UTD Battle Of The Brains Fall 2024

Rules:

1) **DO NOT TURN THE PAGE UNTIL THE CONTEST HAS STARTED**
2) No electronic devices aside from the computer your team is assigned (Cell phones off, no tablets, etc...).
3) No communication between teams.
4) Programming language reference books are permitted.
5) Use of the internet aside from accessing the contest site is not permitted. Failure to follow this rule will result in disqualification.
6) Time and memory limits are stated in the corresponding problem.
7) If any issues arise during the contest, contact the proctors.

Scoring:

1) The team to solve the most problems wins.
2) Ties will be broken by penalty points.
3) Wrong submissions result in 20 minutes of penalty points.
4) Correct submissions give you penalty points equal to the elapsed time in the contest.

Good Luck!

This Page Is Intentionally Left Blank

Fast I/O:

Several of the problems this year are I/O-intensive. As a result, we have provided a short description on how to quickly take in input and process output in each suite of languages...

C++:

Add the following to the beginning of your main method:

```
1: ios_base::sync_with_stdio(false);
2: cin.tie(nullptr);
```

Also, only use `std::endl` sparingly since it flushes output, which can be expensive if done repeatedly.

Java:

Add the following to the beginning of your main method:

```
1: import java.io.BufferedReader;
2: import java.io.IOException;
3: import java.io.InputStreamReader;
4: import java.io.PrintWriter;
5: import java.util.StringTokenizer;
6: ⋮
7: BufferedReader file = new BufferedReader(new InputStreamReader(System.in));
8: PrintWriter out = new PrintWriter(System.out);
9: ⋮
10: StringTokenizer st = new StringTokenizer(file.readLine());          // Must redefine for a new line of input
11: int a = Integer.parseInt(st.nextToken());          // Pulls the next token from the line of input from line 9
12: ⋮
13: file.close();                                                      // Closes standard input
14: out.close();                                          // Flushes output and closes standard output
```

Use the method `file.readLine()` to read in an entire line of input (null if no such line exists), and use a `st.nextToken()` to read in the next contiguous block of data from a whitespace-separated line.

Python:

Use `sys.stdin` and `sys.stdout` instead of the built-in `input()` and `print()` as this can speed up I/O operations. Add the following imports:

```
1: import sys
```

To read input, use:

```
1: data = sys.stdin.read()                                    // Reads all input at once
2: lines = data.splitlines()                                  // Splits input into lines
```

For output, use:

```
1: sys.stdout.write(str(output) + "\n")          // Use sys.stdout.write() instead of print()
```

Kotlin:

Kotlin supports fast I/O through essentially the same I/O classes that Java uses. Add the following to your main function:

```
import java.io.BufferedReader
import java.io.IOException
import java.io.InputStreamReader
import java.io.PrintWriter
```

To read input:

```
val reader = BufferedReader(InputStreamReader(System.`in`))
val out = PrintWriter(System.out)
⋮
val line = reader.readLine()
val tokens = line.split(" ")
val a = tokens[0].toInt()
```

To output results:

```
out.println(result)                    // Use PrintWriter for fast output
out.flush()                            // Ensure to flush the output
```

In Kotlin, avoid using the built-in `readLine()` and `println()` too often, as they are relatively slower. Use `BufferedReader` and `PrintWriter` for more efficient I/O operations.

C:

In C, the standard functions `scanf()` and `printf()` are typically used for I/O. Here are a few tips for fast I/O in C:

```
#include <stdio.h>
⋮
int a;
scanf("%d", &a);                       // Use scanf() for input
printf("%d\n", a);                     // Use printf() for output
```

Tips for fast I/O in C:

- Use `scanf()` and `printf()` instead of other more complex I/O methods for speed.
- Avoid flushing output buffers repeatedly. Using `fflush(stdout)` inside loops can be costly, so only flush when necessary.
- Batch input and output when possible to minimize the number of function calls.

Bishop Attack

Time Limit: 1 second
Memory Limit: 512 megabytes
Standard Input, Standard Output

Problem Statement

A chess board has columns 'a' through 'h' and rows 1 through 8 resulting in a board with a total of 64 squares. The adjacent squares alternate colors between black and white: square 'a1' (column 'a' and row 1) is black, 'a2' is white, 'b1' is white, and so on. While many people's favorite piece is the Queen, Temoc's favorite piece remains as the Bishop.

A bishop is a piece that attacks all squares on the diagonals on which the bishop is placed. A bishop placed on the square 'd4' attacks the squares 'd4', 'e5', 'c5', 'c3', 'e3', 'f6', etc. as long as there is a direct line of sight along a diagonal.

You are given three distinct positions: an attacking bishop, A, a target bishop, B, and a pawn, P. Note that if A and B are in the same diagonal but P is in between them, then A cannot reach B in one move. Additionally, P can neither be jumped over nor captured. Temoc wants to know the minimum number of moves needed for A to reach B under this constraint if it is possible to do so.

Input Format

The input consists of three lines. Each line consists of a single character, c, a space, and a single integer i denoting the square that the piece is on. Line 1 denotes the position of A, line 2 for B, and line 3 for P.

Constraints

$c \in \{\mathrm{a}, \mathrm{b}, \ldots, \mathrm{h}\}$
$i \in \{1, 2, \ldots, 8\}$

Output Format
Print a single integer n indicating the minimum number of moves A needs to reach B, or 0 if it is not possible to do so.

Sample I/O 1

Input	Output
a 1 h 8 d 4	5

Explanation 1
A possible valid solution of minimal distance is a1 → c3 → b4 → d6 → e5 → h8. Note that since we cannot capture the pawn, we must go around it.

Sample I/O 2

Input	Output
a 1 h 8 g 7	0

Explanation 2
Since we cannot capture the pawn, we are unable to reach B from A.

Chess Board
A sample chessboard has been provided below for your convenience.

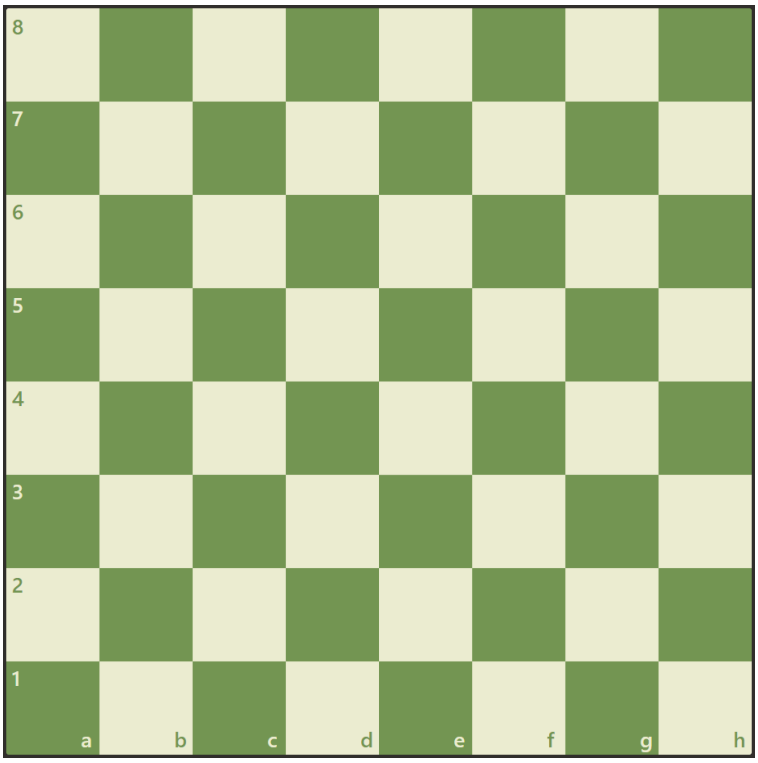

Figure 1: A Sample Chess Board

Two Brains Are Better Than One (Easy)

Time Limit: 1 second
Memory Limit: 512 megabytes
Standard Input, Standard Output

This is the easy version of the problem. The only difference from the hard version is the constraint on n and r_i. The easy version appears in both packets, while the hard version only appears in the advanced packet.

Problem Statement
The students at UTD are known as Comtets, and each Comet has a rating, r. A "sub-rating" of some Comet's rating, r_i, is a value v such that r_i is an integer multiple of v. Note: values 1 and r are both considered sub-ratings of r.

Let x be the number of values that are a sub-rating of r. A Comet's rating is said to be "cool" if x is an odd number ("cool" refers to the temperature of a Comet, not any kind of perceived social status).

Comets are paired to form teams. The rating of a team R is the product of the rating of the two Comets on the team. The rating of a team is said to be cool if, when separate, the two Comet's ratings are not cool, but the team's rating as a whole is cool.

Temoc, being the caring mascot that he is, is interested in knowing how many unique teams are considered to be cool given a set of ratings for various different Comets.

Input Format
The first line of input consists of an integer n denoting the number of Comet ratings to follow. The second line of input consists of n space-separated integers, $r_1, r_2, ..., r_n$, denoting the ratings of the n Comets. Rating r_i is considered to be the rating of the i^{th} Comet.

Constraints
$1 \leq n \leq 10^4$
$2 \leq r_i \leq 10^9$

Output Format
Output a single non-negative integer, p, where p is the number of unique team pairings that are considered to be cool.

Sample I/O 1

Input	Output
5 2 3 4 7 2	1

Explanation 1
The only non-cool number here is $r_2 = 4$. However, the only cool team is $(r_0, r_4) = (2, 2)$.

Sample I/O 2

Input	Output
8 32 2 5 20 3 242 242 363	8

Two Brains Are Better Than One (Hard)

Time Limit: 1 second
Memory Limit: 512 megabytes
Standard Input, Standard Output

This is the hard version of the problem. The only difference from the easy version is the constraint on n and r_i. A copy of the problem description has been provided to you for your convenience.

Problem Statement

The students at UTD are known as Comtets, and each Comet has a rating, r. A "sub-rating" of some Comet's rating, r_i, is a value v such that r_i is an integer multiple of v. Note: values 1 and r are both considered sub-ratings of r.

Let x be the number of values that are a sub-rating of r. A Comet's rating is said to be "cool" if x is an odd number ("cool" refers to the temperature of a Comet, not any kind of perceived social status).

Comets are paired to form teams. The rating of a team R is the product of the rating of the two Comets on the team. The rating of a team is said to be cool if, when separate, the two Comet's ratings are not cool, but the team's rating as a whole is cool.

Temoc, being the caring mascot that he is, is interested in knowing how many unique teams are considered to be cool given a set of ratings for various different Comets.

Input Format

The first line of input consists of an integer n denoting the number of Comet ratings to follow. The second line of input consists of n space-separated integers, $r_1, r_2, ..., r_n$, denoting the ratings of the n Comets. Rating r_i is considered to be the rating of the i^{th} Comet.

Constraints

$1 \leq n \leq 2 \cdot 10^5$
$2 \leq r_i \leq 10^6$

Output Format

Output a single non-negative integer, p, where p is the number of unique team pairings that are considered to be cool.

Sample I/O 1

Input	Output
5 2 3 4 7 2	1

Explanation 1

The only non-cool number here is $r_2 = 4$. However, the only cool team is $(r_0, r_4) = (2, 2)$.

Sample I/O 2

Input	Output
8 32 2 5 20 3 242 242 363	8

Cylinder Volume

Time Limit: 1 second
Memory Limit: 512 megabytes
Standard Input, Standard Output

Problem Statement

Temocs live in a land where food can be grown only for a part of the year, and due to the harsh temperatures, they need to stay indoors during the winter. Because of this, they must store their food, grains, water, fuel, etc. so that they can survive the harsh winter.

Their method of storage is to make cylinders whose total height equals that of the town's storage godown, stacking the cylinders one on top of each other. Moreover, the cylinder at the bottom of the pile must have the greatest radius and the the one at the top must have the least radius. The same applies to the height of the cylinders. More generally, if cylinder x is on top of cylinder y, then cylinder y must have a radius that is strictly greater than x's radius, and it must also have a height that is strictly greater than x's height. All heights and radii of any cylinder are positive integers measured in centimeters.

Figure 2: An Example of a Storage Godown

Every year a set of lucky numbers is given to the population of Temocs by the food Oracle who states the exact number of cylinders they should make, as well as the sum of the radii of all of those cylinders, in order to increase their fortune during the winter time. Your goal is to maximize the volume of the cylinder at the top of the stack while abiding by all design restrictions.

Input Format

The first line consists of the number of test cases, t. The remaining input denotes the t test cases.

Each test case consists of a single line with three space-separated positive integer values h, r, and k. Here, h is the sum of the height of all k cylinders, r is the sum of the radii of all k cylinders, and k is the number of cylinders to be made.

Constraints

$1 \leq t \leq 5 \cdot 10^5$
$1 \leq h, r \leq 10^{18}$
$1 \leq k \leq 10^9$

Output Format

For each test case print the volume of the topmost cylinder divided by $\pi \cdot r_{\text{top-most}}$ in a line by itself. Since this value can get quite large, print your answer modulo $10^9 + 7$. If the arrangement is impossible, then print 0.

Sample I/O 1

Input	Output
2 4 5 1 875 561 8	20 6930

Explanation 1

In the first test case, there is only a single cylinder, so we are free to let it be the sum of the height and radii.

In the last test case, we can make a topmost cylinder with height 105 and radius 66.

A Big Bang Theory

Time Limit: 4 second
Memory Limit: 512 megabytes
Standard Input, Standard Output

Problem Statement

Dr. Sheldon is working on a chemistry experiment, instead of something physics related. Namely, he has N different types of double-bond chemical elements and M different types of single-bond chemical elements, and he is trying to figure out for each single-bond element x, how many double-bond elements can be paired with x.

Formally, a double-bond element is defined by a tuple (x, y) and a single-bond chemical element is defined by a single value z. A single-bond element x can be paired with a double-bond element (y, z) if and only if $gcd(y, z)$ is a multiple of x. Can you help Dr. Sheldon write a program to help him determine for each single-bond element the number of double-bond elements that can be paired with it.

Input Format

The first line of the input contains 1 integer N. The next N lines contain 2 integers x_i, y_i, denoting the i^{th} double-bond element. The $(N + 2)^{th}$ line contains 1 integer M. The last M lines denote the single-bond elements that Dr. Sheldon has, the i^{th} of which denotes that Dr. Sheldon has element z_i.

Constraints

$1 \leq N, M \leq 2 * 10^5$
$1 \leq x_i, y_i, z_i \leq 2 * 10^5$

Output Format

Print M lines, the i^{th} of which contains a single integer denoting number of double-bond elements that can be pair with the i^{th} single-bond element.

Sample I/O

Input	Output
5 3 2 20 15 6 12 15 18 14 21 2 3 2	2 1

All The Rage

Time Limit: 1 second
Memory Limit: 512 megabytes
Standard Input, Standard Output

Problem Statement

Billy is a psychology researcher at UTD looking into an unknown phenomenon that he has called "Engulfing Rage", a condition that spreads through social connection. He is studying an isolated group of N individuals. With his flawless (and kind of creepy) investigative skills, has made note of all M friendships between these individuals. He also knows all the people that have ER on day 1.

Billy hypothesizes that when a person has Engulfing Rage (ER), their rage tends to influence their friends, giving them ER as well. Formally, if Person 1 is friends with Person 2, and Person 1 has ER on day i, Person 2 will have ER on day $i + 1$.

Assuming his hypothesis is correct, Billy wants to know for each person, by when should they have ER?

Input Format

The first line of input contains two space-separated integers: N, the number of people, and M, the number of friendships.

The next line contains N integers, the i^{th} of which is 1 if person i is infected on day 1, or 0 if not.

The next M lines contain two integers, u and v, representing a friendship between person u and person v. There are no duplicate edges or self-loops.

Constraints

$3 \leq N \leq 10^5$
$0 \leq M \leq 2 \cdot 10^5$
$1 \leq u, v \leq N$, $u \neq v$

Output Format

The output should contain N lines, the i^{th} line having the number of the day person i gets ER, or -1 if they never get ER.

Sample I/O 1

Input	Output
7 6 0 0 0 0 1 0 0 1 2 2 3 3 4 1 4 2 5 6 7	3 2 3 4 1 -1 -1

Explanation 1

1) Person 5 is initially infected
2) On day 2, person 2 is newly infected
3) On day 3, person 1 and person 3 are newly infected
4) On day 4, person 4 are newly infected
5) Person 6 and Person 7 are never infected

Sample I/O 2

Input	Output
4 2 1 1 0 0 1 3 2 4	1 1 2 2

Comet Camp Crisis

Time Limit: 2 second
Memory Limit: 512 megabytes
Standard Input, Standard Output

Problem Statement

The UTD administration maintains a network of $n-1$ covered walkway paths with different lengths connecting n buildings across campus. The network of paths ensure that you can reach a building from any other building, with building indexed 1 the central hub. In other words, the buildings and paths structure on campus make up a rooted tree with weighted edges.

You are asked to solve q independent problems for the administration team. In each problem, building x and all buildings in its subtree are closed down, they need you to calculate the longest path between any two accessible buildings.

Input Format

The first line consists of two space-separated integers, n and q - the number of buildings and the number of problems.

The next $n-1$ lines describe the edges of the tree, where the i-th line consists of three space-separated integers u_i, v_i, w_i, representing a path of length w_i between building u_i and v_i.

The next q lines describe the problems by UTD administration, where the i-th line consists of a single integer x_i, requiring you to solve the problem for building x_i.

Constraints

$1 \leq n, q \leq 2 \times 10^5 \ \forall i,\ 1 \leq i \leq n-1 :\ 1 \leq w_i \leq 100 \ \forall i,\ 1 \leq i \leq q :\ 1 \leq x_i \leq n, x_i \neq 1$

Output Format

Output q lines where i^{th} line has the length of the longest path between any two accessible buildings if building x and all buildings in its subtree are closed down.

Sample I/O 1

Input	Output
5 5 1 3 1 4 2 4 3 5 2 1 4 3 3 5 2 2 3	7 8 6 6 7

Explanation 1

Query 1: Building 3 and all buildings in its subtree are blocked off, the longest path is from building 1 to building 2 with total length 7.

Query 2: Building 3 and all buildings in its subtree are blocked off, the longest path is from building 2 to building 3 with total length 8.

Query 3: Building 2 and all buildings in its subtree are blocked off, the longest path is from building 5 to building 4 with total length 6.

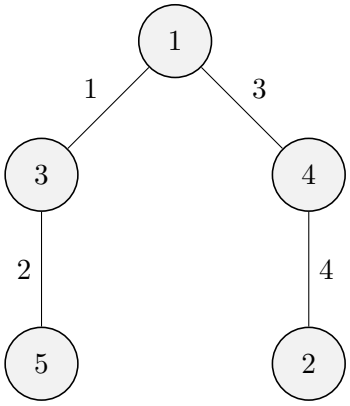

Figure 3: Visualization of Sample I/O 1

Sample I/O 2

Input	Output
10 6 9 1 5 9 6 1 2 10 2 6 7 4 9 3 2 2 3 3 8 1 1 9 4 7 7 5 6 10 7 5 3 2 2	18 14 14 18 18 18