University of Central Florida

2025 Local Programming Contest (Final Round)

Problems

Problem#	Difficulty Level	Filename	Problem Name
1	Easy	blank	Intentional Blank Page
2	Easy	poker	Full House
3	Easy-Medium	swags	Duck Stress Balls
4	Easy-Medium	twins	Twin Numbers
5	Medium	house	Simple House
6	Medium	moose	Pathfinder and Wolves
7	Medium	xorprimes	XOR Primes
8	Medium-Hard	festival	Festival of Blossoms
9	Medium-Hard	zigzag	Ancient Zigzag
10	Medium-Hard	birthday	Birthday Bash
11	Medium-Hard	things	Dark Things
12	Hard	voyage	Galactic Voyage
13	Hard	degenerate	Degenerating Walkways

Call your program file: filename.c, filename.cpp, filename.java, or filename.py

For example, if you are solving Pathfinder and Wolves:

Call your program file: moose.c, moose.cpp, moose.java, or moose.py

UCF Local Contest (Final Round) — September 13, 2025

Intentional Blank Page

filename: `blank`
Difficulty Level: `Easy`
Time Limit: `2 seconds`

Some contests copy the problem set double-sided (i.e., front and back). They do not, however, start a problem on a back page. If a problem requires an odd number of pages (1, 3, 5, 7, …), they leave the last page for that problem (a back page) blank so that the next problem starts on a front page (they usually put the message "*This page is intentionally left blank.*" on that last page for the problem).

The Problem:

Given number of problems and number of pages for each problem, determine how many pages are left blank in the problem set, i.e., how many pages have the special message.

The Input:

The first input line contains an integer, p ($1 \leq p \leq 20$), indicating the number of problems in the set. Each of the next p input lines contains an integer (between 1 and 100, inclusive) indicating the number of pages for a problem.

The Output:

Print the number of blank pages.

Sample Input	Sample Output
5 13 10 70 1 1	3
3 6 12 8	0
1 1	1

Full House

filename: `poker`
Difficulty Level: `Easy`
Time Limit: `2 seconds`

A poker hand consists of 5 cards. The hand is called a *full house* when three cards match in value and the remaining two cards also match in value (but the second value is different from the first value).

The Problem:

Given an integer consisting of exactly five digits, determine if it is a *full house* (three digits one value, two digits another value).

The Input:

There is only one input line; it provides an integer between 10,000 and 99,999, inclusive.

The Output:

Print YES or NO, indicating whether or not the input integer is a *full house*.

Sample Input	**Sample Output**
12121	YES
12345	NO
88888	NO
77555	YES

Duck Stress Balls

filename: `swags`
Difficulty Level: `Easy-Medium`
Time Limit: `2 seconds`

The UCF Programming Team organizes an onsite contest for the high schools in Florida. This contest is modeled after ICPC and each team consists of three students. In the most recent contest, the contest organizers purchased some stress balls (ducks) to give away as swags. The ducks happen to be in several different colors and the organizers decided to give each team three ducks of different colors, i.e., the three students on a team will have ducks of different colors. The process became a little cumbersome so here you are to help!

The Problem:

Given number of duck colors and the count for each color, determine if the three students on each team can get ducks of different colors. Assume that there are $3n$ total ducks to be divided among n teams.

The Input:

The first input line contains an integer, c ($3 \le c \le 10$), indicating the number of different duck colors. Each of the next c input lines contains an integer (between 1 and 60, inclusive) indicating the number of ducks of a given color. Again, there will be a total of $3n$ ducks to divide among n teams.

The Output:

Print YES or NO, indicating whether or not each team can get three ducks of different colors.

(Sample Input/Output on the next page)

Sample Input	Sample Output
5 2 3 3 2 2	YES
3 59 1 60	NO
4 3 3 3 3	YES

Note: In the second Sample, there are 120 ducks to be divided among 40 teams. It is not possible for each team to get three ducks of different colors.

Twin Numbers

filename: `twins`
Difficulty Level: `Easy-Medium`
Time Limit: `1 second`

An integer is a *twin* if every digit at position $2n$-1 in the number is the same as the digit at position $2n$, e.g., 88, 8855, 775588, 8888. Note that a *twin* number must have an even number of digits.

The Problem:

Given a range (lower bound and upper bound), determine how many *twins* there are in that range.

The Input:

There are two input lines; the first line provides the lower bound for the range and the second line provides the upper bound. Assume that $10 \leq$ lower bound $\leq$ upper bound $\leq 10^{100,000}$.

The Output:

Print the number of *twins* in the given range. Since the answer can be large, print the value *mod* 10,007.

Sample Input	Sample Output
`20` `30`	`1`
`22` `99`	`8`
`1000` `4489`	`39`
`330210` `1234567890`	`9683`

Note: The number of *twins* for the last Sample range is 19,690 so the output will be 19,690 *mod* 10,007 = 9683.

UCF Local Contest (Final Round) — September 13, 2025

Simple House

filename: `house`
Difficulty Level: `Medium`
Time Limit: `1 second`

You have purchased a piece of land that is rectangular. The land boundaries are not necessarily parallel to *x* and *y* axes. You want to build a rectangular house on your land but you'd like the house boundaries to be parallel to *x* and *y* axes. The house can touch the land boundaries but cannot go beyond the land. You would also like the house to look nice so the house must have a fixed ratio of its width (*x*-axis) to its height (*y*-axis).

The Problem:

Given the coordinates of the four corners of the piece of land, as well as two integers *w* and *h* (which represent the ratio of the width of the house to the height of the house), determine the area of the largest house that you can build.

The Input:

There are five input lines:

- Each of the first four input lines contains two integers, *x* and *y* ($-10^4 \le x, y \le 10^4$), providing one corner of the land. The four corners are provided in a clockwise order, with the bottom-most, left-most corner first.
- The fifth input line contains two integers, *w* and *h* ($1 \le w, h \le 20$), representing the ratio of the width to height.

The Output:

Print the area of the largest house that you can build. Any answer within an absolute or relative error of 10^{-6} of the correct answer will be accepted.

Sample Input	Sample Output
3 0 0 4 8 10 11 6 2 1	12.5
0 0 0 5 100 5 100 0 20 1	500.0

Pathfinder and Wolves

filename: `moose`
Difficulty Level: `Medium`
Time Limit: `3 seconds`

The famous explorer, Moose The Pathfinder (MTP), is at cell position (1,1) in an $r \times c$ rectangular maze and needs to get to cell position (r,c). MTP can go in any of the four directions (north, south, east, west) except when in a border cell (the border cells have fewer move options since MTP cannot move across the outer walls of the maze). Unfortunately, wolves are hiding in some cells and, obviously, MTP wants to stay as far away as possible from these cells.

The Problem:

Given the maze, determine the closest MTP will have to get to any cell with wolves while trying to go from position (1,1) to position (r,c). Note that MTP would travel longer if it helps to be farther away from the cells with wolves, i.e., MTP would take a longer path if it means being farther away from the cells with wolves. MTP distance to a cell with wolf is defined as the number of steps (in the four directions) from where MTP is to the cell with wolf.

Assume that there will be a path for MTP that will not go thru any cell with wolf, i.e., the closest MTP will have to get to a cell with wolf will be of distance 1. Also assume that there is at least one cell with wolf in each maze.

The Input:

The first input line contains two integers: r ($2 \le r \le 10^3$), indicating the number of rows in the maze and c ($2 \le c \le 10^3$), indicating the number of columns in the maze. The maze is provided in the following r input lines. Each of these input lines contains c characters starting in column 1. The cells with a wolf are indicated with the letter W and the empty cells are indicated with hyphen (-). Assume that the starting and ending positions will be empty.

The Output:

Print the closest MTP will have to get to any cell with wolf.

(Sample Input/Output on the next page)

Sample Input	Sample Output
5 5 ----W --W-W ----W ----- W----	2
3 5 ---WW ----- -----	2
3 3 --- --W ---	1
3 3 -W- --- ---	1

Notes:

In the third Sample, MTP is of distance 1 to a cell with wolf when at destination.
In the last Sample, MTP is of distance 1 to a cell with wolf when at the starting position.

XOR Primes

filename: `xorprimes`
Difficulty Level: `Medium`
Time Limit: `1 second`

Did we steal this problem from CSES or Project Euler? Surprisingly not!

The Problem:

Given a positive integer n, determine 100 or fewer **prime numbers**, each less than 2.1×10^9, such that the bitwise XOR of those prime numbers equals n. Recall that a prime number is an integer 2 or greater that has exactly two positive divisors: 1 and itself.

The Input:

There is only one input line; it provides the integer n ($1 \le n \le 2\times10^9$).

The Output:

Print, on a single line, 100 or fewer space separated prime numbers, each less than 2.1×10^9, such that the bitwise XOR of the prime numbers equals n. The numbers can be listed in any order and any set of prime numbers that satisfies the given constraints will count as an accepted response.

Sample Input	Sample Output
`12`	`31 19`
`19`	`19`

Festival of Blossoms

filename: `festival`
Difficulty Level: `Medium-Hard`
Time Limit: `3 seconds`

In the Kingdom of Floralia, the annual Festival of Blossoms marks the arrival of spring with a cherished tradition: every household must display a string of exactly N flowers on their door. Each year, the king announces K different flower colors available at the royal gardens, and citizens may use any combination of these colors to craft their decoration (a string of flowers).

This year's festival carries special significance. The royal oracle has emerged from the Temple of Petals with a prophetic vision: a sacred substring S of colors that promises to bring prosperity and good fortune to the kingdom. According to the prophecy, every flower-string must contain this blessed pattern (contiguous substring) somewhere along its arrangement, meaning the colors of S must appear consecutively and in the exact same order somewhere within the N flowers. Only then will the divine blessing reach each home in Floralia.

The king's assistant, a meticulous man who takes great pride in the festival's organization, faces his annual challenge. He firmly believes that every household deserves a unique decoration (a string of flowers), and no two families should receive identical arrangements on this special day. In his careful records, two strings of flowers are considered the same if each of the N flowers are identical in color when viewed left to right.

With this year's values of N and K announced, and the oracle's prophetic substring S revealed at dawn, the assistant turns to you with an urgent question: exactly how many distinct strings of N flowers containing S are possible?

The answer will determine whether the kingdom has enough unique designs for all its households to participate in this blessed tradition. Time is short and the festival begins at sunset.

The Problem:

Given N, K, and S, determine the number of unique flower strings which contain the sacred contiguous substring.

The Input:

There is only one input line; it contains two integers N ($1 \le N \le 10^4$) and K ($1 \le K \le 26$), followed by substring S ($1 \le |S| \le min(N, 1000)$).

Assume that the flower colors are represented by the first K lowercase letters, e.g., if $K = 4$, the flower colors will be a, b, c, and d. The substring S is also made up of the first K lowercase letters.

The Output:

Print one integer, the number of unique strings of *N* flowers containing the sacred substring *S*. Since the answer can be quite large, your program should report the number as the smallest non-negative remainder when divided by 1,000,000,007.

Sample Input	**Sample Output**
`4 2 a`	`15`
`2 3 ca`	`1`
`10 4 abc`	`127248`
`5 26 wow`	`2027`
`10 26 wow`	`176416529`
`31 2 a`	`147483633`

Ancient Zigzag

filename: `zigzag`
Difficulty Level: `Medium-Hard`
Time Limit: `3 seconds`

Two ancient civilizations, separated by light-years, have each passed down a sequence of sacred numbers. A *sequence* is a list of numbers such as {1, 7, 14, 8, 2} and {7, 9, 1, 8, 14, 2, 1}.

While the sequences in these civilizations differ, their scholars are fascinated by a particular pattern present in these sequences, the *zigzag*, a sequence that constantly switches direction and never becomes too predictable.

A sequence is called *zigzag* if **no three consecutive elements are monotonic**. Formally, a sequence A of length n is *zigzag* if, for all i such that $1 \le i \le n - 2$, neither of the following holds:

- $A_i \le A_{i+1} \le A_{i+2}$
- $A_i \ge A_{i+1} \ge A_{i+2}$

You are tasked with finding the longest such zigzag *subsequence* that is common to both civilizations.

A *subsequence* of a given *sequence* is a sequence that can be derived from the given sequence by deleting some or no elements without changing the order of the remaining elements. For example, {7, 14, 2} is a *subsequence* of each of the two *sequences* listed above.

The Problem:

Given two sequences of integers, your task is to find the longest subsequence that is present in both sequences and forms a zigzag. Note that the two input sequences are not necessarily zigzag themselves.

The Input:

The first input line contains two integers, n and m ($1 \le n, m \le 5000$), the sizes of the two sequences. The second input line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$), the first sequence. The third input line contains m integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le 10^9$), the second sequence.

The Output:

Print two lines. On the first line, output a single integer L, the length of the longest common zigzag subsequence. On the second line, output a list of L space separated integers representing a valid common zigzag subsequence, in order. If multiple valid solutions exist, you may print any one of them.

(Sample Input/Output on the next page)

Sample Input	**Sample Output**
1 3 5 1 5 2	1 5
6 6 1 3 2 4 6 5 2 3 2 4 5 6	3 3 2 5
7 10 827 53 1923 456 731 88 619 53 934 729 1087 456 615 731 777 88 1000	3 53 731 88

Birthday Bash

filename: `birthday`
Difficulty Level: `Medium-Hard`
Time Limit: `5 seconds`

Bob loves birthdays. In fact, he loves them so much that he's taken his celebrations interplanetary. Whenever he lands on a new planet, his first mission (before food, water, or breathable air) is to throw the biggest birthday bash the galaxy has ever seen. The catch? Bob only parties when every single birthday is represented.

On this planet, the year consists of exactly n days, and each person Bob meets has an equal and independent chance of being born on any of those days. Starting from an empty guest list, Bob adds people to his party one by one, selecting each guest uniformly at random from the population. He stops as soon as his guest list includes someone born on *every* of the n possible birthdays.

What is the **expected number of people** Bob needs to invite until he has a complete set of birthdays?

The Problem:

Given an integer n, find the expected number of randomly selected people needed until every one of the n possible birthdays is represented at least once. Each person's birthday is uniformly and independently distributed across the n days. Calculate this answer *mod* 1,000,000,007.

The Input:

There is only one input line; it provides an integer n ($1 \le n \le 2 \times 10^9$), the number of days in a year on this planet.

The Output:

Print the expected number of people *mod* 1,000,000,007. If the answer is a fraction, output it as an integer in the form $(A \times B^{-1})$ *mod* 1,000,000,007, where B^{-1} is the modular inverse of B *mod* 1,000,000,007.

(Sample Input/Output on the next page)

Sample Input **Sample Output**

1	1
2	3
6	900000021
10	289682571
14	980613883
20	26325398
1075	742969056
99998	924541619

Explanation of the Sample Input/Output:

For the Sample Input $n = 6$, we will have:

$A = 147$

$B = 10$

Answer = $(A \times B^{-1})$ *mod* 1000000007 = 900000021

For the Sample Input $n = 20$, we will have:

$A = 279175675$

$B = 3879876$

Answer = $(A \times B^{-1})$ *mod* 1000000007 = 26325398

Dark Things

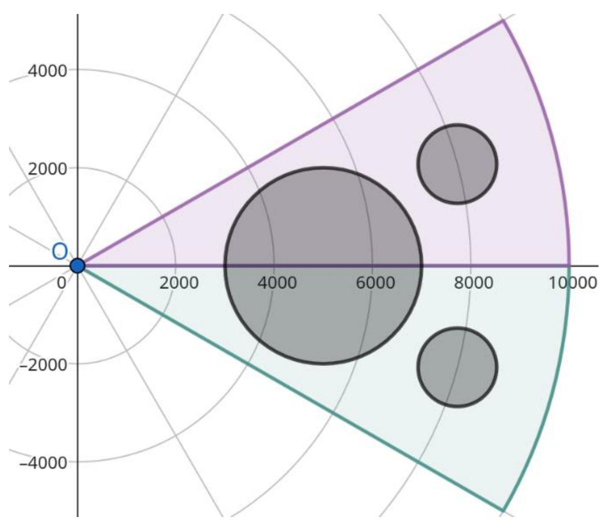
filename: `things`
Difficulty Level: `Medium-Hard`
Time Limit: `1 second`

As you sit in the observatory gazing upon the stars, several alerts go off drawing your attention to the main console which reads:

```
Warning: Several unidentified objects spotted!
```

These objects are quite peculiar, not matching any physical properties of any known or theorized substance or energy, not even dark matter or dark energy! Of course, your first thought is about how many great papers this could lead you to publishing and just like many great physicists, you started by deciding to give these objects a perfectly apt name: "Dark Things!".

Now that you have named these objects, you have decided to focus on the less important details, such as how the dark things interact with surrounding masses and energy. To study these in further detail, you decide to employ k sensors on the observatory's telescope systems and you want each of these sensors to do an equivalent amount of work. To determine how to allocate the search space of each sensor, you have decided to model your situation using geometry as illustrated below:



In the cartesian plane, you model the Earth as a point in the plane located at the origin. At the current point in time, you know that any point which is within R parsecs from earth and is at most θ degrees away from the positive x-axis can be seen by the observatory. The dark things can also be modelled as non-intersecting circles whose centers are r_i parsecs away from the earth, are φ_i degrees away from the positive x-axis, and have a diameter of d_i. Due to the limitation in the observatory's technology, you also know that each dark thing must be fully contained within the set of observable points.

Each sensor will scan all points that are not contained within any dark thing, are within R parsecs of the earth, and are within some range of angles measured from the positive x-axis. To make sure all sensors do an equal amount of work, you must find k-1 angles ψ_j such that the first sensor scans from $-\theta$ to ψ_1, the second sensor scans from ψ_1 to ψ_2, the third sensor scans from ψ_2 to ψ_3, and so on until the last scanner which scans from ψ_{k-1} to θ and the areas of the set of points each sensor scans are equal.

The Problem:

Given the region of observable points and the location of each dark thing, partition the search space into k equal areas, where any point within a dark thing is ignored.

The Input:

The first input line contains four integers: n ($1 \leq n \leq 10^3$), indicating the number of dark things, k ($2 \leq k \leq 50$), indicating the number of sensors, R ($10^4 \leq R \leq 10^6$), indicating the maximum distance of any observable point, and θ ($30 \leq \theta \leq 60$), indicating the maximum angle of any observable point when measured from positive x-axis.

Each of the following n input lines will contain three integers: r_i ($1 \leq r_i \leq R$), φ_i ($|\varphi_i| \leq \theta$), and d_i ($1 \leq d_i \leq 2000$), indicating the location and size of the i^{th} dark thing. It is guaranteed that no two dark things touch or intersect and that each dark thing is fully contained within the set of observable points.

The Output:

Print k-1 floating point numbers, one per output line, indicating the angle of each partition. Any answer within an absolute or relative error of 10^{-6} will be accepted.

Sample Input	Sample Output
3 2 10000 30 5000 0 2000 8000 15 800 8000 -15 800	0.0000000000
3 3 10000 60 9000 55 50 9500 -59 10 9250 57 30	-20.0009600000 19.9979900000

Note: The image on the previous page corresponds to the first sample, where we are splitting the observable set of points into two equal regions (the green and purple sections). In this scenario, both sensors will cover an area of roughly 5.0286×10^7 parsecs^2.

Galactic Voyage

filename: `voyage`
Difficulty Level: `Hard`
Time Limit: `3 seconds`

You are the navigator aboard the starship Celestial Wanderer, bound for Sagittarius A*, the supermassive black hole at the galaxy's center. Your vessel is equipped with a state-of-the-art Quantum Warp Drive. Unlike conventional propulsion systems that provide continuous thrust, this warp drive operates through discrete spatial translations. The drive has been calibrated with S distinct warp configurations, where each configuration can transport your ship exactly s_i astronomical units forward along your trajectory. This drive only works while traversing towards the center of the galaxy, so you'll have to equip a different drive for the return trip.

In the observation lounge, you overhear a group of veteran explorers discussing an ancient network of wormholes that permeates the galaxy along radial paths toward its center. According to these explorers, there are P distinct series of wormholes along the straight-line path from our current location to the center of the galaxy. Each of the P series forms a regular pattern of wormhole entrances, with wormholes in series i being located at distances of $0, p_i, 2p_i, 3p_i, \ldots$ astronomical units from your current position. It just so happens we are at a location where all P series converge.

The group of veteran explorers went on to detail exactly how these wormholes work. You can enter any wormhole from any series and exit from any wormhole in any series (including the same one), with two critical restrictions. First, you can only travel forward toward the galactic center, meaning if you enter a wormhole at distance X then you must exit at a distance greater than X. Second, gravitational distortions limit each wormhole jump to at most K astronomical units, meaning the distance between your entry and exit points cannot exceed K.

Mission control plans to place a resupply checkpoint station along this heavily traveled route to Sagittarius A*. To ensure the station is accessible to all vessels using the same warp drive as yours, they want you to calculate the maximum distance D that you cannot reach from your starting point.

The Problem:

Given the ship's warp configurations S, the set of wormhole series P, and the value K in astronomical units, determine the maximum distance D from your current location that you cannot reach via some combination of warp drive and wormhole traversals.

The Input:

There are three input lines:

- The first input line contains three integers: S, P, *and* K ($1 \leq S, P, K \leq 10$), indicating (respectively) the number of warp configurations, the number of wormhole intervals, and the value of K.

- The second input line contains S integers, $s_1, s_2, \ldots s_s$ ($1 \le s_i \le 10^4$), the set of warp drive distances.
- The third input line contains P integers, $p_1, p_2, \ldots p_p$ ($1 \le p_i \le 10^4$), the set of wormhole intervals.

The Output:

Print the distance D, in astronomical units, representing the maximum unreachable distance. If all locations are reachable or there are infinitely many unreachable locations, print -1 instead.

Sample Input	Sample Output

`2 2 4`
`3 7`
`4 11`
	`5`

`3 2 1`
`6 9 15`
`4 10`
	`-1`

`3 2 5`
`8 17 20`
`10 15`
	`63`

Explanation of the first Sample Input/Output:

$S = \{3, 7\}$: This means that from your starting position 0, you can advance to 3, 6, 9, ...; you can also advance from your starting position 0 to 7, 14, 21, ... Additionally, you can combine these forward steps in any order, e.g., from starting position 0 you can traverse 3 steps followed by 7 steps to reach position 10.

$P = \{4, 11\}$: This means that there are wormholes at position 0, 4, 8, ... as well as at positions 0, 11, 22, ... With $K = 4$, you can traverse from the 0 portal to position 4, then from 4 to 8, then from 8 to 11; you can go from 8 to 11 because the distance between your entry point (8) and your exit point (11) does not exceed K (4). Note that you cannot traverse from the 0 portal to position 11 directly (even though P contains 11) because the distance between your entry and exit points exceeds K.

The largest value which is completely unreachable via some combination of warp configurations in S or wormholes in P is 5 (this means we can reach 6, 7, 8, 9, 10, …).

(Please see the "Notes" on the next page)

Notes:

- When at a position (regardless of how you got there), you can use steps in *S* to advance. For example, after entering a wormhole, you can use steps in *S* to advance next.
- The values in *P* cannot be used as steps to advance, i.e., wormhole values cannot be used as steps. For example, in the first Sample above, let's assume you move to position 7; from there you cannot use 11 (a value in *P*) to advance to position 18 (of course, there may be other ways to get to position 18).
- You can advance from one wormhole position to any other wormhole position (assuming the constraint *K* is met).

Degenerating Walkways

filename: `degenerate`
Difficulty Level: `Hard`
Time Limit: `7 seconds`

You are exploring an old abandoned space station looking for a way to get home after your spaceship malfunctioned. The space station consists of n hubs and n-1 bidirectional walkways. Each bidirectional walkway connects two different hubs and any pair of hubs are directly or indirectly connected to each other.

After quite a bit of time exploring the station, the space station's power generators activate, revealing a Sustained Technological Astro-Relic (STAR) in each hub. There is exactly one STAR (an integer) in each hub. These STARs are just what you have been looking for, as you can use them to stabilize some portal technology you brought from your spaceship and will help you to get home! More specifically, you will attach (append) these STARs to your portal following the pattern (order) in a target sequence. The task is simplified by having no duplicate values in the target sequence, i.e., you know that $x_i \neq x_j$ in the target sequence for any indices i and j such that $i \neq j$.

STARs come in one of k forms (k different integer values) and from your engineering background, you know that certain sets of STARs can be attached to portal technology resulting in various stabilities. More specifically, if the i^{th} STAR appended to the portal is of form x_i (i.e., integer value x_i) and this value matches the next item in the target sequence, the stability of the portal increases by one. Note that the target sequence dictates what can be attached to the portal. If the i^{th} item in the target sequence is x_i, the i^{th} item attached to the portal must be x_i; taking any other STAR will result in the complete decay of the portal.

Your goal is to now collect STARs to stabilize your portal technology and travel home. Due to being really bad at directions, you are not sure where (which hub) you are in the space station, however you do know what the layout of the space station is, so to collect a STAR set, you are going to randomly traverse the station.

Unfortunately, the power being reactivated has made traversing the space station more difficult. Due to the age and the lack of maintenance of the walkways, traversing any walkway will result in it degenerating and it will no longer be able to be used again, i.e., each walkway can be used at most once. Thus, you will collect STARs with the following strategy:

1. In the current hub, take the STAR in the hub if it is of the form you need, i.e., matches the next item in the target sequence.
2. If you have collected t total STARs so far, you attempt to use the portal. The portal will work with a probability of success of $1 - (p/q)^{t+1}$, where p and q are given constants.
3. If Step 2 does not succeed, traverse to a new hub. Of the walkways that are currently functioning, choose one equiprobably. If there are no walkways connected to the hub you are currently in, you will stay in the current hub and repeat Step 2 (i.e., attempt to use the portal again) until you succeed, i.e., the portal successfully brings you home.

You now want to determine what the expected stability of the portal will be when you take it home. Since you are unsure of which hub you are currently in, you will assume that each hub has an equal probability of being the starting hub.

The Problem:

Given the space station layout, the target sequence of STARs to be collected, and which type of STAR each hub contains, determine the expected stability of a portal that can be created.

The Input:

The first input line contains five integers: n ($1 \le n \le 10^5$), indicating the number of hubs in the space station, m ($1 \le m \le 10^5$), indicating the size of the target sequence of STARs, k ($1 \le k \le 10^9$), indicating the number of types of STARs, and p and q ($0 < p < q <$ 998,244,353), indicating the given constants used for computing the probability of success.

The second input line provides the target sequence, i.e., this input line will consist of m integers where the i^{th} integer corresponds to the i^{th} target STAR. Each of these integers will be between 1 and k, inclusive, and no pair of integers will be equal.

The next input line will be composed of n integers, where the i^{th} integer indicates the STAR type in the i^{th} hub. Each STAR type will be denoted by an integer between 1 and k, inclusive. Note that different hubs may have the same STAR type.

Each of the following n-1 input lines will contain two integers u and v ($1 \le u, v \le n$) denoting that there is a walkway connecting hub u and hub v. Note that the space station is completely connected, i.e., you can reach any hub from any other hub.

The Output:

Print the expected stability of the portal after successfully using it. Since the answer can be represented as a/b where a and b are integers, output $a(b^{-1})$ *mod* 998,244,353, where b^{-1} denotes the multiplicative inverse of b *mod* 998,244,353.

(Sample Input/Output on the next page)

Sample Input	**Sample Output**
2 2 2 1 2 2 1 1 2 1 2	124780545
4 5 6 4 5 3 5 2 1 6 5 2 3 2 1 2 1 3 1 4	343422678

Note: The answer to the first sample corresponds to the fraction 7/8 and the answer to the second sample corresponds to the fraction 26947/37500.