Problem A. Entangled Coins

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1s
Memory limit:	512MB

Given n coins with two sides (facing downwards or upwards), s of them are facing upwards while the remaining are the opposite.

You can operate coins **any number of times (including zero)**; in each operation, you should arbitrarily choose **exactly** k coins to flip (facing up becomes facing down and vice versa).

Your goal is to change the number of coins facing up (from s) to t . Output the **minimum** number of operations or report that it's impossible.

Input

The first line contains the number of test cases t ($1 \leq t \leq 2 \times 10^5$). The description of the test cases follows.

Each test case is represented by one line, containing four integers n, k, s, t ($1 \leq k \leq n \leq 10^9, 0 \leq s, t \leq n$) meaning as above.

Output

For each test case, answer it on a full line:

If there exists a solution, output an integer representing the minimum number of operations; or if not, output -1 as the report for impossible.

Examples

standard input	standard output
8 8 3 4 7 9 7 1 0 16 15 1 0 4 2 3 3 6 6 2 4 7 6 2 5 98257693 98257692 24 43850682 98257693 98257692 24 43850681	1 5 15 0 1 -1 43850658 -1

Problem B. Ice Pigeon vs. Fire Pigeon: Extra Training!

Input file:	standard input
Output file:	standard output
Time limit:	5s
Memory limit:	1024MB

Since Ice Pigeon taught Fire Pigeon the KMP algorithm, Fire Pigeon has been inspired to become a string master like Ice Pigeon.

Fire Pigeon asked Ice Pigeon how to achieve a high level of string problem, and Ice Pigeon replied: Extra training!

Every day Ice Pigeon trains extra-hard on string problems.

Today he is solving a problem that goes like this:

Given a string S of length n. You need to perform the following operation exactly k times:

1. For the i-th operation ($1 \le i \le k$), Choose a substring of S as S_i.

$S[l_i, r_i]$ is substring of S. (i.e., the contiguous substring of S starting at position l_i and ending at position r_i , where $1 \le l_i \le r_i \le n$). In additonal, the empty string is also a substring of S.

2. Concatenate the chosen substrings $S_i (1 \le i \le k)$ in order to form a new string $S_1 + S_2 + ... + S_k$.

Let X be the number of distinct strings that can be formed by concatenating k substrings in this way. Output X mod 998244353.

Input

The first line contains two integers n and k ($1 \le n \le 5 \times 10^5, 1 \le k \le 10^9$).

The second line contains a string S of length n, only consisting of both uppercase and lowercase English letters.

Note that uppercase and lowercase letters are treated as distinct.

Output

Output a single integer, the number of distinct strings that can be formed by concatenating k substrings (selected in order from the 1st to the k-th operation) modulo 998244353.

Examples

standard input	standard output
2 2 ab	12
3 3 abb	96
2 10 Aa	28656
9 6 IcePigeon	811212467

Note

For the first sample, the substrings of “ab” are: “” (empty string), “a”, “b”, and “ab”.

(1) “” + “” = “”

(2) “” + “a” = “a”

(3) “” + “b” = “b”

(4) “” + “ab” = “ab”

(5) “a” + “” = “a”

(6) “a” + “a” = “aa”

(7) “a” + “b” = “ab”

(8) “a” + “ab” = “aab”

(9) “b” + “” = “b”

(10) “b” + “a” = “ba”

(11) “b” + “b” = “bb”

(12) “b” + “bab” = “bab”

(13) “ab” + “” = “ab”

(14) “ab” + “a” = “aba”

(15) “ab” + “b” = “abb”

(16) “ab” + “ab” = “abab”

Among the sixteen strings above, only twelve are distinct:

“”,“a”,“b”,“ab”,“aa”,“aab”,“ba”,“bb”,“bab”,“aba”,“abb”,“abab”

Problem C. Array Deletion Game

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3s
Memory limit:	512MB

Alice and Bob play a game on an array A of length N, where all elements are positive integers. The rules are as follows:

1. Players take turns, with Alice going first.
2. On each turn, a player can:
   - Remove the leftmost element of the array, or
   - Remove the rightmost element of the array.
3. If after a player's move, the sum of the remaining elements $\leq s$, that player loses the game.

Given the initial array, you need to process Q queries. For each query with a different s, determine whether Alice has a winning strategy.

Input

The first line contains an integer N ($1 \leq N \leq 10^5$), the length of the array.

The second line contains N integers A_i ($1 \leq A_i \leq 10000$), the elements of the array.

The third line contains an integer Q ($1 \leq Q \leq 10^5$), the number of queries.

The next Q lines each contain an integer s ($1 \leq s < \sum A$), the threshold for the current query.

Output

For each query s, output one line:

- If Alice has a winning strategy, output "Alice".
- Otherwise, output "Bob".

Examples

standard input	standard output
5 1 3 5 7 9 3 10 15 20	Alice Alice Bob

Problem D. Prime XOR Permutation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2s
Memory limit:	512MB

Given an integer N, you need to construct a permutation P of integers from 0 to $N-1$ such that for all $1 \leq i < N$, $P_i \oplus P_{i+1}$ is a prime number. Here, $\oplus$ denotes the bitwise XOR operation.

Input

The first line contains an integer T ($1 \leq T \leq 2 \times 10^5$), the number of test cases.

For each test case, there is a single integer N ($1 \leq N \leq 10^6$).

It is guaranteed that the sum of N across all test cases does not exceed 10^6.

Output

For each test case:

- If a valid permutation exists, output a single line containing N space-separated integers representing the permutation P.
- If no such permutation exists, output -1.

Examples

standard input	standard output
2 4 5	3 1 2 0 4 1 3 0 2

Problem E. Mysterious XOR Operation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1s
Memory limit:	512MB

We define a special mysterious XOR operation $\oplus_m$ with the following rules:

For two numbers a and b, first compute their regular XOR result $c = a \oplus b$. Then process c's binary representation as follows:

1. Scan c's binary representation from the least significant bit to the most significant bit
2. Initialize a counter $count = 0$
3. For each bit:
   - If the bit is 1:
     - Increment *count* by 1
     - If *count* is odd, keep the bit
     - If *count* is even, clear the bit (set to 0)
   - If the bit is 0, leave it unchanged

Example: $(101001)_2 \oplus_m (10010)_2 = (101001)_2$

Given an array A of length N, compute the sum of the mysterious XOR results of A_i and A_j for all unordered pairs (i, j) where $i \neq j$.Formally, this can be expressed as $\sum_i^N \sum_{j>i}^N A_i \oplus_m A_j$.

Input

The first line contains an integer N ($2 \leq N \leq 10^5$).

The second line contains N integers A_i ($0 \leq A_i \leq 10^8$).

Output

Output a single integer representing the sum of mysterious XOR results for all unordered pairs.

Examples

standard input	standard output
3 5 3 9	8

Note

$5 \oplus_m 3 = 2$

$5 \oplus_m 9 = 4$

$3 \oplus_m 9 = 2$

So the answer equals $4 + 2 + 2 = 8$

Problem F. Grid Survival

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2s
Memory limit:	512MB

Alice and Bob are playing a game with one piece on a grid board with n rows and m columns. The rules of the game are as follows:

- The grid in the l-th line and the c-th column is called of position (l, c) , and is colored **white** when l and c have the same parity and **black** when different.
- The piece must be within a certain grid at any moment; in other words, the position of the piece can also be represented by an integer pair (l, c) , which means that the piece is in the grid of the l-th line and the c-th column, and $1 \le l \le n, 1 \le c \le m$ always holds.
- On the board there are k grid(s) specialized in advance, the i-th special grid is of position (l'_i, c'_i) and has a positive integer value w_i .
- The game lasts for a number of round(s) (starting with 1):
  - Initially, the board is empty and the piece is not placed yet;
  - In the first round, Bob chooses a color *col* either white or black, and for each special grid (of the i-th) , he can make it **active** at the cost of w_i or **inactive** with no cost;
  - In the second round, Alice chooses an arbitrary grid of color *col* to place the piece;
  - In the third round and further odd rounds, Bob must move the piece along the line by 1 unit, from position (l, c) to either $(l + 1, c)$ or $(l - 1, c)$. Note that after the move $1 \le l \le n$ should still be satisfied, and Bob can't leave the piece not moved in the round.
  - In the fourth round and further even rounds, Alice must move the piece along the column by 1 unit, from position (l, c) to either $(l, c - 1)$ or $(l, c + 1)$. Similarly, after the move $1 \le c \le m$ should still be satisfied, and Alice can't leave the piece not moved in the round.
- Whenever the piece lies in any **active** grid, Alice loses and Bob wins; else if the game lasts for 10^{100} rounds (or forever), Bob loses and Alice wins.

Assuming that both players are smart enough, and given the parameters of the grid board (n and m) and special grid(s) (l'_i, c'_i and w_i); for each chosen color, please calculate the minimal total cost for Bob to activate some special grid(s) to win, or report that it's impossible.

Note that you may answer multiple queries.

Input

The first line contains the number of test cases t ($1 \le t \le 2 \times 10^5$). The description of the test cases follows.

The first line contains three integers n, m, k ($2 \le n, m \le 10^6, 0 \le k \le \min(2 \times 10^5, n \times m)$) meaning as above;

Then following k lines, the i-th line contains three integers l'_i, c'_i ($1 \le l'_i \le n, 1 \le c'_i \le m$) and w_i ($1 \le w_i \le 10^9$), representing the located row and column and the value of the i-th special grid; it's guaranteed that in the same test case no two special grids fall in the same position ($\forall\ 1 \le i < j \le k \Rightarrow l'_i \ne l'_j \vee c'_i \ne c'_j$).

It's also guaranteed that the sum of k in each test file doesn't exceed 2×10^5 ($\sum k \le 2 \times 10^5$); note that there are no restrictions with the sum of n and m .

Output

For each test case, answer it on a full line, containing two integers representing the answer for the chosen color (white or black in order): the minimal total cost if possible, or -1 if impossible.

Examples

standard input	standard output
7 2 3 4 1 1 52 1 2 33 2 1 47 2 2 95 2 3 2 1 2 90 1 3 30 5 5 4 4 2 30 4 4 10 2 4 40 2 2 20 6 6 4 6 2 10 6 4 10 1 3 10 1 5 10 5 6 4 5 2 10 5 4 10 1 3 10 1 5 10 11 5 5 2 1 1 2 3 1 6 3 1 10 3 1 10 5 1 2 2 0	33 33 90 90 40 30 -1 40 40 40 5 -1 -1 -1

Problem G. Line of Sight

Input file:	`standard input`
Output file:	`standard output`
Time limit:	8s
Memory limit:	512MB

Now SATSKY transformed the stories of Moon Cat into a programming geometry problem, in commemoration of the memorable year with them, the treasure of his life.

Formally:

- Give you an n-sided polygon P, of which each point $P_{1\sim n}$ is of integer coordinates and given in **counterclockwise** order;
- And give you two more integer points A and B, inside the polygon (**boundary not included**) guaranteed;
- Call "point X can be seen from point Y (in P)" **if and only if** the whole segment connecting two points, **except two endpoints**, is inside P strictly (and doesn't intersect with any of edge of P naturally) ;
- You need to find out:
  - For each point of polygon (i from 1 to n), can it (P_i) be seen from point A ?
  - For each point of polygon (i from 1 to n), can it (P_i) be seen from point B ?
  - Is there a point C inside the polygon (**boundary not included**) to see A and B simultaneously ?

Input

The first line contains the number of test cases t ($1 \le t \le 2 \times 10^5$). The description of the test cases follows.

The first line contains two integers n ($3 \le n \le 2 \times 10^5$), indicating the number of points of P;

The second line contains four integers x_A, y_A, x_B, y_B , representing the x and y coordinates of point A and B;

Then following n lines, the i-th line contains two integers x_{P_i}, y_{P_i} , representing the x and y coordinates of point P_i.

It's guaranteed that:

- $P_{1\sim n}$ are given in **counterclockwise** order and must form a polygon;
- For A,B and each point of P ($P_{1\sim n}$), all the coordinates satisfy $|x|, |y| \le 10^9$;
- The sum of n in each test file doesn't exceed 10^6 ($\sum n \le 10^6$).

Some emphasis:

- Adjacent edges of P may be collinear, so are not adjacent pairs;
- Any three points within A,B and $P_{1\sim n}$ may be on the same straight line, A and B may even coincide;
- The coordinates of C you choose can be **decimals**.

Output

For each test case, answer it on three lines:

In the first line, output n integers separated by spaces: the i-th integer should be 1 if P_i can be seen from point A, or 0 if it can't;

In the second line, output n integers separated by spaces: the i-th integer should be 1 if P_i can be seen from point B, or 0 if it can't;

In the third line, if there exists a point C meeting the requirements above, output a string "Yes" (without quotes); otherwise output a string "No" (without quotes).

Examples

standard input	standard output
2 8 -1 1 1 -1 2 -1 2 1 1 2 -1 2 -2 1 -2 -1 -1 -2 1 -2 13 1 1 6 1 0 1 1 0 2 1 3 1 4 0 5 0 6 0 7 1 6 2 5 1 4 1 3 2 1 2	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 Yes 1 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 1 1 1 1 1 0 0 0 No

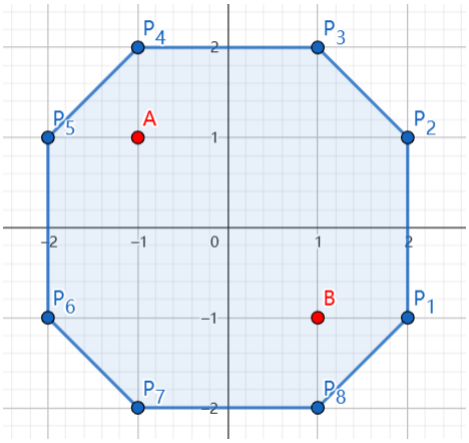
standard input	standard output
4 8 -2 0 2 0 -3 -1 -1 -1 -1 1 1 1 1 -1 3 -1 3 2 -3 2 8 -2 0 2 0 -3 -1 -1 -1 -1 1 1 1 1 -1 3 -1 3 3 -3 3 12 -4 -4 4 -4 -2 1 -2 -1 -5 -5 -1 -2 1 -2 5 -5 2 -1 2 1 5 5 1 2 -1 2 -5 5 8 -4 -4 4 -4 -5 2 -2 -1 -5 -5 -1 -2 1 -2 5 -5 2 -1 5 2	1 1 1 0 0 0 0 1 0 0 0 1 1 1 1 0 No 1 1 1 0 0 0 0 1 0 0 0 1 1 1 1 0 Yes 0 1 1 1 0 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 0 0 1 1 Yes 0 1 1 1 0 0 0 0 0 0 0 0 1 1 1 0 Yes

standard input	standard output
2 26 6 20 20 6 23 8 19 6 16 5 13 5 10 6 8 7 7 8 6 10 5 13 5 16 6 19 8 23 5 21 3 19 2 17 1 14 1 11 2 7 3 5 5 3 7 2 11 1 14 1 17 2 19 3 21 5 18 -15 20 15 20 -3 16 -6 15 -15 22 -18 13 -18 7 -21 -2 -21 -8 -18 -14 -6 -20 6 -20 18 -14 21 -8 21 -2 18 7 18 13 15 22 6 15 3 16	0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 No 0 1 1 1 1 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 1 1 1 1 0 Yes

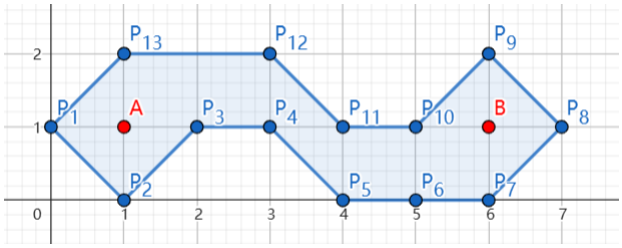
Note

Graphs of all samples are shown below in order:

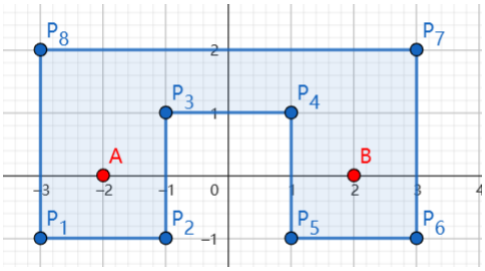
sample 1-1:



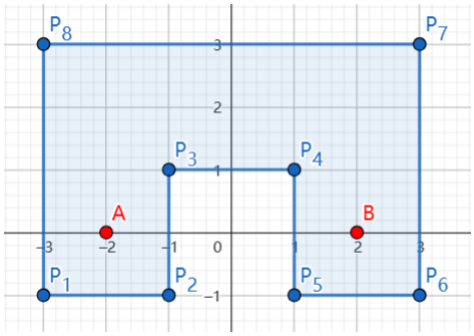
sample 1-2:



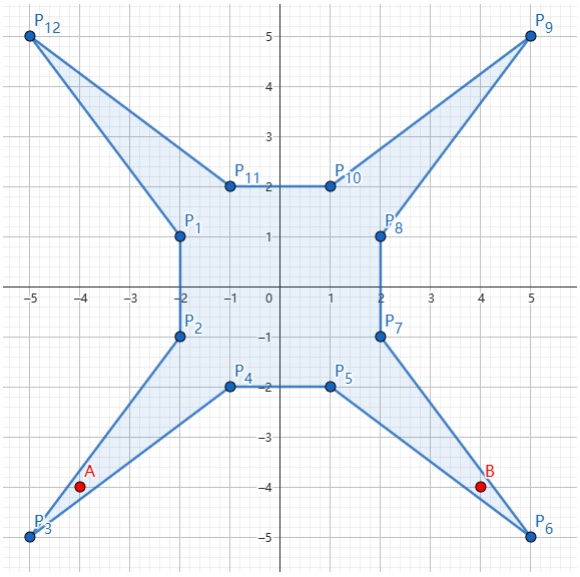
sample 2-1:



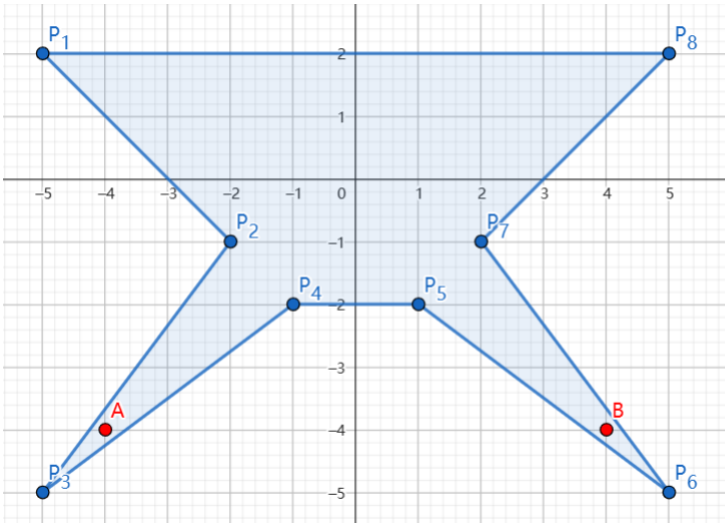
sample 2-2:



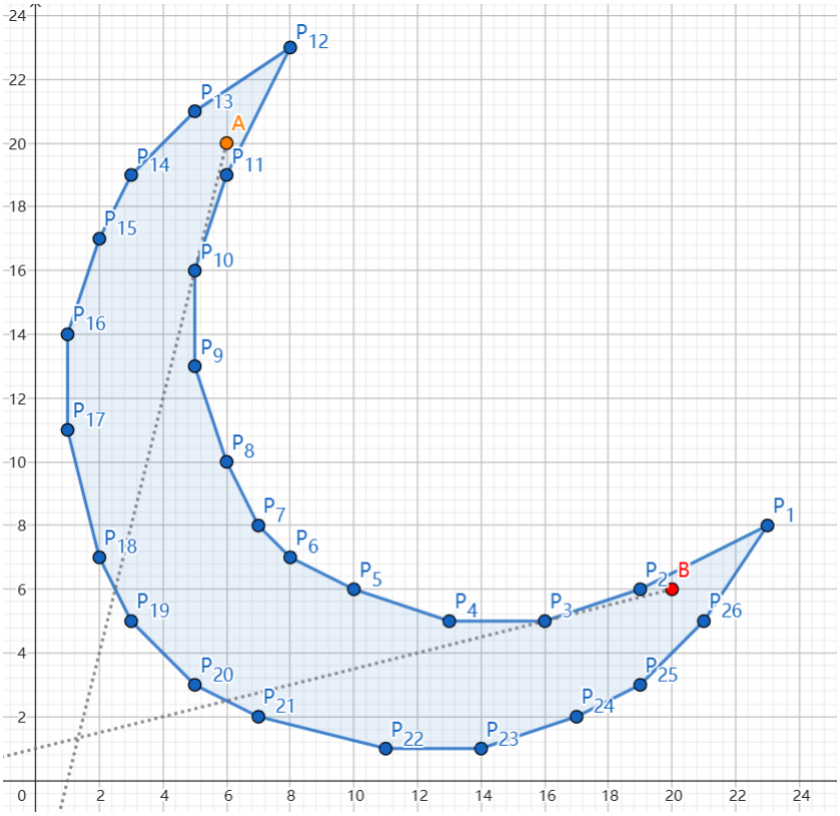
sample 2-3:



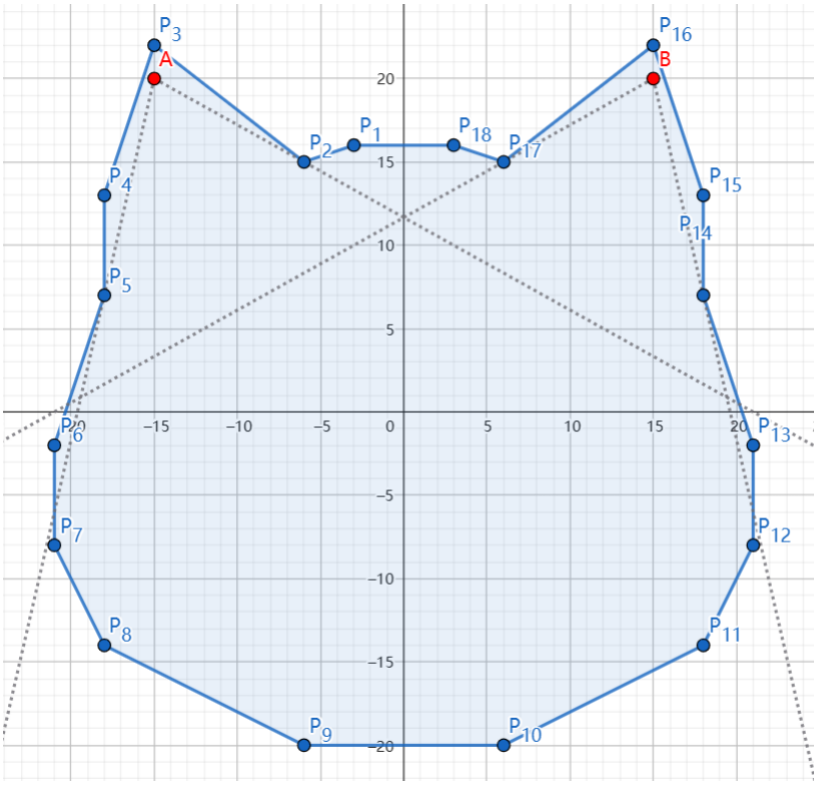
sample 2-4:



sample 3-1:


P12
P13
A
P14
P11
P15
P10
P16
P9
P17
P8
P7
P1
P18
P6
P5
P2
B
P4
P3
P26
P19
P25
P20
P24
P21
P22
P23

sample 3-2:


P3
A
P16
B
P2
P1
P18
P17
P4
P15
P14
P5
P6
P13
P7
P12
P8
P11
P9
P10

Problem H. VI Civilization

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1s
Memory limit:	512MB

In the game VI Civilization, the player needs to achieve a science victory: accumulating at least s science points in the science victory slot within t turns.

There are n technologies in the game. Initially, only the first technology, $Tech_1$, is unlocked and can be completed. All other technologies are locked. The player must complete the technologies in a fixed sequence: $Tech_1 \to Tech_2 \to \cdots \to Tech_n$. This order cannot be skipped or changed. Specifically, technology $Tech_i$ is unlocked only after all preceding technologies ($Tech_1$ to $Tech_{i-1}$) have been completed.

Completing each technology requires a certain amount of science points. The player can allocate production to trigger the technology's "Eureka" moment, which reduces the science points required for completion. **Each technology's Eureka can only be triggered once**. Upon completing a technology, the science points gained per turn will increase.

Each technology $Tech_i$ has four parameters:

- a_i: The science points required to complete this technology.
- k_i: The increase in science points per turn after completion.
- b_i: The production required to trigger its Eureka.
- c_i: The reduction in required science points after triggering the Eureka ($0 \le c_i < a_i$).

VI Civilization is a turn-based game. In each turn, the player first gains science points and production, and then allocates them. The allocation of science points and production must be **indivisible (cannot be split among multiple tasks)**, and the science points and production gained in the current turn **are not saved for the next turn**.

The game proceeds as follows:

1. At the start of each turn, the player gains:
   - Science points m (after completing technology i, m permanently increases by k_i).
   - A fixed amount of production p (remains constant throughout the game).
2. Then, the player performs actions:
   - **Science Point Allocation:**
     (a) Allocate the entire amount of science points m gained this turn to either an unlocked technology or the science victory slot.
     (b) When allocating to a technology, any excess points are wasted and do not carry over to the next technology. After completing technology $Tech_i$, m permanently increases by k_i.
     (c) When allocated to the science victory slot, the points are directly added to its total.
   - **Production Allocation:**
     (a) Allocate the entire amount of production p gained this turn **indivisibly (cannot be split among multiple Eurekas)** to the Eureka of **any technology (regardless of whether the technology is unlocked)**.
     (b) When allocating to a Eureka, any excess production is wasted. After a Eureka is triggered, the science point required to complete the corresponding technology is reduced.

Find the minimum non-negative integer production p such that there exists a strategy to achieve the science victory (accumulating $\geq s$ science points in the science victory slot) within t turns. If it's impossible to win within t turns, output -1.

Input

The first line contains three integers m, s, and t ($1 \leq m \leq 100$, $1 \leq s \leq 10^9$, $1 \leq t \leq 100$).

The second line contains an integer n ($0 \leq n \leq 100$).

The next n lines each contain four integers a_i, k_i, b_i, and c_i ($1 \leq a_i \leq 10^6$, $0 \leq k_i \leq 1000$, $1 \leq b_i \leq 10000$, $0 \leq c_i < a_i$).

Output

Output the minimum non-negative integer p. If it is impossible to win within t turns, output -1.

Examples

standard input	standard output
10 100 9 2 50 10 20 25 60 10 30 20	4
22 970 8 3 85 24 9 27 81 20 85 44 30 80 75 7	-1

Note

In the sample case, a valid strategy for $p = 4$ is as follows:

Turn 1: Gain 10 science and 4 production. Allocate production to $Tech_1$'s Eureka and science to $Tech_1$.

Turn 2: Gain 10 science and 4 production. Allocate production to $Tech_1$'s Eureka and science to $Tech_1$.

Turn 3: Gain 10 science and 4 production. Allocate production to $Tech_1$'s Eureka and science to $Tech_1$.

Turn 4: Gain 10 science and 4 production. Allocate production to $Tech_1$'s Eureka and science to the science victory slot.

Turn 5: Gain 10 science and 4 production. Allocate production to $Tech_1$'s Eureka and science to the science victory slot. At this turn, $Tech_1$'s Eureka has accumulated 20 production, triggering the Eureka. The science cost for $Tech_1$ is now $50 - 25 = 25$. Since 30 science points have already been allocated, $Tech_1$ is completed. The science per turn increases to $10 + 10 = 20$.

Turn 6: Gain 20 science and 4 production. Allocate science to the science victory slot.

Turn 7: Gain 20 science and 4 production. Allocate science to the science victory slot.

Turn 8: Gain 20 science and 4 production. Allocate science to the science victory slot.

Turn 9: Gain 20 science and 4 production. Allocate science to the science victory slot. The science victory slot has accumulated a total of 10(T4) + 10(T5) + 20(T6) + 20(T7) + 20(T8) + 20(T9) = 100 science points. Science victory is achieved!

Problem I. Block Combination Minimal Perimeter

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1s
Memory limit:	512MB

Given n rectangular blocks, where the i-th block has dimensions $1 \times i$. You need to combine **all** blocks to form a solid rectangle (no overlaps or empty spaces allowed). Find the minimal perimeter of the resulting rectangle.

It is guaranteed that you can always form a solid rectangle under the problem constraints.

Input

The first line contains an integer n ($1 \leq n \leq 2 \times 10^5$), representing the number of blocks.

Output

Output a single integer representing the minimal perimeter of the formed rectangle.

Examples

standard input	standard output
1	4
6	20
10	32

Problem J. Fastest Coverage Problem

Input file:	standard input
Output file:	standard output
Time limit:	2s
Memory limit:	512MB

Given an $n \times m$ binary matrix where 1 represents a black cell and 0 represents a white cell. Every second, each black cell will convert its four neighboring white cells (top, bottom, left, right) to black.

You may change **at most** one white cell to black (convert 0 to 1) to minimize the time required for the entire matrix to become completely black. Find this minimum time.

Input

The first line contains two integers n and m ($1 \leq n \times m \leq 2 \times 10^5$), representing the number of rows and columns of the matrix.

The next n lines each contain m digits (0 or 1), representing the initial state of the matrix.

Output

Output a single integer representing the minimum time required to turn the entire matrix black after adding one black cell.

Examples

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

Note

For sample 1, You can choose position (1,1).

Matrix at second 0:

1 0 0
0 0 0
0 0 1

Matrix at second 1:

1 1 0
1 0 1
0 1 1

Matrix at second 2:

1 1 1
1 1 1
1 1 1

Problem K. Perfect Journey

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2s
Memory limit:	512MB

In a country with n cities, there are $n-1$ bidirectional roads connecting these cities, forming a tree.You are now visiting this country, and there are m specific roads that you definitely want to travel through.The travel agency offers k optional tour routes. Each route starts from city s_i and follows the shortest path to reach city t_i.

Your goal is to select as few routes as possible from these k tour routes, ensuring that all m key roads are traveled at least once.

Please calculate the minimum number of tour routes you need to select, and the number of possible ways to achieve this minimum, with the answer modulo 998244353. A plan is defined as the set of tour routes you choose. Two plans are considered different if and only if there exists a tour route that is selected in one plan but not in the other.

If it's impossible to travel through all the specific roads, output -1.

The problem guarantees that the answer is non-zero modulo 998244353.

Input

The first line contains three integers $2 \le n \le 2 \times 10^5$, $1 \le m \le 22$, $1 \le k \le 2 \times 10^5$, representing the number of cities, the number of specific roads, and the number of tour routes.

The next $n-1$ lines each contain two integers $1 \le u_i, v_i \le n$, guaranteeing that the given graph is a tree.

The next line contains m distinct integers $1 \le x_i \le n-1$ representing the indices of the specific roads (according to the input order).

The next k lines each contain two integers $1 \le s_i, t_i \le n$, indicating that the tour route goes from s_i to t_i.

Output

Output the minimum number of tour routes you need to select, and the number of ways to achieve this minimum, with the answer modulo 998244353.

If it's impossible to travel through all the specific roads, output -1.

Examples

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

Note

For sample 2, we need to select at least 2 routes, and there are two possible plans:

1. Route 1 and Route 2, Route 1 covers road 1 and road 3, Route 2 covers road 1 and road 5.
2. Route 2 and Route 3, Route 2 covers road 1 and road 5, Route 3 covers road 3.

Problem L. Float

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1s
Memory limit:	512MB

There are n levels, the i-th level allows one traveling from point $i-1$ to point i ; you start at point 0 , and want to pass through each level sequentially to reach point n. Each time you attempt a level, you have a probability p of passing and a probability $(1-p)$ of failing.

Initially you have some number of coins, if you fail and have at least one coin, you will stay at the current level and retry by consuming one coin (you won't leave it unused even if you are at point 0); and if you fail and have no coins left, you will be kicked back to the starting point (point 0) .

Given m, calculate the expected number of steps to reach the endpoint for every possible initial number of coins from 0 to m, modulo 998244353.

Input

The first line contains three integers n, m, and p ($1 \le n \le 10^9$, $1 \le m \le 2 \times 10^5$, $0 < p < 998244353$), representing the number of levels, the maximum number of coins, and the probability of passing a level; the probability has been moduloed by 998244353.

Output

A single line containing $m+1$ integers, where the i-th number represents the expected number of steps to reach the endpoint when starting with $i-1$ coins modulo 998244353.

Examples

standard input	standard output
2 2 499122177	6 499122182 5
1 3 332748118	3 3 3 3

Note

Under modulo 998244353, the value 332748118 corresponds to a probability of $\frac{1}{3}$.

In this scenario, the result is independent of m, because regardless of whether coins are available or not, failure always results in returning to the starting point.

Problem M. Mysterious Spacetime

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4s
Memory limit:	512MB

In a mysterious spacetime, there are x unknown energies, numbered from 1 to x. These energies appear at specific times and positions, following these rules:

1. **Spacetime Appearance Rule**:

- There are n different time points, numbered from 1 to n.
- At time point t, all energies in the interval $[l, r]$ will appear and disappear at time point $t + 1$.

2. **Generation Rule**:

- There are m generators, numbered from 1 to m. Each generator i requires all energies in the interval $[L_i, R_i]$.
- If at some integer time point t, at least k_i energies in the interval $[L_i, R_i]$ appear simultaneously, then generator i will be activated at time point t.
- Each generator is activated only once at the earliest moment when requirements are met.

You need to process q queries. Each query gives `tl tr l r`, asking:

- Within the time range [tl, tr], what is the earliest time point t such that: there exists some generator i satisfying $l \le L_i \le R_i \le r$ that is activated at time point t.
- If no such t exists, output -1.

Input

The first line contains a number $1 \le T \le 10^5$, indicating there are T test cases, each in the following format:

- The first line contains four integers $1 \le n, m, x, q \le 5 \times 10^5$.
- The next n lines, each containing three integers $1 \le t \le 10^9$, $1 \le l \le r \le x$, indicating that at time point t, all energies in interval $[l, r]$ appear. (The problem guarantees all time points are different)
- The next m lines, each containing three integers $1 \le L_i \le R_i \le x$, $1 \le k_i \le R_i - L_i + 1$, describing the requirements of generator i.
- The next q lines, each containing four integers $1 \le tl \le tr \le 10^9$, $1 \le l \le r \le x$, representing a query.

The problem guarantees $\sum n, \sum m, \sum q, \sum x \le 5 \times 10^5$

Output

For each query, output one integer per line, representing the answer.

Examples

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