Problem A
Olympic Ranking

Time limit: 1 second

Memory limit: 1024 megabytes

Problem Description

The Olympic Games are the most important sporting events in human history. Usually, the Olympic Games are held every four years. However, the 2020 Olympic Games just finished in August 2021 due to the COVID-19 outbreaks.

There are more than 200 nations participating the Olympic Games. Thousands of athletes around the world compete in various sports. Athelete represent their countries or National Olympuc Committees (NOCs) to compete for medals. Therefore, each country or NOC may win medals in the competitions.

There are three types of medals: gold medals, silver medals, and bronze medals. Typically, the gold medals are awarded to the winner of the competitions, and silver medals are awarded to the runner-up. Most of the bronze medals are awarded to the second runner-up. However, there can be no second runner-up in some sports. The bronze medals are awarded in different manners. For example, in a few tournament sports, such as wrestling, boxing, and judo, two bronze medals are awarded to the eliminated semi-finalists.

A country or NOC has a better rank than another country or NOC if one of the following conditions holds.

1. It wins more gold medals.
2. It wins the same ammount of gold medals, and it wins more silver medals.
3. It wins the same ammount of gold medals and silver medals, and it wins more bronze medals.

Please write a program to find the country or NOC which has the best rank among all countries and NOCs.

Input Format

The first line of the input contains one positive integer n. Then n lines follow. Each of the following lines contains three non-negative integers g, s, b, and the name of a country or NOC. They are separated by blanks.

Output Format

Print the name of the country or NOC of the best rank.

Technical Specification

- $1 \leq n < 300$

- $g, s, b \in \{0, 1, \ldots, 999\}$
- There is only one country or NOC of the best rank.
- The names of countries and NOCs consists of only printable ASCII characters.
- The size of an input file does not exceed 3 megabytes.

Sample Input 1

```
4
22 21 22 Great Britain
27 14 17 Japan
39 41 33 United States of America
20 28 23 ROC
```

Sample Output 1

```
United States of America
```

Sample Input 2

```
3
999 999 998 Malaysia
999 999 999 Thailand
999 998 999 Indonesia
```

Sample Output 2

```
Thailand
```

Problem B
Aliquot Sum

Time limit: 10 seconds
Memory limit: 1024 megabytes

Problem Description

A divisor of a positive integer n is an integer d where $m = \frac{n}{d}$ is an integer. In this problem, we define the aliquot sum $s(n)$ of a positive integer n as the sum of all divisors of n other than n itself. For examples, $s(12) = 1 + 2 + 3 + 4 + 6 = 16$, $s(21) = 1 + 3 + 7 = 11$, and $s(28) = 1 + 2 + 4 + 7 + 14 = 28$.

With the aliquot sum, We can classify positive integers into three types: abundant numbers, deficient numbers, and perfect numbers. The rules are as follows.

1. A positive integer x is an abundant number if $s(x) > x$.
2. A positive integer y is a deficient number if $s(y) < y$.
3. A positive integer z is a perfect number if $s(z) = z$.

You are given a list of positive integers. Please write a program to classify them.

Input Format

The first line of the input contains one positive integer T indicating the number of test cases. The second line of the input contains T space-separated positive integers $n_1, \ldots, n_T$.

Output Format

Output T lines. If n_i is an abundant number, then print `abundant` on the i-th line. If n_i is a deficient number, then print `deficient` on the i-th line. If n_i is a perfect number, then print `perfect` on the i-th line.

Technical Specification

- $1 \le T \le 10^6$
- $1 \le n_i \le 10^6$ for $i \in \{1, 2, \ldots, T\}$.

Sample Input 1

```
3
12 21 28
```

Sample Output 1

```
abundant
deficient
perfect
```

Almost blank page

Problem C
A Sorting Problem

Time limit: 3 seconds

Memory limit: 1024 megabytes

Problem Description

You are given an array $[p[1],\ p[2],\ \ldots,\ p[n]]$ where all the numbers in the array are distinct. In addition, the numbers are positive integers between 1 and n. You can only perform the following operations on the array: Pick two indices x and y such that $|p[x]-p[y]| = 1$, and then swap the values of $p[x]$ and $p[y]$. We now want to sort this array in ascending order. That is, to make $p[i] = i$ for all $i \in \{1, 2, \ldots, n\}$. For example, we can sort the array $[p[1] = 2, p[2] = 3, p[3] = 1]$ in two operations:

1. Swap $p[1]$ and $p[3]$. The array becomes $[p[1] = 1, p[2] = 3, p[3] = 2]$.
2. Swap $p[2]$ and $p[3]$. The array becomes $[p[1] = 1, p[2] = 2, p[3] = 3]$ which is sorted in ascending order.

Please write a program to compute the minimum number of operations to sort a given array.

Input Format

The input contain two lines. The first line contains one integer n. The second lines contain n space-saparated numbers $p[1], p[2], \ldots, p[n]$ representing the array $[p[1],\ p[2], \ldots,\ p[n]]$

Output Format

Output only one number that denotes the minimum number of operations required to sort the given array.

Technical Specification

- $1 < n \leq 200000$.
- $1 \leq p[i] \leq n$.
- All $p[i]$ are distinct.

Almost blank page

Problem D
Drunk Passenger

Time limit: 2 seconds

Memory limit: 1024 megabytes

Problem Description

Due to COVID-19, social distancing is applied in our daily life to prevent the spread of the disease. It changes our living styles a lot, especially the way of traveling. Now, many carriers cancel non-reserved seats and introduce seating rules to ensure that the distance between any two passengers is long enough.

On your trip to the 2022 ICPC World Finals, you take a flight. The airplane provides n reserved seats to n passengers. The passengers must queue up first, then they board the airplane one by one. You are the last passenger to board, since you are at the end of the queue. Unfortunately, the first passenger is drunk. The drunk passenger randomly goes to another passenger's seat and then sit there. You may assume the following.

1. The drunk passenger never takes their own seat.
2. The probability of any other seat taken by the drunk passenger is uniform.

Luckily, all the other passenger are not drunk. However, they don't want to move any passenger from a taken seat. If a passenger's seat has been taken by another passenger when boarding, the passenger would randomly take a vacant seat with equal probability. Otherwise, the passengers just take their own seats.

Please write a program to compute the probability that your seat is taken by another passenger.

Input Format

The input contains only one positive integer n.

Output Format

Output the probability that your seat is taken by another passenger. It is acceptable if the difference between your output and the answer is less then 10^{-6}.

Technical Specification

- $1 < n \leq 300$

Sample Input 1

```
2
```

Sample Output 1

```
1
```

Sample Input 2

```
3
```

Sample Output 2

```
0.75
```

Sample Input 3

```
4
```

Sample Output 3

```
0.6666666666666666666666667
```

Note

The problem statement is a fiction.

Problem E
Mountain Park

Time limit: 5 seconds

Memory limit: 1024 megabytes

Problem Description

Input Format

Output Format

Technical Specification

Almost blank page

Problem F
F

Time limit: 3 seconds

Memory limit: 1024 megabytes

Problem Description

Suppose you are given an array of n entries where each entry of the array is either 0 or 1. A subarray $[a[\ell], a[\ell+1], \ldots, a[r]]$ of an array $[a[1], a[2], \ldots, a[n]]$ if $1 \leq \ell \leq r \leq n$. An alternating subarray $[a[\ell], a[\ell+1], \ldots, a[r]]$ of $[a[1], a[2], \ldots, a[n]]$ if $a[\ell] \neq a[\ell+1] \neq \cdots \neq a[r]$. I.e., an arbitrary entry in the subarray is different from its neighbors in the subarray.

In this problem, two types of operations will be applied on the given array.

- $1\ \ell\ r$: for every $i \in [\ell, r]$, change $a[i]$ into $1 - a[i]$.
- $2\ \ell\ r$: report the number of pairs (x, y) such that $\ell \leq x \leq y \leq r$ and subarray $[a[x], a[x+1], \ldots, a[y]]$ is an alternating subarray.

Please write a program to maintain the given array. Your program must report the numbers efficiently.

Input Format

The first line contains two integers n and q indicating the length of the given array and the number of operations. The second line contain n space saparated numbers $a[1], a[2], \ldots, a[n]$ representing the given array $[a[1], a[2], \ldots, a[n]]$. Then q lines follows, and the i-th of them contains 3 integers t_i, ℓ_i, r_i where the i-th operation is $t_i\ \ell_i\ r_i$.

Output Format

For each operation of the second type， output the reported number on one line.

Technical Specification

- $1 \leq n \leq 200000$
- $1 \leq q \leq 200000$
- $a[i] \in \{0, 1\}$ for all $i \in \{1, 2, \ldots, n\}$.
- $t_j \in \{1, 2\}$ for all $j \in \{1, 2, \ldots, q\}$.
- $1 \leq \ell_j \leq r_j \leq q$ for all $j \in \{1, 2, \ldots, q\}$.

Sample Input 1

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

Sample Output 1

```
6
```

Almost blank page

Problem G
G

Time limit: 3 seconds

Memory limit: 1024 megabytes

Problem Description

In a forest park, there are n places of interest (numbered from 1 to n) and $n-1$ trails (numbered from 1 to $n-1$) connecting the places of interest. For every $i \in \{1, 2, \ldots, n-1\}$, trail i has two ends at place a_i and place b_i, and the trail does not pass any place of interest except its ends. Moreover, the trails do not have any intersection except the ends.

To protect the forest, visitors may only walk along the trails (in any direction) and inside the places of interest. For any pair of places of interest (x, y) where $x \neq y$, there exists a sequence of trails $s_1, s_2, \ldots, s_k$ satisfying the following conditions.

- Place x is an end of trail s_1.
- Place y is an end of trail s_k.
- For $1 \leq i < k$, trail s_i and trail s_{i+1} have a common end.
- If place z is the common end of trails s_i and s_{i+1} for some $i \in \{1, \ldots, k-1\}$, then z cannot be a common end of any other pairs of trails in $s_1, \ldots, s_k$.

In other words, a visitor move from x to y by walking along the trails $s_1, s_2, \ldots, s_k$ without visiting a place of interest twice. Such sequence is called a simple path from x to y.

The administration division of the park plans to host an event in the park. It puts labels on the trails. For trail t, the label on t is an integer $\ell(t)$, and a visitor can learn $\ell(t)$ by walk through trail t. A simple path $s_1, s_2, \ldots, s_k$ from x to y is with strictly increasing labels if $\ell(s_1) < \ell(s_2) < \cdots < \ell(s_k)$. By reporting m distinct simple paths with strictly increasing labels to the administration division, a visitor may win m free tickets for future visits.

You friend George just visit the park, and learn all labels on the trails. He wants to win free tickets for future visits with you. Please write a program to compute the number of distinct simple paths with strictly increasing labels in the forest park.

Input Format

The first line contains one integers n . The $(i+1)$-th line contains three integers a_i, b_i, c_i. Trail i connects place a_i and b_i, and the label $\ell(i)$ on trail i is c_i.

Output Format

Output the number of distinct simple paths with strictly increasing labels in the forest park.

Technical Specification

- $1 \le n \le 2 \times 10^5$
- $1 \le a_i \le n$ for $i \in \{1, 2, \ldots, n\}$.
- $1 \le b_i \le n$ for $i \in \{1, 2, \ldots, n\}$.
- $0 \le c_i \le 10^9$ for $i \in \{1, 2, \ldots, n\}$.
- $a_i \ne b_i$ for $i \in \{1, 2, \ldots, n\}$.

Problem H

H

Time limit: 3 seconds

Memory limit: 1024 megabytes

Problem Description

You are given a simple undirected graph consisting of n nodes and m edges. Each node i has a value V_i and the weight of each edge (u, v) is V_u xor V_v.

There are q constraints represented in five variables `t u i v j`

- if $t = 0$, then $getBit(V_u, i) = getBit(V_v, j)$
- if $t = 1$, then $getBit(V_u, i) \neq getBit(V_v, j)$

where $getBit(x, i)$ is `(x >> i) & 1`.

Some values of nodes are missing. You need to assign them to make $\sum_{(u,v)\in E} \text{popcount}(V_u \text{ xor } V_v)$ as small as possible under the all given constraints where popcount(x) is the number of bit 1 in x.

Input Format

The first line of the input contains two positive integers n and m. Then m lines follow. Each of the following lines contains two integers u and v. Then one line containing n integers follow. Each integer denotes the value of each node. If the value of the node is missing, it will be `-1`. Then one integer q and q lines follow. Each line contains five integers t, u, i, v, j

Output Format

Output an integer which is the minimum value under the q constraints. If it is not possible to satisfy all the constraints, output `-1`.

Technical Specification

- $1 \le n \le 1000$
- $1 \le m \le 5000$
- $-1 \le V_i < 2^{16}$
- $0 \le q \le 12$
- $t \in \{0, 1\}$
- $0 \le u, v < n$
- $0 \le i, j < 16$

Sample Input 1

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

Sample Output 1

```
13
```

```
3 2
0 3
-1 -1 60091 51514
2
1 2 0 1 5
0 2 6 0 15
```

Sample Input 2

```
3 2
0 1
1 2
-1 -1 -1
2
1 2 0 1 5
0 1 5 2 0
```

Sample Output 2

```
-1
```

Problem I
I

Time limit: 3 seconds

Memory limit: 1024 megabytes

Problem Description

The ICPC kingdom has a n city, and there are m roads connecting these cities. Each road connected two cities and can travel in both directions. There are a_i residents living in the i-th city. The j-th road connects the u_j city and v_j city. And the economic benefit of the j-th road is $\lfloor\sqrt{a_{u_j} + a_{v_j}}\rfloor$.

One day, the enemy comes to the ICPC kingdom and destroys all the roads. The king of ICPC hired w workers to fix the road. The i-th worker can only fix the roads of index $b_1, b_2, b_3 ... b_{x_i}$. And each worker can only fix one road. Now the king wants to fix the road to make the kingdom normal. Considering the cost, the king does not want the fix plan to contain the useless roads. A plan exists for useless roads which means that there is a pair of city a, b, there is more than one simple path from a to b. A simple path is a path $c_1, c_2, c_3, ..c_z$ that $c_i \neq c_j$ for each $1 \leq i, j \leq z, i \neq j$. The king asked you to calculate that if we want to fix exactly k road without useless roads, how much is the maximum economic benefit. You need to calculate for each k between 1 and $n - 1$.

Input Format

The first line contains n and m seperated with white space indicating the number of city and number of road. The second line contains n number $a_1, a_2, a_3 ... a_n$ seperated with white space indicating the resident in each city. There are m lines following, each line contains u_j and v_j separated with white space indicating the road with index j connecting the city u_j and city v_j. The index of the road starts from 1. Next line contains a number w indicating the number of workers. The following w line indicates the kind of road which the work can fix. The i-th line contains a number x_i indicating the number of road that the i-th worker can fix and following contains x_i distinct number $b_1, b_2, b_3 ... b_{x_i}$ seperated with white space indicating the index of these road.

Output Format

Print $n - 1$ numbers, the i-th number indicating that if we wiant to fix exactly the i roads, how much is the maximum economic benefit. If there is no plan containing exactly i roads, you should print -1.

Technical Specification

- $1 \leq n \leq 100$
- $0 \leq m \leq 100$
- $1 \leq a_i \leq 10^9$
- $1 \leq u_i, v_i \leq n, u_i \neq v_i$

- $0 \leq w \leq 5000$

Sample Input 1

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

Sample Output 1

```
2 3 4 -1
```

Problem J
J

Time limit: 3 seconds

Memory limit: 1024 megabytes

Problem Description

Input Format

Output Format

Technical Specification