The 2023 Guangdong Provincial Collegiate Programming Contest

Contest Session

May 14, 2023

Problem List

A	Programming Contest
B	Base Station Construction
C	Trading
D	New Houses
E	New but Nostalgic Problem
F	Traveling in Cells
G	Swapping Operation
H	Canvas
I	Path Planning
J	X Equals Y
K	Peg Solitaire
L	Classic Problem
M	Computational Geometry

This problem set should contain 13 (thirteen) problems on 21 (twenty-one) numbered pages. Please inform a runner immediately if something is missing from your problem set.

Prepared by SUA Programming Contest Problem Setter Team.
https://sua.ac/

Hosted by

Problem Set Prepared by

It's against the rules to open non-contest websites during the contest.
If you're interested (which is our pleasure),
please scan the QR code only after the contest.

Problem A. Programming Contest

Guangdong Province is one of the earliest province in China which holds its own provincial collegiate programming contest. Sun Yat-sen University hosted the first Guangdong Collegiate Programming Contest in year 2003. After that, other universities in Guangdong, such as South China Agricultural University, South China University of Technology and South China Normal University, also hosted the contest. The contest is held once a year except for year 2020 due to the epidemic. In year 2023, Shenzhen Technology University will host the twentieth Guangdong Collegiate Programming Contest. We are looking forward to seeing participants' outstanding performance!

The beautiful campus of Shenzhen Technology University.

In another world, a programming contest has been held once a year since year y_1, except for the n years $s_1, s_2, \cdots, s_n$ when it was not held due to special reasons.

Calculate the number of times the competition has been held up to year y_2 (inclusive).

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \le T \le 20$) indicating the number of test cases. For each test case:

The first line contains an integer y_1 ($1970 \le y_1 \le 9999$) indicating the first year when the contest was held.

The second line first contains an integer n ($0 \le n \le 100$) indicating the number of years the contest was not held. Then n integers $s_1, s_2, \cdots, s_n$ ($y_1 < s_i \le 9999$) follow, indicating the years when the contest was not held. These years are given in increasing order and have no duplicates.

The third line contains an integer y_2 ($y_1 \le y_2 \le 9999$). It's guaranteed that y_2 is not a year when the contest was not held.

Output

For each test case output one line containing one integer, indicating the number of times the competition has been held up to year y_2 (inclusive).

Example

standard input	standard output
4 2003 1 2020 2023 2003 1 2020 2003 2345 0 3456 3000 4 3001 3003 3004 3008 3007	20 1 1112 5

Note

For the first sample test case, as described in the problem description, the answer is 20.

For the second sample test case, because year 2003 is the 1-st year when the contest was held, the answer is 1.

For the third sample test case, because the contest was held every year, the answer is $3456 - 2345 + 1 = 1112$.

For the fourth sample test case, the first 5 years when the contest was held is 3000, 3002, 3005, 3006 and 3007. So the answer is 5.

Problem B. Base Station Construction

China Mobile Shenzhen Branch was registered in 1999. Four years later, Guangdong Collegiate Programming Contest was held for the first time. China Mobile Shenzhen Branch, along with Guangdong Collegiate Programming Contest, witnesses the prosperity and development of the computer industry in Guangdong.

The 5G base station of China Mobile.

During the construction of a communication line, it is critical to carefully choose the locations for base stations. The distance from west to east of a city is n kilometers. The engineers have investigated the cost to build a base station at $1, 2, \cdots, n$ kilometers from west to east, which are $a_1, a_2, \cdots, a_n$ respectively.

To ensure communication quality for the residents, the locations of base stations also need to meet m requirements. The i-th requirement can be represented as a pair of integers l_i and r_i ($1 \le l_i \le r_i \le n$), indicating that there must be at least 1 base station between l_i kilometers and r_i kilometers (both inclusive) from west to east.

As the chief engineer, you need to decide the number of base stations to build and their locations, and finally calculate the minimum total cost to satisfy all requirements.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains an integer n ($1 \le n \le 5 \times 10^5$) indicating the distance from west to east of the city.

The second line contains n integers $a_1, a_2, \cdots, a_n$ ($1 \le a_i \le 10^9$) where a_i indicates the cost to build a base station at i kilometers from west to east.

The third line contains an integer m ($1 \le m \le 5 \times 10^5$) indicating the number of requirements.

For the following m lines, the i-th line contains two integers l_i and r_i ($1 \le l_i \le r_i \le n$) indicating that there must be at least 1 base station between l_i kilometers and r_i kilometers (both inclusive) from west to east.

It's guaranteed that neither the sum of n nor the sum of m of all test cases will exceed 5×10^5.

Output

For each test case output one line containing one integer indicating the minimum total cost to satisfy all requirements.

Example

standard input	standard output
2 5 3 2 4 1 100 3 1 3 2 4 5 5 5 7 3 4 2 2 3 1 4 2 3 4 5	102 5

Note

For the first sample test case the optimal solution is to build base stations at 2 kilometers and 5 kilometers from west to east. The total cost is $2 + 100 = 102$.

For the second sample test case the optimal solution is to build base stations at 2 kilometers and 4 kilometers from west to east. The total cost is $3 + 2 = 5$.

Problem C. Trading

Twenty years ago, the northern section of Beijing Road Pedestrian Street in Guangzhou unearthed eleven layers of pavement from the Tang Dynasty to the Republic of China, and the southern section excavated the foundation of Gongbei Building with five layers from the Song Dynasty to the Ming and Qing Dynasties. This proves that Beijing Road has a long history as a commercial pedestrian street since the Song Dynasty. At the same time, the first Guangdong Province Collegiate Programming Contest was also held at Sun Yat-sen University in Guangzhou. Today, twenty years later, Beijing Road Pedestrian Street has become one of Guangzhou's most famous tourist attractions and shopping destinations, and the Guangdong Province Collegiate Programming Contest is also celebrating its twentieth birthday.

Beijing Road Pedestrian Street in Guangzhou.

There are n stores in the pedestrian street which buy and sell the same type of product. The buying and selling price of one such product in the i-th store are both a_i. To avoid over-trading, the pedestrian street has a regulation, that one can only trade b_i times in the i-th store (each buy or each sell both count as a trade) and can only trade one product each time.

You're going to earn money by buying and selling the products in the pedestrian street. If you have infinite amount of money at the beginning (that is to say, you can't be short of money when buying a product), what's the maximum total profit you can make? More precisely, *profit* means the total amount of money earned by selling the products, minus the total amount of money spent for buying the products.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains an integer n ($1 \le n \le 10^5$) indicating the number of stores.

For the following n lines, the i-th line contains two integers a_i and b_i ($1 \le a_i, b_i \le 10^6$) indicating the price and the maximum number of trades in the i-th store.

It's guaranteed that the sum of n of all test cases will not exceed 10^6.

Output

For each test case output one line containing one integer indicating the maximum total profit.

Example

standard input	standard output
2 4 10 2 30 7 20 4 50 1 2 1 100 1 1000	100 0

Note

For the first sample test case, the optimal strategy is to buy 2 products from the 1-st store, buy 4 products from the 3-rd store, sell 5 products to the 2-nd store and sell 1 product to the 4-th store. The total profit is $30 \times 5 + 50 \times 1 - 10 \times 2 - 20 \times 4 = 100$.

For the second sample test case, because all stores have the same price, there is no profit.

Problem D. New Houses

With the construction and development of Guangdong, more and more people choose to come to Guangdong to start a new life. In a recently built community, there will be n people moving into m houses which are arranged in a row. The houses are numbered from 1 to m (both inclusive). House u and v are neighboring houses, if and only if $|u - v| = 1$. We need to assign each person to a house so that no two people will move into the same house. If two people move into a pair of neighboring houses, they will become neighbors of each other.

Some people like to have neighbors while some don't. For the i-th person, if he has at least one neighbor, his happiness will be a_i; Otherwise if he does not have any neighbor, his happiness will be b_i.

As the planner of this community, you need to maximize the total happiness.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains two integers n and m ($1 \le n \le 5 \times 10^5$, $1 \le m \le 10^9$, $n \le m$) indicating the number of people and the number of houses.

For the following n lines, the i-th line contains two integers a_i and b_i ($1 \le a_i, b_i \le 10^9$) indicating the happiness of the i-th person with and without neighbors.

It's guaranteed that the sum of n of all test cases will not exceed 10^6.

Output

For each test case output one line containing one integer indicating the maximum total happiness.

Example

standard input	standard output
3 4 5 1 100 100 1 100 1 100 1 2 2 1 10 1 10 2 3 100 50 1 1000	400 2 1050

Note

For the first sample test case, the optimal strategy is to let person 1 move into house 1 and let person 2 to 4 move into house 3 to 5. Thus, person 1 have no neighbors while person 2 to 4 have neighbors. The answer is $100 + 100 + 100 + 100 = 400$. Of course, we can also let person 2 to 4 move into house 1 to 3 and let person 1 move into house 5. This will also give us 400 total happiness.

For the second sample test case, as there are only 2 houses, person 1 and 2 have to be neighbors. The answer is $1 + 1 = 2$.

For the third sample test case, the optimal strategy is to let person 1 move into house 1 and let person 2 move into house 3. Thus, both of them have no neighbors. The answer is $50 + 1000 = 1050$.

Problem E. New but Nostalgic Problem

Given n strings $w_1, w_2, \cdots, w_n$, please select k strings among them, so that the lexicographic order of string v is minimized, and output the optimal string v. String v satisfies the following constraint: v is the longest common prefix of two selected strings with different indices. Also, v is the lexicographically largest string among all strings satisfying the constraint.

More formally, let $\mathbb{S}$ be a set of size k, where all the elements in the set are integers between 1 and n (both inclusive) and there are no duplicated elements. Let $\text{lcp}(w_i, w_j)$ be the longest common prefix of string w_i and w_j, please find a set $\mathbb{S}$ to minimize the lexicographic order of the following string v and output the optimal string v.

$$v = \max_{i \in \mathbb{S}, j \in \mathbb{S}, i \neq j} \text{lcp}(w_i, w_j)$$

In the above expression, max is calculated by comparing the lexicographic order of strings.

Recall that:

- String p is a prefix of string s, if we can append some number of characters (including zero characters) at the end of p so that it changes to s. Specifically, empty string is a prefix of any string.
- The longest common prefix of string s and string t is the longest string p such that p is a prefix of both s and t. For example, the longest common prefix of "abcde" and "abcef" is "abc", while the longest common prefix of "abcde" and "bcdef" is an empty string.
- String s is lexicographically smaller than string t ($s \neq t$), if
  - s is a prefix of t, or
  - $s_{|p|+1} < t_{|p|+1}$, where p is the longest common prefix of s and t, $|p|$ is the length of p, s_i is the i-th character of string s, and t_i is the i-th character of string t.

  Specifically, empty string is the string with the smallest lexicographic order.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains two integers n and k ($2 \leq n \leq 10^6$, $2 \leq k \leq n$) indicating the total number of strings and the number of strings to be selected.

For the following n lines, the i-th line contains a string w_i ($1 \leq |w_i| \leq 10^6$) consisting of lower-cased English letters.

It's guaranteed that the total length of all strings of all test cases will not exceed 10^6.

Output

For each test case output one line containing one string indicating the answer. Specifically, if the answer is an empty string, print `EMPTY`.

Example

standard input	standard output
2 5 3 gdcpc gdcpcpcp suasua suas sususua 3 3 a b c	gdcpc EMPTY

Problem F. Traveling in Cells

There are n cells arranged in a row. The i-th cell has a color c_i and contains a ball with value v_i.

You're going to travel several times in the cells. For each travel, you'll be given an integer x and a set of colors $\mathbb{A} = \{a_1, a_2, \cdots, a_k\}$ where $c_x \in \mathbb{A}$. The travel starts from cell x. During the travel, if you're located in cell i you can next move to cell $(i-1)$ or $(i+1)$. Note that you can't move out of these n cells. Also at any time, the color of cell you're located in must belong to set $\mathbb{A}$.

When you're in cell i, you can choose to remove the ball in the cell and gain its value v_i. As there is only one ball in each cell, you can only remove the ball from each cell once.

Your task is to process q operations in order. Each operation is one of the following three types:

- $1\ p\ x$: Change c_p to x.
- $2\ p\ x$: Change v_p to x.
- $3\ x\ k\ a_1\ a_2\ \ldots\ a_k$: Given the starting cell x and the color set $\mathbb{A} = \{a_1, a_2, \cdots, a_k\}$ of a travel, imagine that you're going on this travel, calculate the maximum total value you can gain. Note that this travel is only an imagination, thus the balls won't be truely removed. That is, all queries are independent.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains two integers n and q ($1 \le n \le 10^5$, $1 \le q \le 10^5$) indicating the number of cells and the number of operations.

The second line contains n integers $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le n$) where c_i is the initial color of the i-th cell.

The third line contains n integers $v_1, v_2, \ldots, v_n$ ($1 \le v_i \le 10^9$) where v_i is the initial value of ball in the i-th cell.

For the following q lines, the i-th line describes the i-th operation. The input format is listed as follows:

- $1\ p\ x$: $1 \le p \le n$ and $1 \le x \le n$.
- $2\ p\ x$: $1 \le p \le n$ and $1 \le x \le 10^9$.
- $3\ x\ k\ a_1\ a_2\ \ldots\ a_k$: $1 \le x \le n$, $1 \le a_1 < a_2 < \ldots < a_k \le n$ and $c_x \in \{a_1, a_2, \cdots, a_k\}$.

It's guaranteed that neither the sum of n nor the sum of q of all test cases will exceed 3×10^5. Also the sum of k of all test cases will not exceed 10^6.

Output

For each operation of type 3 output one line containing one integer, indicating the maximum total value you can gain.

Example

standard input	standard output
2 5 10 1 2 3 1 2 1 10 100 1000 10000 3 3 1 3 3 3 2 2 3 2 5 20000 2 3 200 3 3 2 1 3 3 3 3 1 2 3 1 3 4 2 1 100000 1 2 2 3 1 2 1 2 4 1 1 2 3 4 1000000 1000000 1000000 1000000 3 4 4 1 2 3 4	100 110 1200 21211 100010 4000000

Problem G. Swapping Operation

Given a non-negative integer sequence $A = a_1, a_2, \dots, a_n$ of length n, define

$$F(A) = \max_{1 \le k < n} ((a_1 \,\&\, a_2 \,\&\, \cdots \,\&\, a_k) + (a_{k+1} \,\&\, a_{k+2} \,\&\, \cdots \,\&\, a_n))$$

where & is the bitwise-and operator.

You can perform the swapping operation at most once: choose two indices i and j such that $1 \le i < j \le n$ and then swap the values of a_i and a_j.

Calculate the maximum possible value of $F(A)$ after performing at most one swapping operation.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains an integer n ($2 \le n \le 10^5$) indicating the length of sequence A.

The second line contains n integers $a_1, a_2, \cdots, a_n$ ($0 \le a_i \le 10^9$) indicating the given sequence A.

It's guaranteed that the sum of n of all test cases will not exceed 10^5.

Output

For each test case output one line containing one integer indicating the maximum possible value of $F(A)$ after performing at most one swapping operation.

Example

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

Note

For the first sample test case, we can swap a_4 and a_6 so the sequence becomes $\{6, 5, 4, 6, 5, 3\}$. We can then choose $k = 5$ so that $F(A) = (6 \,\&\, 5 \,\&\, 4 \,\&\, 6 \,\&\, 5) + (3) = 7$.

For the second sample test case, we can swap a_2 and a_4 so the sequence becomes $\{1, 1, 1, 2, 2, 2\}$. We can then choose $k = 3$ so that $F(A) = (1 \,\&\, 1 \,\&\, 1) + (2 \,\&\, 2 \,\&\, 2) = 3$.

For the third sample test case we do not perform the swapping operation. We can then choose $k = 2$ so that $F(A) = (1 \,\&\, 1) + (2 \,\&\, 2 \,\&\, 2) = 3$.

Problem H. Canvas

There is a sequence of length n. At the beginning, all elements in the sequence equal to 0. There are also m operations, where the i-th operation will change the value of the l_i-th element in the sequence to x_i, and also change the value of the r_i-th element in the sequence to y_i. Each operation must be performed exactly once.

Find the optimal order to perform the operations, so that after all operations, the sum of all elements in the sequence is maximized.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains two integers n and m ($2 \le n, m \le 5 \times 10^5$) indicating the length of the sequence and the number of operations.

For the following m lines, the i-th line contains four integers l_i, x_i, r_i and y_i ($1 \le l_i < r_i \le n$, $1 \le x_i, y_i \le 2$) indicating the i-th operation.

It's guaranteed that neither the sum of n nor the sum of m of all test cases will exceed 5×10^5.

Output

For each test case, first output one line containing one integer, indicating the maximum sum of all elements in the sequence after all operations. Then output another line containing m integers $a_1, a_2, \cdots, a_m$ separated by a space, indicating the optimal order to perform the operations, where a_i is the index of the i-th operation to be performed. Each integer from 1 to m (both inclusive) must appear exactly once. If there are multiple valid answers, you can output any of them.

Example

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

Note

For the first sample test case, after performing operations $4, 1, 3, 2$ in order, the sequence becomes $\{2, 2, 2, 1\}$. The sum of all elements is 7.

For the second sample test case, after performing operations $2, 1$ in order, the sequence becomes $\{2, 0, 2, 1\}$. The sum of all elements is 5.

Problem I. Path Planning

There is a grid with n rows and m columns. Each cell of the grid has an integer in it, where $a_{i,j}$ indicates the integer in the cell located at the i-th row and the j-th column. Each integer from 0 to $(n \times m - 1)$ (both inclusive) appears exactly once in the grid.

Let (i, j) be the cell located at the i-th row and the j-th column. You now start from $(1, 1)$ and need to reach (n, m). When you are in cell (i, j), you can either move to its right cell $(i, j + 1)$ if $j < m$ or move to its bottom cell $(i + 1, j)$ if $i < n$.

Let $\mathbb{S}$ be the set consisting of integers in each cell on your path, including $a_{1,1}$ and $a_{n,m}$. Let $\text{mex}(\mathbb{S})$ be the smallest non-negative integer which does not belong to $\mathbb{S}$. Find a path to maximize $\text{mex}(\mathbb{S})$ and calculate this maximum possible value.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains two integers n and m ($1 \le n, m \le 10^6$, $1 \le n \times m \le 10^6$) indicating the number of rows and columns of the grid.

For the following n lines, the i-th line contains m integers $a_{i,1}, a_{i,2}, \cdots, a_{i,m}$ ($0 \le a_{i,j} < n \times m$) where $a_{i,j}$ indicates the integer in cell (i, j). Each integer from 0 to $(n \times m - 1)$ (both inclusive) appears exactly once in the grid.

It's guaranteed that the sum of $n \times m$ of all test cases will not exceed 10^6.

Output

For each test case output one line containing one integer indicating the maximum possible value of $\text{mex}(\mathbb{S})$.

Example

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

Note

For the first sample test case there are 3 possible paths.

- The first path is $(1, 1) \to (1, 2) \to (1, 3) \to (2, 3)$. $\mathbb{S} = \{1, 2, 4, 5\}$ so $\text{mex}(\mathbb{S}) = 0$.
- The second path is $(1, 1) \to (1, 2) \to (2, 2) \to (2, 3)$. $\mathbb{S} = \{1, 2, 0, 5\}$ so $\text{mex}(\mathbb{S}) = 3$.
- The third path is $(1, 1) \to (2, 1) \to (2, 2) \to (2, 3)$. $\mathbb{S} = \{1, 3, 0, 5\}$ so $\text{mex}(\mathbb{S}) = 2$.

So the answer is 3.

For the second sample test case there is only 1 possible path, which is $(1, 1) \to (1, 2) \to (1, 3) \to (1, 4) \to (1, 5)$. $\mathbb{S} = \{1, 3, 0, 4, 2\}$ so $\text{mex}(\mathbb{S}) = 5$.

Problem J. X Equals Y

For positive integers X and $b \geq 2$, define $f(X, b)$ as a sequence which describes the base-b representation of X, where the i-th element in the sequence is the i-th least significant digit in the base-b representation of X. For example, $f(6, 2) = \{0, 1, 1\}$, while $f(233, 17) = \{12, 13\}$.

Given four positive integers x, y, A and B, please find two positive integers a and b satisfying:

- $2 \leq a \leq A$
- $2 \leq b \leq B$
- $f(x, a) = f(y, b)$

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \leq T \leq 10^3$) indicating the number of test cases. For each test case:

The first line contains four integers x, y, A and B ($1 \leq x, y \leq 10^9$, $2 \leq A, B \leq 10^9$).

It's guaranteed that there are at most 50 test cases satisfying $\max(x, y) > 10^6$.

Output

For each test case, if valid positive integers a and b do not exist, output `NO` in one line.

Otherwise, first output `YES` in one line. Then in the next line, output two integers a and b separated by a space. If there are multiple valid answers, you can output any of them.

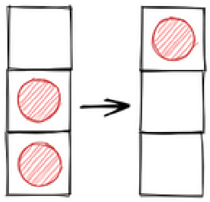
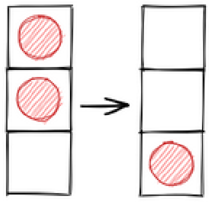
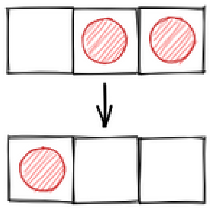
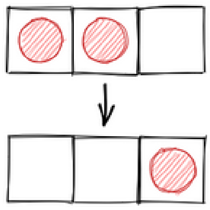
Example

standard input	standard output
6 1 1 1000 1000 1 2 1000 1000 3 11 1000 1000 157 291 5 6 157 291 3 6 10126 114514 789 12345	YES 2 2 NO YES 2 10 YES 4 5 NO YES 779 9478

Problem K. Peg Solitaire

Peg Solitaire is a single-player boardgame on a chessboard with n rows and m columns. Each cell of the chessboard either is empty, or contains a chesspiece. Initially, there are k chesspieces on the chessboard.

During the game, the player can choose a chesspiece, jump it over an adjacent chesspiece into an empty cell, and finally remove the chesspiece which is jumped over. More precisely, let (i, j) be the cell on the i-th row and the j-th column, the player can perform operations of the following four types.

Operation	Description	Figure
Jump Up	Choose (i, j) which satisfies all of the following. • $i \geq 3$. • Both (i, j) and $(i-1, j)$ contain a chesspiece. • $(i-2, j)$ is empty. Jump the chesspiece in (i, j) to $(i-2, j)$, and remove the chesspiece in $(i-1, j)$.	
Jump Down	Choose (i, j) which satisfies all of the following. • $i \leq n-2$. • Both (i, j) and $(i+1, j)$ contain a chesspiece. • $(i+2, j)$ is empty. Jump the chesspiece in (i, j) to $(i+2, j)$, and remove the chesspiece in $(i+1, j)$.	
Jump Left	Choose (i, j) which satisfies all of the following. • $j \geq 3$. • Both (i, j) and $(i, j-1)$ contain a chesspiece. • $(i, j-2)$ is empty. Jump the chesspiece in (i, j) to $(i, j-2)$, and remove the chesspiece in $(i, j-1)$.	
Jump Right	Choose (i, j) which satisfies all of the following. • $j \leq m-2$。 • Both (i, j) and $(i, j+1)$ contain a chesspiece. • $(i, j+2)$ is empty. Jump the chesspiece in (i, j) to $(i, j+2)$, and remove the chesspiece in $(i, j+1)$.	

Given the initial state of the chessboard, the player can perform the operations any number of times (including zero times). Calculate the minimum possible number of chesspieces remaining on the chessboard.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \leq T \leq 20$) indicating the number of test cases. For each test case:

The first line contains three integers n, m and k ($1 \leq n, m \leq 6$, $1 \leq k \leq \min(6, n \times m)$) indicating the number of rows and columns of the chessboard and the initial number of chesspieces.

For the following k lines, the i-th line contains two integers x_i and y_i ($1 \leq x_i \leq n$, $1 \leq y_i \leq m$) indicating that there is a chesspiece in the cell on the x_i-th row and the y_i-th column at the beginning. Except from these k cells, all other cells are empty at the beginning. The positions of these k cells contain no duplicate.

Output

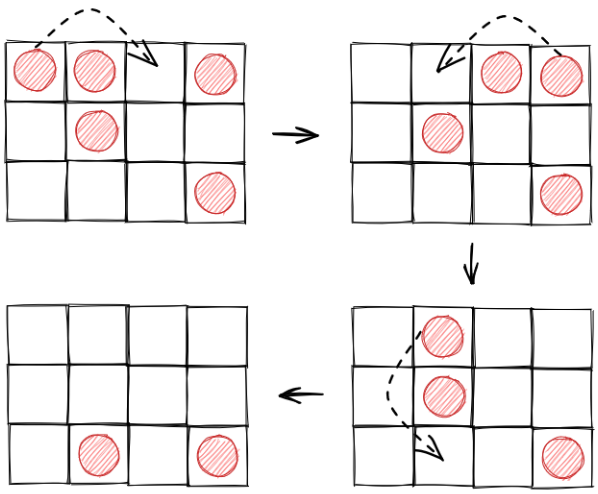
For each test case output one line containing one integer indicating the minimum possible number of chesspieces remaining on the chessboard.

Example

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

Note

The first sample test case is explained as follows.



For the second sample test case, as the chessboard does not contain empty cell at the beginning, the player cannot perform any operation.

For the third sample test case, as the chessboard has less than three cells, the player cannot perform any operation.

Problem L. Classic Problem

Given an undirected complete graph with n vertices and m triples $P_1, P_2, \cdots, P_m$ where $P_i = (u_i, v_i, w_i)$, it's guaranteed that $1 \le u_i < v_i \le n$, and for any two triples P_i and P_j with different indices we have $(u_i, v_i) \ne (u_j, v_j)$.

For any two vertices x and y in the graph ($1 \le x < y \le n$), define the weight of the edge connecting them as follows:

- If there exists a triple P_i satisfying $u_i = x$ and $v_i = y$, the weight of edge will be w_i.
- Otherwise, the weight of edge will be $|x - y|$.

Calculate the total weight of edges in the minimum spanning tree of the graph.

Input

There are multiple test cases. The first line of the input contains an integer T ($1 \le T \le 10^5$) indicating the number of test cases. For each test case:

The first line contains two integers n and m ($1 \le n \le 10^9$, $0 \le m \le 10^5$) indicating the number of vertices in the graph and the number of triples.

For the following m lines, the i-th line contains three integers u_i, v_i and w_i ($1 \le u_i < v_i \le n$, $0 \le w_i \le 10^9$) indicating the i-th triple. It's guaranteed that for all $1 \le i < j \le m$ we have $(u_i, v_i) \ne (u_j, v_j)$.

It's guaranteed that the sum of m of all test cases will not exceed 5×10^5.

Output

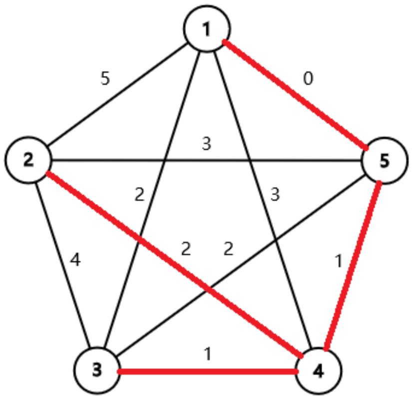
For each test case output one line containing one integer indicating the total weight of edges in the minimum spanning tree of the graph.

Example

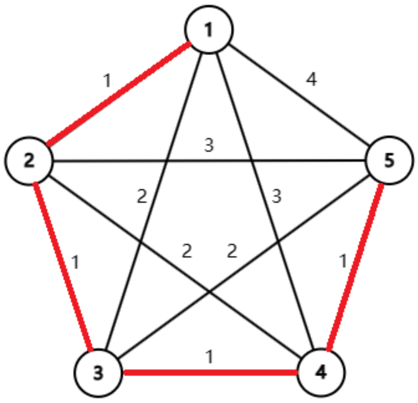
standard input	standard output
3 5 3 1 2 5 2 3 4 1 5 0 5 0 5 4 1 2 1000000000 1 3 1000000000 1 4 1000000000 1 5 1000000000	4 4 1000000003

Note

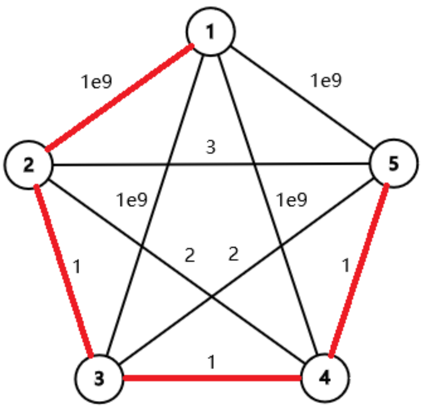
The first sample test case is illustrated as follows. The minimum spanning tree is marked by red segments.



The second sample test case is illustrated as follows. The minimum spanning tree is marked by red segments.



The third sample test case is illustrated as follows. The minimum spanning tree is marked by red segments.



Problem M. Computational Geometry

Given a convex polygon P with n vertices, you need to choose two vertices of P, so that the line connecting the two vertices will split P into two smaller polygons Q and R, both with positive area.

Let $d(Q)$ be the diameter of polygon Q and $d(R)$ be the diameter of polygon R, calculate the minimum value of $(d(Q))^2 + (d(R))^2$.

Recall that the diameter of a polygon is the maximum distance between two points inside or on the border of the polygon.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains an integer n ($4 \le n \le 5 \times 10^3$) indicating the number of vertices of the convex polygon P.

For the following n lines, the i-th line contains two integers x_i and y_i ($0 \le x_i, y_i \le 10^9$) indicating the i-th vertex of the convex polygon P. Vertices are given in counter-clockwise order. It's guaranteed that the area of the convex polygon is positive, and there are no two vertices with the same coordinate. It's possible that three vertices lie on the same line.

It's guaranteed that the sum of n of all test cases will not exceed 5×10^3.

Output

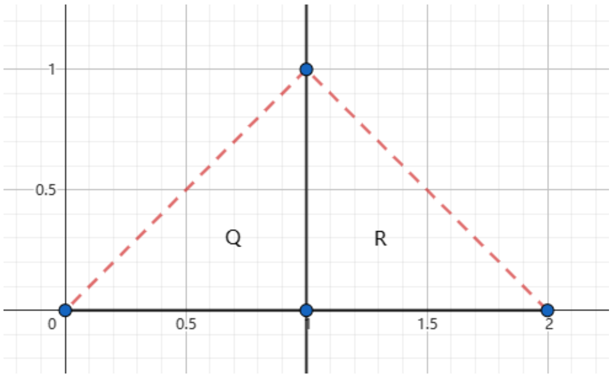
For each test case output one line containing one integer indicating the answer.

Example

standard input	standard output
2 4 1 0 2 0 1 1 0 0 6 10 4 9 7 5 7 4 5 6 4 9 3	4 44

Note

The first sample test case is shown as follows. The diameter of smaller polygons are marked by red dashed segments. In fact, $(1, 0)$ and $(1, 1)$ are the only pair of vertices we can choose in this test case. You can't choose $(0, 0)$ and $(2, 0)$, or $(0, 0)$ and $(1, 1)$, because they can't split P into two smaller polygons both with positive area.



The second sample test case is shown as follows. The diameter of smaller polygons are marked by red dashed segments.