MCPC 2025

Jackson & Parsa

Division A


A: *Potion Seller*


B: *Tickets*


C: *Toad Optimism*


D: *Divide & Conquer (Literally)*


E: *Rock Piles*


F: *Cats and Mouse*


G: *Lucas The Wizard*


H: *Picking Petals*


I: *The Last Leaf*


J: *GiDi Up*


K: *Downstream*

October 4, 2025

This page is intentionally left blank

A. Potion Seller

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Ali is a knight going into battle, and he needs a very particular potion to ensure victory - not too weak or short-lived to be useful in battle, but also not too potent or long-lasting that the potion may kill him.

On his travels, Ali stumbles into a tavern. Inside he finds a merchant, a potion seller, who agrees to sell him their finest potions.

Each potion has two stats: the potion's strength and duration. Both are measured in positive integers.

However, Ali is not strictly limited to the potions offered by the merchant. Ali is also able to mix potions.

Take two potions p_1 and p_2, with strength ratings of s_1 and s_2, and durations of d_1 and d_2 respectively. If Ali creates a mixture potion p_3 which is 30% p_1 and 70% p_2, then the strength and duration of this potion would be:

$$s_3 = 0.3 \times s_1 + 0.7 \times s_2$$
$$d_3 = 0.3 \times d_1 + 0.7 \times d_2$$

In general, Ali can mix two potions at any ratio, not just the 30:70 ratio depicted above, and the strength/duration values will be interpolated accordingly.

This means that even if many of the merchant's potions are so strong that Ali cannot handle them, by repeatedly mixing these potions, Ali may be able to create the perfect strength and duration potion he needs.

However, each potion the seller provides has a cost, and Ali would like to spend the least amount of money required to get the particular potion he is after. Can you help him?

Input

Input will begin with three integers: the number of potions on sale n ($1 \leq n \leq 1000$), and the strength and duration of the potion Ali is trying to make s_t and d_t.

n lines will follow, detailing the potions on sale. Each line will contain three integers:

- The strength of the potion s ($0 \leq s \leq 10^9$)
- The duration of the potion d ($0 \leq d \leq 10^9$)
- The cost of the potion c ($1 \leq c \leq 10^9$)

Output

If it is not possible to acquire the potion Ali is after, print `IMPOSSIBLE` on a single line.

If it is possible with a certain selection of potions, then first print how many potions are needed to minimise cost. Then, for each potion needed, print the strength and duration of these potions on a new line.

If there are multiple possible selections of potions that use the least cost, printing any valid selection will be accepted.

Note

TLDR: Using floating point math probably won't break this problem.

You are guaranteed that, if any of the strength values or duration values were shifted by any amount that would keep the relative and absolute error within 10^{-8} (relative here means relative when taking the difference in stats to the target potion), then any collection of potions where it would be impossible to generate the target potion would remain impossible to do so. In short, if we change the duration/strength of particular potions by an extremely small amount, this is guaranteed to never make the potion easier to formulate, only harder.

Examples

standard input	standard output
4 5 5 7 13 3 8 6 8 1 5 4 4 1 7	2 4 1 7 13

For Example 1, it is possible to create the 5 strength, 5 duration potion by purchasing the 7 strength, 13 duration potion, as well as the 4 strength, 1 duration potion, then mixing them in a ratio of 1:2 (there are other possible selections of potions that work, but this is the cheapest).

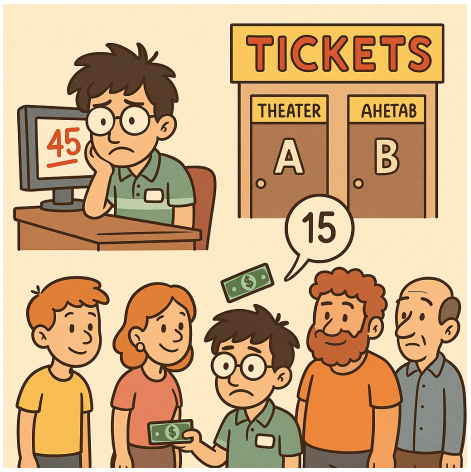
standard input	standard output
4 5 5 7 13 20 8 6 8 1 5 1 4 1 7	3 4 1 8 6 1 5

For Example 2, purchasing the final 3 potions makes for the cheapest way to generate Ali's potion. We can create a new potion p by mixing the 2nd and 3rd potions with a ratio of 16:11, with an approximate strength of 5.15 and duration of 5.6. Mixing this with the 4th potion with a ratio of 27:4 would then give a potion with a strength of 5 and duration of 5, at a cost of 16.

Alternatively, you could mix all three potions with a ratio of 16:11:4 to get the result potion in a single mixture. (However, you can prove that anything achievable by mixing k potions at once can be obtained by repeatedly mixing pairs of potions, so this isn't necessary.)

B. Tickets

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes



After receiving his FIT1054 mark for assignment 1, Parsa decided to retire from competitive programming and pursue his true passion: selling cinema tickets.

Unfortunately, without a strong résumé, he could only find work at a very small cinema with two theatres. Each theatre can accommodate at most B people.

Every evening, n people line up to buy tickets. However, they have a peculiar habit: each person wants to watch the same movie as the person directly in front of them in the queue. The only way to make a person choose the other movie is to pay them exactly the amount of money they have in their pocket, at which point all subsequent people in line also change their mind.

The amount of money the i-th person has in their pocket is denoted by a_i. Parsa's boss has instructed him to sell tickets under the following conditions:

1. No theatre should contain more than B people.
2. Parsa should spend the minimum possible amount of money.

Since Parsa is traumatised by anything that reminds him of competitive programming, he has asked you to determine the minimum amount of money he must spend.

It is guaranteed that there exists a way to assign tickets such that each theatre has at most B people.

Input

The first line of input contains two integers:

- n ($1 \leq n \leq 3000$) – the number of people in the queue.
- B ($n \leq 2B$) – the capacity of each theatre.

The second line contains n integers, where the i-th integer represents a_i, the amount of money the i-th person has in their pocket.

- a_i ($0 \leq a_i \leq 10^9$).

Output

Print a single integer – the minimum amount of money Parsa must spend.

Examples

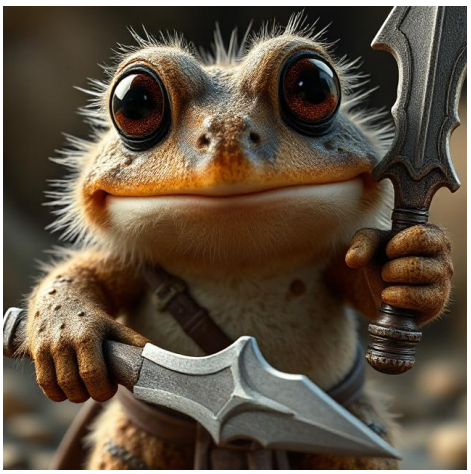
standard input	standard output
1 1 0	0
2 1 1000 50	50
3 2 50 10 1000	10
6 3 0 1 2 3 4 5	3

In example 3, we have three patrons, and need to assign them so that no theatre has more than 2 people. The optimal way to do this is to pay only the second person to change their mind, then one theatre will have patron #1, while the other theatre will have patrons #2 and #3.

In example 4, we can simply put three patrons in each theatre by paying just patron #4.

C. Toad Optimism

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes



Ali is preparing for a great battle, where he'll lead an army of battle-ready toads. These toads need to be organised into battalions.

There are many types of battle toads, and you want to make the instructions for the toads easy, so you want each battalion to have exactly the same amount of any particular toad type (so, all teams might have 3 warriors, 2 shields, and 1 archer). You also want every toad to fit neatly into one of these battalions (no toad left behind!)

You've just received a list of all the toad types and their population sizes. However, you know that exactly one of the numbers is wrong! Ali is an optimist, so he wants to know, at best, how many battalions could he draft for the toads?

Input

Input will begin with a single integer n ($2 \leq n \leq 2 \times 10^5$), the number of toad types.

A single line will follow, containing n integers p_i ($0 \leq p_i \leq 10^9$), representing the population amounts of each individual toad type.

At least 2 population values will be non-zero.

Output

Output a single integer b, the maximum number of battalions that Ali could draft, given that any one of the population numbers could be wrong (and therefore a more useful number).

Examples

standard input	standard output
5 54 72 20 24 12	6

For Example 1, the best possible outcome is that the 3rd population value is incorrect, and this was instead something like 30. This would then allow 6 battalions, of the following composition:

- 9 type-1s,
- 12 type-2s,
- 5 type-3s,
- 4 type-4s, and
- 2 type-5s.

standard input	standard output
4 0 5 0 7	7

For Example 2, if the 2nd population value was the incorrect one (instead it was 7), then we could make 7 battalions, each with 1 of the type-2 and type-4 toads.

D. Divide & Conquer (Literally)

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

Eric, Saksham, and Annabel from the MAPS marketing team have been asked to manage the Instagram page for MCPC. Their task is to prepare a set of posts, each of which has a certain difficulty level that reflects how polished or complex it must be.

They know that every post will create stress, and that each team member can only handle a limited amount of stress in the month before burning out. At the same time, they have carefully measured each member's ability for different types of posts. If post i has difficulty h_i and member j has ability $s_{i,j}$ for that post, then if member j completes the post alone, it will take them exactly $\frac{h_i}{s_{i,j}}$ hours and they will endure h_i units of stress.

Posts, however, do not have to be done by a single person. They can be split between multiple members. If a member does only a fraction α of post i, they spend $\alpha \times \frac{h_i}{s_{i,j}}$ hours and incur $\alpha \times h_i$ stress. Stress is always proportional to the fraction of the post handled, and no member may exceed their personal monthly stress limit.

Since the MAPS budget is tight and each member is paid directly according to the number of hours they work, Eric, Saksham, and Annabel want to find the most efficient way to finish all posts. Their goal is to complete every post while **minimising the total hours worked across the entire team**, ensuring that no one goes over their stress capacity.

It is guaranteed that there exists at least one way to assign the work so that all posts are completed without exceeding any member's limits.

Input

The first line of input contains one integer:

- p $(1 \leq p \leq 100)$ – the number of posts.

The second line of input contains p integers, $h_1, h_2, \ldots, h_p$:

- h_i $(1 \leq h_i \leq 100)$ – the difficulty of the i-th post.

The third line of input contains one integer:

- n $(1 \leq n \leq 100)$ – the number of team members.

The fourth line contains n integers, $m_1, m_2, \ldots, m_n$:

- m_i $(1 \leq m_i \leq 100)$ – the maximum stress the i-th member can endure this month.

Then follow p lines. In the i-th line there are n integers, $s_{i,1}, s_{i,2}, \ldots, s_{i,n}$

- $s_{i,j}$ $(1 \leq s_{i,j} \leq 100)$ – the ability of member j on post i.

Output

Print a single real number — the minimum total hours (equivalently, total payment) required to complete all posts without exceeding any member's stress capacity. Your answer will be accepted if it has an absolute or relative error of at most 10^{-6}.

Examples

standard input	standard output
1 10 2 6 7 4 1	5.500000000000

In this example, we can assign $\frac{3}{5}$ of post 1 to member 1, and the remaining $\frac{2}{5}$ to member 2.

This gives $\frac{3\times10}{5} = 6$ stress to member 1, and $\frac{2\times10}{5} = 4$ stress to member 2. It takes a total of $\frac{6}{4} + \frac{4}{1} = 5.5$ billable hours.

standard input	standard output
3 7 9 11 2 10 17 5 4 8 5 3 2	7.383333333330

In this example, we can assign all of posts 1 and 2 to member 2, and $\frac{10}{11}$ of post 3 to member 1, leaving $\frac{1}{11}$ of post 3 for member 2.

This gives 10 stress to member 1, and 17 stress to member 2. It takes a total of $\frac{10}{3} + \frac{1}{2} + \frac{9}{5} + \frac{7}{4} \approx 7.38333333333$ billable hours.

E. Rock Pile

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Ali decides to take residence among the toad civilization for a while. This understandably takes some adjusting, and one of the ways this presents itself is in their currency.

Toads use rocks as currency to trade for goods and services, like toad-school or toad-restaurants. Luckily for Ali, he already had r rocks in his pocket before arriving in the toad civilization.

Unfortunately for the toads, Ali knows how to break rocks, which allows him to turn few rocks into many.

So for Ali, there are two ways to increase his rock collection:

1. He can work a job i, earning him b_i rocks.
2. He can break all of his available rocks in two, doubling the amount of rocks in his collection.

However, this has three stipulations:

1. The job market in toad civilization is weak, and so for any job i you actually have to pay upfront a_i rocks to secure the job. (As the saying goes, you've got to spend rocks to make rocks.)
2. For a similar reason, you can only complete job i at most c_i times (each time, you need to pay a_i rocks).
3. Because Ali does not want to inflate the rock currency too much, he will only allow himself to break all of his rocks k times.

Within these stipulations, Ali wants to end up with the most rocks possible.

Input

Input will begin with 3 space-separated integers:

- r, the initial amount of rocks Ali has ($0 \leq r \leq 100$),
- n, the number of jobs available to Ali ($0 \leq n \leq 10^5$),
- k, the number of times Ali is allowed to break all of his rocks ($0 \leq k \leq 20$).

n lines will then follow, each containing 3 space-separated integers:

- a_i, the upfront cost Ali has to pay to complete this job ($0 \leq a_i \leq 10^9$)
- b_i, the rocks Ali will receive for completing this job ($0 \leq b_i \leq 10^9$)
- c_i, the number of times Ali can complete this job ($0 \leq c_i \leq 10^9$)

It is guaranteed that the sum of $(b_i - a_i) \times c_i$ over all jobs will not exceed 10^9.

Output

Your output should be a single integer, representing the largest amount of rocks that Ali could ever attain.

Examples

standard input	standard output
4 4 4 7 10 3 1000000 10000000 100 20 50 2 10 4 1000	376

One possible way to amass a rock pile would be like so:

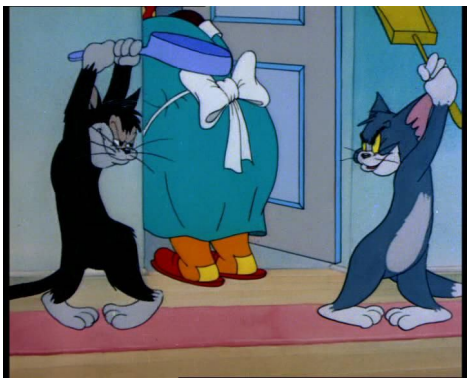
1. Currently, all jobs require more upfront rocks than Ali owns, so for now his only option is to double his rocks. Let's do this twice. We go from 4 to 16 rocks.
2. Now, we have access to 2 jobs, one costing 7 rocks and providing 10, and another costing 10 rocks and providing 4. Let's work the first job 3 times. We then end up with 25 rocks.
3. Let's double our rocks a third time, taking us to 50 rocks.
4. We now might work the 3rd job (20 rocks for 50 rocks) just once, leaving us at 80 rocks.
5. We can now use our last rock-doubling, to net us 160 rocks.
6. We can now work the 3rd job a second time, getting us to 190 rocks.

That is one possible way, but it is not the optimal way. The optimal way would net us '376' rocks in total.

standard input	standard output
5 4 0 0 10 1 10 20 1 40 50 5 20 30 2	95

F. Cats and Mouse

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



After a hard fought battle, Ali now finds himself against a pair of undefeatable foes - two human sized cats.

These cats are inseparable, making it impossible for Ali to face either of them in one-on-one combat. However, their inseparability is also their weakness, for the cats need to always be within earshot of each other.

In particular, the battlefield Ali and the cats are battling on can be represented by a collection of positions, connected by tunnels. The outcome of the battle is determined in the following manner:

1. The two cats first get to pick the single location where they will start.
2. Ali then picks his start location. To keep it fair, this location must be reachable from the cat's starting location using the tunnels.
3. Now the foes take turns, repeatedly:
   (a) One of the cats may move to any location, with the restriction that their new location must be at the same position, or adjacent via a tunnel, to the stationary cat.
   (b) Ali can then move to an adjacent location by using a tunnel, or stand still.
   (c) If after he moves/stays still, he is in the same location as a cat, then he is defeated.
4. If Ali can evade the cats indefinitely (he has a strategy that means the cats will never catch Ali), then the cats are defeated due to exhaustion.

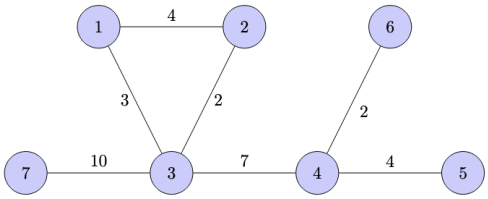
However, Ali was cunning, and knew this day would come. In fact, the tunnels in the battlefield are not natural - these were his own creation!

Digging tunnels takes energy however, and Ali wants to minimise his energy exerted while still guaranteeing victory.

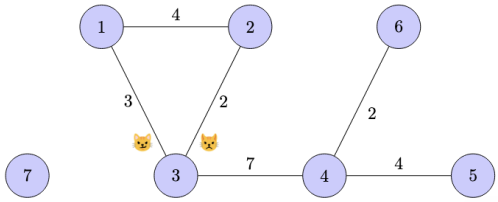
Your task is, given a number of locations, and the possible tunnels between them, find the cheapest way to ensure that Ali can always defeat the cats, if it is possible.

Explanation

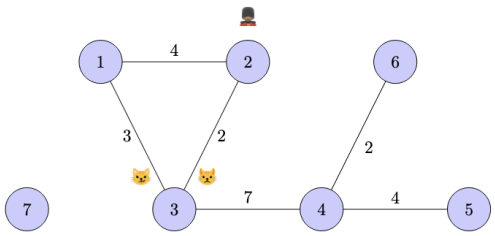
Let's play the game, step by step.
Suppose the battlefield looked like this, and Ali dug all tunnels except 3-7:



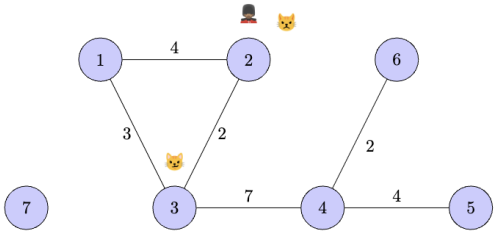
First, the cats get to pick their starting position.
Let's say they chose position 3:



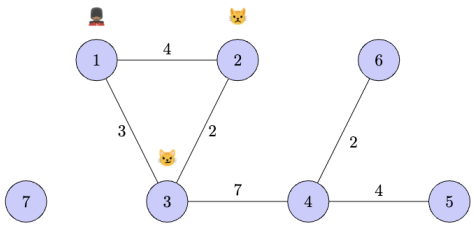
Next, Ali can pick any location to start which the cats can reach (so anywhere but 7).
Let's start at 2:



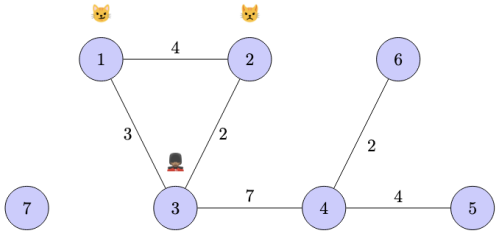
Now, we go into the repetitive part of the game.
Let's suppose that the pouting cat went to position 2:



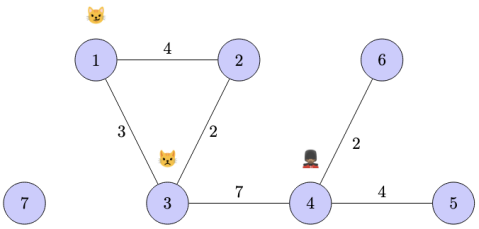
Now, Ali needs to pick an adjacent location not occupied by the cats.
So he can move to position 1:



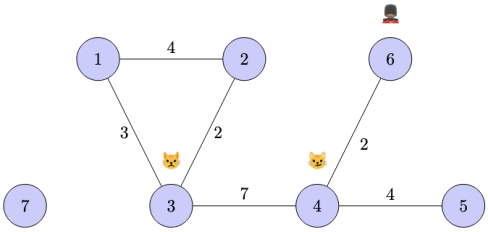
The smirking cat then moves to position 1, allowing Ali to move into position 3:



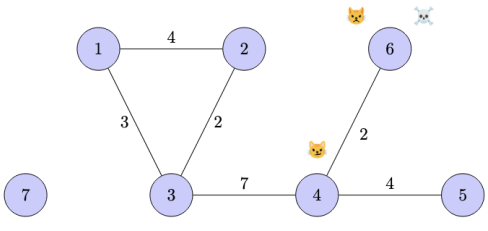
Next, the pouting cat moves back to position 3, and Ali then needs to move to either position 2 or 4.
He picks 4:



Smirking cat then moves to position 4, and Ali moves to position 6:



Now the pouting cat moves to position 6, and Ali has no move that doesn't land him in the same spot as a cat.
Ali is defeated:



Input

Input will begin with two integers n ($1 \leq n \leq 10^5$), the number of locations, and m ($0 \leq m \leq 10^5$), the number of possible tunnels.

m lines will follow, each representing a possible tunnel that Ali could dig. This line contains 3 integers:

- a ($1 \leq a \leq n$), the location on one side of the tunnel,
- b ($1 \leq b \leq n$), the location on the other side of the tunnel,
- c ($1 \leq c \leq 10^9$), the energy cost required to dig the tunnel.

Output

If it is impossible for Ali to guarantee victory, then simply print 'NO'.

If it is possible for Ali to guarantee victory, then find the collection of tunnels whose total energy cost is minimised while still guaranteeing victory.

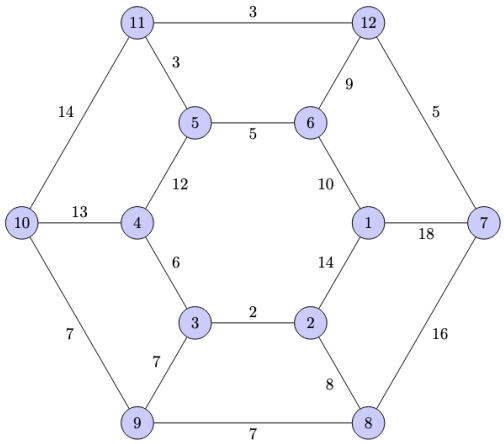
Start with a single line 'YES x' where x is the number of tunnels used.

x lines should then follow, each containing 3 integers 'a b c' representing the tunnel used (matching the input the tunnel was specified with). If there are multiple cheapest ways to dig tunnels, then choosing any valid selection will be accepted.

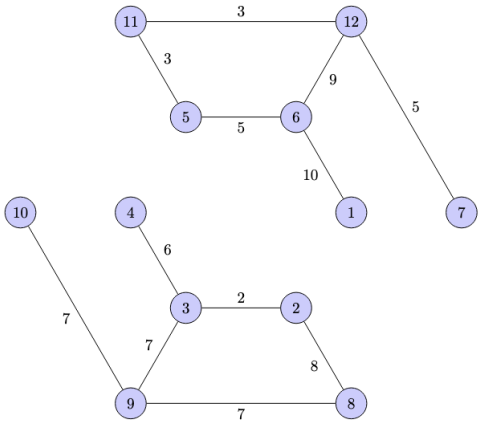
Examples

standard input	standard output
12 18 1 7 18 8 9 7 3 9 7 6 1 10 2 3 2 4 10 13 2 8 8 5 6 5 7 8 16 6 12 9 1 2 14 11 12 3 5 11 3 3 4 6 4 5 12 10 11 14 9 10 7 12 7 5	YES 12 3 9 7 6 1 10 2 3 2 9 10 7 3 4 6 8 9 7 5 6 5 6 12 9 12 7 5 11 12 3 2 8 8 5 11 3

This example is represented by the following diagram:

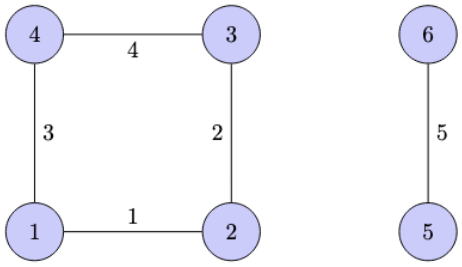


This can be made winnable for Ali by digging the following tunnels:



standard input	standard output
6 5 1 2 1 5 6 5 4 1 3 3 4 4 2 3 2	NO

This example is represented by the diagram on the right, which, no matter what tunnels Ali digs, the cats can force him to lose.



G. Lucas The Wizard

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds
Memory limit:	1024 megabytes

Lucas had done the impossible: he broke an unbreakable rule (the details of which we shall not discuss) and somehow became a Candidate Master.

Tans, who was desperately chasing the same title, confronted him in disbelief. *"What kind of wizardry did you use to bypass the unwritten rule?"*, he demanded.

Lucas smirked. *"It wasn't wizardry. It was a problem — a magical one. Solve it, and you too may rise."*

He revealed the challenge:

You are given an array a of integers of size n and an integer k.

- A pair (i, j) with $i < j$ is called mystical if $a_i > a_j$.
- A pair (i, j) with $i < j$ is called magical if $a_i \oplus a_j > k$. (The symbol $\oplus$ represents the binary XOR operation.)

A pair is called a *candidate* if it is either mystical or magical, but not both.

Lucas leaned closer and whispered: *"To become a Candidate Master, you must first **master** the art of counting **candidate** pairs."* But Lucas hadn't anticipated that Tans was a student of Will—our legendary alumnus. And every student of Will knows the ancient truth:

"If there's a Will, there's a Wei."

Armed with this powerful spell, Tans solved the problem in seconds. Now it's your turn to do the same!

Input

The first line of input contains two integers.

- n $(1 \leq n \leq 10^6)$.
- k $(0 \leq k < 2^{30})$

The second line of input contains n integers.

- a_i $(1 \leq a_i < 2^{30})$

Output

Print an integer, the number of *candidate* pairs in the array.

Examples

standard input	standard output
4 4 5 3 4 1	4

standard input	standard output
9 5 1 2 3 4 5 6 7 8 9	20

H. Picking Petals

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Ali, forlorn from the many lost toads in the great battle, picks petals from a flower to calm his mind.

The flower sports a beautiful combination of coloured petals, and Ali picks each petal one by one, placing it on a stack. However, there are a few petals which are jet black, and when Ali encounters a black petal, he always habitually places it on the stack, which stains the top petal. Therefore, he must remove 2 petals - both the black petal and the stained petal - from the stack before continuing. If the stack was empty when he encounters a black petal, then he just removes the black petal and the stack remains empty.

Ali is interested in how many possible resulting stacks of petals are possible, given that two petals of the same colour are interchangeable. Can you soothe Ali's mind by helping him with this problem?

Input

Input will begin with 3 integers: n ($1 \leq n \leq 10^5$) the number of petals on the flower, k ($1 \leq k \leq \min(n, 100)$) the number of unique colours of the petals (excluding black), and b ($0 \leq b \leq \min(n, 100)$) the number of black petals.

A single line will then follow, containing k integers, representing the counts of each colour of petal.

The total counts will be valid (the sum of the second line, plus b, will equal n).

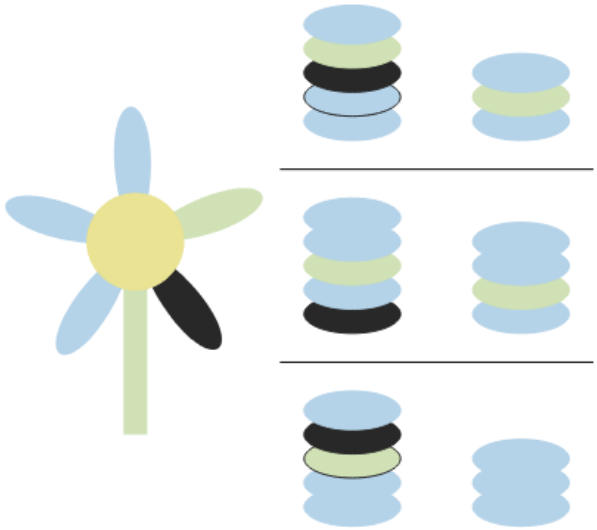
Output

Output the number of possible final stacks of petals possible. Since this number can be large, output the number modulo 1000000007.

Examples

standard input	standard output
5 2 1 1 3	8

For Example 1, picking some random representative colours, this represents a flower with 5 petals: 1 green, 3 blue, and 1 black.



One possible way of picking the petals would be blue, blue, black, green, blue. This would result in a final stack of blue, green, blue. Another possible ordering of the petals would be black, blue, green, blue, blue. This would result in a final stack of blue, green, blue, blue.

Over all possible initial orderings, the possible final stacks are:

- green-blue-blue-blue
- blue-blue-green-blue
- blue-blue-green
- blue-blue-blue
- blue-blue-blue-green
- blue-green-blue
- blue-green-blue-blue
- green-blue-blue

Since there are 8 distinct possible final stacks, the output is 8.

standard input	standard output
25 4 3 5 8 3 6	449246857

For Example 2, there are 157449247956 possible end stacks. Modulo 1000000007 this is 449246857.

I. The Last Leaf

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds
Memory limit:	1024 megabytes

Winson has recently taken a break from competitive programming to clear his mind, and instead he has taken up gardening. Winter has now arrived, and it's time for him to prune his trees. Of course, Winson's algorithmic mindset never leaves him — even in the garden he wants to prune in the most *optimal* way.

Winson's trees can be represented as individual vertices connected by branches. Each tree starts at a root vertex, which then has branches extending to other vertices, which in turn may connect to more vertices, and so on.

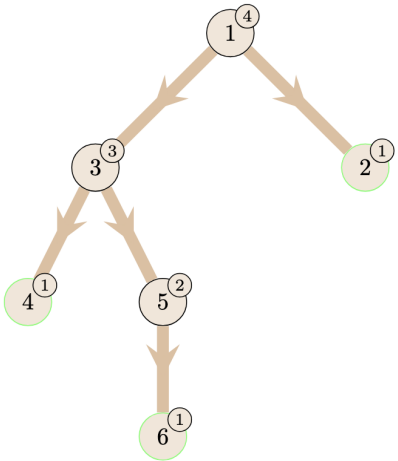
Some more concrete concepts and definitions:

- These branches will never form a cycle.
- All vertices are connected to the root vertex eventually via some combination of branches.
- The children of a vertex are the vertices that are directly connected by a branch but are further away from the root vertex.
- A *leaf* is a vertex that has no children.

In short, Winson's trees are rooted trees, in the mathematical sense. He wants to prune so that the *"dangling depth"* of his rooted trees is minimised.

- The *dangling depth* of a leaf is defined as 1.
- For any other vertex, its *dangling depth* is 1 plus the maximum *dangling depth* among its children.

Pictured on the right is Winson's third tree in the sample input. The small annotations for each vertex are the *dangling depth* of that vertex.



Now, here's the challenge:

- Winson refuses to cut his trees too much — he will perform at most k pruning operations.
- In each pruning operation, he removes exactly one leaf together with the branch connecting it.
- A vertex that was not initially a leaf may become a leaf after some pruning operations.
- The root (vertex 1) is special: it can never be pruned.

Winson loves his trees but also loves efficiency. He asks for your help: after pruning at most k times, what is the minimum possible *dangling depth* of his tree?

Input

The first line of input contains one integer:

- t ($1 \leq t \leq 10{,}000$) – the number of trees Winson owns.

The description of the t trees then follows. For each tree r:

- The first line contains two integers n_r and k_r, where n_r is the number of vertices in the tree and k_r is the maximum number of pruning operations allowed.
- The next $n_r - 1$ lines each contain two integers u_{ri} and v_{ri}, indicating that there is a branch between vertices u_{ri} and v_{ri}.

It is guaranteed that the given graph represents a tree, which can be rooted at vertex 1.

- The sum of n_i over all test cases will not exceed $500\,000$. $1 \leq n_i \leq 500\,000$
- $0 \leq k_r \leq n_r - 1$
- $1 \leq u_{ri}, v_{ri} \leq n_r$

Output

For each tree, output on a separate line the minimum *dangling depth* of the root after pruning at most k leaves.

Examples

standard input	standard output
3 3 2 1 2 1 3 6 1 1 2 1 3 3 4 3 5 4 6 6 2 1 2 3 1 3 4 5 3 5 6	1 3 3

J. GiDi-Up

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

In order to fund his battles, Ali starts a chariot-share service called GiDi-Up.

Since this is occurring in medieval times, village-to-village rides don't take hours, but days to months. And so intermediary tavern stays are a necessity.

Each village-to-village connection is bidirectional and takes a full day to traverse, and he'd like to get the patron from village 1 to village n in the least days possible.

Ali's particular service is all expenses paid, so anytime he crosses through a village with a tavern, he is obligated to let the patron stay there free of charge, adding 1 full day to his journey (and the patron *always* wants a free tavern stay). He is therefore incentivised to find his way through villages that don't have taverns.

That being said, he doesn't want to get a bad review from a noble and end up on the wrong end of a guillotine. If the patron spends more than k days on the road without seeing a tavern, then it'll be off with his head.

Can you help figure out how to get Ali and his patron from 1 to n efficiently, without getting a bad review?

Input

The first line will contain four integers:

- n ($2 \leq n \leq 10^5$), the number of villages,
- t ($1 \leq t \leq n$), the number of villages that contain a tavern,
- r ($1 \leq r \leq 10^5$), the number of village to village connections that exist,
- k ($1 \leq k \leq 5$), the number of days the patron can go without sleeping at a tavern.

The next line will contain t integers t_i ($1 \leq t_i \leq n$), detailing which villages have a tavern.

The next r lines will contain 2 integers, representing the start and end villages for each bidirectional connection.

Output

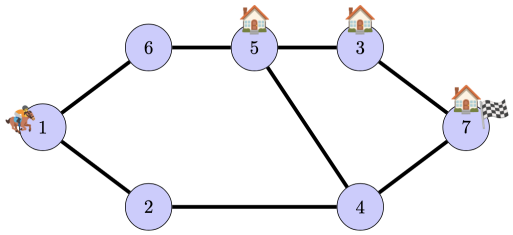
Your output should consist of only a single integer. If the trip is impossible within the constraints given, output -1.

Otherwise, print the minimum number of days required for the journey.

Examples

standard input	standard output
7 3 8 2 3 5 6 1 6 6 5 1 2 4 2 4 5 5 3 7 3 4 7	6

This is represented by the following diagram:

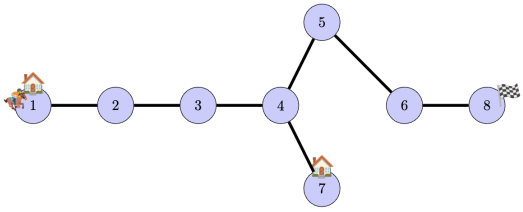


The optimal path takes 6 days to complete.

We can do this by traversing from village 1 to village 6 to village 5, where we have to stay at the tavern, then to village 4, and then to village 7 (where we then also need to stay for the night at the tavern).

standard input	standard output
8 2 7 5 1 7 2 1 2 3 3 4 7 4 4 5 6 5 8 6	10

This is represented by the following diagram:



The optimal path takes 10 days to complete.

First we have to stay at the tavern in the first village, then we can begin by traversing from village 1 to village 2 to village 3 to village 4 to village 7, where we have to stay at the tavern, then to village 4, 5, 6 and 8.

K. Downstream

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Long after his battle against the cats, Ali comes back to the battlefield later in life to realise that the tunnels he had dug had become rivers! A bit too enthusiastic about this, Ali gets too close and falls into these rivers, floating downstream to a peculiar current.

Ali is floating along a weird river current in the shape of a torus. This means that the current can be represented by a grid, whose left and right boundaries connect, as well as the top and bottom boundaries. This torus also has fish swimming in each grid square, which the ever-resourceful Ali can then catch and consume to give him energy.

At every time step, Ali is shifted c_x units to the right and c_y units down the grid, overflowing into the left/top boundaries of the grid. The coordinate system of the grid is represented by two integers (x, y), where x ranges from 1 to m (left to right) and y ranges from 1 to n (bottom to top).

Ali however does have some control in this situation - between each time step, he can move as many units left/right/up/down in the current as he would like, but he expends 1 unit of energy for every grid tile he traverses.

Suppose that the river current is represented by a 3x3 grid, Ali is always shifted left 2 units and down 1 unit between timestamps, and the energy value of the fish at each timestep were as depicted below. Then a possible path that Ali could take is depicted on the right:

```
TIMESTAMP 0
5 4 1
2 4 7
8 2 1

TIMESTAMP 1
5 4 1
3 4 7
4 6 4

TIMESTAMP 2
5 5 3
8 4 5
6 3 7
```

TIMESTAMP 0

- Begin at cell (1, 1) (Bottom-Left), and collect 8 energy
- Ali is then washed downstream to cell (2, 3) (Top-Middle)

TIMESTAMP 1

- Ali can then swim 1 unit up to (2, 1) (Bottom-Middle) and collect 6 energy (but expending 1)
- Ali is then washed downstream to cell (3, 3) (Top-Right)

TIMESTAMP 2

- Ali can then swim down 1 unit and right 1 unit to (1, 2) (Middle-Left) and collect 8 energy (but expending 2)

This results in Ali collecting a total of 8+6+8-3 = 19 energy. In short, Ali collected fish from cells (1, 1), (2, 1) and (1, 2) to collect 19 energy.

Another possible route is (1, 1), (3, 2) and (1, 2). This takes 2 energy and 1 energy to swim between the locations at timestamp 1 and 2, but collects 1 more energy in the fish tiles, resulting in 20 total energy. This is in fact the optimal route Ali could take.

Can you determine the best possible path Ali can take?

Input

Input will begin with 5 integers:

- n ($1 \leq n \leq 70$), the number of rows in the grid
- m ($1 \leq m \leq 70$), the number of columns in the grid
- t ($1 \leq t \leq 25$), the number of timestamps
- c_x ($-10^9 \leq c_x \leq 10^9$), the amount of units Ali is shifted right after each timestamp
- c_y ($-10^9 \leq c_y \leq 10^9$), the amount of units Ali is shifted down after each timestamp

$n \times t$ lines then follow, each containing m integers. The first n lines correspond to the grid at timestamp 0, the next n correspond to the grid at timestamp 1, and so on. Each fish value f will satisfy $0 \leq f \leq 10^9$.

Output

Output should begin with a single integer, representing the maximum energy Ali can achieve. t lines should follow, detailing the positions Ali should take - each line should contain two integers, Ali's coordinates. As stated previously, Bottom-Left, Bottom-Right, Top-Left and Top-Right are represented by coordinates 11, $m1$, $1n$ and mn respectively.

If there are multiple optimal paths, then print any one.

Examples

standard input	standard output
3 3 3 -2 1 5 4 1 2 4 7 8 2 1 5 4 1 3 4 7 4 6 4 5 5 3 8 4 5 6 3 7	20 1 1 3 2 1 2
2 4 3 10 -8 0 8 7 9 3 2 8 3 5 5 0 4 9 3 7 7 6 1 6 2 5 9 4 4	25 3 1 1 1 2 1