However, this condition is not met in Andrew's data, causing your program to crash!

In an attempt to help your beloved mentor out, you count how many size-**k** intervals don't satisfy his problem statement's promise. Unfortunately, your work is not done yet. Having difficulty fixing his data, Andrew makes ***m*** sequential updates, Andrew makes ***m*** sequential updates, where each update involves selecting some position ***a*** in the number sequence, and changing its value to some value ***b***. After each change, he wants to know how many invalid size-**k** intervals his new sequence contains. As if that wasn't enough, after the m^{th} update, Andrew decides that the data is good enough, and wants you to solve his problem with the resulting sequence of integers! His problem statement is as follows:

Given a sequence of integers, compute their sum.

The Input

There will be several test cases in the input. Each test case will begin with a line with three integers, ***n*** ($1 \le n \le 100{,}000$), ***k*** ($1 \le k \le n$) and ***m*** ($1 \le m \le 100{,}000$), where ***n*** is the size of Andrew's list, ***k*** is the size of the consecutive intervals of interest, and ***m*** is the number of changes Andrew makes. On each of the next ***n*** lines will be a single integer ***v*** ($1 \le v \le 100{,}000$), which is a value in Andrew's input. The ***v***'s will be in the order that they appear in Andrew's list. On each the following ***m*** lines will be a pair of integers ***a*** ($1 \le a \le n$) and ***b*** ($1 \le b \le 100{,}000$), indicating that Andrew has changed value V_a to be ***b***. The input will end with a line with three 0s.

The Output

For each test case, output ***m***+2 integers. The first integer should be the number of size-**k** intervals in Andrew's original list which fail the pairwise-coprime constraint. Each of the next ***m*** integers should be the number of size-**k** intervals that fail the constraint after each of Andrew's ***m*** changes, in order. The final integer should be the sum of the numbers in the final list. Output each integer on its own line, with no spaces. Do not print any blank lines between outputs.

Sample Input	Sample Output
6 3 4 7 2 3 4 5 6 4 3 5 9 4 10 6 11 0 0 0	2 3 3 3 2 42

E: GCDs

Given a sequence **A** of ***n*** numbers, define f(**lo**,**hi**), 1≤**lo**≤**hi**≤***n***, as the Greatest Common Divisor of all the numbers $\mathbf{A_{lo}}$ through $\mathbf{A_{hi}}$, inclusive. Note that **lo** and **hi** are indices, not members of the list. Given an array, considering all possible values of **lo** and **hi**, how many unique values of f(**lo**,**hi**) will there be?

Input

There will be several test cases in the input. Each test case will begin with a line with a single integer ***n*** (1≤***n***≤100,000) representing the length of the sequence. The next ***n*** lines will each have an integer ***a*** (1≤***a***≤100). These are the numbers in the sequence, in sequence order. The input will end with a line with a single 0.

Output

For each test case, output a single integer denoting the number of unique values f(**lo**,**hi**) can have for the input sequence. Do not output any spaces, and do not print any blank lines between answers.

Sample Input	Sample Output
2 4 6 3 3 6 8 0	3 5

F: Gold Bandits

In the hills of a distant land there is a valley with many villages. The villages are all ruled by a cruel king who demands that each village pay him tribute in gold each year. The villages are required to bring gold to the king as quickly as possible when he demands it.

There is a bandit village in the kingdom. The bandit village has saved no gold for the king, because they spent every bit of gold that they had! What are they going to do? Well, since they're bandits, when they travel through other villages to get to the castle, they may (or may not) choose to steal the gold from any of those villages on their path to the castle, and present it to the king as their own.

The bandits travel through as few villages as possible to reach the castle. They also have one other consideration–after delivering the gold to the king the bandits must be able to return home. They consider it unsafe to return home traveling through a village from whom they stole gold. The bandits do not otherwise care how long it takes to return home.

Determine the maximum amount of gold the bandits can steal on their way to the king's castle and still be able to return home safely.

Input

There will be several test cases in the input. Each test case will begin with a line with two integers, n ($3 \le n \le 36$) and m ($n-1 \le m \le n(n-1)/2$), where n is the number of villages, and m is the number of roads. The villages are numbered 1..n. Village 1 is the bandit's home, and village 2 holds the king's castle. On the next line will be n-2 space-separated integers g ($1 \le g \le 5{,}000$), which are the amount of gold in each village 3, 4, …, n, skipping the bandit's home and the king's castle. On each of the following m lines will be two integers, a and b ($1 \le a < b \le n$) indicating that there is a road between villages a and b. All roads are two-way. All m (a,b) pairs will be unique. There will be a path, direct or indirect, from every village to every other village. The input will end with a line with two 0s.

Output

For each test case, output a single integer, indicating the maximum amount of gold that the bandits can purloin and still get safely home. Output each integer on its own line, with no spaces. Do not print any blank lines between outputs.

Sample Input

```
3 3
1
1 2
2 3
1 3
4 4
24 10
1 3
2 3
2 4
1 4
6 8
100 500 300 75
1 3
1 4
3 6
4 5
3 5
4 6
2 5
2 6
7 7
90 1000 700 2000 800
1 3
1 4
1 5
3 7
5 6
2 6
3 6
0 0
```

Sample Output

```
0
24
800
700
```

G: Integer Estate Agent

After recent riots in Flatland, the leader has granted permission to purchase real estate at the famous Zero street to everyone. Now, not only the noble polygons are allowed to live there, but pentagons, squares, and even triangles as well (if they can afford a house at Zero street, of course)!

As an employee of the "Integer Estate Agency", you are in charge of the one side of the street. House numbering starts right after Zero Square. House #1 is right off of the square, House #2 comes right after House #1, House #3 comes after House #2, and so on towards Infinity (where, as rumors say, the family of noble Perfect Circles live). House #***k*** costs exactly ***k***+1 coins. There is no House #0, since that is the square.

A promising customer is willing to spend exactly ***n*** coins to purchase a single block of consecutive houses to group together as a condo complex. How many options does he have? For example, if he wishes to spend 5 coins, he may buy houses #1 and #2 (which costs 2+3=5), or he may just buy house #4, so he has two options.

Input

There will be several test cases in the input. Each test case will consist of a single integer ***n*** ($1 \le n \le 1{,}000{,}000$) on its own line, indicating the number of coins a customer is willing to spend. The input will end with a line with a single 0.

Output

For each customer, output the number of ways he can buy consecutive houses spending exactly ***n*** coins. Output each number on its own line, with no spaces. Do not print any blank lines between outputs.

Sample Input	Sample Output
1	0
2	1
5	2
0	

H: Reconnaissance

You have located a major supply line that the enemy has been using. With satellite imaging, you've been able to determine the current location and velocity of every vehicle along the supply line, which is for all practical purposes an infinitely long straight line. Furthermore, you know that each vehicle moves at constant velocity, and that they can pass by each other without incident along this supply line. What you need to do now is deploy an air drone with special sensors that can give you readings of the contents of the vehicles. The sensor is capable of reading everything in its range instantaneously, but power limitations allow it to do so only once. In order to minimize the required range, you want to deploy it when the vehicles are as close to each other as possible. Given knowledge of the current location and velocity of all the vehicles, what is the closest that the vehicles will get to each other?

Input

There will be several test cases in the input. Each test case will begin with a line with a single integer n ($1 \le n \le 100{,}000$) representing the number of vehicles. Each of the next n lines will have two integers, x and v ($-100{,}000 \le x, v \le 100{,}000$), indicating the position (x, in meters) and velocity (v, in meters/hour) of that vehicle. The sign of velocity indicates direction. The input will end with a line with a single 0.

Output

For each test case, output a single number equal to the minimum distance that will cover all of the vehicles at some time, in meters, given to exactly two decimal places, rounded. Output each number on its own line, with no spaces. Do not print any blank lines between outputs.

Sample Input	Sample Output
2 -100 1 100 -1 3 -100 1 100 -1 101 -1 3 -100 -1 0 0 100 1 0	0.00 1.00 200.00

I: Super Mario 169

Siggy is quite the video game enthusiast, and he's been playing lots of Super Mario 169 lately (the highly obscure sequel to the more popular Super Mario 64). This game takes place entirely in an ocean, which can be modelled with a 3 dimensional coordinate system. The player's objective is to swim around as the titular character, Mario, and collect all of the coins, of which there can be up to 169.

The coins are not simply in plain sight, however! Instead, there are up to 13 switches which Mario can press by touching them. Pressing any switch causes up to 13 coins to appear. Additionally, each switch can only be pressed once, and when it's pressed, any other uncollected coins (which had been revealed by the previously-pressed switch, if any) disappear, meaning that Mario will be unable to collect them in the future. All switches and coins are small enough that they can be treated as points.

To make sure that he's playing the game as optimally as possible, Siggy wants to know the minimum possible distance required to collect all of the coins.

Input

There will be several test cases in the input. Each test case will begin with a single line with four integers:

n mx my mz

where ***n*** (1≤***n***≤13) is the number of switches, and the 3D point (***mx***,***my***,***mz***) is Mario's starting point.

The following pattern is then repeated ***n*** times, once for each switch. The pattern begins with a single line with four integers:

k sx sy sz

where ***k*** (1≤***k***≤13) is the number of coins activated by this switch, and the 3D point (***sx***,***sy***,***sz***) is the point where the switch is found. Following this there will be ***k*** lines with three integers:

cx cy cz

where (***cx***,***cy***,***cz***) is the 3D point of one of the coins activated by this switch.

All coordinates **x**, **y**, **z** of all points will be in the range -1,000≤**x**,**y**,**z**≤1,000, and all points in a test case, whether Mario's starting point, switch or coin, will be unique. The input will end with a line with four 0s.

Output

For each test case, output a single number equal to the minimum distance for Mario to travel in order to collect all of the coins, given to exactly two decimal places, rounded. Output each number on its own line, with no spaces. Do not print any blank lines between outputs.

Sample Input	Sample Output
2 5 5 0 4 6 0 0 7 0 0 -11 -1 0 -11 1 0 -10 0 0 2 5 0 0 0 0 0 0 5 0 0 0 0 0	44.22

J: Two Knight's Poem

Two chess knights have decided to collaborate on writing short, one-line poems. They have obtained the use of a laptop to type their poetry. The laptop keyboard is composed of 4 rows of 10 keys. 30 of these are symbol keys, 4 are Shift keys, and 6 are Space keys.

Q **q**	**W** **w**	**E** **e**	**R** **r**	**T** **t**	**Y** **y**	**U** **u**	**I** **i**	**O** **o**	**P** **p**
A **a**	**S** **s**	**D** **d**	**F** **f**	**G** **g**	**H** **h**	**J** **j**	**K** **k**	**L** **l**	**:** **;**
Z **z**	**X** **x**	**C** **c**	**V** **v**	**B** **b**	**N** **n**	**M** **m**	**<** **,**	**>** **.**	**?** **/**
Shift	**Shift**	**Space**	**Space**	**Space**	**Space**	**Space**	**Space**	**Shift**	**Shift**

Note that the **Shift** and **Space** keys, which are usually extra-wide keys on a keyboard, are treated here as multiple individual keys that each have the same effect.

The knights will type the poem by making moves, one at a time, that are valid for a chess knight. A chess knight can move two positions vertically and then one horizontally, or one position vertically and then two horizontally. For example, from the **D** key, a knight can move to any of these keys: **Q**, **Z**, **T**, **B**, the second **Shift** key from the left, and the second **Space** key from the left.

One knight always begins each poem on the left-most Shift key. The other knight always begins on the right-most Shift key. Either knight may move first, and either may make multiple consecutive moves. The knights cannot occupy the same key.

Each move of a knight will type at most one character, adding to the poem. Landing on a symbol key or **Space** key will type one character. A knight landing on a symbol key types the upper value of that key when the other knight is on a **Shift** key; otherwise, the lower value is typed. Landing on a **Space** key always types a single space character, regardless of whether the other knight is on a **Shift** key. Landing on a **Shift** key does not add anything to the poem.

Input

There will be several test cases in the input. Each test case will consist of a string on a single line, representing a poem. Each poem will consist of 1 to 100 characters inclusive, using only characters from the symbol keys on the keyboard and spaces. No poem will begin or end with a space. The input will end with a line with a single asterisk ('*****').

Output

For each poem, output `1` if the knights can type the poem, or `0` if they cannot. Output each number on its own line, with no spaces. Do not print any blank lines between outputs.

Sample Input	Sample Output
S,veA,eVE,aU S,veA,eVE,aUc CAlmimg eventa CAL *	1 0 1 1