Problem A. Golden Spirit

Time limit: 1 second

It's a sunny day with good scenery, and you come to the park for a walk. You feel curious that there are many old guys gathering by a bridge, and want to take a look at what happened.

There are exactly n old guys on each side of the bridge, and they all want to go across the bridge, take some time for relaxing on the other side, and finally go across the bridge back to the original side. However, they are too old to cross the bridge by themselves.

Driven by the golden spirit in your heart, you want to help these $2n$ old guys. Initially, you are on one side of the bridge. It takes t minutes for you to go across the bridge and x minutes for an old guy relaxing. You may help an old guy when you cross the bridge, which doesn't take extra time.

As a master of time management, you want to know the minimum time needed to help all these $2n$ old guys. Please write a program to calculate this minimum time.

Input

The first line contains one integer T ($1 \leq t \leq 10^4$), indicating the total number of test cases.

For each of the next T lines, there are three integers n, x, t ($1 \leq n, x, t \leq 10^9$), as explained in problem statement.

Output

You should output exactly T lines. Each line should contain exactly one integer, indicating the minimum time you needed in each test case.

Sample Input 1	Sample Output 1
3 2 2 2 3 1 10 11 45 14	16 120 616

Note

For the first case of the sample data, the optimal plan is shown below, where a numerical digit denotes an old guy, `|` denotes the bridge, and `x` denotes you.

```
Time
  0     x 1 2 | 3 4
  2         2 | 3 4 1 x
  4     x 3 2 | 4 1
  6         3 | 4 1 2 x
  8     x 4 3 | 1 2 x
 10         4 | 1 2 3 x
 12     x 1 4 | 2 3
 14         1 | 2 3 4 x
 16     x 2 1 | 3 4
```

This page is intentionally left blank.

Problem B. Labyrinth

Time limit: 5 seconds

Due to the challenging problems, some of the contestants decide to escape from this contest. However, to prevent this from happening, the EVIL problem setters made a labyrinth at the stadium's exit. The labyrinth is made of an $n \times m$ grid, on which lie the entrance and the exit, and k black holes. Contestants who accidentally step into any black hole will fall into it and thus can never escape from the contest.

What's worse, the problem setters may also adjust the coordinates of the entrance and the exit. You, a poor contestant, who start from the entrance and wish to reach the exit without stepping into any of the black holes, can only move to one of the four adjacent cells in each step. You want to know, after each time the problem setters change the coordinates of the entrance and the exit, what's the minimum number of steps needed to reach the exit starting from the entrance?

Input

The first line of the input contains four integers n, m, k, q ($1 \le n, m \le 200\,000, nm \le 200\,000, 0 \le k \le 42$, $1 \le q \le 100\,000$), denoting the number of rows, the number of columns, the number of black holes in the labyrinth, and the number of queries, respectively.

The following k lines contain the description of the black holes. Each of these lines contains two integers x, y ($1 \le x \le n, 1 \le y \le m$), denoting the coordinates of a black hole. No two black holes are located at the same position.

The last q lines contain the description of the queries. Each of the q lines contains four integers x_s, y_s, x_t, y_t ($1 \le x_s, x_t \le n, 1 \le y_s, y_t \le m$), where (x_s, y_s) is the coordinates of the entrace and (x_t, y_t) is the exit.

Output

For each query, output a number in a line, denoting the minimum number of steps needed to reach the exit starting from the entrance. If it is impossible to reach the exit, output `-1` instead. It should be considered impossible when the entrace or the exit coincides with a black hole.

Sample Input 1

```
5 5 4 7
2 2
2 3
3 2
3 3
2 1 3 4
1 1 1 1
2 2 2 2
1 1 1 5
2 2 5 5
2 1 2 4
1 1 3 3
```

Sample Output 1

```
6
0
-1
4
-1
5
-1
```

Sample Input 2

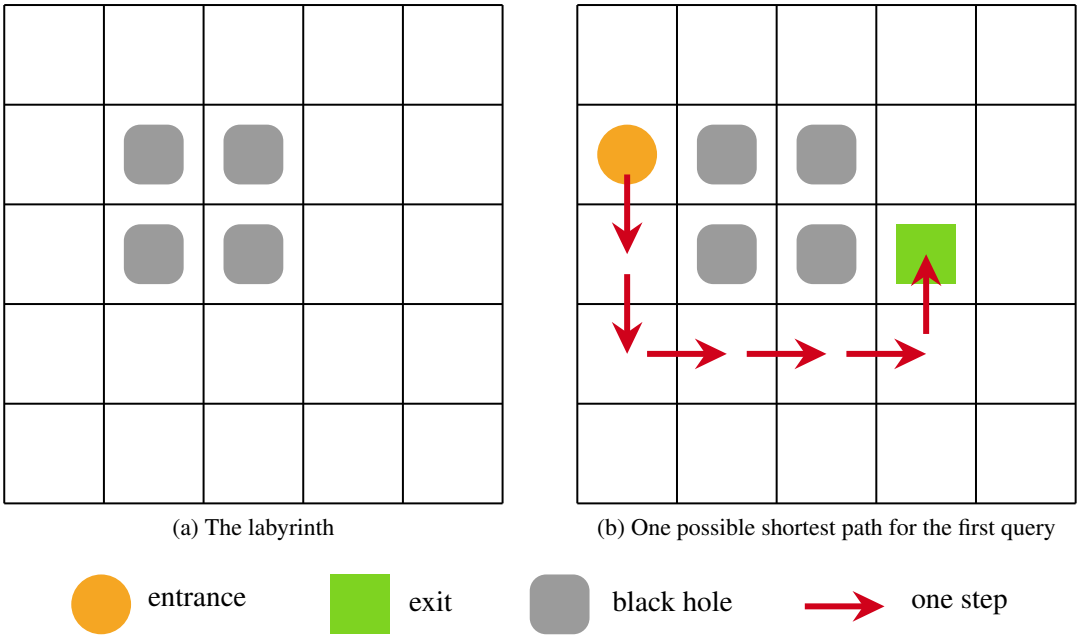
```
2 3 2 1
1 2
2 1
1 1 2 3
```

Sample Output 2

```
-1
```

Note

The plots for the labyrinth and the first query of the first sample data are shown below.



(a) The labyrinth

(b) One possible shortest path for the first query

Figure 1: Plots for sample test data

Problem C. Rencontre

Time limit: 2 seconds

Located at the easternmost tip of Shandon Peninsula, Weihai is one of the most famous tourist destinations all over China. There are beautiful hills, seas, bays, springs, islands, and beautiful beaches in Weihai. It is also a coastal city abundant in seafood, including prawn, sea cucumber, abalone, shellfish, and algae.

Attracted by the distinctive scenery and pleasant environment, three theoretical computer scientists plan to have a trip to Weihai. However, they cannot reach a consensus on accommodation, since some people prefer some hotels while other people like others. They decide to stay in possibly different hotels at night and meet in one hotel the next day. The hotel they meet may not necessarily be one of the hotels they stay in.

There are some roads connecting the hotels in Weihai. The roads are specially designed such that there is a unique path between every pair of hotels. Every theoretical computer scientist has prepared a list of candidate hotels before their trip starts. When they arrive in Weihai, each of them will uniformly and independently choose one hotel from the candidate hotel list. Also, they will meet in a hotel such that the total length of their routes is minimized. As a member of the theoretical computer science group, can you tell the expected total length of their routes?

Input

The first line of the input contains a single integer n $(1 \leq n \leq 200\,000)$, denoting the number of hotels in Weihai. Then follow $n-1$ lines, describing the roads connecting the hotels. Each of the $n-1$ lines contains three integers u, v, w $(1 \leq u, v \leq n, u \neq v, 1 \leq w \leq 1000)$, denoting a road of length w connecting the hotels numbered u and v. It is guaranteed that there is a unique path between every pair of hotels.

The last three lines of the input specify the candidate hotel lists, one for each theoretical computer scientist. Each line begins with a single integer m $(1 \leq m \leq n)$ and m distinct integers $a_1, a_2, \cdots, a_m$ $(1 \leq a_i \leq n)$, meaning that the candidate hotel list contains the hotels numbered $a_1, a_2, \cdots, a_m$.

Output

Print the expected total length of their routes within an absolute or relative error of no more than 10^{-6}.

Sample Input 1

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

Sample Output 1

```
3
```

Sample Input 2

```
5
1 2 3
1 3 5
2 4 7
2 5 11
3 2 4 5
4 1 2 3 5
2 1 3
```

Sample Output 2

```
13.958333333333
```

This page is intentionally left blank.

Problem D. ABC Conjecture

Time limit: 3 seconds

The ABC conjecture (also known as the Oesterlé–Masser conjecture) is a famous conjecture in number theory, first proposed by Joseph Oesterlé and David Masser. It is formally stated as follows:

> For every positive real number ε, there are only finitely many positive integer triples (a, b, c) such that
>
> 1. a and b are relatively prime;
> 2. $a + b = c$; and
> 3. $c > \mathrm{rad}(abc)^{1+\varepsilon}$,
>
> where
>
> $$\mathrm{rad}(n) = \prod_{\substack{p \mid n \\ p \in \text{Prime}}} p$$
>
> is the product of all distinct prime divisors of n.

Figure 2: Shinichi Mochizuki

Shinichi Mochizuki claimed to have proven this conjecture in August 2012. Later, Mochizuki's claimed proof was announced to be published in *Publications of the Research Institute for Mathematical Sciences* (RIMS), a journal of which Mochizuki is the chief editor.

Spike is a great fan of number theory and wanted to prove the ABC conjecture as well. However, due to his inability, he turned to work on a weaker version of the ABC conjecture, which is formally stated as follows:

> Given a positive integer c, determine if there exists positive integers a, b, such that $a + b = c$ and $\mathrm{rad}(abc) < c$.

Note that in the original ABC conjecture, the positive integers a and b are required to be relatively prime. However, as Spike is solving an easier version of the problem, this requirement is removed.

Input

The first line of input contains one integer T $(1 \le T \le 10)$, the number of test cases.

The next lines contain description of the t test cases. Each test case contains one line, including an integer c $(1 \le c \le 10^{18})$.

Output

For each test case, if there exist two positive integers a, b satisfying $a + b = c$ and $\mathrm{rad}(abc) < c$, then output `yes` in a line, otherwise output `no` instead.

Sample Input 1	Sample Output 1
3 4 18 30	yes yes no

Note

For the first test case, we have $2 + 2 = 4$ and $\mathrm{rad}(2 \times 2 \times 4) = 2 < 4$.

For the second test case, we have $6 + 12 = 18$ and $\mathrm{rad}(6 \times 12 \times 18) = 6 < 18$.

For the third test case, there's no solution.

Problem E. So Many Possibilities...

Time limit: 3 seconds

Hearthstone is a free-to-play online digital collectible card game developed and published by Blizzard Entertainment. The game is a turn-based card game between two opponents, using constructed decks of 30 cards along with a selected hero with a unique power. Players use their limited mana crystals to cast spells or summon minions to attack the opponent, with the goal of destroying the opponent's hero.

Kurisu loves playing Hearthstone and plays her Highlander Mage deck very well. However, sometimes she isn't lucky enough, so that when she plays the card *Reno the Relicologist* (as shown in the image below), it always fails to kill many of the enemy minions. Now she wonders, given the n enemy minions on board and their remaining health, if she may deal m damage randomly split among them, what is the expected number of enemy minions that would be killed?

Figure 3: Reno the Relicologist

One may think of dealing m damage randomly split among minions as m separate one-damage hits, processed one by one. Each one-damage hit targets a minion with positive health uniformly at random, and then decreases its health by one. Note that the one-damage hit will never be distributed to a dying minion, i.e., a minion whose health is already zero.

Since there are so many possibilities in this process, please help her calculate the expected number of minions with zero health after the process above.

Input

The first line of input contains two integers n ($1 \le n \le 15$) and m ($1 \le m \le 100$), the number of minions and the total amount of damage to distribute.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 100, \sum a_i \ge m$), where a_i denotes the remaining health of the i-th minion.

Output

Output the expected number of minions killed during the process within an absolute error of no more than 10^{-6}.

Sample Input 1	Sample Output 1
2 2 2 2	0.5

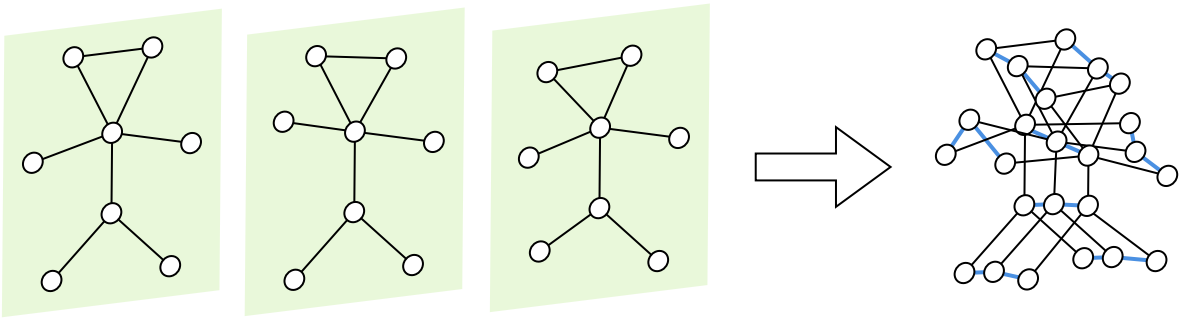
Sample Input 2	Sample Output 2
3 3 1 2 3	1.0833333333333333

Problem F. Skeleton Dynamization

Time limit: 3 seconds

The skeleton data has been widely used in computer vision tasks such as action recognition. In the skeleton model, the human body is represented by a set of body joints interconnected by bones. This naturally forms an undirected graph model: vertices are joints and edges are bones.

To incorporate the dynamics of the skeletons in a video, we may keep track of the joints across the frames and build a *spatial-temporal graph* for the video. The spatial-temporal graph consists of the skeleton graphs of every frame, with additional inter-frame edges connecting the same joints in two adjacent frames. Note that the skeleton graph should keep the same in all frames of the video. The following picture exhibits how a spatial-temporal graph is formed.



skeleton graphs of 3 adjacent frames

spatial-temporal graph

Figure 4: Formation of a spatial-temporal graph

Your task is to reverse-engineer a spatial-temporal graph. Formally, you should assign a pair of integers (f_v, s_v) to every vertex v of the graph, where f_v is the frame number and s_v is the index of the joint. Let T, S denote the number of frames and the number of joints, respectively, then your labeling must simultaneously satisfy the following conditions:

- for every $1 \le t \le T$ and $1 \le i \le S$, exactly one vertex is labeled (t, i);
- there is an edge between vertices labeled (t, i) and $(t + 1, i)$ for every $1 \le t < T$ and $1 \le i \le S$; there are no inter-frame edges other than those mentioned before;
- for every $1 \le t_1 < t_2 \le T$ and $1 \le i < j \le S$, there is an edge between vertices labeled (t_1, i) and (t_1, j) if and only if so between (t_2, i) and (t_2, j).

Input

The first line of the input consists of two integers n, m ($1 \le n \le 100\,000, 0 \le m \le 200\,000$), denoting the number of vertices and edges in the given spatial-temporal graph respectively. The vertices of the graph are indexed 1 through n.

Each of the remaining m lines of the input contains two integers u, v ($1 \le u, v \le n, u \ne v$), denoting an undirected edge connecting vertices indexed u and v. Each pair of vertices is connected by at most one edge. The input graph is guaranteed to be connected.

Output

Print two integers T, S in the first line of your output, denoting the number of frames and the number of joints, respectively. Then print T lines, each containing S integers; the s-th integer of the t-th line is the index of the vertex labeled (t, s).

If multiple valid labelings exist, print the one with the maximum number of frames. If there are still multiple, any one is acceptable.

Sample Input 1

```
12 20
5 12
6 10
8 1
11 3
5 1
12 4
12 2
11 8
2 8
6 4
7 11
9 1
8 10
9 6
4 1
2 5
10 3
7 2
8 4
9 10
```

Sample Output 1

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

Sample Input 2

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

Sample Output 2

```
1 3
1 2 3
```

Sample Input 3

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

Sample Output 3

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

Problem G. Caesar Cipher

Time limit: 10 seconds

Have you ever heard of the Caesar cipher? It is one of the simplest and best-known encryption techniques. Named after Julius Caesar, he used this cipher to communicate with his generals.

Caesar cipher is a type of substitution cipher in which each letter in the plaintext is *shifted* a certain number of places down the alphabet. The alphabet is considered wrapped around. For example, with a shift of 1 in the Latin alphabet, A would be replaced by B, B would become C, Z would be A, and so on.

Kazusa now has an array a of integers $a_1, a_2, \ldots, a_n$ of length n, with each a_i in the range $[0, 65\,536)$, and she wants to encrypt it using Caesar cipher several times. She selects an interval $[l, r] (1 \le l \le r \le n)$ each time, and makes an encryption using Caesar cipher with shift of 1 to the numbers in the interval. Formally, for all $l \le i \le r$, this transforms a_i into $(a_i + 1) \bmod 65\,536$.

However, while Kazusa is encrypting the array, her sister, Setsuna, raises some questions about the array. Each query asks on the current copy of the array, where zero or more encryptions using Caesar cipher has been done. Each query is given by three integers x, y, L, which asks whether the two strings $a_x, a_{x+1} \ldots a_{x+L-1}$ and $a_y, a_{y+1} \ldots a_{y+L-1}$ are same.

While Kazusa is busying doing the encryption, she has no time to answer these queries. Could you please help her?

Input

The first line contains two integers n, q $(1 \le n, q \le 500\,000)$, denoting the size of the array and the number of operations, respectively. The second line contains n integers $a_1, a_2, \ldots, a_n$ $(0 \le a_i < 65\,536)$, denoting the initial array. The next q lines describe the operations, each operation is in one of the two following types:

- Operation of type 1 contains three integers $1, l, r$ $(1 \le l \le r \le n)$, meaning that Kazusa made an encryption using Caesar cipher with shift 1 to the numbers in the interval $[l, r]$;
- Operation of type 2 contains four integers $2, x, y, L$ $(1 \le x, y \le n, \max\{x, y\} + L - 1 \le n)$, meaning that Setsuna asked a query whether the two strings are the same.

Output

For each operation of type 2, if the two strings are same, please print `yes` in one line; otherwise print `no`.

Sample Input 1

```
5 6
1 2 1 2 1
2 1 2 2
2 1 3 3
1 1 1
1 3 5
2 1 2 4
2 1 2 2
```

Sample Output 1

```
no
yes
no
yes
```

Sample Input 2

```
3 3
0 65535 65535
2 1 2 2
1 2 3
2 1 2 2
```

Sample Output 2

```
no
yes
```

Note

The first test case is explained below.

Operation	Array	Description
1	`[1,2,1,2,1]`	`[1,2]` and `[2,1]` are different
2	`[1,2,1,2,1]`	`[1,2,1]` and `[1,2,1]` are same
3	`[2,2,1,2,1]`	the first element is shifted by one
4	`[2,2,2,3,2]`	the third to the fifth elements are shifted by one
5	`[2,2,2,3,2]`	`[2,2,2,3]` and `[2,2,3,2]` are different
6	`[2,2,2,3,2]`	`[2,2]` and `[2,2]` are same

Problem H. Message Bomb

Time limit: 5 seconds

While we enjoy chatting with friends on the internet, it is always annoying that we are overwhelmed by lots of messages in various chat groups. A great majority of these messages are actually not interesting to us, but we may miss some important notices if we silence these groups. How many messages do we receive from all online chat groups? Nobody has ever seriously gone into this question.

As an assistant researcher in the school of informatics, you are required to investigate the number of online messages we receive every day. We have already sampled n groups and m students. Every group contains a subset of the m students, which is possibly empty. Also, the members of the groups are constantly evolving; old members may quit, and new members may join in a chat group. Members can send messages in the group; the message is broadcast to all other members currently in the same group.

Now we have collected the log of these chat groups. The log is a sequence of events, which may be a student joining in a group, quitting a group, or sending a message in a group. Your task is to compute the total number of messages received by every student.

Input

The first line of the input contains three integers n, m, s ($1 \leq n \leq 100\,000, 1 \leq m \leq 200\,000, 1 \leq s \leq 1\,000\,000$), denoting the number of groups, the number of students and the number of events in the log.

The next s lines give the events in the log in chronological order. Each of them contains three integers t, x, y ($t \in \{1, 2, 3\}, 1 \leq x \leq m, 1 \leq y \leq n$) specifying an event, which may fall into one of the following three categories:

- If $t = 1$, it means that the x-th student joined in the y-th group. It is guaranteed that the student was not in the group before.
- If $t = 2$, it means that the x-th student quitted the y-th group. It is guaranteed that the student was currently in the group.
- If $t = 3$, it means that the x-th student sent a message in the y-th group. It is guaranteed that the student was in the group now.

Initially, all groups were empty.

Output

Output m lines. The i-th line contains an integer, denoting the total number of messages the i-th student received.

Sample Input 1	Sample Output 1
3 3 10 1 3 2 1 3 1 1 1 2 1 2 1 3 1 2 2 3 1 3 3 2 3 2 1 3 3 2 3 2 1	2 0 1

Sample Input 2	Sample Output 2
2 5 10 1 1 2 3 1 2 2 1 2 1 3 2 1 1 2 3 1 2 3 3 2 1 4 2 3 3 2 1 5 1	2 0 1 1 0

Problem I. Sean the Cuber

Time limit: 30 seconds

Sean is one of the best cubers in China. Today, he is participating in the Fewest Moves Count event of the Chinese Competition of Pocket Cube (CCPC).

A pocket cube is the $2 \times 2 \times 2$ equivalent of the Rubik's cube. Each face of the cube can be rotated, either clockwise or counterclockwise. A standard solved pocket cube has the following color scheme (as shown in Figure 5): the top face is yellow, the bottom face is white, the left face is orange, the right face is red, the front face is blue, the back face is green.

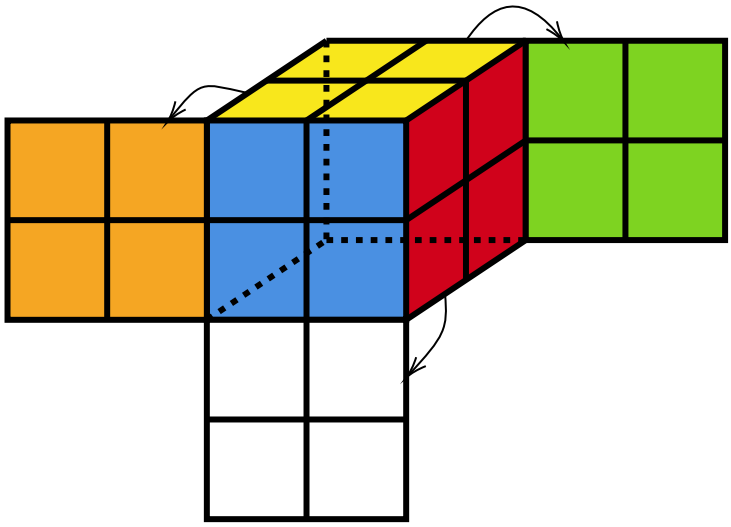


Figure 5: The solved state of a standard pocket cube

The Fewest Moves Count event requires a player to find a sequence of moves to transform one configuration of the pocket cube into another, and the player who finds the shortest such sequence among all participants wins the event. In this event, the player can twist any of the six faces in either direction; also, rotating the entire pocket cube is allowed. Note that a quarter twist counts as one move, and a half twist counts as two moves; however, rotating the entire cube doesn't count into the total number of moves.

Since there are millions of essentially different configurations, the task is too hard for Sean and he can't solve it independently. He wants you to write a program to find the minimum number of moves, given the initial and final configurations. Can you help him?

Input

The first line of input contains a single integer T ($1 \le T \le 250\,000$), denoting the number of the test cases.

Each test case begins with an empty line followed by six lines, representing the initial and final configurations. A configuration is represented as a net of the pocket cube in the following format:

```
  XX
  XX
XXXXXXXX
XXXXXXXX
  XX
  XX
```

where `X` is one of the uppercase letters in {`G`, `O`, `Y`, `R`, `W`, `B`}, denoting green, orange, yellow, red, white, and blue, respectively, and all other characters are whitespaces. The initial and final configurations are horizontally stacked together, with each line split by a vertical bar `|`. In other words, the 1st to the 8th columns represent the initial

configuration, the 9th column is vertical bars, and the 10th to the 17th columns represent the final configuration. It is guaranteed that both configurations are legal, that is, they can be recovered to the solved state by a finite sequence of moves.

Output

For each test case, output a single integer in a line, denoting the minimum number of moves required to transform the initial configuration into the final one.

Sample Input 1	Sample Output 1

2
 	
  GO | WG
  GO | WY
OOYBYWBW|ROGBRYRG
RRYRBWRB|BOGBWYBW
  GY | YO
  WG | RO
 	
  WR | YR
  OR | OW
OYGBWGYB|OWBRBYBG
BWGWRBYY|YOWRYGWO
  OG | GG
  OR | BR
	2
	11

Note

In the first case of the sample data, an optimal move sequence is shown in Figure 6. The total number of moves is 2 (note that the first step does not count into the number of moves), and it can be shown that it can't be done in fewer than 2 moves.

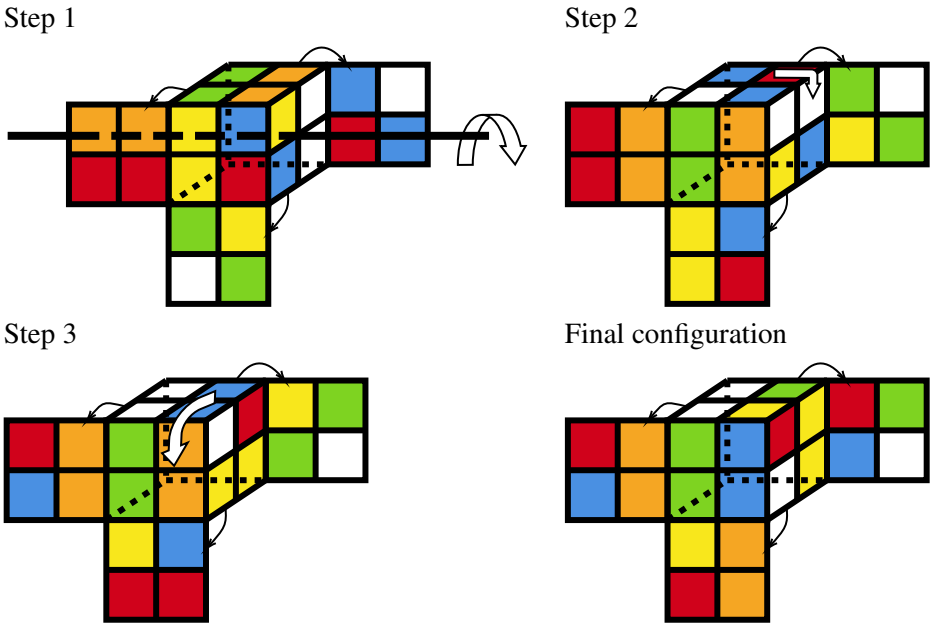


Figure 6: Optimal move sequence for the first case

Problem J. Steins;Game

Time limit: 1 second

Have you ever heard of the Nim game? Probably your answer is yes, but I bet you've never heard a new game called "Steins;Game"! The game is named so because it is a game played with many piles of stones, and *Stein* means *stone* in German.

Alice and Bob want to play this game now. The game is played as follows:

> There are n piles of stones arranged in a row. Each pile of stones is colored either white or black. The i-th pile consists of a_i stones. Starting with Alice, the two players take turns to choose to do one of the following things
>
> - take any positive number of stones from the smallest black pile (i.e., the smallest one among all black piles); or
> - take any positive number of stones from any white pile.
>
> The game ends when no stones are remaining, and the player who takes the next move loses the game.

Now that all piles of stones are fixed, but not yet colored. Bob has bribed the referee to get the chance to paint all piles of stones by himself. Now he wonders how many ways to paint the piles of stones so that he can win the game, assuming Alice and Bob play optimally? Since the answer may be too large, you only need to output it modulo $1\,000\,000\,007$.

Input

The first line of the input contains one integer n ($1 \le n \le 10^5$), denoting the number of piles in the game.

The second line contains n integers $a_1, a_2, \cdots, a_n$ ($1 \le a_i \le 10^{18}$), denoting the number of stones in each pile.

Output

Print one integer in a line, denoting the number of ways of painting the stone piles for Bob to win the game, modulo $1\,000\,000\,007$.

Sample Input 1	Sample Output 1
2 1 1	4

Sample Input 2	Sample Output 2
2 1 2	1

Sample Input 3	Sample Output 3
1 3	0

Note

For the first test case, Bob can win under any of the four ways of paintings.

For the second test case, the only way for Bob to win is to paint both two piles to black, so that Alice is forced to take the first pile in the first move, and then Bob can take the whole second pile and win the game.

For the third test case, no matter how Bob paints the only pile, Alice can take the whole pile and win the game.

This page is intentionally left blank.

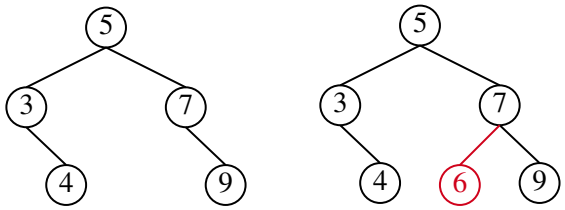
Problem K. Tree Tweaking

Time limit: 1 second

In computer science, a binary search tree is a binary tree where each node is associated with a key; the key of a node is greater than all keys in its left subtree but smaller than all in its right subtree. Binary search trees are commonly used to implement dynamic sets, which support inserting or deleting data items, as well as searching for them.

When searching for a specific key from a set of keys, a binary search tree usually performs better than the naive search algorithm, since when comparing with the value in a node and walking down into a subtree, we don't need to compare the key with all values in the other subtree. The average number of comparisons made is logarithmic to the size of the tree if the binary search tree is balanced. However, it is not so effective if the tree is unbalanced; in extreme cases, the time complexity degenerates to linear. Malicious attackers may leverage this vulnerability by crafting a bad key insertion sequence. If the binary search tree is not self-balanced, the resulting tree may be biased, which significantly undermines the performance of the application.

Consider a scenario where you run a server that maintains a dynamic set implemented by a binary search tree. The binary search tree adopts a simple insertion algorithm, which inserts the key as a new leaf in the appropriate position without moving existing nodes. Figure 7 shows how a new key is inserted into a binary search tree using this simple insertion algorithm.



(a) Before insertion (b) After insertion

Figure 7: A running example of the simple insertion algorithm.

.

Now a client is ready to send a stream of keys to the server, and the server inserts the keys one by one. You are allowed to reorder a specific range of insertions, such that the resulting tree is as balanced as possible. Please report the minimum possible sum of depths of the nodes in the resulting tree after reordering part of the insertions. The depth of a node is defined as the number of nodes in the path from that node to the root.

Input

The first line of the input contains a single integer n $(1 \leq n \leq 10^5)$, denoting the number of keys to insert. The second line of the input contains n integers $a_1, a_2, \cdots, a_n$ $(1 \leq a_i \leq n)$, denoting the keys to insert in order. All keys inserted are distinct. The third line contains two integers l, r, $(1 \leq l \leq r \leq n, r - l + 1 \leq 200)$, denoting the range of insertions you can reorder, i.e., you can permute the insertions of keys $a_l, a_{l+1}, \cdots, a_r$.

Output

Output a single integer in one line, denoting the minimum possible sum of depths of the nodes in the resulting binary search tree.

Sample Input 1	Sample Output 1
8 2 4 5 7 1 3 8 6 3 6	24

Sample Input 2	Sample Output 2
5 5 1 2 3 4 3 5	14

Sample Input 3	Sample Output 3
7 3 2 4 6 7 5 1 1 7	17

Note

For the first sample data, one optimal insertion sequence after rearrangement is `[2,4,1,7,3,5,8,6]`. The resulting binary search tree is shown in Figure 8.

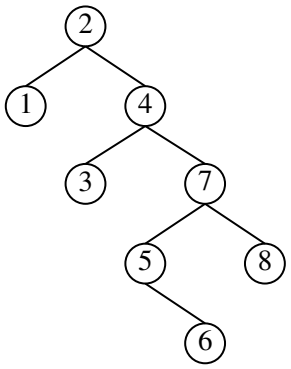


Figure 8: The optimal binary search tree in the first sample data.

Problem L. Clock Master

Time limit: 1 second

With the rapid development of society, the demand for high-precision clocks is constantly rising. Recently, the China Clock Production Company is developing a new type of clock, which can represent a wide range of times.

The novel clock displays the current time in an unusual fashion. The clock consists of several pointers, each controlled by a gear. All gears rotate synchronously – one tooth per period. However, the numbers of teeth of the gears may differ. If a gear has t teeth, then the corresponding pointer can point to t different directions, denoted $0, 1, 2, \cdots, t-1$, respectively, where 0 is the initial direction. Furthermore, if a clock is equipped with n pointers, the i-th of which is controlled by a t_i-tooth gear, then the i-th pointer will point to $k \bmod t_i$ after k periods of time.

The price for a t-tooth gear is t yuan. Given a total budget of b yuan, you need to design a combination of gears, such that the number of valid combinations of directions of pointers is maximized, and the total cost on gears does not exceed the budget. A combination of directions $(d_1, d_2, \cdots, d_n)$ is valid, if it can be written

$$(k \bmod t_1, k \bmod t_2, \cdots, k \bmod t_n)$$

for some nonnegative integer k, where t_i is the number of teeth of the i-th gear. Since the answer may be too large, output the answer in natural logarithm (logarithm with base $e = 2.718281828\cdots$).

Input

The first line of input is a single integer T ($1 \le T \le 30\,000$), indicating the number of test cases. Each test case is a single line of an integer b ($1 \le b \le 30\,000$), denoting the total budget.

Output

For each test case, print the natural logarithm, within an absolute or relative error of no more than 10^{-6}, of the maximum number of valid combinations, in a single line.

Sample Input 1	Sample Output 1
3 2 7 10	0.693147181 2.484906650 3.401197382

Note

For the second sample data, a 3-tooth gear along with a 4-tooth gear may yield 12 different combinations of directions, with total cost exactly being 7. So you should print the value of $\ln 12$, which is approximately 2.484906650.

This page is intentionally left blank.