Problem A. Azalea Garden

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

In a serene garden of azaleas, n dangerous creatures have appeared. Each creature possesses an *attack power* and a *defense power*. Initially, the i-th creature ($1 \le i \le n$) has an attack power of a_i and a defense power of b_i.

You, the guardian of the azalea garden, can mentally imagine a *war* between them. A *war* consists of several (possibly, 0) *battles*. In each imagined *battle*, you choose two living creatures i and j ($1 \le i, j \le n$, $i \ne j$), and:

- If the attack power of creature i is greater than or equal to the defense power of creature j (i.e. $a_i \ge b_j$), then i can defeat j in the battle, and j is considered eliminated.
- Otherwise, nothing happens.

Note that the *wars* are imaginary; that is, a creature eliminated cannot be used for future *battles* in the same *war*, but it regains its vigor at the beginning of the next *war*, and thus can participate in future *battles* of consequent *wars*.

The creatures are volatile and undergo mutations over time. You are given q mutations. After each of the mutations, the attributes of a creature change. Specifically, in the i-th mutation, the v_i-th creature has its attack power updated to x_i and its defense power updated to y_i. Note that the mutations are persistent; After the i-th mutation, the impacts of the first $i-1$ mutations are accumulated.

Since each remaining creature poses an ongoing threat to the flowers, you want to find the minimum possible number of creatures that remain after an optimal sequence of an imagined *war*. You need to answer the question for all states before all mutations and after each mutation.

Input

The first line of the input contains two integers n and q ($1 \le n \le 4 \cdot 10^5$, $0 \le q \le 4 \cdot 10^5$), where n is the number of creatures and q is the number of mutations.

The next n lines of the input describe all the creatures. The i-th line of these contains two integers a_i and b_i ($1 \le a_i, b_i \le 10^9$), where a_i is the attack power and b_i is the defense power of the i-th creature.

The next q lines of the input describe all the mutations. The i-th line of these contains three integers v_i, x_i, and y_i ($1 \le v_i \le n$, $1 \le x_i, y_i \le 10^9$), where x_i is the new attack power of creature v_i and y_i is the new defense power of creature v_i.

Output

Output $q+1$ lines, each containing a single integer, which are the answers before any mutations and after each mutation in sequence.

Example

standard input	standard output

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

Note

In the example, before the mutations begin, the third creature can defeat the first and second creatures. Clearly, this is the optimal sequence of imagined battles, so the answer is 1.

After the first mutation ends, the attack power of the second creature becomes 2, and its defense power becomes 4. Now the third creature can only defeat the first creature, leaving 2 creatures remaining. It can be proved that no better solution exists, so the answer is 2.

Problem B. Beautiful Dangos

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Little Cyan Fish likes to eat *Tricolor Dango*!

Figure 1: *Little Cyan Fish*

Now there are n dangos arranged in a string, and each dango is colored cyan (`C`), white (`W`), or pink (`P`). The dangos are numbered from 1 to n.

Little Cyan Fish considers a string of dangos to be *beautiful* if and only if any two adjacent dangos have different colors.

To make this string of tricolor dango more beautiful, Little Cyan Fish decides to select an interval $[l, r]$ $(1 \le l \le r \le n)$, and rearrange all dangos in this interval arbitrarily, so that the entire string of dango becomes beautiful after the rearrangement.

Little Cyan Fish wants to make the interval he selects as short as possible. Can you help him? You need to output the optimal interval, as well as the whole string after rearrangement.

Note that the original string may already be beautiful, or it might be impossible to make it beautiful through any rearrangement.

Input

The input consists of multiple test cases. The first line contains an integer t $(1 \le t \le 10^5)$, the number of test cases. For each test case:

- The first line contains an integer n $(1 \le n \le 2 \cdot 10^6)$, which is the number of dangos in this string.
- The second line contains a string of length n, where the i-th character denotes the color of the i-th dango, with "`C`" representing cyan, "`W`" representing white, and "`P`" representing pink.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^6$.

Output

For each test case:

- If the string of dangos is already beautiful, output a single line "`Beautiful`".
- Otherwise, if it is impossible to make the string of dangos beautiful through any rearrangement, output a single line "`Impossible`".

- Otherwise, output three lines:
  - The first line should contain the word "`Possible`".
  - The second line should contain two integers l and r, representing the selected interval $(1 \le l \le r \le n)$.
  - The third line should contain a string of length n, representing the colors of all dangos after rearrangement.

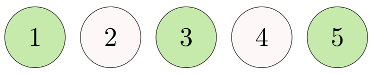
  If there are multiple possible solutions, you may output any of them.

Example

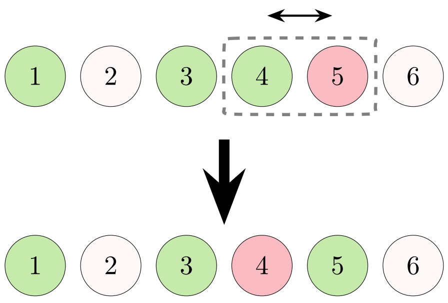
standard input	standard output
4 5 CWCWC 6 CWCCPW 3 PPP 8 CWPPCWWC	Beautiful Possible 4 5 CWCPCW Impossible Possible 4 6 CWPWCPWC

Note

In the first test case, the string of dangos is already beautiful.



In the second test case, initially, the string of dangos is not beautiful because two adjacent dangos, the third and the fourth, are both cyan. But Little Cyan Fish can resolve this by selecting the interval $[4, 5]$ and swapping the two dangos within it.



It can be easily shown that this solution selects an interval of the shortest possible length.

In the third test case, Little Cyan Fish can't make it beautiful through any rearrangement.

Problem C. Catch the Monster

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

A prehistoric monster has arrived on Earth via Dr. Peter's time machine. You need to help Dr. Peter catch it.

The monster hides in a forest with n vertices and m edges. Here, a forest is an acyclic undirected graph that may consist of multiple trees. You can perform the following operations to catch the monster:

- First, you choose a vertex x ($1 \le x \le n$).
- Then, if the monster is currently at vertex x, it is caught.
- Otherwise, the monster remains uncaught, and it may move to any adjacent vertex to its current vertex after the operation, except vertex x. Or it may choose not to move and stay at the same vertex.

We define a forest as *nice* if and only if there exists a finite sequence a of vertices, such that regardless of the monster's initial position and how it moves, performing operations by selecting vertices in the order of a can guarantee that the monster will be caught.

Now, you need to answer q questions from Dr. Peter. In each question, he gives you an interval $[l, r]$ ($1 \le l \le r \le n$). You need to tell him whether the subforest induced by the vertices whose indices are in $[l, r]$ (i.e., the graph formed by retaining only these vertices and the edges between them) is *nice*.

Input

The first line of the input contains three integers n, m, and q ($2 \le n \le 10^6$, $1 \le m \le n-1$, $1 \le q \le 10^6$), where n is the number of vertices in the forest, m in the number of edges in the forest, and q is the number of queries.

The next m lines of the input each contain two integers u and v ($1 \le u, v \le n$, $u \ne v$), representing an edge in the forest.

The next q lines of the input each contain two integers l and r ($1 \le l \le r \le n$), representing a query.

Output

For each query, output a single line "`Yes`" if the subforest is *nice*; otherwise, output a single line "`No`".

You can output the answer in any case (upper or lower). For example, the strings "`yEs`", "`yes`", "`Yes`", and "`YES`" will be recognized as positive responses.

Examples

standard input	standard output

10 9 3
1 2
1 3
1 8
2 5
2 6
2 7
3 4
8 9
8 10
1 3
2 6
1 10
	Yes
	Yes
	No

100000 1 1
1 2
1 9999
	Yes

Note

In the first test case:

- In the first query, for the subforest $[1, 3]$, you can set $a = [3, 1, 2]$.
- In the second query, for the subforest $[2, 6]$, you can set $a = [3, 4, 5, 2, 6]$.
- In the third query, for the subforest $[1, 10]$, it can be proven that there does not exist a finite valid sequence a.

In the second test case:

- In the only query, for the subforest $[1, 9999]$, you can set $a = [1, 2, \ldots, 9999]$.

Problem D. Directed Acyclic Graph

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Given two integers n and m, you need to construct a directed acyclic graph (DAG) $G = (V, E)$ with exactly n vertices and m edges.

In the graph G, a vertex v is called *reachable* from vertex u if and only if there exists a path in the graph that starts at vertex u and ends at vertex v.

For a non-empty set of vertices $A \subseteq V$, a vertex w is defined as *good* for A if and only if it is reachable from **every** vertex in A, and we denote $f(A)$ as the set of all good vertices for A.

The graph you constructed should satisfy both of the following constraints:

- For every vertex i ($1 \le i \le n$), it is reachable from vertex 1.
- There exist k distinct non-empty sets of vertices $S_1, S_2, \ldots, S_k$, such that $f(S_1), f(S_2), \ldots, f(S_k)$ are pairwise distinct. Note that $f(S_i)$ can be empty.

To prove the graph G you constructed satisfies the second constraint, you also need to provide k sets $S_1, S_2, \ldots, S_k$ that satisfy the second constraint.

Input

The only line of the input contains three integers n, m, and k.

There are only 2 tests in this problem:

1. $n = 5$, $m = 6$, $k = 6$;
2. $n = 100$, $m = 128$, $k = 16\,000$.

Output

The first m lines of the output describe the graph G you construct. Each line contains two integers u, v representing an edge from u to v in the graph.

The next k lines of the output describe the k sets of vertices you provide. The i-th line first contains the size of the set S_i, followed by the $|S_i|$ numbers representing each vertex in the set.

Example

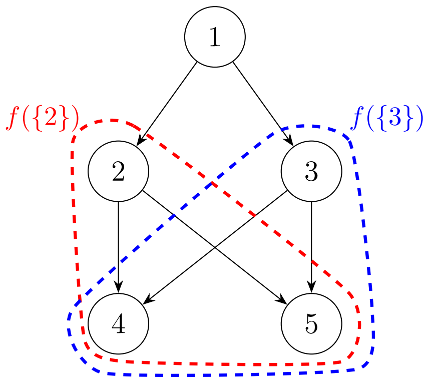
standard input	standard output

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

Note

In the example, the output constructs a graph with $n = 5$ vertices and $m = 6$ edges. The corresponding $k = 6$ sets are $S_1 = \{1\}, S_2 = \{2\}, S_3 = \{3\}, S_4 = \{4\}, S_5 = \{5\}, S_6 = \{2, 3\}$.

Here, vertices 4 and 5 can both be reached from any element in $S_6 = \{2, 3\}$, so $f(\{2, 3\}) = \{4, 5\}$. Together with $f(S_1) = \{1, 2, 3, 4, 5\}, f(S_2) = \{2, 4, 5\}, f(S_3) = \{3, 4, 5\}, f(S_4) = \{4\}, f(S_5) = \{5\}$, these sets are all distinct, satisfying the constraints.



Problem E. Epilogue of Happiness

Input file:	`standard input`
Output file:	`standard output`
Time limit:	7 seconds
Memory limit:	1024 megabytes

As the competition was nearing its end, a banquet was held. For decoration, there was a tree with n light bulbs numbered from 1 to n on it, where the i-th bulb has a *beauty* of w_i. The bulbs are connected by $n-1$ circuits. Specifically, for each i from 2 to n, there is a circuit connecting the i-th bulb and the f_i-th bulb $(1 \le f_i < i)$.

There is a row of m switches that control the bulbs' lighting. Pressing the i-th switch toggles the state of all bulbs on the simple path from bulb 1 to bulb o_i on the tree (i.e., bulbs that are on turn off, and those that are off turn on).

q children will interact with the tree. The process for the i-th child is described by three integers l_i, r_i, and x_i:

- Initially, all bulbs are off.
- Then, the switches from the l_i-th to the r_i-th are pressed in sequence.
- Finally, a photo is taken of the bulbs on the simple path from 1 to x_i.

The *total beauty* of a photo is the sum of the *beauty* of all bulbs that are on in the photo. Your task is to compute this value for each of the q children.

Input

The first line of the input contains three integers n, m, and q $(1 \le n, m, q \le 5 \cdot 10^5)$, where n is the number of bulbs, m is the number of switches, and q is the number of children.

The second line of the input contains $n-1$ integers $f_2, f_3, \ldots, f_n$ $(1 \le f_i < i)$, where f_i represents a circuit between the i-th bulb and the f_i-th bulb.

The third line of the input contains n integers $w_1, w_2, \ldots, w_n$ $(0 \le w_i \le 1000)$, where w_i is the *beauty* of the i-th bulb.

The next m lines of the input describe the switches. The i-th line of these contains a single integer o_i $(1 \le o_i \le n)$.

The next q lines of the input describe the interaction process of the children. The i-th line of these contains three integers l_i, r_i, and x_i $(1 \le l_i \le r_i \le m,\ 1 \le x_i \le n)$.

Output

Output q lines, each containing an integer, representing the *total beauty* of each photo.

Example

standard input	standard output

5 3 7
1 2 3 3
907 609 48 670 184
2
3
5
1 2 5
1 3 3
1 3 2
2 3 1
2 3 3
2 3 4
3 3 5
	48
	1516
	1516
	0
	0
	0
	1748

Note

For the first child:

- First, he pressed the first switch, causing the first and the second bulbs to turn on.
- Then, he pressed the second switch, causing the first and the second bulbs to turn off while the third bulb turns on.
- He took a photo of bulbs $1, 2, 3$, and 5. In the photo, only the 3rd bulb was lit, resulting in a beauty of 48.

Problem F. Follow the Penguins

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

There are n penguins standing on a number line. The i-th penguin is initially located at coordinate a_i. It is guaranteed that all a_i-s are pairwise distinct.

Each penguin chooses a target penguin, denoted by t_i ($1 \le t_i \le n$, $t_i \ne i$). At time 0, all penguins start moving simultaneously. Each one runs towards the current position of its target penguin at a constant speed of 0.5 units per second.

When penguin i meets penguin t_i, it stops immediately. For every penguin, determine the time when it stops moving. Here, penguin i meets penguin t_i if and only if they are at the same coordinate at the same time.

It can be proven that every penguin will stop moving within a finite amount of time, and the stopping time is always an integer.

Input

The first line of the input contains a single integer n ($2 \le n \le 5 \cdot 10^5$), which is the number of penguins.

The second line of the input contains n integers $t_1, t_2, \ldots, t_n$ ($1 \le t_i \le n$, $t_i \ne i$), where t_i is the target chosen by the i-th penguin.

The third line of the input contains n distinct integers $a_1, a_2, \ldots, a_n$ ($-5 \cdot 10^8 \le a_i \le 5 \cdot 10^8$), where a_i is the initial coordinate of the i-th penguin.

Output

Output a single line containing n integers, where the i-th integer represents the time (in seconds) when the i-th penguin stops moving.

Examples

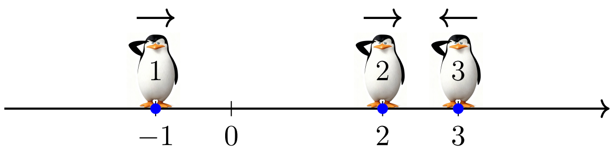
standard input	standard output

3
2 3 1
-1 2 3
	7 1 4

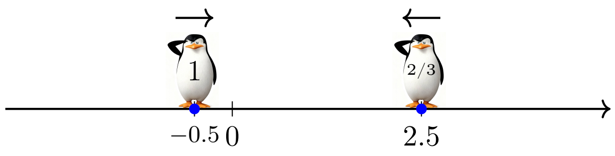
10
8 3 6 7 1 8 2 10 8 1
0 -14 5 -3 14 -12 11 8 -18 17
	25 21 17 14 14 49 29 9 61 17

Note

In the example, initially, since the second penguin is in the positive direction of the first penguin, the first penguin runs in the positive direction. Similarly, the second penguin runs in the positive direction, while the third penguin runs in the negative direction. The initial positions of the three penguins on the number line are shown in the figure below:



At second 1, the second penguin and the third penguin meet at $x = 2.5$, at which point the second penguin stops moving.



At this moment, the first penguin is at −0.5, and the second penguin is at 2.5. The distance between them is 3. Since the first penguin runs at a speed of 0.5 units per second, it will take 6 more seconds to reach the second penguin. Therefore, the first penguin stops moving at second 7.

Before the first penguin stops, the third penguin meets it at second 4. So the answers are 7, 1, and 4, respectively.

Problem G. Grand Voting

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Dada organized a contest, but it received heavy downvotes. He decided to start manipulating the comments.

This contest has s votes, initially set to 0.

There are n participants, each with a voting parameter a_i. When it's their turn to vote:

- If $s \geq a_i$, they cast an upvote, incrementing s by 1.
- If $s < a_i$, they cast a downvote, decrementing s by 1.

Dada can control the voting order of these n people. He wants to know the maximum and minimum possible vote count s in this contest.

Input

The first line of input contains a single integer n ($1 \leq n \leq 10^5$), representing the number of voters.

The next line of input contains n integers $a_1, a_2, \cdots, a_n$ ($|a_i| \leq 10^5$), separated by spaces.

Output

Output one line containing two integers separated by a space, representing the maximum and minimum vote count s in this contest.

Example

standard input	standard output
5 -1 0 1 2 3	5 -5

Note

For example, if you rearrange a to $[-1, 0, 1, 2, 3]$, initially $s = 0$. Since $s \geq a_1 = -1$, the first voter casts an upvote, making $s = 1$. Similarly, the remaining four voters also satisfy $s \geq a_i$, so all cast upvotes. The final value of s is 5, which is the maximum possible.

Conversely, if you rearrange a to $[1, 2, 0, 3, -1]$, then for each voter from left to right, $s < a_i$ holds, so all cast downvotes, resulting in $s = -5$. This is the minimum possible. Another arrangement such as $[3, 2, 1, 0, -1]$ also leads to $s = -5$.

Problem H. Heart of Darkness

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

For a tree T, define $v(T)$ as the number of schemes that stain vertices of T in black and white, and satisfy the following conditions:

- For all black vertices u, v in T, vertices in the simple path from u to v are all black.
- There are at least k undirected edges (u, v) satisfy u and v has different color.

For all n vertices labeled unrooted tree T, calculate the sum of $v(T)$ modulo 998244353.

Input

A single line contains two positive integers n, k ($1 \le n \le 10^7$, $1 \le k \le 5000$).

Output

A single integer as the answer.

Examples

standard input	standard output
`3 1`	`15`
`6 2`	`17286`
`30 9`	`434031055`
`114514 2520`	`136362204`

Note

For the first test case, there are only 3 different T those are chains, so they have the same $v(T)$. Denote 0 as black and 1 as white, there are 5 schemes: $(0, 0, 1), (1, 0, 0), (1, 1, 0), (0, 1, 1), (1, 0, 1)$. We emphasize that schemes $(1, 1, 1)$ and $(0, 0, 0)$ don't satisfy the second condition, and $(0, 1, 0)$ doesn't satisfy the first one.

Therefore, the answer is $3 \cdot 5 = 15$.

Problem I. Imagined Holly

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Given a non-negative integer matrix A of size $n \times n$. For a tree with n vertices, numbered from 1 to n, we call it a *holly tree* if and only if:

- For any pair of vertices u and v ($1 \le u, v \le n$), $A_{u,v}$ equals the bitwise XOR sum of the indices of the vertices on the simple path from u to v in the tree.

Your task is to construct a holly tree. It is guaranteed that such a holly tree always exists.

Input

The first line of the input contains an integer n ($2 \le n \le 2000$), which is the size of matrix A.

The next n lines of the input describe the matrix A. The i-th line contains $n - i + 1$ integers $A_{i,i}, A_{i,i+1}, \ldots, A_{i,n}$ ($0 \le A_{i,j} < 2^{11}$). Note that $A_{i,j} = A_{j,i}$ holds for all $1 \le i, j \le n$.

It is guaranteed that such a holly tree always exists.

Output

Output $n - 1$ lines, each containing two integers u and v ($1 \le u, v \le n$), representing an edge of the holly tree.

Examples

standard input	standard output

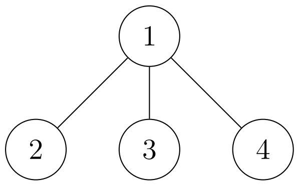
2
1 3
2
	1 2

4
1 3 2 5
2 0 7
3 6
4
	1 2
	1 3
	1 4

6
1 7 4 5 2 3
2 1 6 7 0
3 5 4 3
4 3 2
5 5
6
	4 1
	2 3
	6 4
	5 2
	4 2

Note

In the second example, the tree in the output is shown in the following figure:



This tree is a holly tree. For example, for the pair of vertices $(2, 4)$, the simple path from 1 to 4 includes vertices $2, 1$, and 4, with a bitwise XOR sum of $2 \oplus 1 \oplus 4 = 7$, which satisfies the constraint $A_{2,4} = 7$ given in the input.

Problem J. January's Color

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Given a rooted tree with n vertices, where the root is at vertex 1. It is guaranteed that no vertex has exactly one child in the tree. In other words, each vertex is either a leaf or has **at least two** children. You own some of the vertices in the tree.

You may have some vertices in your hand. You can obtain new vertices in the following two ways:

- Directly purchase a vertex i from the bank at a cost of c_i.
- Select **two different** vertices from your hand that share the same parent, discard them, and obtain their parent for free.

Now, you have m queries. In each query, you initially have exactly one vertex x, and you want to end up with exactly one vertex y with no other vertices left. Find the minimum cost needed to achieve this, or report no solution.

Input

The input consists of multiple test cases. The first line contains an integer t ($1 \le t \le 10^5$), the number of test cases. For each test case:

- The first line contains two integers n and m ($3 \le n \le 3 \cdot 10^5, 1 \le m \le 3 \cdot 10^5$), where n is the number of vertices in the tree and m is the number of queries.
- The second line contains n integers $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le 10^9$), which are the costs to obtain each vertex from the bank.
- The next $n-1$ lines each contain two integers u and v ($1 \le u, v \le n, u \ne v$), representing an edge in the tree.
- The next m lines each contain two integers x and y ($1 \le x, y \le n$, $x \ne y$), representing a query.

It is guaranteed that the sum of n and the sum of m over all test cases do not exceed $3 \cdot 10^5$.

Output

For each query, output a single integer representing the minimum additional cost for that query. If there is no way to end up with a single vertex y, output -1.

Example

standard input

```
3
5 5
1 2 3 4 5
1 2
1 3
2 4
2 5
3 1
2 1
4 1
5 1
5 2
5 5
1 5 1 1 1
1 2
1 3
2 4
2 5
3 1
2 1
4 1
5 1
2 5
6 5
9 9 8 2 4 4
1 2
1 3
1 4
1 5
1 6
2 1
3 1
4 1
5 1
6 1
```

standard output

```
2
3
8
7
4
2
1
2
2
-1
2
2
4
2
2
```

Note

For the first query in the first test case, you initially have vertex 3 and wish to obtain vertex 1. The cheapest solution is to directly purchase vertex 2 at a cost of $c_2 = 2$, then discard vertices 2 and 3 to obtain their common parent 1 for free. It can be proven that this is the minimum-cost solution.

For the first query in the second test case, you again start with vertex 3 and want to obtain vertex 1. Instead of purchasing vertex 2, you choose to purchase vertices 4 and 5 at a total cost of $c_4 + c_5 = 2$, then discard vertices 4 and 5 to obtain their parent vertex 2 for free. Next, discard vertices 2 and 3 to obtain their parent vertex 1 for free. It can be proven that this is the minimum-cost solution.

Note that you can not purchase another vertex 3 and then discard it with the original vertex 3 to obtain vertex 1, since these two vertices 3 do not satisfy the requirement of being two different vertices.

Problem K. Killing Bits

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	512 megabytes

You are given two arrays a and b, both consisting of n non-negative integers. You can perform the following operation on the array a an arbitrary number of times (possibly, zero):

- First, you select a permutation p of $0, 1, \ldots, n-1$;
- Then, for each $1 \le i \le n$, you set a_i to $a_i \,\&\, p_i$. Here, & denotes the bitwise AND operation.

You have to determine whether it is possible to transform a into b.

Input

The input consists of multiple test cases. The first line contains an integer t ($1 \le t \le 10^4$), the number of test cases. For each test case:

- The first line contains a single integer n ($1 \le n \le 5 \cdot 10^4$), which is the length of arrays a and b.
- The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le n-1$), which are the elements of a.
- The third line contains n integers $b_1, b_2, \ldots, b_n$ ($0 \le b_i \le n-1$), which are the elements of b.

It is guaranteed that the sum of n over all test cases does not exceed $5 \cdot 10^4$.

Output

For each test case, print "`Yes`" in a single line if it is possible to transform a into b. Otherwise, print "`No`".

You can output the answer in any case (upper or lower). For example, the strings "`yEs`", "`yes`", "`Yes`", and "`YES`" will be recognized as positive responses.

Example

standard input	standard output

4
3
0 1 2
2 1 0
5
1 0 1 3 4
0 0 1 1 4
8
1 2 3 4 5 6 7 7
1 2 3 4 5 6 7 7
8
7 7 7 7 7 7 7 7
1 2 3 4 5 6 7 7
	No
	Yes
	Yes
	No

Note

In the first test case, we need to use at least one operation to transform a into b. Note that $a_1 \,\&\, p_1$ is always 0 because $a_1 = 0$. However, $b_1 > 0$, so it is impossible to make $a_1 = b_1$, no matter how the permutations are selected during the operations.

In the second test case, you can select $p = [2, 0, 3, 1, 4]$. After this operation, a is transformed into b.

In the third test case, $a = b$, so we do not need any operations.

Problem L. Let's Make a Convex!

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

Kevin is the chief judge of the *International Convex Polygon Championship (ICPC)*. He proposed a geometry task for the contest. However, due to his lack of experience in geometry, he was unable to generate the correct convex polygons for the task.

To prove his geometry skills, Kevin starts playing with sticks. He has n sticks, the i-th of which has a length a_i. He would like to select k sticks so that they can be arranged as a non-degenerate convex polygon.

Since stronger test data are needed, Kevin wants to maximize the perimeter of the polygon (i.e., maximize the sum of a_i of all sticks in the subset). Could you help him find out the value for all integers k from 1 to n? If no such polygon exists, tell him a single integer 0 instead.

Input

The input consists of multiple test cases. The first line contains an integer t ($1 \le t \le 10^5$), the number of test cases. For each test case:

- The first line contains a single integer n ($1 \le n \le 2 \cdot 10^5$), which is the number of sticks.
- The second line contains n integers $a_1, \ldots, a_n$ ($1 \le a_i \le 10^9$), which are the lengths of each stick.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output a single line containing n integers, representing the maximal perimeter of the polygon. Specifically, if no such polygon exists, output a single integer 0.

Example

standard input	standard output

```
7
5
1 2 3 4 5
5
1 2 4 8 16
5
2 2 2 2 2
4
1 4 10 7
2
1 2
3
2 3 4
4
3 1 2 6
```
	```
	0 0 12 14 15
	0 0 0 0 0
	0 0 6 8 10
	0 0 21 22
	0 0
	0 0 9
	0 0 0 0
	```

Note

In the first test case, it can be shown that there does not exist a convex polygon of 1 or 2 sides. When $k = 3$, the maximal perimeter of the convex polygon is 12, since sticks of side lengths 3, 4, and 5 are known for forming a right triangle. Similarly, when $k = 4$, the maximal perimeter of the polygon is $2+3+4+5 = 14$; when $k = 5$, selecting all sticks is a valid scheme and obtains a perimeter of $1 + 2 + 3 + 4 + 5 = 15$.

In the second test case, it can be proven that no matter how the sticks are selected, they cannot form a convex polygon.

Problem M. Mystique as Iris

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

The following two steps, on an array x consisting of positive integers, are called an operation:

1. Select any two adjacent elements in x, decrease one of them by 1, and set the other to 0.
2. Remove all zeros from x.

We call x *mystic* if and only if it can be transformed into an empty sequence after a finite number of operations (possibly zero).

You are given an array a consisting of n integers, as well as an integer m. Each element of a is either an integer from 1 to m or -1. Your task is to replace every occurrence of -1 in a with any integer from 1 to m.

Determine the number of distinct mystic arrays a that can be obtained after the replacement. Since the answer can be very large, output it modulo $10^9 + 7$.

Input

The first line of the input contains two integers n and m ($2 \le n \le 10^6$, $1 \le m \le 10^8$).

The second line of the input contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le m$ or $a_i = -1$).

Output

Output a single integer, representing the number of distinct mystic sequences a that can be obtained, modulo $10^9 + 7$.

Examples

standard input	standard output

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

`6 10`
`-1 -1 -1 -1 1 7`
	`9125`

Note

In the first test, the array a is $[-1, -1]$. By replacing both -1-s, one possible result is $[1, 2]$. In this case, we select the two adjacent numbers: decrease the first by 1 and set the second to 0, obtaining $[0, 0]$. After removing all zeros, the sequence becomes empty. Hence, $[1, 2]$ is a mystic sequence.

Similarly, if we replace the -1-s with $[1, 1]$ or with $[2, 1]$, both sequences can also be reduced to empty. Therefore, the total number of distinct mystic sequences is 3.