The 39th Annual
ACM International Collegiate Programming Contest
Asia Regional - Daejeon

Problem A
Algorist Club
Time Limit: 0.1 Second

The pride of the Algorist Club is that its members really know one another well. In May of every year when it is time for new members to come in, they have a chicken and beer party where the new members are acquainted with the current members and also with one another. The rule is that, if two members $\mathcal{A}$ and $\mathcal{B}$ do not know each other then they have to have a talk session for 15 minutes to really get to know each other. If a member $\mathcal{A}$ does not know members $\mathcal{B}$ and C then $\mathcal{A}$ must have two separate sessions.

Of course the party has to start as soon as possible and the leaders of Algorist Club have decided to plan the get-to-know-each-other sessions. The time for the sessions will be divided into 15-minute slots. An unlimited number of sessions can go on simultaneously in one slot. However, it is clear that one person can be in only one session in a slot. Let's say that $\mathcal{A}$ is the one who does not know the largest number of people in the Club and k is the number of people that $\mathcal{A}$ does not know. It is obvious that at least k slots are needed. But we really do not know if exactly k slots will be enough. So the leaders have decided to allow $k + 1$ slots for the sessions.

Given the pairs of people who do not know each other, write a program to find a schedule for the sessions which has $k + 1$ slots, where k is defined as above.

Input
Your program is to read from standard input. The input consists of $\boldsymbol{T}$ test cases. The number of test cases $\boldsymbol{T}$ is given in the first line of the input. Each test case starts with a line containing two integers $\boldsymbol{n}$ and $\boldsymbol{m}$, where $\boldsymbol{n}$ is the number of people in the club and $\boldsymbol{m}$ is the number of pairs of people who do not know each other ($\boldsymbol{2 \le n \le 400}$, $\boldsymbol{1 \le m \le (n(n-1))/2}$). The people are numbered from $\boldsymbol{1}$ to $\boldsymbol{n}$. In each of the next $\boldsymbol{m}$ lines, a pair of integers $\boldsymbol{a}$ and $\boldsymbol{b}$ is given, indicating a pair of people who do not know each other. The pairs are given in lexicographical order. That is, they are given in such way that $\boldsymbol{a < b}$ is always true, pairs with smaller $\boldsymbol{a}$ are given earlier, and within the pairs with the same value for $\boldsymbol{a}$, those with smaller $\boldsymbol{b}$'s are given earlier.

Output
Your program is to write to standard output. For each test case, you should print the pairs given in the input *exactly as they were given*, with the slot number added as the third integer for each pair. That is, for each of the pair given in the input, you should print the pair and the slot that the pair is assigned to. The slots are numbered from 1 to $k + 1$. If it is not possible to finish all the sessions using $k + 1$ slots, print a `0` for the slots of *all* pairs.

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

Sample Input	Output for the Sample Input
2 3 3 1 2 1 3 2 3 4 5 1 2	1 2 1 1 3 2 2 3 3 1 2 1 1 4 2 2 3 3 2 4 4

1 4 2 3 2 4 3 4	3 4 1

The 39th Annual
ACM International Collegiate Programming Contest
Asia Regional - Daejeon

Problem B
Deduction
Time Limit: 1 Second

Mr. Hong, a private detective, has been summarizing information about the murder cases that he is currently responsible for. Collecting the conduct information of the persons associated with the incident, he was able to construct n statements $S_1, S_2, \ldots, S_n$. Mr. Hong does not know whether or not each statement is true. Therefore, he called each statement an incident variable. The following six statements show an example of incident variables.

S_1 : Cheolsu Kim has a dog.
S_2 : Sucheol Park likes cats.
S_3 : Heeyoung Ahn likes Sucheol Park.
S_4 : (*omitted*)
S_5 : (*omitted*)
S_6 : (*omitted*)

Investigating the truth of each incident variable and the relationship between variables, he has constructed his deduction based on the result. His deduction consists of one or more assertions which belong to one of the following three types. The first type is that an incident variable S_i is true, the second type is that a set with one or more incident variables has at least one which is false. The last type is that an incident variable S_i proves to be true if one or more variables associated with S_i are all true. These types are represented as follows.

Type 1: S_i	(S_i is true.)
Type 2: $S_{i_1}, S_{i_2}, \cdots, S_{i_k} \rightarrow \emptyset$	(At least one among $S_{i_1}, S_{i_2}, \cdots$, and S_{i_k} is false.)
Type 3: $S_{j_1}, S_{j_2}, \cdots, S_{j_k} \rightarrow S_i$	(If $S_{j_1}, S_{j_2}, \cdots$, and S_{j_k} are all true, then S_i is also true.).

For example, let's assume that Mr.Hong's deduction consists of the following eight assertions:

① S_1
② S_2
③ $S_1, S_2, S_6 \rightarrow \emptyset$
④ $S_1 \rightarrow S_6$
⑤ $S_2 \rightarrow S_6$
⑥ $S_1, S_6 \rightarrow S_3$
⑦ $S_2, S_4 \rightarrow S_1$
⑧ $S_5, S_6 \rightarrow S_2$

The assertions ① and ② belong to Type 1, the assertion ③ belongs to Type 2, and the assertions ④, ⑤, ⑥, ⑦, ⑧ belong to Type 3. In order to prove that his deduction is valid, he should show that there aren't any contradictions between the assertions of the deduction and that there are no assertions that belong to Type 3 such that the variable on the right side of '→' must be false even though all variables on the left side are all

true. (Notice that if there exists a variable on the left side of '→' which is false, the truth of the variable on the right side does not infringe the validity of the deduction.)

A valid assignment means a truth assignment to the incident variables that makes a deduction valid. Mr. Hong wants to know if there is a valid assignment that makes his deduction valid.

In the above example, since both of S_1 and S_2 are true by assertions ① and ②, S_6 must be false by assertion ③. However this leads to a contradiction because S_6 should be true by assertion ⑤. Hence, Mr. Hong's deduction is not valid. If the assertion ③ is excluded from his deduction, there exists a valid assignment, which is that all variables are assigned as true or that only S_4 is assigned as false.

Given Mr. Hong's deduction with n incident variables, write a program that determines whether or not there is a valid assignment.

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 starts with a line containing four integers n, m_1, m_2, m_3 ($1 \le n \le 1{,}500$, $1 \le m_1 < n$, $0 \le m_2, m_3 \le 1{,}500$), where n is the number of incident variables and m_1, m_2, m_3 are the numbers of Type 1, Type 2, and Type 3 assertions of Mr. Hong's deduction, respectively. Each of the following m_1 lines contains an integer i ($1 \le i \le n$) representing a Type 1 assertion, i.e., an incident variable S_i. Each of the following m_2 lines contains $k+1$ integers $k, i_1, i_2, \dots,$ and i_k ($1 \le k, i_1, i_2, \dots, i_k \le n$, $i_r \ne i_s$ for $r \ne s$) representing a Type 2 assertion "$S_{i_1}, S_{i_2}, \cdots, S_{i_k} \rightarrow \emptyset$". Each of the following m_3 lines contains $k+2$ integers $k, j_1, j_2, \dots, j_k,$ and i ($1 \le k \le n-1$, $1 \le j_1, j_2, \dots, j_k, i \le n$, $j_r \ne j_s$ for $r \ne s$, $i \ne j_r$ for $1 \le r \le k$) representing a Type 3 assertion "$S_{j_1}, S_{j_2}, \cdots, S_{j_k} \rightarrow S_i$".

Output

Your program is to write to standard output. Print exactly one line for each test case. The line should contain "YES" if the deduction is valid, otherwise, "NO".

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

Sample Input	Output for the Sample Input
2 6 2 1 5 1 2 3 1 2 6 1 1 6 1 2 6 2 1 6 3 2 2 4 1 2 5 6 2 6 2 0 5 1 2 1 1 6 1 2 6 2 1 6 3 2 2 4 1 2 5 6 2	NO YES

The 39th Annual

ACM International Collegiate Programming Contest
Asia Regional - Daejeon

Problem C
Eureka Theorem
Time Limit: 0.1 Second

A triangle number $T_n (n \geq 1)$ is a figurate number that can be represented by a regular geometric arrangement of equally spaced points as illustrated in Figure 1.

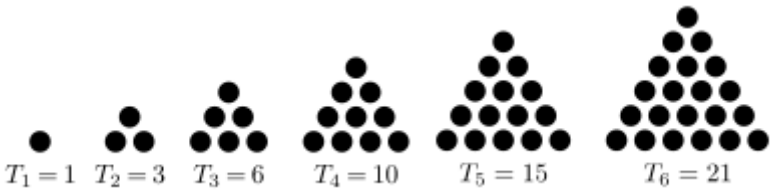


Figure 1.

The triangle number T_n for any positive integer $n \geq 1$ is given by the explicit formula:

$$T_n = 1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}.$$

In 1796, Gauss proved that every positive integer can be represented as a sum of *at most* three triangle numbers. For examples,

$$4 = T_1 + T_2$$
$$5 = T_1 + T_1 + T_2$$
$$6 = T_2 + T_2 \text{ or } 6 = T_3$$
$$10 = T_1 + T_2 + T_3 \text{ or } 10 = T_4$$

This result is known as the Eureka theorem since he wrote in his diary "Eureka! num = Δ + Δ + Δ" for commemorating the proof. We wonder if some positive integer can be represented as a sum of *exactly* three triangle numbers. As shown in the above examples, integers 5 and 10 can be represented as a sum of exactly three triangle numbers, but integers 4 and 6 cannot.

Given a positive integer, write a program to test whether or not the integer can be represented as a sum of exactly three triangle numbers that may not be distinct.

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 line containing a positive integer $\boldsymbol{K}$ ($\mathbf{3} \leq \boldsymbol{K} \leq \mathbf{1,000}$).

Output
Your program is to write to standard output. Print exactly one line for each test case. Print `1` if the input number *K* can be represented as a sum of exactly three triangle numbers, and print `0` (zero), otherwise.

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

Sample Input	Output for the Sample Input
3 10 20 1000	1 0 1

The 39th Annual
ACM International Collegiate Programming Contest
Asia Regional - Daejeon

Problem D
Exploration
Time Limit: 0.5 Second

Global warming is caused by air pollution in the sky and too much carbon dioxide in the atmosphere. Because of this, the North Pole ice plates are melting and the sea level is rising. As a result, a lot of cities will be under the sea. Also, there will be abnormal climate all over the world. To explore the exact status of the North Pole, ICPC (International Climate Protection Committee) would like to organize a team for the exploration. ICPC made a list of applicants for the exploration and acquired friendship information between the applicants. In order to select a team of closer cooperation, ICPC made the following rule:

Eligibility: For each member of the team, at least k friends of the member should be in the team.

Among eligible teams, ICPC will select a team of maximum size. For example, suppose there are 5 applicants and their friendship relations are represented by a graph in Figure 1. In the graph each vertex represents an applicant, and an edge between two vertices represents two applicants are friends. If $k = 2$, any of $\{2, 3, 4\}$, $\{2, 4, 5\}$, and $\{2, 3, 4, 5\}$ satisfies the ICPC rule. Among the teams, the maximum size is 4. Therefore, ICPC will select $\{2, 3, 4, 5\}$ as an exploration team. If $k = 3$, there is no team satisfying the rule.

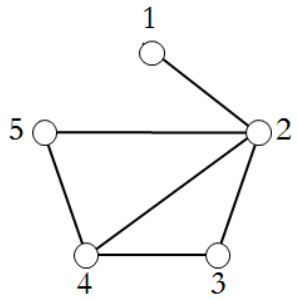


Figure 1.

Given the friendship relations between n applicants and an integer k, you are to write a program to find the maximum size of the exploration team satisfying the ICPC rule.

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 starts with a line containing three integers, $\boldsymbol{n}, \boldsymbol{k}$ and $\boldsymbol{f}$ ($\mathbf{1} \leq \boldsymbol{k} < \boldsymbol{n} \leq \mathbf{2,000}$, $\mathbf{1} \leq \boldsymbol{f} \leq \boldsymbol{n}(\boldsymbol{n}-\mathbf{1})/\mathbf{2}$), where $\boldsymbol{n}$ is the number of applicants, $\boldsymbol{k}$ is the number specified in the ICPC rule, and $\boldsymbol{f}$ is the number of friendship relations. Each of the next $\boldsymbol{f}$ lines contains two integers representing two friends in the applicants. There is a single space between two integers in the same line.

Output
Your program is to write to standard output. Print exactly one line for each test case. The line should contain the maximum size of the exploration team. If there is no team satisfying the ICPC rule, your program should print `0`.

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

Sample Input	Output for the Sample Input
2 5 2 6 1 2 3 2 3 4 4 5 5 2 2 4 5 3 6 1 2 3 2 3 4 4 5 5 2 2 4	4 0

The 39th Annual
ACM International Collegiate Programming Contest
Asia Regional - Daejeon

Problem E
Marbles
Time Limit: 0.3 Second

Three children A, B, and C play marbles. There are marbles with colors red, blue, and green. At the beginning of the game, the marbles are thrown on the ground, and then they are dispersed.

Each of A, B, and C draws a rectangular area by his hand on the ground. The rectangular areas are closed regions containing their boundaries and all of them should be disjoint with each other. And the sides of the rectangles are axis parallel.

Then the child A counts only the number of red marbles contained in his/her area, B does this for the number of blue marbles, and C does it for the number of green marbles. The number of marbles a child counts in his/her area may be zero. The goal of the game is to maximize the total number of the counted marbles.

Figure 1 shows an example of marbles thrown on the ground. If three rectangular areas are drawn as in Figure 2, then there are 2 red marbles on the A's area, 2 blue marbles on the B's area, and 3 green marbles on the C's area. The total number of the counted marbles is 7 and this is the maximum that can be achieved.

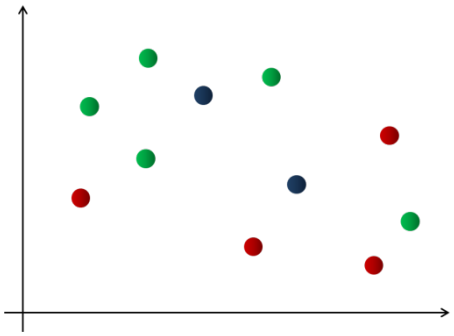


Figure 1.

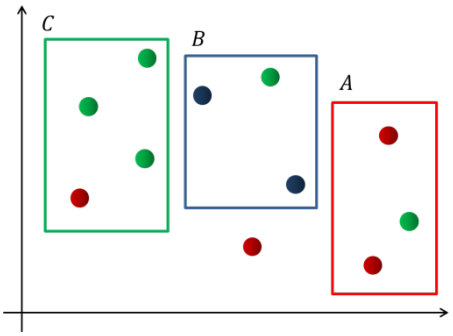


Figure 2.

Your program should compute the maximum total number of marbles countable by the children.

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 starts with a line containing three integers, a, b, and c ($1 \leq a, b, c \leq 10{,}000$), where a, b, and c are the numbers of red, blue and green marbles, respectively. Each of the following a lines contains two integers x_i and y_i, representing the coordinate (x_i, y_i) of a red marble ($1 \leq i \leq a$). Each of the following b lines contains two integers x_i and y_i, representing the coordinate (x_i, y_i) of a blue marble ($a + 1 \leq i \leq a + b$). And each of the following c lines also contains two integers x_i and y_i, representing the coordinate (x_i, y_i) of a green marble ($a + b + 1 \leq i \leq a + b + c$). Here, $0 \leq x_i, y_i \leq 10{,}000{,}000$. Note that the locations of marbles satisfy the condition that at most one marble can be located on any horizontal or vertical line in the plane.

Output

Your program is to write to standard output. Print exactly one line for each test case. The line should contain the maximum total number of the counted marbles.

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

Sample Input	Output for the Sample Input
2 2 2 1 1 1 4 4 2 2 5 5 3 3 3 4 2 10 2 15 9 14 10 12 1 4 5 2 15 6 12 5 8 3 3	3 7

The 39th Annual

ACM International Collegiate Programming Contest
Asia Regional - Daejeon

Problem F
Permutation Cycles
Time Limit: 0.1 Second

There are several ways to represent a permutation of the n integers from 1 to n. For example, when the permutation of 8 integers is (3,2,7,8,1,4,5,6), one way to represent it as an array is $\begin{pmatrix}1&2&3&4&5&6&7&8\\3&2&7&8&1&4&5&6\end{pmatrix}$. Another way to represent it in cycle-arrow form is shown in Figure 1.

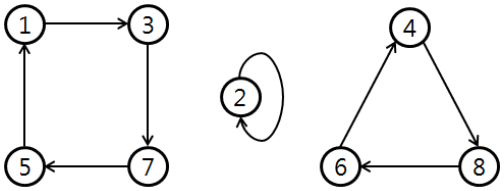


Figure 1.

If we represent a permutation as an array $\begin{pmatrix}1 & \dots & i & \dots & n\\ \pi_1 & \dots & \pi_i & \dots & \pi_n\end{pmatrix}$ then there is a directed edge from i to π_i in its corresponding cycle-arrow form for each i.

As shown in Figure 1, there are 3 cycles when we represent permutation (3,2,7,8,1,4,5,6) in cycle-arrow form. We call these cycles 'permutation cycles.'

You are to write a program which counts the number of permuation cycles in a given permutation of n integers.

Input
Your program is to read from standard input. The input consists of $\boldsymbol{T}$ test cases. The number of test cases $\boldsymbol{T}$ is given in the first line of the input. Each test case starts with a line containing an integer $\boldsymbol{n}$ ($\boldsymbol{2 \le n \le 1,000}$). In the following line there is a permutation of $\boldsymbol{n}$ integers from 1 to $\boldsymbol{n}$. Each integer in a permutation is separated by a blank.

Output
Your program is to write to standard output. Print exactly one line for each test case. The line should contain the number of permutation cycles in the given permutation.

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

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

The 39th Annual
ACM International Collegiate Programming Contest
Asia Regional - Daejeon

Problem G
Road Repair
Time Limit: 0.2 Second

In a city, there is a huge network of roads, each of which connects two districts. The network seems like a tree, in other words, for two arbitrary districts, there is exactly one path between them. Here, the path is a sequence of roads on which we can travel from one district to the other.

Many cars pass roads in the network day after day and so the roads are damaged seriously. The city department of transportation repairs roads regularly. Each road R has a cost c and a benefit b. We can repair the road R with the cost c and obtain the (social) benefit b when R is repaired.

Repairing all the roads belonging to a path, the cost and benefit of the path are the total cost and benefit of its roads, respectively. Given an upper bound of cost C, you should find a path in the network that maximizes its benefit while having a cost of at most C.

For example, a network is given in Figure 1. The circles and the solid lines represent districts and roads, respectively. The pair of integers $\langle c, b \rangle$ on a road describes that the road has the cost c and the benefit b. In this example, if the bound of cost is 8, then the red path given in Figure 2 has the benefit 13 with the cost 8 and it maximizes the benefit.

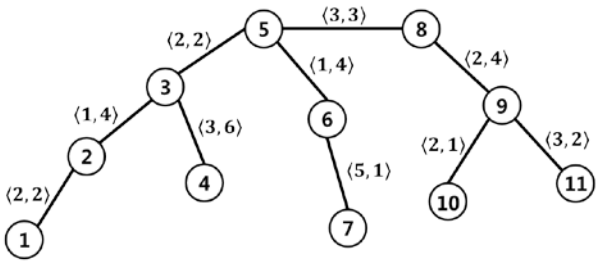


Figure 1.

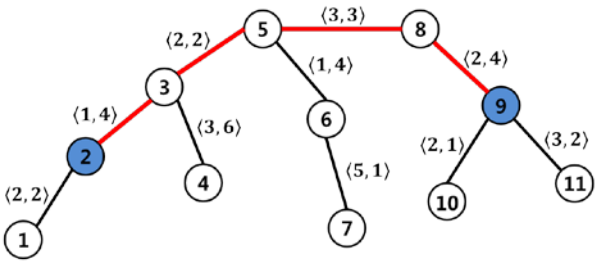


Figure 2.

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 starts with an integer $\boldsymbol{n}$, the number of districts in the network, where $\mathbf{2} \leq \boldsymbol{n} \leq \mathbf{20,000}$. The districts are numbered from $\mathbf{1}$ to $\boldsymbol{n}$. There are always exactly $\boldsymbol{n} - \mathbf{1}$ roads in the network. Each of the following $\boldsymbol{n} - \mathbf{1}$ lines contains four integers $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{c}$ and $\boldsymbol{b}$, representing that the road connecting districts $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ has the cost $\boldsymbol{c}$ and the benefit $\boldsymbol{b}$. Here, $\mathbf{1} \leq \boldsymbol{\alpha}, \boldsymbol{\beta} \leq \boldsymbol{n}$ and $\mathbf{1} \leq \boldsymbol{c}, \boldsymbol{b} \leq \mathbf{1,000}$. Finally, the last line contains an integer $\boldsymbol{C}$ to represent the upper bound of cost, where $\mathbf{1} \leq \boldsymbol{C} \leq \mathbf{2 \times 10^7}$.

Output
Your program is to write to standard output. Print exactly one line for each test case. The line should contain an integer to represent the maximum benefit which a path can obtain, while having a cost of at most C. If such a path never exists, then the line contains `0`.

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

Sample Input	Output for the Sample Input
2 11 1 2 2 2 2 3 1 4 3 4 3 6 3 5 2 2 5 6 1 4 5 8 3 3 6 7 5 1 8 9 2 4 9 10 2 1 9 11 3 2 8 18 1 9 1 2 9 14 2 3 10 14 6 2 2 9 5 2 3 10 1 3 4 11 2 6 11 15 3 3 12 15 4 4 5 12 1 5 6 12 2 6 17 18 3 4 16 18 2 5 7 13 2 3 13 16 1 2 8 16 1 2 15 17 4 1 14 17 2 3 10	13 18

The 39th Annual
ACM International Collegiate Programming Contest
Asia Regional - Daejeon

Problem H
String Transformation
Time Limit: 0.1 Second

A string is called to be well-formed if it can be generated by the following rules:

(a) `ab` is a well-formed string.
(b) If *S* is a well-formed string, `a`*S*`b` is also well-formed.
(c) If *S* and *T* are well-formed strings, *ST* is also well-formed.

For example, `aabbabab`, `abababab`, and `aaaabbbb` are well-formed strings.

For two well-formed strings, *A* and *B*, we want to transform *A* into *B* by performing successively the operation of exchanging adjacent characters. After performing each operation, the string should be well-formed. Let *A* = `aabbabab` and *B* = `aaaabbbb`. Then *A* can be transformed into *B* by five operations as follows:

`aabbabab` → `aabbaabb` → `aababbb` → `aabaabbb` → `aaababbb` → `aaaabbbb`.

Given two well-formed strings, *A* and *B*, you have to find the minimum number of operations which are needed for transforming *A* into *B*.

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 line containing two well-formed strings *A* and *B* which are separated by a single space. The length of *A* is the same as the length of *B*, and is at least 2 and at most 100,000.

Output
Your program is to write to standard output. Print exactly one line for each test case. The line should contain the minimum number of the operations which are needed for transforming *A* into *B*. If the transformation is impossible, print -1.

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

Sample Input	Output for the Sample Input
2 aabbabab aaaabbbb aabbab abaabb	5 2

The 39th Annual

ACM International Collegiate Programming Contest
Asia Regional - Daejeon

Problem I
Three Squares
Time Limit: 1 Second

A square is called axis-parallel if its sides are vertical or horizontal, and two squares are said to be congruent if their side lengths are equal. You are given a set P of n points in the plane. Any real number $x \geq 0$ is called *3SQ-sufficient* for P if there exist three congruent axis-parallel squares of side length x such that the union of the three squares contains all points in the set P. If a square has side length zero, it is degenerated to a point, so you can consider a point itself as a square of side length zero. Your program is to find the smallest 3SQ-sufficient number for a given input set P of points.

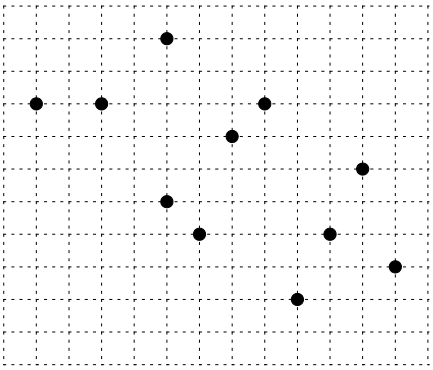
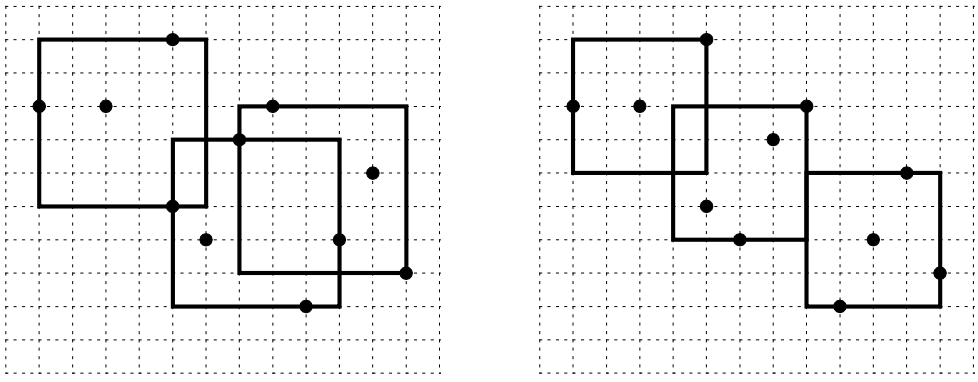


Figure 1. 11 points in the plane

For example, let P be the set of 11 points in the plane as shown in Figure 1. Then, one can find three congruent axis-parallel squares of side length 5 whose union contains all these 11 points in the set P, as shown in Figure 2(a). This means that number 5 is 3SQ-sufficient for this set P of 11 points. Notice that points on the boundary of a square are considered to be contained in the square.



(a) **(b)**

Figure 2. Two numbers 5 and 4 are 3SQ-sufficient

You can find an even smaller 3SQ-sufficient number for this case; there exist three axis-parallel squares of side length 4 whose union contains all the 11 points as illustrated in Figure 2(b). More effort to find a further smaller 3SQ-sufficient number will be however worthless because number 4 is in fact the smallest 3SQ-sufficient number for this set P. Therefore, if these 11 points of Figure 1 are given as input of your program, then your program must output 4 as the answer.

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 starts with a line containing an integer, n ($1 \leq n \leq 100{,}000$), where n is the number of points in the data set P. In the following n lines, each of the n points in P is given line by line. Each point is represented by two numbers separated by a single space, which are the x-coordinate and the y-coordinate of the point, respectively. Each coordinate is given as an integer between $-1{,}000{,}000{,}000$ and $1{,}000{,}000{,}000$, inclusively. Note that there may be two or more points having the same coordinates in the input data set P.

Output

Your program is to write to standard output. Print exactly one line for each test case. The line should contain an integer representing the smallest 3SQ-sufficient integer for a given input set P of points.

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

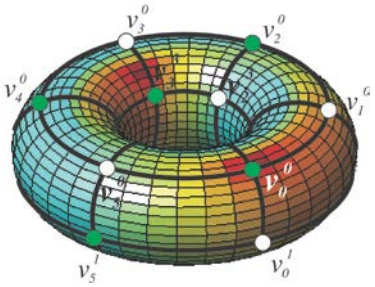
Sample Input	Output for the Sample Input
3 5 1 0 0 1 5 2 2 3 3 2 11 1 8 3 8 5 5 5 10 6 4 7 7 8 8 9 2 11 6 12 3 10 4 4 1 1 2 8 -21 45 1 1	1 4 0

The 39th Annual

ACM International Collegiate Programming Contest
Asia Regional - Daejeon

Problem J
Torus
Time Limit: 0.1 Second

As is well known, a *regular polygon* is a polygon having all sides of the same length and all equal angles. A *regular tiling* of the Euclidean plane is a covering of the entire plane with non-overlapping, identical regular polygons, in which the polygons are placed vertex-to-vertex. There exist only three types of polygons that admit a regular tiling: square, equilateral triangle and regular hexagon. A *lattice* is an infinite graph whose drawing on the plane forms a regular tiling. In Figure 1, three lattices are illustrated: square lattice, triangular lattice, and hexagonal lattice (from left to right).



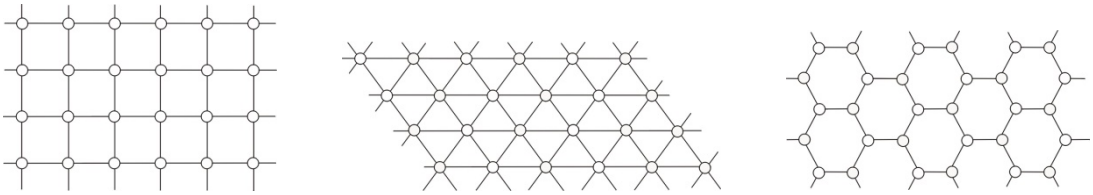


Figure 1. The square, triangular and hexagonal lattices.

One day, Heechul, an eminent researcher in the community of graph theory, discovered the interesting fact that such lattices can be converted into finite graphs (graphs with finite numbers of vertices and edges), conveying their inherent characteristics, e.g. the beautiful symmetrical structures. In particular, the finite graphs allow effective drawings on the surface of a torus without edge crossings, also resulting in the tiles (i.e., the regions bounded by the minimum-length cycles) that are very similar in their shapes and sizes. So, he delightedly called such drawings *toroidal lattices*, which are now defined as follows (refer to Figure 2):

Definition 1. For two integers $m, n \geq 3$, the $m \times n$ *toroidal square lattice* is a graph whose vertex set is $\{v_j^i : 0 \leq i \leq m-1, 0 \leq j \leq n-1\}$ and edge set is $\{(v_j^i, v_{j'}^{i'}) : (i' = i \text{ and } j' \equiv j+1 \pmod n) \text{ or } (j' = j \text{ and } i' \equiv i+1 \pmod m)\}$.

Definition 2. For two integers $m, n \geq 3$ with m even, the $m \times n$ *toroidal triangular lattice* is the graph constructed from the $m \times n$ toroidal square lattice by adding edges of $E_0 \cup E_1 \cup \cdots \cup E_{m-1}$, where

$$E_i = \begin{cases} \{(v_j^i, v_{j'}^{i'}) : i' \equiv i+1 \pmod m \text{ and } j' \equiv j-1 \pmod n\} & \text{if } i \text{ is even,} \\ \{(v_j^i, v_{j'}^{i'}) : i' \equiv i+1 \pmod m \text{ and } j' \equiv j+1 \pmod n\} & \text{if } i \text{ is odd.} \end{cases}$$

Definition 3. For two integers $m, n \geq 4$ with both even, the $m \times n$ *toroidal hexagonal lattice* is the graph whose vertex and edge sets are $\{v_j^i : 0 \leq i \leq m-1, 0 \leq j \leq n-1\}$ and $\{(v_j^i, v_{j'}^{i'}) : (j' = j \text{ and } i' \equiv i+1 \pmod m) \text{ or } (i' = i \text{ and } j' \equiv j+1 \pmod n \text{ and } i+j \equiv 0 \pmod 2)\}$, respectively.

To celebrate the 60th birthday of Heechul, a project was initiated by his colleagues to build a C/C++ library for lattice-related graph algorithms. In order to help them, you are going to write a program that, given an $m \times n$ toroidal lattice, finds a cycle that visits every vertex exactly once. (Note that a cycle of a graph is represented as a sequence $(u_1, u_2, \ldots, u_{mn})$ of mn distinct vertices such that u_k and u_{k+1} are adjacent in the graph for all

$k \in \{1, \ldots, mn-1\}$ and moreover, u_{mn} and u_1 are also adjacent.) Since the toroidal square lattice is rather simple, we will only consider the toroidal triangular/hexagonal lattices in this coding.

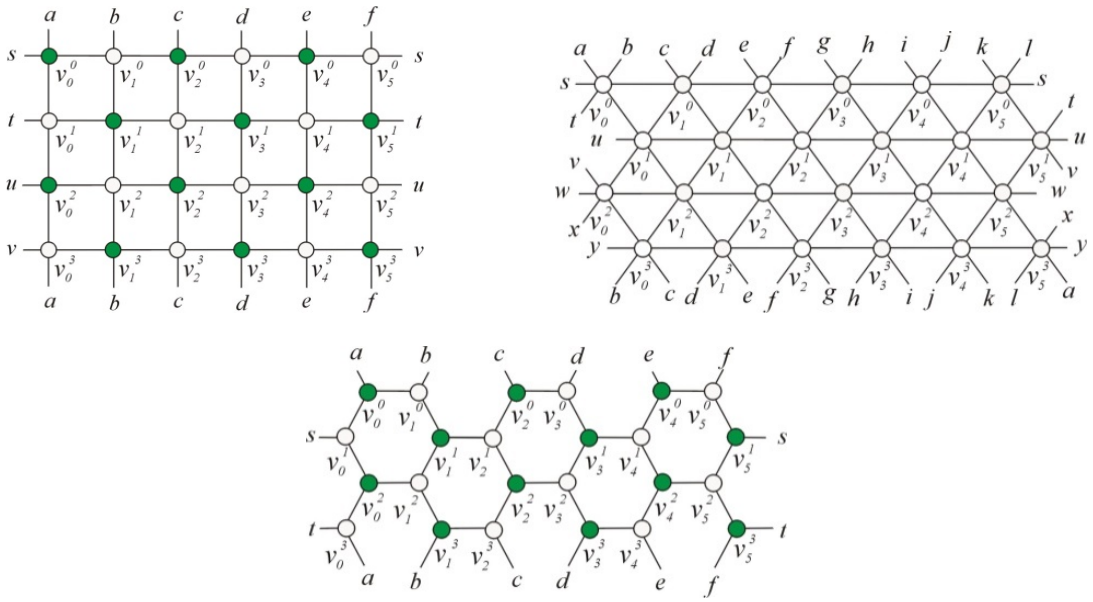


Figure 2. The 4×6 toroidal square, triangular and hexagonal lattices.

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. Then, *T* lines are followed, where each line contains three integers, $\boldsymbol{m}$, $\boldsymbol{n}$ and $\boldsymbol{p}$, where $\mathbf{3} \leq \boldsymbol{m}, \boldsymbol{n} \leq \mathbf{100}$ and $\boldsymbol{p} \in \{\mathbf{3}, \mathbf{6}\}$, indicating that the corresponding input graph is an $\boldsymbol{m} \times \boldsymbol{n}$ toroidal lattice of type triangular (if $\boldsymbol{p} = \mathbf{3}$) or hexagonal (if $\boldsymbol{p} = \mathbf{6}$). You are always given a well-defined toroidal lattice.

Output

Your program is to write to standard output. The output consists of the results for the *T* test cases in the given order. For each test case, the first line must contain an integer indicating whether there exists a feasible solution. If yes, the integer must be `1`; otherwise `-1`. When and only when the first line is `1`, it must be followed by mn lines, describing the sequence of vertices of the found cycle, where the index of vertex v_j^i is to be output as `(i,j)`. In case multiple solutions are possible, just output any one of them. No whitespace characters (blanks and/or tabs) are allowed inside a line.

The following shows a sample output for an input with one test case.

Sample Input	Output for the Sample Input
1 4 3 3	1 (0,0) (1,0) (2,0) (3,0) (0,1) (1,1) (2,1) (3,1) (3,2) (2,2) (1,2) (0,2)

The 39th Annual
ACM International Collegiate Programming Contest
Asia Regional - Daejeon

Problem K
Travel Card
Time Limit: 0.3 Second

You correctly think that owning a car can cost a lot of money and can be a great trouble due to congestion. The public transportation system of your city is made up of buses and trains. The fare rule is as follows.

1. Each bus ride costs $1, and each train ride costs $2.
2. No free transfer: if you want to transfer between buses/trains, you are supposed to buy a new ticket.
3. A daily bus card costs $3 and gives you limitless bus rides within a day (but you need to pay for train rides).
4. A daily travel card costs $6 and gives you limitless bus and train rides within a day.
5. A 7 days bus card costs $18 and gives you limitless bus rides for seven days (but you need to pay for train rides).
6. A 7 days travel card costs $36 and gives you limitless bus and train rides for seven days.
7. A 30 days bus card costs $45 and gives you limitless bus rides 30 days (but you need to pay for train rides).
8. A 30 days travel card costs $90 and gives you limitless bus and train rides for 30 days.

You thought that the rule was too complicated and you did not know how many times you would ride buses or trains. Instead of buying bus card or travel card, you paid single ride fare each time. Now you think that if you bought travel cards, then you would have spent less money.

For example, suppose your travel log for three days is as follows.

Day	Number of bus rides	Number of train rides
1	1	0
2	5	0
3	0	2

If you do not buy any travel card, you will spend $10 in total as you spend $1 on day 1, $5 on day 2, and $4 on day 3. However, you can spend $1 on day 1, $3 on day 2 by buying a daily bus card, and $4 on day 3. In total, you will spend $8, which is smaller than $10 and optimal.

Given the travel logs of $\boldsymbol{n}$ days, write a program which calculates the minimum amount of fare.

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 starts with a line containing one integers $\boldsymbol{n}$ ($\mathbf{1} \le \boldsymbol{n} \le \mathbf{10,000}$), where $\boldsymbol{n}$ is the number of days. Each of the following $\boldsymbol{n}$ lines contains two integers $\boldsymbol{a}$ and $\boldsymbol{b}$ where $\boldsymbol{a}$ is the number of bus rides and $\boldsymbol{b}$ is that of train rides on that day ($\mathbf{0} \le \boldsymbol{a}, \boldsymbol{b} \le \mathbf{100,000}$). There is a single space between two integers in the same line.

Output
Your program is to write to standard output. Print exactly one line for each test case. The line should contain the minimum amount of fare.

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

Sample Input	Output for the Sample Input
2 3 1 0 5 0 0 2 6 2 3 4 2 1 5 2 2 3 4 5 5	8 36

The 39th Annual
ACM International Collegiate
Programming Contest
Asia Regional – Daejeon

Problem L
Two Yachts
Time Limit: 0.1 Second

The ICPC marine resort has planned an event to invite bids for the use of two luxurious yachts for the upcoming season. The participants of the bidding must submit a sealed proposal, giving the period of use and the bidding price. After bidding, the event manager of the resort chooses as many proposals as possible to get maximum profit as long as the periods of use do not overlap for each yacht. A participant whose proposal is chosen uses a yacht during the submitted period. We assume that the season consists of m days numbered from 1 to m.

For example, let's assume that there are five proposals as follows:

No	period of use		bidding price
	beginning day	ending day	
1	10	18	40,000
2	1	12	50,000
3	2	7	60,000
4	9	16	30,000
5	5	20	80,000

The manager cannot choose all proposals for the above example, because there exists some days at which three or more periods are overlapped. If the manager chooses the proposals 2 and 5, the profit will be 130,000. In order to get the maximum profit, he has to choose the proposals 1, 3, and 5. Then, the profit will be 180,000.

Given n proposals for the bidding, write a program that finds a set of proposals which gives the maximum profit.

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 starts with an integer $\boldsymbol{n}$, the number of the proposals for the bidding, where $\boldsymbol{1 \le n \le 10,000}$. Each of the following $\boldsymbol{n}$ lines contains three integers $\boldsymbol{s}$, $\boldsymbol{t}$, and $\boldsymbol{p}$ representing the beginning day, the ending day of the period of use, and the bidding price, respectively, where $\boldsymbol{1 \le s \le t \le 10,000,000}$ and $\boldsymbol{1 \le p \le 100,000}$. You may assume the number of proposals that overlap on the same day does not exceed $\boldsymbol{100}$.

Output
Your program is to write to standard output. Print exactly one line for each test case. The line should contain an integer representing the maximum profit which the resort will be obtained.

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

Sample Input	Output for the Sample Input
2 5 10 18 40000 1 12 50000 2 7 60000 9 16 30000 5 20 80000 7 1 3 100 3 5 100 5 7 100 7 9 100 1 4 100 5 5 100 6 9 100	180000 500