The 35th Annual
ACM International Collegiate Programming Contest
ASIA Regional - Daejeon

Problem A
Sales

Mr. Cooper, the CEO of CozyWalk Co., receives a report of daily sales every day since the company has been established. Starting from the second day since its establishment, on receiving the report, he compares it with each of the previous reports in order to calculate the number of previous days whose sales amounts are less than or equal to it. After obtaining the number of such days, he writes it in a list.

This problem can be stated more formally as follows. Let $A=(a_1, a_2, ..., a_n)$ denote the list of daily sales amounts. And let $B=(b_1, b_2, ..., b_{n-1})$ be another integer list maintained by Mr. Cooper, each value representing the number of such previous days. On the i-th day $(2 \le i \le n)$, he calculates b_{i-1}, the number of a_k s such that $a_k \le a_i$ $(1 \le k < i)$.

For example, suppose that $A=(20, 43, 57, 43, 20)$. For the fourth day's sales amount, $a_4 = 43$, the number of previous days whose sales amounts are less than or equal to it is 2 since $a_1 \le a_4$, $a_2 \le a_4$, and $a_3 > a_4$. Therefore, $b_3 = 2$. Similarly, b_1, b_2, and b_4 can be obtained and it results in $B=(1, 2, 2, 1)$.

Given an array of size n for the list of daily sales amounts, write a program that prints the sum of the $n-1$ integers in the list B.

Input
Your program is to read the input from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case starts with a line containing an integer n $(2 \le n \le 1{,}000)$, which represents the size of the list A. In the following line, n integers are given, each represents the daily sales amounts a_i $(1 \le a_i \le 5{,}000$ and $1 \le i \le n)$ for the test case.

Output
Your program is to write to standard output. For each test case, print the sum of the $n-1$ integers in the list B which is obtained from the list A.

The following shows sample input and output for two test cases.

Sample Input	Output for the Sample Input
2 5 38 111 102 111 177 8 276 284 103 439 452 276 452 398	9 20

The 35th Annual
ACM International Collegiate
Programming Contest
ASIA Regional - Daejeon

Problem B
String Popping

We are given a string *s* of two characters ‘a’ and ‘b’. Let a *group* be a maximal consecutive substring of the same character. Any group *g* of *s* of length at least two can be removed (or *popped*) and a new string is constructed by concatenating the remaining left and right substrings of *s*. We repeat this process until either the string becomes the empty string or there is no more group of length at least two.

For example, string *s* = *babbbaaabb* has 5 groups *b*, *a*, *bbb*, *aaa*, and *bb*. The string *s* can be turned into the empty string by popping groups in the following sequence (the underlined group is to be popped in the sequence):

$$ba\underline{bbb}aaabb \to b\underline{aaaa}bb \to \underline{bbb} \to \text{empty string}$$

But the group may not turn to an empty string by a different sequence of pop operations:

$$babbbaaa\underline{bb} \to ba\underline{bbb}aaa \to b\underline{aaaa} \to b$$

Given a string, write a program to decide whether the string can be turned into the empty string by some sequence of popping operations.

Input

Your program is to read from standard input. The input consists of *T* test cases. The number of test cases *T* is given in the first line of the input. Each test case consists of a single line containing a string of two characters ‘a’ and ‘b’. The minimum and maximum length of the string is 1 and 25, respectively.

Output

Your program is to write to standard output. Print one line for each test case. The line of each test case should contain 0 or 1. Print 1 if the input string can be turned to an empty string by a sequence of popping operations. Otherwise print 0.

The following shows sample input and output for three test cases.

Sample Input	Output for the Sample Input
3 babbbaaabb aabbaabb abab	1 1 0

Problem C
Password

Shoulder-surfing is the behavior of intentionally and stealthily watching the screen of another person's electronic device, such as laptop computer or mobile phone. Since mobile devices prevail, it is getting serious to steal personal information by shoulder-surfing.

Suppose that we have a smart phone. If we touch the screen keyboard directly to enter the password, this is very vulnerable since a shoulder-surfer easily knows what we have typed. So it is desirable to conceal the input information to discourage shoulder-surfers around us. Let me explain one way to do this.

You are given a 6×5 grid. Each column can be considered the visible part of a wheel. So you can easily rotate each column wheel independently to make password characters visible. In this problem, we assume that each wheel contains the 26 upper letters of English alphabet. See the following Figure 1.

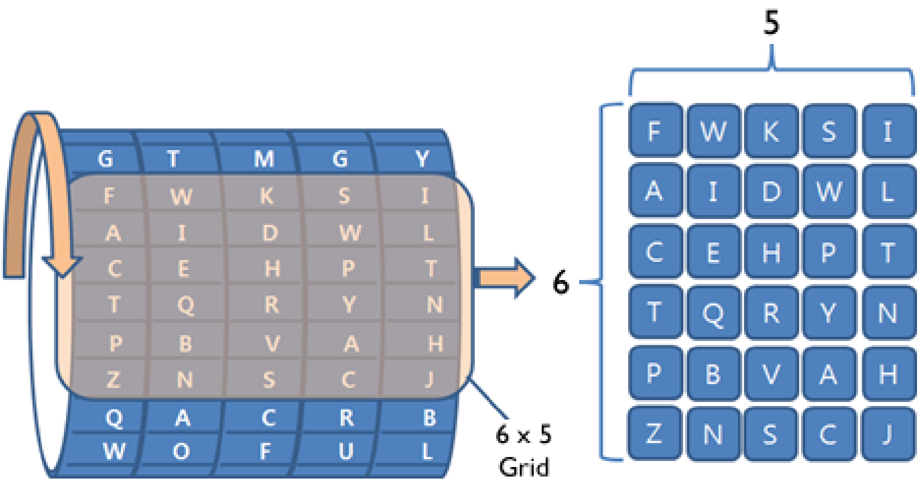


Figure 1. 6×5 window clips a valid grid representation for a password.

Assume that we have a length-5 password such as $p_1\ p_2\ p_3\ p_4\ p_5$. In order to pass the authentication procedure, we should construct a configuration of grid space where each p_i appears in the i-th column of the grid. In that situation we say that the user password is accepted.

Let me start with one example. Suppose that our password was set 'COMPU'. If we construct the grid as shown in Figure 2 on next page, then the authentication is successfully processed.

A	*Y*	*G*	*S*	*U*
D	*O*	*M*	*R*	*A*
C	*P*	*F*	*A*	*S*
X	*B*	*O*	*D*	*G*
W	*D*	*Y*	*P*	*K*
R	*B*	*C*	*D*	*E*

Figure 2. A valid grid representation for password 'COMPU'.

In this password system, the position of each password character in each column is meaningless. If each of the 5 characters in $p_1\ p_2\ p_3\ p_4\ p_5$ appears in the corresponding column, that can be considered the correct password. So there are many grid configurations allowing one password. Note that the sequence of letters on each wheel is randomly determined for each trial and for each column. In practice, the user is able to rotate each column and press "Enter" key, so a should-surfer cannot perceive the password by observing the 6×5 grid since there are too many password candidates. In this 6×5 grid space, maximally 6^5 =7,776 cases are possible. This is the basic idea of the proposed password system against shoulder-surfers.

Unfortunately there is a problem. If a shoulder-surfer can observe more than two grid plate configurations for a person, then the shoulder-surfer can reduce the searching space and guess the correct password. Even though it is not easy to stealthily observe other's more than once, this is one weakness of implicit grid passwords.

Let me show one example with two observed configurations for a grid password. The user password is 'COMPU', but 'DPMAG' is also one candidate password derived from the following configuration.

A	*Y*	*G*	*S*	*U*
D	*O*	*M*	*R*	*A*
C	*P*	*F*	*A*	*S*
X	*B*	*O*	*D*	*G*
W	*D*	*Y*	*P*	*K*
P	*R*	*X*	*W*	*O*

C	*B*	*O*	*P*	*T*
D	*O*	*S*	*B*	*G*
G	*T*	*R*	*A*	*R*
A	*P*	*M*	*M*	*S*
W	*S*	*X*	*N*	*U*
E	*F*	*G*	*H*	*I*

Figure 3. Both of 'COMPU' and 'DPMAG' are feasible password .

You are given two configurations of grid password from a shoulder-surfer. Suppose that you have succeeded to stealthily record snapshots of the target person's device (e.g. smart phone). Then your next task is to reconstruct all possible passwords from these two snapshots. Since there are lots of password candidates, you are asked for the *k*-th password among all candidates in lexicographical order. In Figure 3, let us show the first 5 valid password. The first 5 valid passwords are 'ABGAG' , 'ABGAS', 'ABGAU', 'ABGPG' and 'ABGPS'.

The number *k* is given in each test case differently. If there does not exist a *k*-th password since *k* is larger than the number of all possible passwords, then you should print 'NO' in the output.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. The first line of each test case contains one integer, K, the order of the password you should find. Note that $\mathbf{1 \le K \le 7,777}$. Next the following 6 lines show the 6 rows of the first grid and another 6 lines represent the 6 rows of the second grid.

Output

Your program is to write to standard output. Print exactly the k-th password (including 'NO') in one line for each test case.

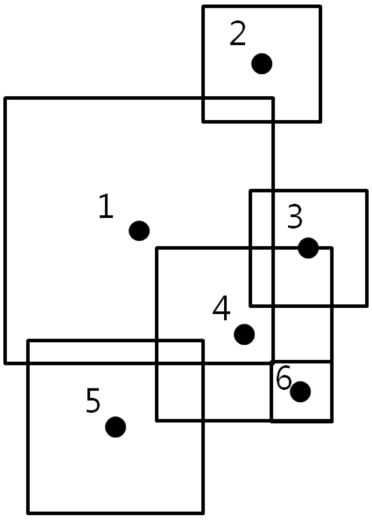
The following shows sample input and output for three test cases.

Sample Input	Output for the Sample Input
3 1 AYGSU DOMRA CPFAS XBODG WDYPK PRXWO CBOPT DOSBG GTRAR APMMS WSXNU EFGHI 5 AYGSU DOMRA CPFAS XBODG WDYPK PRXWO CBOPT DOSBG GTRAR APMMS WSXNU EFGHI 64 FGHIJ EFGHI DEFGH CDEFG BCDEF ABCDE WBXDY UWYXZ XXZFG YYFYH EZWZI ZGHIJ	ABGAG ABGPS NO

Problem D
Mines

There are N mines in an old battlefield. Each mine affects an axis-parallel square area depending on its performance. Assume that the location of the mine is the center of the square area. When a mine explodes, all mines in the square area of the explosion will explode as well. As a chain reaction, all the mines in the square area of the following explosion will also explode. Assume that when a mine is exploding, a mine on the edge of the exploding square area will also explode.

In the following figure, if mine 4 initiates, mines 3 and 6 will explode. If mine 1 initiates, mine 4 will explode. The following explosion will result mines 3 and 6 to explode. Therefore, initiating mines 1, 2, and 5 will cause all the mines to explode.



Given N mines with their explosion performance as square areas in a two-dimensional plane, write a program that determines the minimum number of mines that needs to be initiated to explode all given mines.

Input

Your program is to read the input from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case starts with a line containing an integer N ($3 \le N \le 2{,}000$), which represents the number of mines. In the following N lines, each line contains three integers x, y and d, where x and y are the coordinates of the mine in the plane and d is the size of one side of the square which representing the explosion performance ($1 \le x, y \le 10{,}000{,}000$, $1 \le d \le 1{,}000{,}000$).

Output

Your program is to write to standard output. Print exactly one line for each test case. Print the minimum number of mines that needs to be initiated to explode all given mines.

The following shows sample input and output for two test cases.

Sample Input	Output for the Sample Input
2 6 6 11 10 10 17 4 12 10 4 10 7 6 5 4 6 12 5 2 4 6 7 8 9 10 4 11 5 4 15 9 8	3 2

Problem E
Binary Search Tree

A binary search tree is a binary tree. It may be empty. If it is not empty, it satisfies the following properties:

(1) Every node has a key, and no two nodes have the same key.
(2) The keys in a nonempty left subtree must be smaller than the key in the root of the subtree.
(3) The keys in a nonempty right subtree must be larger than the key in the root of the subtree.
(4) The left and right subtrees are also binary search trees.

Sample binary search trees are shown in Figure 1.

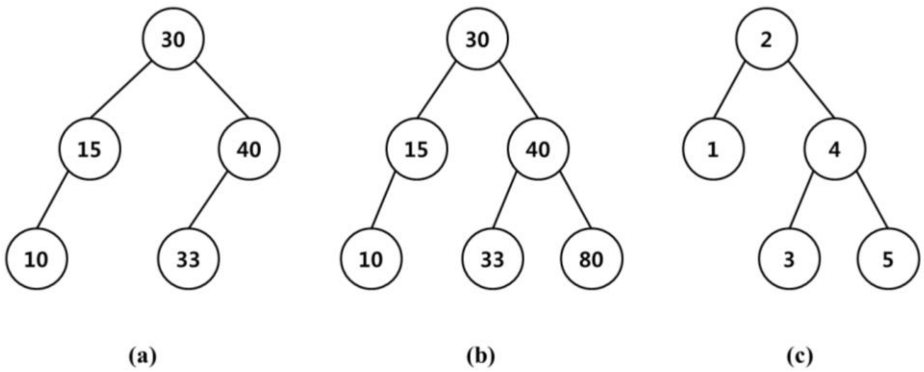


Figure 1. binary search trees

To search for a node with a key k in a binary search tree T, we begin at the root. If T is empty, T contains no keys and the search is unsuccessful. Otherwise, we compare k with the key in root. If k equals root's key, then the search terminates successfully. If k is less than root's key, we search the left subtree of the root. If k is larger than root's key, we search the right subtree of the root. In the same way, we can proceed the search in the left or right subree of T.

To insert a new key k into a binary search tree T where k is different from those of existing keys in T, we first search the tree T. The search will be unsuccessful, then we insert the key at the point the search terminated. For instance, to insert a key 80 into the Figure 1(a), we first search the tree for 80. This search terminates unsuccessfully, and the last node examined has key 40. We insert a new node containing 80 as the right child of the node. The resulting search tree is shown in Figure 1(b).

In this problem, we consider binary search trees with N keys 1, 2, ... , N. For a permutation $a_1\, a_2 \dots a_N$ of {1, 2, ... , N }, inserting $a_1\, a_2 \dots a_N$ successively into an initially empty binary search tree will produce a binary search tree. For instance, the permutation 2 1 4 3 5 will produce the tree in Figure 1(c). Also, 2 4 3 1 5 will produce the same tree. Actually, 8 permutations among all possible permutations of {1, 2, 3, 4, 5} will produce the same tree to the tree in Figure 1(c).

We are interested in finding the number of permutations of {1, 2, … , N } such that all those permutations produce a binary search tree identical to the tree produced by a given permutation P. Given N and P, you are to write a program that calculates the number of permutations satisfying the above condition.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line. Each test case starts with a line containing an integer N representing the number of keys, $1 \le N \le 20$. In the next line, a permutation of length N is given. There is a single space between the integers representing keys in the permutation.

Output

Your program is to write to standard output. Print exactly one line for each test case as follows: Let B be the number of permutations that produce the binary search tree identical to the tree produced by the input permutation. Print B mod 9,999,991 for each test case. For example, if $B =$ 20,000,000 , the output should be 18 for that test case.

The following shows sample input and output for three test cases.

Sample Input	Output for the Sample Input
3 5 2 1 4 3 5 4 2 4 1 3 12 1 2 3 4 5 6 7 8 9 10 11 12	8 3 1

The 35th Annual
ACM International Collegiate
Programming Contest
ASIA Regional - Daejeon

Problem F
Tour Belt

Korea has many tourist attractions. One of them is an archipelago (Dadohae in Korean), a cluster of small islands scattered in the southern and western coasts of Korea. The Korea Tourism Organization (KTO) plans to promote a new tour program on these islands. For this, The KTO wants to designate two or more islands of the archipelago as a tour belt.

There are n islands in the archipelago. The KTO focuses on the synergy effect created when some islands are selected for a tour belt. Synergy effects are assigned to several pairs of islands. A synergy effect $\mathrm{SE}(u,v)$ or $\mathrm{SE}(v,u)$ between two islands u and v is a positive integer which represents a value anticipated when both u and v are included in a tour belt. The KTO wants to select two or more islands for the tour belt so that the economic benefit of the tour belt is as high as possible.

To be precise, we define a connected graph $G=(V,E)$, where V is a set of n vertices and E is a set of m edges. Each vertex of V represents an island in the archipelago, and an edge (u,v) of E exists if a synergy effect $\mathrm{SE}(u,v)$ is defined between two distinct vertices (islands) u and v of V. Let A be a subset consisting of at least two vertices in V. An edge (u,v) is an *inside edge* of A if both u and v are in A. An edge (u,v) is a *border edge* of A if one of u and v is in A and the other is not in A.

A vertex set B of a connected subgraph of G with $2 \leq |B| \leq n$ is called a *candidate* for the tour belt if the synergy effect of every inside edge of B is larger than the synergy effect of any border edge of B. A candiate will be chosen as the final tour belt by the KTO. There can be many possible candidates in G. Note that V itself is a candidate because there are no border edges. The graph in Figure 1(a) has three candidates $\{1,2\},\{3,4\}$ and $\{1,2,3,4\}$, but $\{2,3,4\}$ is not a candidate because there are inside edges whose synergy effects are not larger than those of some border edges. The graph in Figure 1(b) contains six candidates, $\{1,2\}$, $\{3,4\}$, $\{5,6\}$, $\{7,8\}$, $\{3,4,5,6\}$ and $\{1,2,3,4,5,6,7,8\}$. But $\{1,2,7,8\}$ is not a candidate because it does not form a connected subgraph of G, i.e., there are no edges connecting $\{1,2\}$ and $\{7,8\}$.

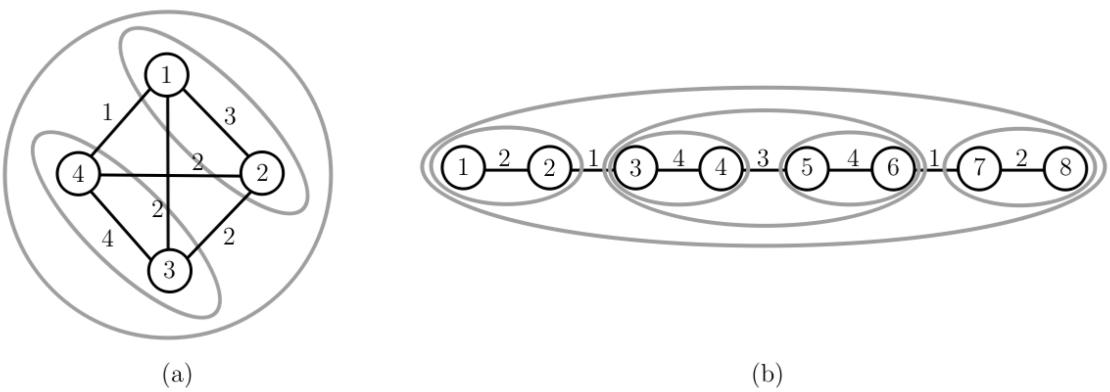


Figure 1. Graphs and their good subsets marked by gray ellipses.

The KTO will decide one candidate in G as the final tour belt. For this, the KTO asks you to find all candidates in G. You write a program to print the sum of the sizes of all candidates in a given graph G. For example, the graph in Figure 1(a) contains three candidates and the sum of their sizes is $2 + 2 + 4 = 8$, and the graph in Figure 1(b) contains six candidates and the sum of their sizes is $2 + 2 + 2 + 2 + 4 + 8 = 20$.

Input

Your program is to read input from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case starts with a line containing two integers n $(2 \le n \le 5{,}000)$ and m $\left(1 \le m \le \frac{n(n-1)}{2}\right)$, where n represents the number of vertices (islands) and m represents the number of edges of a connected graph G. Islands are numbered from 1 to n. In the following m lines, the synergy effects assigned to m edges are given; each line contains three integers, u, v, and k $(1 \le u \neq v \le n$, $1 \le k \le 10^5)$, where k is the synergy effect between two distinct islands u and v, i.e., $\mathrm{SE}(u, v) = \mathrm{SE}(v, u) = k$.

Output

Your program is to write to standard output. Print exactly one line for each test case. Print the sum of the sizes of all candidates for a test case.

The following shows sample input and output for two test cases.

Sample Input	Output for the Sample Input
2 4 6 1 2 3 2 3 2 4 3 4 1 4 1 2 4 2 1 3 2 8 7 1 2 2 2 3 1 3 4 4 4 5 3 5 6 4 6 7 1 7 8 2	8 20

Problem G
String Phone

Believe it or not, the streets of ACM city are laid out in an exact grid pattern, as shown in Figure 1. Each crossing in the streets is indentified with its vertical and horizontal street number. Some buildings in this city are so important that they need to be under surveillance by the police at all time. Since each building occupies an entire block of the streets, we can simply think of it as a cell of the grid. For an example, the black cells in Figure 1 represent important buildings. The building numbered 1 is located at a block surrounded by four crossings (3,7), (3,8), (4,7) and (4,8).

The APD (ACM Police Department) decided to assign one police officer to each of the important buildings and let them keep watch of the buildings. The police officer must reside at one of the four crossings surrounding the building he is in charge of. For an example, the officer in charge of building 1 in Figure 1 has to reside at one of the four crossings (3,7), (3,8), (4,7), and (4,8).

The officers should be able to communicate with each other. Since the chief of police was a huge fan of toys, he got rid of all walkie-talkies in the department, and instead forced the officers to use “string phones”. As you may know, a string phone is made of two paper cups (or, sometimes, two steel cans) linked by a string. For a string phone to work properly, the string should be kept tight. Since strings cannot pass though the buildings, they always run along the streets. Therefore, for two officers located at crossings (x_1,y_1) and (x_2,y_2), respectively, to be able to communicate, the length of the string phone they share must be equal to $|x_1-x_2|+|y_1-y_2|$, which is called *the shortest distance along the streets* between the two crossings.

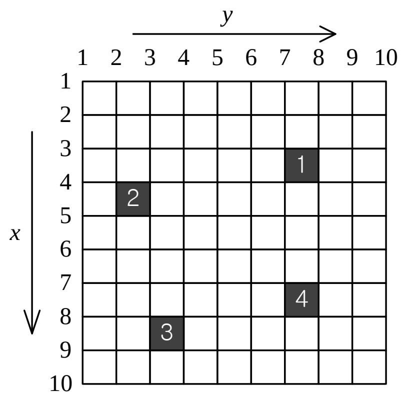


Figure 1. Streets of ACM city and locations of important buildings

Now, you have to help the officers to find their locations so that every string phone they have works properly. You are given the locations of n important buildings which are numbered from 1 through n. The buildings are apart from each other enough for no two buildings to have a crossing surrounding them in common. You are also given information about which couples of the officers share string phones and what the lengths of the strings are. This information is given as a form of weighted connected graph as depicted in Figure 2(a). Node i, $i=1,2,\ldots,n$, of the graph represents the officer who is in charge of building i. The existence of an edge (i,j) implies that two officers in charge of buildings i and j, respectively, share a string phone, and the weight of the edge represents the length of the string phone. This graph is always a connected graph. Your goal is to determine if it is possible to locate n officers so that every string phone works properly. Figure 2(b) shows an example of such locations for our example. In Figure 2(b), the small circles denote the crossings at which the officers are located. You can easily verify that the shortest distances along the streets between them are all met by the lengths of the string phones they have.

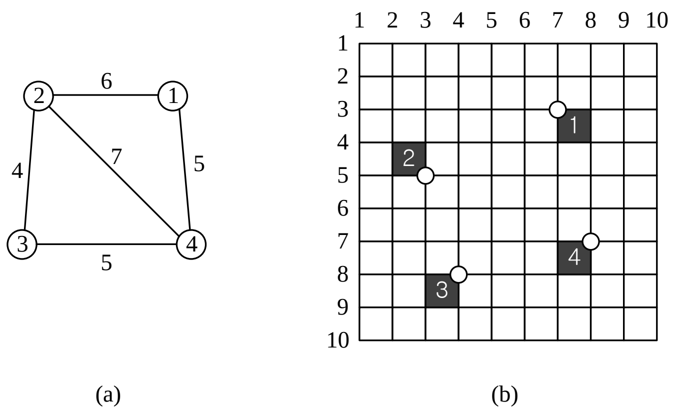


Figure 2. Lengths of string phones and proper locations of officers

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. The first line of each test case contains one integer n, which is the number of important buildings, where $1 \le n \le 3{,}000$. In the following n lines, each line contains two integers x, and y which mean, assuming that it is the i-th line among those n lines, that building i is located so that it is surrounded by the four street crossings (x,y), $(x+1,y)$, $(x,y+1)$, and $(x+1,y+1)$, where $1 \le x, y \le 3{,}000{,}000$. The next line contains another integer m which is the number of edges of the graph, where $1 \le m \le 300{,}000$. In the following m lines, each line contains three integers u, v, and d which represent that there is an edge between vertex u and v of which the weight is d, where $1 \le d \le 3{,}000{,}000$.

Output

Your program is to write to standard output. Print exactly one line for each test case. The output for each test case should be either `possible` or `impossible` depending on whether it is possible to place all officers satisfying given conditions or not.

The following shows sample input and output for two test cases.

Sample Input	Output for the Sample Input
2 4 3 7 4 2 8 3 7 7 5 1 2 6 1 4 5 2 3 4 2 4 7 3 4 5 3 1 1 1 4 4 1 3 1 2 7 2 3 6 3 1 4	possible impossible

The 35th Annual

ACM International Collegiate

Programming Contest

ASIA Regional - Daejeon

Problem H
Installations

In the morning, service engineers in a telecom company receive a list of jobs which they must serve today. They install telephones, internet, ipTVs, etc and repair troubles with established facilities. A client requires a deadline when the requested job must be completed. But the engineers may not complete some jobs within their deadlines because of job overload. For each job, we consider, as a penalty of the engineer, the difference between the deadline and the completion time. It measures how long the job proceeds after its deadline. The problem is to find a schedule minimizing the sum of the penalties of the jobs with the two largest penalties.

A service engineer gets a list of jobs J_i with a serving time s_i and a deadline d_i. A job J_i needs time s_i, and if it is completed at time C_i, then the penalty of J_i is defined to be $\max\{0, C_i - d_i\}$. For convenience, we assume that the time t when a job can be served is $0 \le t < \infty$ and s_i and d_i are given positive integers such that $0 < s_i \le d_i$. The goal is to find a schedule of jobs minimizing the sum of the penalties of the jobs with the two largest penalties.

For example, there are six jobs J_i with the pair (s_i, d_i) of the serving time s_i and the deadline d_i, $i = 1,\ldots,6$, where (s_1, d_1) = (1, 7), (s_2, d_2) = (4, 7), (s_3, d_3) = (2, 4), (s_4, d_4) = (2, 15), (s_5, d_5) = (3, 5), (s_6, d_6) = (3, 8). Then Figure 1 represents a schedule which minimizes the sum of the penalties of the jobs with the two largest penalties. The sum of the two largest penalties of an optimal schedule is that of the penalties of J_2 and J_6, namely 6 and 1, respectively, which is equal to 7 in this example.

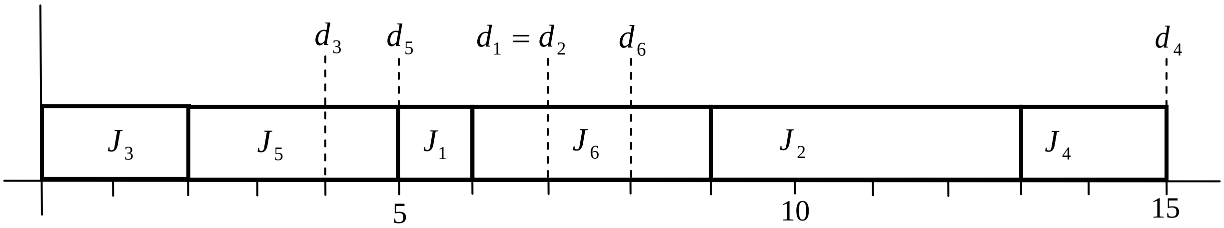


Figure 1. The optimal schedule of the example

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given on the first line of the input. The first line of each test case contains an integer n $(1 \le n \le 500)$, the number of the given jobs. In the next n lines of each test case, the i-th line contains two integer numbers s_i and d_i, representing the serving time and the deadline of the job J_i, respectively, where $1 \le s_i \le d_i \le 10{,}000$.

Output

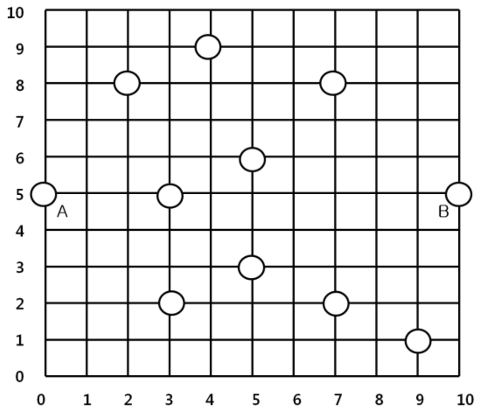
Your program is to write to standard output. Print exactly one line for each test case. The line contains the sum of the penalties of the jobs with the two largest penalties.

The following shows sample input and ouput for three test cases.

Sample Input	Output for the Sample Input
3 6 1 7 4 7 2 4 2 15 3 5 3 8 7 2 17 2 11 3 4 3 20 1 20 4 7 5 14 10 2 5 2 9 5 10 3 11 3 4 4 21 1 7 2 9 2 11 2 23	7 0 14

Problem I
Restaurant

Mr. Kim is planning to open a new restaurant. His city is laid out as a grid with size $M \times M$. Therefore, every road is horizontal or vertical and the horizontal roads (resp., the vertical roads) are numbered from 0 to M-1. For profitability, all restaurants are located near road junctions. The city has two big apartments which are located on the same horizontal road. The figure below shows an example of a city map with size 11×11. A circle represents an existing restaurant and a circle labeled with '*A*' or '*B*' represents the location of an apartment. Notice that a restaurant is already located at each apartment. Each road junction is represented by the coordinate of the ordered pair of a vertical road and a horizontal road. The distance between two locations (x_1, y_1) and (x_2, y_2) is computed as $|x_1\text{-}x_2| + |y_1\text{-}y_2|$. In the figure below, the coordinates of *A* and *B* are (0, 5) and (10, 5), respectively.



Mr. Kim knows that the residents of the two apartments frequently have a meeting. So, he thinks that the best location of a new restaurant is halfway between two apartments. Considering lease expenses and existing restaurants, however, he can't select the optimal location unconditionally. Hence he decides to regard a location satisfying the following condition as a *good place*. Let *dist*(*p, q*) be the distance between *p* and *q*.

> A location *p* is a *good place* if for each existing restaurant's location *q*, *dist*(*p*, *A*) < *dist*(*q*, *A*) or *dist*(*p*, *B*) < *dist*(*q*, *B*). In other words, *p* is not a good place if there exists an existing restaurant's location *q* such that *dist*(*p*, *A*) ≥ *dist*(*q*, *A*) and *dist*(*p*, *B*) ≥ *dist*(*q*, *B*).

In the above figure, the location (7, 4) is a good place. But the location *p* = (4, 6) is not good because there is no apartment which is closer to *p* than the restaurant at *q* = (3, 5), i.e., *dist*(*p*, *A*) = 5 ≥ *dist*(*q*, *A*) = 3 and *dist*(*p*, *B*) = 7 ≥ *dist*(*q*, *B*) = 7. Also, the location (0, 0) is not good due to the restaurant at (0, 5). Notice that the existing restaurants are positioned regardless of Mr. Kim's condition.

Given *n* locations of existing restaurants, write a program to compute the number of good places for a new restaurant.

Input

Your program is to read the input from standard input. The input consists of *T* test cases. The number of test cases *T* is given in the first line of the input. Each test case starts with a line containing two integers *M* and *n*

($2 \leq M \leq 60{,}000$ and $2 \leq n \leq 50{,}000$), which represent the size of a city map and the number of existing restaurants, respectively. The (i+1)-th line of a test case contains two integers x_i and y_i (i =1, 2, ... , n and $0 \leq x_i, y_i < M$), which represents the coordinate of the i-th existing restaurant. Assume that all restaurants have distinct coordinates and that the two apartments A and B are positioned at the locations of 1-st restaurant and 2-nd restaurant. Notice that A and B are placed on the same horizontal line.

Output

Your program is to write to standard output. Print exactly one line for each test case. Print the number of good places which can be found in a given city map.

The following shows sample input and output for two test cases.

Sample Input	Output for the Sample Input
2 6 3 1 3 4 3 0 2 11 11 0 5 10 5 4 9 2 8 7 8 5 6 3 5 5 3 3 2 7 2 9 1	2 16

Problem J
KTX Train Depot

KTX (Korea Train eXpress), Korea’s high-speed rail system operated by KORAIL, connects the capital Seoul to several major cities of the Korean Peninsula. KORAIL’s most recent train sets KTX-II, which were developed in Korea, can travel at over 350 km/h. The next-generation test train is currently being developed, and the train is planned to be able to travel at over 400 km/h by 2012. KTX Goyang Train Depot, which is located behind Haengsin Station, is one of the biggest train depots in Korea. The train depot consists of a set of parallel tracks for storing KTX trains and facilities for carrying out repair jobs.

All trains whose home is KTX Goyang Train Depot enter the depot until the middle of the night and leave it in the morning. Each train might enter the depot either from the eastern side or from the western side and might leave the depot to the eastern side or to the western side. The direction from which it enters or to which it leaves the depot, however, is fixed. Also the arrival and departure times are fixed. Hence, the train is labeled with the time interval $[t_1, t_2]$ in which it stays in the depot and an arrival and departure direction d_1 and d_2, respectively. For example, a train labeled [-6E,13W] is a train that enters the depot at time -6 from the eastern side and leaves the depot at time 13 to the western side. In the meanwhile, it stops on one of several parallel tracks in the depot.

No two trains arrive from the same direction at the same time or depart to the same direction at the same time. There might be several trains waiting at the same track. Each train is assigned to a track such that it can leave the depot on time without being blocked by other trains. That is, the assignment of Figure 1 is not feasible, since train [-2E,2W] would be blocked by train [-1W,4W]. The assignments in Figures 2(a) and 2(b) are both feasible. However, an interesting assignment is the one in Figure 2(b), which uses the minimum number of tracks.

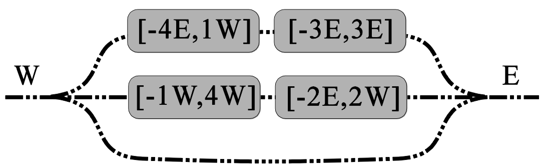
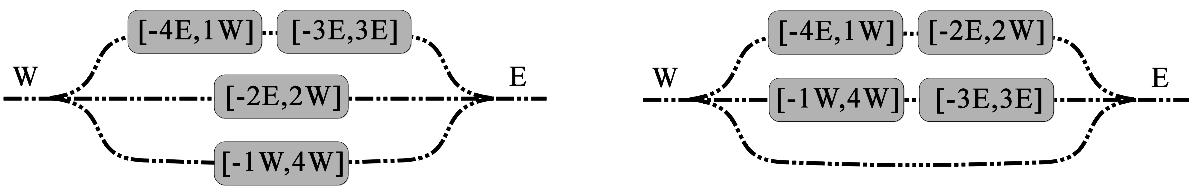


Figure 1. An infeasible track assignment



(a) (b)

Figure 2. Feasible track assignments

It is planned to increase the number of KTX trains whose home is KTX Goyang Train Depot. Dr. Chwa, who is the President of KORAIL Research Institute, foresees that it might cause a difficulty in storing trains in the depot due to the limit on the number of tracks. The length of a track will be no problem since each track is long enough to store even all the trains. Dr. Chwa is eager to determine the minimum number of tracks to store all the trains such that each train can leave the depot on time without being blocked by other trains. Write a program that can help him.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. The first line of each test case contains an integer n which represents the number of trains, where $1 \le n \le 10{,}000$. In the following n lines, each line contains a train label $t_1 d_1 t_2 d_2$, which represents a train that enters the depot at time t_1 from the direction d_1 and leaves the depot at time t_2 to the direction d_2, where t_1 and t_2 are integers such that $-1{,}000{,}000 \le t_1 < 0 < t_2 \le 1{,}000{,}000$ and $d_1, d_2 \in \{E, W\}$. Every line of the input contains no whitespaces except the newline character.

Output

Your program is to write to standard output. Print exactly one line for each test case that contains an integer representing the minimum number of tracks that Dr. Chwa is eager to know.

The following shows sample input and output for three test cases.

Sample Input	Output for the Sample Input
3 4 -4E1W -3E3E -1W4W -2E2W 3 -3E9W -2E8W -1E7W 3 -3E9E -2W8W -1W7W	2 3 1