Problem A. Ant and Sticks

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

You are given two distinct points on an infinite table, $A = (A_x, A_y)$ and $B = (B_x, B_y)$.

You also have M sticks, each of which is a line segment of length W. You may place the sticks anywhere on the table, with arbitrary position and orientation. The sticks may overlap. However, no stick may contain point A or point B.

An ant starts at A and wants to reach B. It may walk anywhere on the table, but it cannot cross any stick.

You want to place the sticks so that the length of the ant's shortest path from A to B is as large as possible.

If it is possible to block the ant completely, so that no path from A to B exists, then the answer is -1.

Otherwise, let L be the supremum[†], over all valid stick placements, of the length of the shortest path from A to B. If there exists a placement for which this value is attained, then L is simply the maximum. If it is not attained by any single placement, but can be approached arbitrarily closely, you must still output L.

[†]The supremum (least upper bound) of a set of real values is the smallest real number that is greater than or equal to every value in the set. It may or may not be attained by some value in the set.

Input

The input is given in the following format:

T
$A_x\ A_y\ B_x\ B_y\ M\ W$
$\vdots$

- All input values are integers.
- $1 \le T \le 10^4$
- $0 \le A_x, A_y, B_x, B_y \le 10^9$
- $1 \le M, W \le 10^9$
- It is guaranteed that $(A_x, A_y) \ne (B_x, B_y)$.

Output

For each test case:

- print -1 if the ant can be blocked completely;
- otherwise print one real number — the value L defined above.

Let X be the correct answer and Y be your output. Your output is accepted if and only if

$$|Y - X| \le 10^{-6} \cdot \max(1, |X|).$$

Examples

standard input	standard output
2 0 0 1 1 343 434 4 5 786331192 707731892 1 639711202	-1 1425074478.231629371643066

Note

Test case 1: It can be proven that it is possible to block the ant completely by placing the sticks optimally.

Problem B. Palindrome and Permutation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

Suppose you have an array B of length M. Your goal is to make B a palindrome.

An array B of length M is said to be a palindrome if $B_i = B_{M+1-i}$ for every $1 \le i \le M$.

To achieve this, you will choose a permutation P of $\{1, 2, \ldots, M\}$ and perform the following process using it:

In the i-th step ($1 \le i \le M$), you must choose exactly one index j ($1 \le j \le M$) and set $B_j = P_i$.

It is allowed to choose the same index j in different steps.

The process ends either after all M indices of P have been processed, or immediately after B becomes a palindrome for the first time.

Define $f(B, P)$ to be the minimum number of steps after which B can become a palindrome, if you choose the indices to operate on optimally.

In particular:

- If B is already a palindrome initially, then $f(B, P) = 0$.
- If it is impossible to make B a palindrome even after processing every element of P, we define $f(B, P) = M + 1$.

You are given an array A of length N.

Calculate the sum of $f(A, P)$ over all possible permutations P of $\{1, 2, \ldots, N\}$.

Since the answer might be large, output it modulo 998 244 353.

Input

The input is given in the following format:

T
N
$A_1\ A_2\ \cdots\ A_N$
$\vdots$

- All input values are integers.
- $1 \le T \le 10^5$
- $1 \le N \le 5000$
- $1 \le A_i \le N$
- It is guaranteed that the sum of N^2 over all test cases does not exceed 5000^2.

Output

For each test case, output a single integer — the value of

$$\sum_P f(A, P)$$

over all permutations P of $\{1, 2, \ldots, N\}$, modulo 998 244 353.

Examples

standard input	standard output
4 3 1 2 3 4 2 3 3 2 6 4 6 4 6 4 6 10 1 5 2 10 2 6 3 1 10 2	8 0 5040 31363200

Note

Test case 1: For $P = [1, 2, 3]$ and $A = [1, 2, 3]$, the minimum number of moves needed is 1, since A can be turned into a palindrome in the first move.

Test case 2: As A is already a palindrome initially, $f(A, P) = 0$ for every permutation P.

Problem C. Score Queries

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	1024 megabytes

Let B be an array of length M. The **score** of B is the number of indices i ($2 \le i \le M-1$) such that there exist indices x and y satisfying $1 \le x < i < y \le M$ and

$$2 \cdot B_i > B_x + B_y$$

You are given an array A of length N.

You need to answer Q queries.

For each query L R ($1 \le L < R \le N$, $R - L + 1 \ge 3$), find

$$\sum_{i=L}^{R-2} \sum_{j=i+2}^{R} \text{score}(A_i, A_{i+1}, \ldots, A_j)$$

In other words, find the sum of scores over all subarrays of length at least 3 fully contained inside $A_L, A_{L+1}, \ldots, A_R$.

Input

The input is given in the following format:

T
N Q
A_1 A_2 $\cdots$ A_N
L_1 R_1
$\vdots$
L_Q R_Q
$\vdots$

- All input values are integers.
- $1 \le T \le 10^4$
- $3 \le N \le 5 \times 10^5$
- $1 \le Q \le 5 \times 10^5$
- $1 \le A_i \le N$
- $1 \le L < R \le N$ and $R - L + 1 \ge 3$.
- It is guaranteed that the sum of N over all test cases does not exceed 5×10^5.
- It is guaranteed that the sum of Q over all test cases does not exceed 5×10^5.

Output

For each query, output one integer: the sum of scores over all subarrays of length at least 3 fully contained inside $A_L, A_{L+1}, \ldots, A_R$.

Examples

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

Note

Test case 1: For query $[1, 5]$, the subarrays of length at least 3 are:

- $[2, 5, 1]$, score $= 1$,
- $[5, 1, 3]$, score $= 0$,
- $[1, 3, 4]$, score $= 1$,
- $[2, 5, 1, 3]$, score $= 1$,
- $[5, 1, 3, 4]$, score $= 1$,
- $[2, 5, 1, 3, 4]$, score $= 2$.

So, the answer is 6.

Problem D. Merging Maximum

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

You are given a permutation P of $\{1, 2, \ldots, N\}$.

That is, P is an array of length N containing every integer from 1 to N exactly once.

You can perform the following operation on it:

- Choose an index i $(1 < i < |P|)$.
- Set $P_i = \max(P_{i-1}, P_i, P_{i+1})$.
- Delete P_{i-1} and P_{i+1} from P. The remaining elements are then re-indexed to start from 1.

Each such operation reduces the length of P by 2. In particular, if the length of P is 1 or 2, then this operation cannot be applied.

Find the number of distinct arrays that can be reached by performing this operation zero or more times starting from P.

Since this value might be large, you only need to find it modulo 998 244 353.

Input

The input is given in the following format:

T
N
$P_1\ P_2\ \cdots\ P_N$
$\vdots$

- All input values are integers.
- $1 \le T \le 10^5$
- $1 \le N \le 5000$
- $1 \le P_i \le N$
- $P_i \ne P_j$ for all $i \ne j$.
- It is guaranteed that the sum of N^2 over all test cases does not exceed 5000^2.

Output

For each test case, output a single integer — the number of distinct arrays that can be reached by performing the operation zero or more times on P, modulo 998 244 353.

Examples

standard input	standard output
5 2 2 1 3 2 1 3 5 3 1 4 5 2 10 1 2 3 4 5 6 7 8 9 10 15 5 7 6 12 2 13 10 1 11 4 3 9 14 15 8	1 2 5 55 365

Note

Test case 1: No operations can be performed.

Test case 2: There are two possibilities:

- Perform no operations. The final array is $[2, 1, 3]$.
- Perform one operation with $i = 2$. The final array is $[3]$.

These are the only reachable arrays.

Problem E. Optimal Splitting

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

You are given a string S of length N consisting of lowercase latin letters.

You need to partition S into two disjoint subsequences X and Y such that each character of S belongs to exactly one subsequence.

Without loss of generality, assume $|X| \geq |Y|$ (otherwise, swap X and Y). Append character `a` to the end of Y until $|Y| = |X|$.

Now define a string Z of length $|X|$ such that $Z_i = \max(X_i, Y_i)$ for every $1 \leq i \leq |X|$.

Find the lexicographically smallest possible string Z.

A sequence is a subsequence of a sequence if it can be obtained from it by deleting several, possibly zero or all, elements from arbitrary positions.

A string U is said to be lexicographically smaller than a string V if and only if one of the following conditions holds:

- U is a prefix of V, but $U \neq V$;
- In the first position where U and V differ, the string U has a letter that appears earlier in the alphabet than the corresponding letter in V.

Input

The input is given in the following format:

T
N
S
$\vdots$

- $1 \leq T \leq 10^4$
- $1 \leq N \leq 5000$
- S is a string of length N consisting of lowercase latin letters.
- It is guaranteed that the sum of N^2 over all test cases does not exceed 5000^2.

Output

For each test case, output one line containing the lexicographically smallest possible string Z.

Examples

standard input	standard output
3 1 a 2 za 5 ababa	a z aba

Note

Test case 3: Choose $X = \texttt{aba}$ and $Y = \texttt{ab}$. After appending one `a` to Y, we get $Y = \texttt{aba}$, so $Z = \max(\texttt{aba}, \texttt{aba}) = \texttt{aba}$. Hence, the answer is `aba`.

Problem F. Restricted Removals

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

Alice and Bob are given an integer array A of length N.

Alice chooses a binary string B of length N with exactly K ones, and this string is fixed for the whole game.

At any moment, index i is protected if and only if $B_i = 1$, and Bob may choose index i if and only if $B_i = 0$.

Bob makes zero or more moves. Bob's score initially equals 0.

In one move, Bob chooses an index i such that:

- $B_i = 0$;
- $1 \le i \le |A|$.

Bob adds the current value A_i to his score and removes this element from A. After each deletion, $|A|$ decreases by 1, and the remaining elements of A are re-indexed from 1 to $|A|$.

Therefore, B is applied to current index positions, not to fixed element identities. After shifts, the same element may become protected or removable.

If Bob chooses indices $P_1, P_2, \ldots, P_m$ across moves, they must be non-increasing, that is,

$$P_1 \ge P_2 \ge \cdots \ge P_m.$$

Bob may choose the same index multiple times.

Alice wants to minimize Bob's score, and Bob wants to maximize it.

Find Bob's final score if both players play optimally.

Input

The input is given in the following format:

T
$N\ K$
$A_1\ A_2\ \cdots\ A_N$
$\vdots$

- All input values are integers.
- $1 \le T \le 10^5$
- $1 \le N \le 6000$
- $0 \le K \le N$
- $-10^9 \le A_i \le 10^9$
- It is guaranteed that the sum of N over all test cases does not exceed 10^6.
- It is guaranteed that the sum of N^2 over all test cases does not exceed 6000^2.

Output

For each test case, output a single integer — Bob's final score.

Examples

standard input	standard output
6 5 2 4 -1 7 -3 2 4 0 -5 -2 -1 -3 4 4 2 -7 5 1 1 0 -1 2 1 0 1 3 1 1 -1 1	7 0 0 0 1 1

Note

Test case 1: Alice can choose $B = \texttt{10001}$. Then, initially, the indices available to Bob are 2, 3 and 4 with values -1, 7 and -3. Bob can guarantee a score of 7 by taking index 3 once (adding value 7). After that, every next chosen index must be at most 3, so index 4 cannot be chosen, and any further legal move adds a non-positive value. Alice can force that Bob cannot do better than 7, so the answer is 7.

Test case 2: As $K = 0$, Alice has no choice, $B = \texttt{0000}$. All indices are available to Bob, but every array value is negative. Any move would decrease Bob's score, so Bob's optimal play is to make no move. Therefore, the answer is 0.

Test case 3: As $K = N$, Alice has no choice, $B = \texttt{1111}$. Every position is protected, so Bob has no legal move at all. Hence, Bob's score is 0.

Problem G. Sequence Domination

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

An integer sequence $V_1, V_2, \ldots, V_N$ is called **super decreasing** if $V_N \geq 0$, and $V_i \geq V_{i+1} + V_{i+2} + \cdots + V_N$ for all $1 \leq i < N$.

Given positive integers N and M, find the number of pairs of integer sequences A and B of length N such that $1 \leq A_i, B_i \leq M$ for all $1 \leq i \leq N$, and for all super decreasing sequences V of length N,

$$\sum_{i=1}^{N} A_i V_i \geq \sum_{i=1}^{N} B_i V_i.$$

Since the number of such pairs can be very large, print the answer modulo 998 244 353.

Input

The input is given in the following format:

T
$N\ M$
$\vdots$

- All input values are integers.
- $1 \leq T \leq 100$
- $1 \leq N, M \leq 5000$
- It is guaranteed that the sum of N over all test cases does not exceed 5000.
- It is guaranteed that the sum of M over all test cases does not exceed 5000.

Output

For each test case, output a single integer — the number of pairs of sequences (A, B) satisfying the conditions, modulo 998 244 353.

Examples

standard input	standard output
4 1 1 2 2 2 1 34 43	1 10 1 711021868

Note

Test case 1: The only possible pair of sequences is $([1], [1])$, and it is valid.

Test case 2: The following are valid pairs of sequences:

- For $B = [1, 1]$, any choice of A is valid. There are four possible choices.
- For $B = [1, 2]$, valid choices of A are $[1, 2]$, $[2, 1]$, $[2, 2]$.
- For $B = [2, 1]$, valid choices of A are $[2, 1]$, $[2, 2]$.

- For $B = [2, 2]$, the only valid choice of A is $[2, 2]$.

So, the number of valid pairs of sequences equals $4 + 3 + 2 + 1 = 10$.

Problem H. Maximum Difference

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

For an array B, define its **score**, denoted by score(B), as follows.

Start with a variable $X = 0$.

In one operation, choose a non-empty subsequence[†] S of the current array B. Then:

- Update $X \leftarrow \max(X, \max(S) - \min(S))$;
- Sort the elements of S in non-decreasing order;
- Write the sorted elements back to the same chosen positions of B.

You may apply operations any number of times.

The value score(B) is the minimum possible final value of X such that, after some sequence of operations, the whole array B becomes non-decreasing.

You are given an array A of length N.

You are also given M updates. In each update, two integers P and Y are given, and you must set $A_P = Y$. The updates are persistent.

After each update, output score(A) for the current array.

[†] A sequence S is a subsequence of a sequence C if S can be obtained from C by deleting several, possibly zero or all, elements from arbitrary positions.

Input

The input is given in the following format:

$$
\begin{array}{l}
T \\
N\ M \\
A_1\ A_2\ \cdots\ A_N \\
P_1\ Y_1 \\
\vdots \\
P_M\ Y_M \\
\vdots
\end{array}
$$

- All input values are integers.
- $1 \le T \le 10^4$
- $1 \le N, M \le 5 \times 10^5$
- $1 \le A_i \le N$
- $1 \le P_i \le N$
- $1 \le Y_i \le N$
- It is guaranteed that the sum of N over all test cases does not exceed 5×10^5.
- It is guaranteed that the sum of M over all test cases does not exceed 5×10^5.

Output

For each test case, after each update, print one integer on a separate line — the value of score(A) after applying this update.

Examples

standard input	standard output
2 4 4 4 4 1 3 4 4 3 4 3 2 2 3 6 1 2 1 4 4 4 5 3 5	3 0 2 1 1

Note

Test case 1: After the first update, the array becomes $[4, 4, 1, 4]$, and the answer is 3.

In the same test case, after the second update, the array becomes $[4, 4, 4, 4]$, which is already sorted, so the answer is 0.

Test case 2: After the only update, the array becomes $[2, 1, 5, 4, 4, 5]$, and the answer is 1.

Problem I. Point Mirror

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

There are N points on a number line. Initially, the i-th point is at coordinate A_i.

You can perform the following operations any number of times, in any order:

- Choose two distinct points i and j, and swap them.
- Choose two distinct points i and j, and mirror point i across point j. Formally, set $A_i := 2A_j - A_i$.

Is it possible to have the i-th point be at B_i after all operations for all $1 \le i \le N$?

Input

The input is given in the following format:

T
N
A_1 B_1
A_2 B_2
$\vdots$
A_N B_N
$\vdots$

- All input values are integers.
- $1 \le T \le 10^4$
- $1 \le N \le 2 \times 10^5$
- $-10^9 \le A_i, B_i \le 10^9$
- It is guaranteed that the sum of N over all test cases does not exceed 2×10^5.

Output

For each test case, output `YES` if it is possible to reach the desired final configuration, and `NO` otherwise.

You can output the answer in any case. For example, the strings `YES`, `yes`, and `yEs` will all be recognized as positive responses.

Examples

standard input	standard output
6 1 1 1 1 0 2 2 0 3 1 2 2 1 5 2 7 3 -3 5 9 13 7 5 3 -3 11 9 25 7 9	YES NO YES NO NO YES

Note

Test case 3: One valid sequence of operations is:

1. Start from coordinates $[0, 1]$.
2. Mirror point 1 across point 2. The coordinates become $[2, 1]$.
3. Swap point 1 with point 2. The coordinates become $[1, 2]$.
4. Mirror point 1 across point 2. The coordinates become $[3, 2]$.

Test cases 2, 4, and 5: It can be proven that it is impossible to have the points at the desired coordinates.

Problem J. Counting Is Fun

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

Let B be an array of length M consisting of positive integers.

In one operation, you may choose a non-empty subsequence[†] of indices

$$1 \le i_1 < i_2 < \cdots < i_k \le M$$

such that the values

$$B_{i_1}, B_{i_2}, \ldots, B_{i_k}$$

are pairwise distinct, and then decrease each chosen value by 1.

Define the **score** of B, denoted by score(B), as the minimum number of operations needed to make every element of B equal to 0.

You are given an array A of length N.

Find the sum of score over all the non-empty subsequences of A.

Since the value might be large, output it modulo 998 244 353.

[†] A sequence C is called a subsequence of a sequence D if it can be obtained from D by deleting several elements, possibly none or all of them, without changing the order of the remaining elements.

Input

The input is given in the following format:

T
N
$A_1\ A_2\ \cdots\ A_N$
$\vdots$

- All input values are integers.
- $1 \le T \le 10^5$
- $1 \le N \le 5000$
- $1 \le A_i \le 10^9$
- It is guaranteed that the sum of N^2 over all test cases does not exceed 5000^2.

Output

For each test case, output a single integer — the sum of score over all the non-empty subsequences of A, modulo 998 244 353.

Examples

standard input	standard output
3 1 5 3 1 1 3 4 2 2 2 2	5 16 47

Note

Test case 1: The only non-empty subsequence is [5], with score 5. Therefore, the answer is 5.

Test case 2: The non-empty subsequences of $[1, 1, 3]$ are:

- two subsequences equal to [1], each with score 1,
- [3], with score 3,
- $[1, 1]$, with score 2,
- two subsequences equal to $[1, 3]$, each with score 3,
- $[1, 1, 3]$, with score 3.

So, the total sum is 16.

Problem K. Prefix MEX Equation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

Suppose you have two arrays X and Y, both of length M. You can perform the following operation however many times you like:

- Choose an index i ($1 \le i \le M$), and swap X_i with Y_i.

The pair of arrays (X, Y) is said to be **good** if it is possible to make their prefix MEX arrays equal by performing several such swaps.

The MEX of a set of integers is the smallest non-negative integer that does not appear in the set.

The prefix MEX array of an array Z is an array P of the same length as Z such that

$$P_i = \text{MEX}(Z_1, Z_2, \ldots, Z_i)$$

for each $1 \le i \le |Z|$.

You are given two arrays A and B, both of length N.

Count the number of pairs (L, R) such that $1 \le L \le R \le N$ and the pair of subarrays

$$(A_L, A_{L+1}, \ldots, A_R) \quad \text{and} \quad (B_L, B_{L+1}, \ldots, B_R)$$

is good.

Input

The input is given in the following format:

T
N
$A_1\ A_2\ \cdots\ A_N$
$B_1\ B_2\ \cdots\ B_N$
$\vdots$

- All input values are integers.
- $1 \le T \le 10^5$
- $1 \le N \le 2 \times 10^5$
- $0 \le A_i \le N$
- $0 \le B_i \le N$
- It is guaranteed that the sum of N over all test cases does not exceed 2×10^5.

Output

For each test case, output a single integer — the number of pairs (L, R) such that $1 \le L \le R \le N$ and the pair of arrays $(A_L, \ldots, A_R)$ and $(B_L, \ldots, B_R)$ is good.

Examples

standard input	standard output
3 4 0 2 1 0 0 1 0 2 3 1 2 1 2 1 1 4 0 0 3 1 2 0 1 1	2 6 4

Note

Test case 1: The valid pairs are $(1, 1)$ and $(2, 2)$.

Test case 2: All pairs are valid.

Test case 3: The valid pairs are $(2, 2)$, $(3, 3)$, $(4, 4)$, and $(3, 4)$.

Problem L. Alice and Bob

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

Alice and Bob have a tree T consisting of $2N$ nodes numbered from 1 to $2N$.

Bob has an integer K, and wants to find N pairs $(X_1, Y_1), (X_2, Y_2), \ldots, (X_N, Y_N)$ such that each integer from 1 to $2N$ appears in exactly one pair and the sum of distances $\sum_{i=1}^{N} \text{dist}(X_i, Y_i) = K$, where $\text{dist}(X_i, Y_i)$ denotes the distance between nodes X_i and Y_i in the tree, i.e. the number of edges on the (unique) path between X_i and Y_i.

Alice doesn't want Bob to succeed. She can perform the following operation:

- Remove an edge (U, V) from T.
- Add a new edge (A, B) such that T remains a tree.

Note that it is allowed to choose the same edge to both delete and insert, which will just result in the original tree.

Help Alice find an edge (U, V) that should be deleted and an edge (A, B) that should be added. If there are multiple solutions, you may find any of them.

It can be proven that under the constraints of this problem, it is always possible to delete an edge and add an edge such that there do not exist N pairs $(X_1, Y_1), (X_2, Y_2), \ldots, (X_N, Y_N)$ where each integer from 1 to $2N$ appears in exactly one pair and the sum of distances equals K.

Input

The input is given in the following format:

T
$N\ K$
$U_1\ V_1$
$U_2\ V_2$
$\vdots$
$U_{2N-1}\ V_{2N-1}$
$\vdots$

- All input values are integers.
- $1 \le T \le 10^4$
- $2 \le N \le 2 \times 10^5$
- $1 \le K \le 10^{12}$
- $1 \le U_i, V_i \le 2N$
- It is guaranteed that the given edges form a tree.
- It is guaranteed that the sum of N over all test cases does not exceed 2×10^5.

Output

For each test case, output two lines:

The first line should contain two integers U and V — meaning that edge (U, V) is to be deleted.

The second line should contain two integers A and B — meaning that edge (A, B) is to be added.

If there are multiple solutions, print any of them.

Examples

standard input	standard output
2 2 20 1 2 2 3 3 4 2 4 1 2 2 3 3 4	1 2 1 2 1 2 1 3

Note

Test case 1: The initial tree already has no way for Bob to obtain a score of $K = 20$, so Alice can simply remove and insert any existing edge.

Test case 2: The initial tree has a way for Bob to obtain a score of $K = 4$, for example by choosing pairs $(1, 4)$ and $(2, 3)$. Alice deletes edge $(1, 2)$ and inserts $(1, 3)$. In this new tree, Bob cannot obtain a score of 4.

Problem M. Vertex Separation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 megabytes

Consider an undirected graph. A pair of vertices (U, V) are said to be **separated** if the following condition holds:

- There exist two simple paths, both from U to V, whose lengths differ by *at least* 2. Note that a simple path is a path where all vertices are distinct.

A vertex U is said to be **separable** if there exists at least one other vertex V such that the pair (U, V) is separated.

You are given three integers N, M, and K. Construct any **simple connected** undirected graph with N vertices and M edges, such that it has **exactly** K separable vertices. If no such graph exists, print -1.

Input

The input is given in the following format:

T
$N\ M\ K$
$\vdots$

- All input values are integers.
- $1 \le T \le 10^5$
- $2 \le N \le 2 \times 10^5$
- $N - 1 \le M \le \min\left(2 \times 10^5, \dfrac{N \cdot (N-1)}{2}\right)$
- $0 \le K \le N$
- It is guaranteed that the sum of N and the sum of M over all test cases each won't exceed 2×10^5.

Output

For each test case:

- If it is impossible to construct such a graph, output a single integer -1.
- Otherwise, output M lines. The i-th of these lines should contain two integers U_i and V_i ($1 \le U_i, V_i \le N$, $U_i \ne V_i$) — the endpoints of the i-th edge. The graph must be **simple** (i.e. no self-loops and no repeated edges), **connected**, and have exactly K separable vertices.

If there are multiple valid graphs, any of them will be accepted.

Examples

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

Note

Test case 1: There's only one possible graph with $N = 2$ and $M = 1$ (which is just a single edge), and it has 0 separable vertices.

Test case 2: We want 0 separable vertices, which the given graph attains.

Test case 3: All vertices are separable. For instance, vertex 1 is separable because of the pair $(1, 2)$, where the two paths $1 \to 2$ and $1 \to 4 \to 3 \to 2$ both exist and differ in length by 2.