



4104 - MODEX

Asia - Singapore - 2007/2008

Many well-known cryptographic operations require modular exponentiation. That is, given integers x , y and n , compute $x^y \bmod n$. In this question, you are tasked to program an efficient way to execute this calculation.

Input

The input consists of a line containing the number c of datasets, followed by c datasets, followed by a line containing the number ``0'`.

Each dataset consists of a single line containing three positive integers, x , y , and n , separated by blanks. You can assume that $1 < x$, $n < 2^{15} = 32768$, and $0 < y < 2^{31} = 2147483648$.

Output

The output consists of one line for each dataset. The i -th line contains a single positive integer z such that

$$z = x^y \bmod n$$

for the numbers x , y , z given in the i -th input dataset.

Sample Input

```
2
2 3 5
2 2147483647 13
0
```

Sample Output

```
3
11
```

Singapore 2007-2008



4105 - JONES

Asia - Singapore - 2007/2008

The time now is 12 noon and Indiana Jones is standing on the western bank of a river. He wants to reach the eastern bank as fast as possible. Across the river is a series of stones, arranged in a straight line, and each stone is 1 meter away from its immediate neighbours or the two river banks.



Let us divide the time into intervals of one minute each, such that Interval 0 starts at 12:00:00, Interval 1 starts at 12:01:00, and so on. At the start of each interval, Indiana Jones can hop once from a stone/river bank to any stone/bank that is not more than 1.5 meters away, or stay put. He can hop so fast that we assume the time taken per hop is negligible.

The tricky part is this: The stones may sink and resurface! Within a time interval, a stone may sink at the middle of the interval, remains submerged and may resurface at the middle of another interval. If Indiana Jones is standing on a sinking stone, he will drown. Of course, Indiana Jones cannot hop to a stone that is submerged. At 12 noon, all stones are above the water and Indiana Jones is ready to hop. He has already derived the pattern of sinking/resurfacing for each stone during the next few intervals. Our task is to find the fastest way to cross the river.

Figure 1 shows an example of sinking and resurfacing stones over time. The example contains three stones, each of which is first sinking, then resurfacing and then sinking. The fastest way for Indiana Jones to cross the river is shown as a black line.

Input

The input consists of a line containing the number c of datasets, followed by c datasets, followed by a line containing the number '0'.

The first line of each dataset contains two positive integers, separated by a blank. The first value s is the number of stones. The second value t is the number of intervals whose movement pattern Indiana Jones can predict. You can assume $1 \leq s \leq 500$ and $1 \leq t \leq 500$. The following t lines of each dataset describe the behavior of the stones in each interval. The i -th line describes the behavior of the stones in Interval i , where $0 \leq i < t$. Within each line, there are s characters, separated by blanks. The j -th character indicates the movement of the j -th stone in the middle of the i -th interval as follow:

``s":

The stone is sinking.

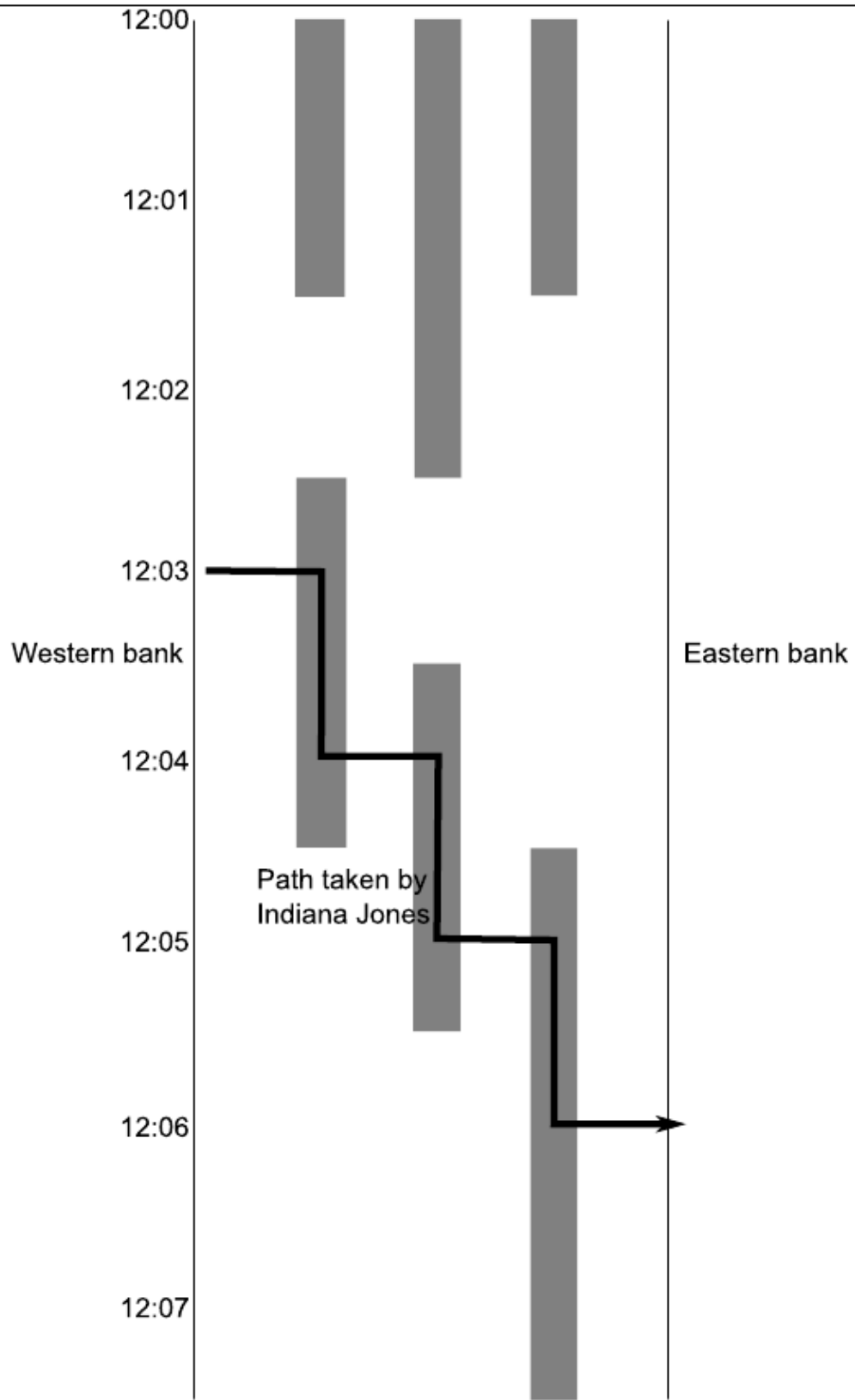
``r":

The stone is resurfacing.

``u":

The stone is not moving.

Figure 1 Example of a Pattern and Indiana Jones's Path



Output

The output consists of one integer, which indicates the earliest interval at the beginning of which Indiana Jones can reach the eastern bank. If there is no way Indiana Jones can cross the river within t minutes, the output should be -1 . Note that Indiana Jones actually has $t + 1$ chances to hop.

Notes for the sample: In this dataset, the fastest way to cross over is as follows:

1. Stay put at the beginning of Interval 0.
2. Stay put at the beginning of Interval 1.
3. Stay put at the beginning of Interval 2.
4. Jump from the western bank to the first stone at the beginning of Interval 3.
5. Jump from the first stone to the second stone at the beginning of Interval 4.
6. Jump from the second stone to the third stone at the beginning of Interval 5.
7. Jump from the third stone to the eastern bank at the beginning of Interval 6.

Thus, Indiana Jones can reach the eastern bank the earliest at the beginning of Interval 6.

Sample Input

```
1
3 7
u u u
s u s
r s u
u r u
s u r
u s u
u u u
0
```

Sample Output

```
6
```

Singapore 2007-2008



4106 - ACORN

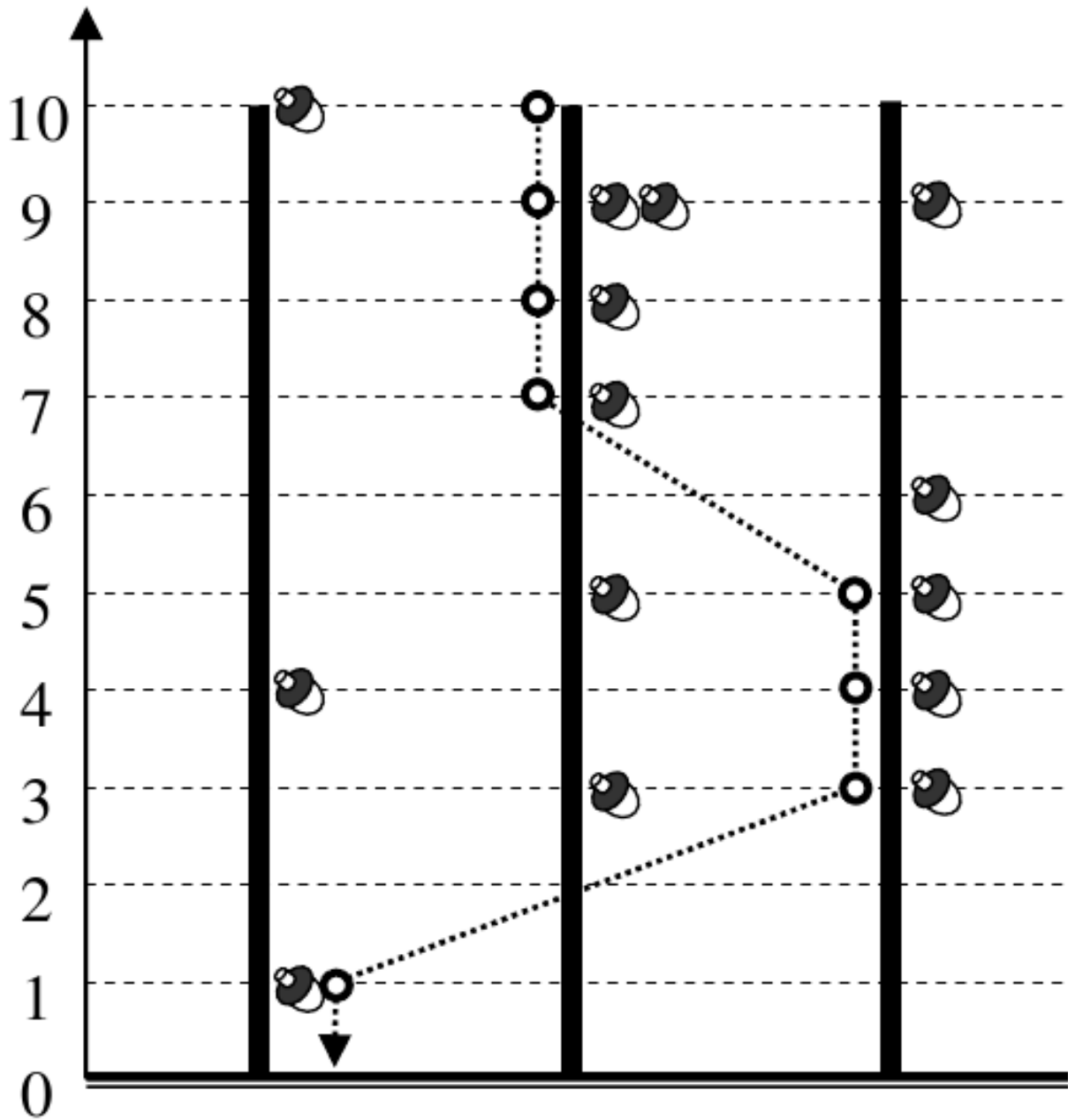
Asia - Singapore - 2007/2008

In the first morning of every summer, when the first ray of sunlight breaks into the oak forest, Jayjay, the flying squirrel, quickly climbs to the top of an oak tree in the forest. From there, he starts his descent to the ground, and tries to gather as many acorns from the trees on his way down. Being a flying squirrel, Jayjay can choose, at any moment, to climb down the tree trunk or to fly from one tree to any other tree on his descending journey. However, he loses f feet of height every time he flies from one tree to another.

Suppose the forest has t oak trees, and all the trees have the same height of h feet. Given the height of every acorn on each tree, write a program to compute the maximal possible number of acorns Jayjay can collect by choosing a tree to climb and descend as described.

Figure 2 shows an example of $t = 3$ oak trees with three, six, and five acorns, respectively. The white circles and grey line indicate a path for Jayjay to collect the maximal possible number of eight acorns, assuming that the height he loses for each flight is $f = 2$.

Figure 2 Example of oak trees with acorns



Input

The input consists of a line containing the number c of datasets, followed by c datasets, followed by a line containing the number '0'.

The first line of each dataset contains three integers, t, h, f , separated by a blank. The first integer t is the number of oak trees in the forest. The second integer h is the height (in feet) of all the oak trees. The third integer, f , is the height (in feet) that Jayjay loses every time he flies from one tree to another. You may assume that $1 \leq t, h \leq 2000$, and $1 \leq f \leq 500$.

The first line of each dataset is followed by t lines. The i -th line specifies the height of every acorn on the i -th tree. The line begins with a non-negative integer a that specifies how many acorns the i -th tree has. Each of the following a integers n indicates that an acorn is at height n on the i -th tree. The positive integers in each line are sorted in ascending order, and repetitions are allowed. Thus, there can be more than one acorn at the same height on the same tree. You can assume that $0 \leq a \leq 2000$, for each i .

Output

The output consists of one line for each dataset. The c -th line contains one single integer, which is the maximal possible number of acorns Jayjay can collect in one single descent for dataset c .

Note: This dataset and Jayjay's path to collect the maximal number of 8 acorns are shown in Figure 2.

Sample Input

```
1
3 10 2
3 1 4 10
6 3 5 7 8 9 9
5 3 4 5 6 9
0
```

Sample Output

```
8
```

Singapore 2007-2008



4107 - TUSK

Asia - Singapore - 2007/2008

A conservation group found a way to scan the cross-section of elephant tusks like the one shown on the left. The scanned image of a tusk is a collection of bright spots, which can be treated as a set of points on the plane, where no three points form a straight line. The group made a database of many tusks and hope that the database can help in tracking illegal ivory trade.



Cross-section of an elephant tusk

To facilitate retrieval from the database, it is desirable to have a feature representation that remains unchanged even if a scanned image is translated or rotated. They decided to use the number of k -sets to represent a set of points, which is defined below.

Consider a set P of n points in the plane, where no three points form a straight line. Any line in the plane that does not contain a point in P will split the set into two sets X , Y , where X contains points on one side of the line and Y contains points on the other side. If the number of points in X is k , then we call the collection $\{X, Y\}$ a k -set in P . Note that $\{X, Y\} = \{Y, X\}$, and thus a k -set is also an $(n - k)$ -set. Given a set P and k , your task is to compute the number of different k -sets in P .

Input

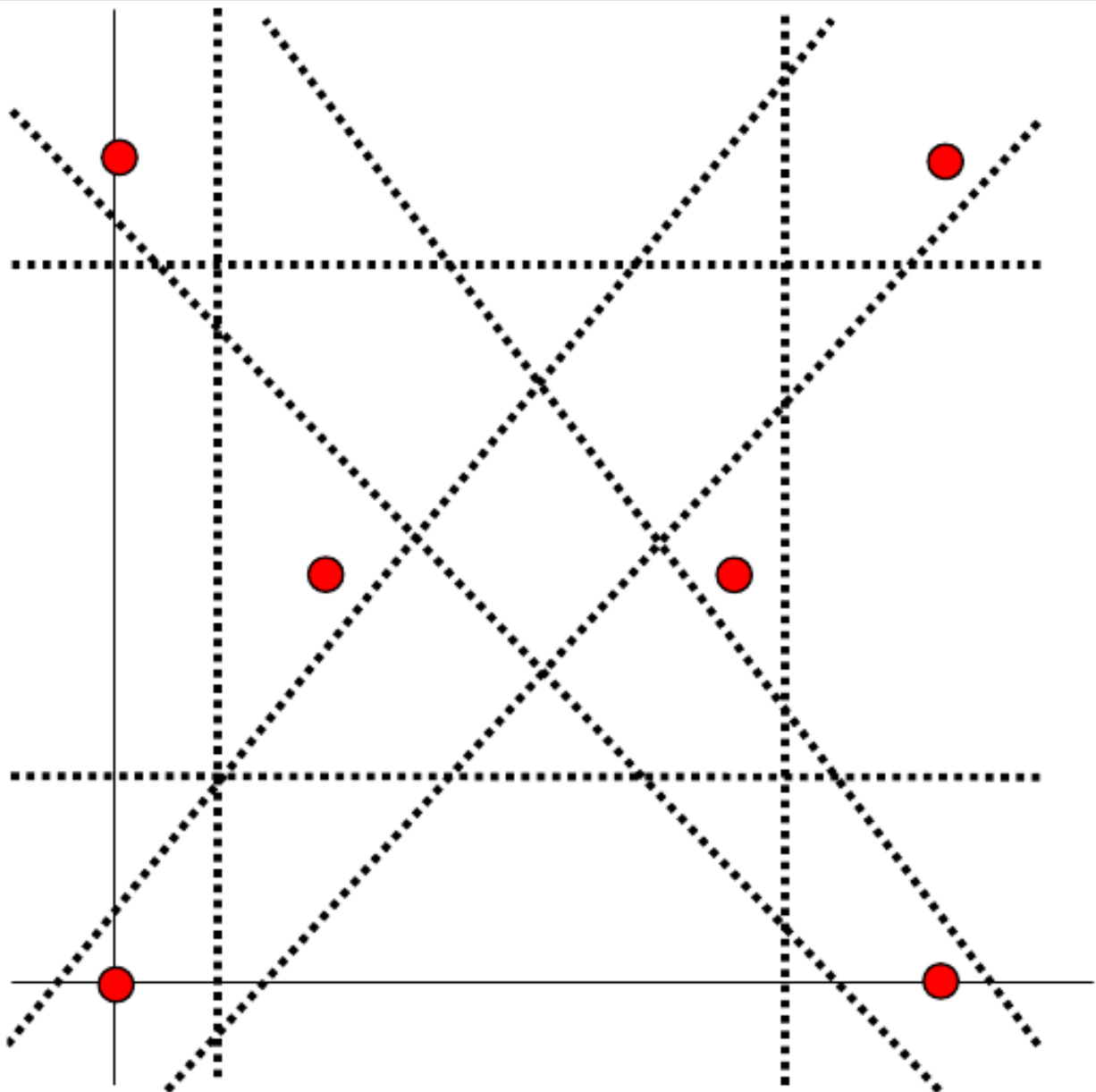
The input consists of a line containing the number c of datasets, followed by c datasets, followed by a line containing the number '0'.

The first line of each dataset contains two integers, separated by a blank. The first integer gives the number n of points in P , and the second integer gives k , where $(0 < k < n \leq 2000)$. The following n lines of each dataset each contains two non-negative integers, indicating the x and y -coordinates of the corresponding point. The x and y -coordinates range from 0 to 10000.

Output

The output consists of one line for each dataset. The c -th line contains the number of k -sets for dataset c .

Figure 3 Illustration of sample input and sample output



Note: This dataset and the lines that form its 2-sets are shown in Figure 3.

Sample Input

```
1
6 2
0 0
4 0
4 4
0 4
1 2
3 2
0
```

Sample Output

```
8
```

Singapore 2007-2008



4108 - SKYLINE

Asia - Singapore - 2007/2008

The skyline of Singapore as viewed from the Marina Promenade (shown on the left) is one of the iconic scenes of Singapore. Country X would also like to create an iconic skyline, and it has put up a call for proposals. Each submitted proposal is a description of a proposed skyline and one of the metrics that country X will use to evaluate a proposed skyline is the amount of overlap in the proposed sky-line.



Skyline of Singapore at Night

As the assistant to the chair of the skyline evaluation committee, you have been tasked with determining the amount of overlap in each proposal. Each proposal is a sequence of buildings, $\{b_1, b_2, \dots, b_n\}$, where a

building is specified by its left and right endpoint and its height. The buildings are specified in back to front order, in other words a building which appears later in the sequence appears in front of a building which appears earlier in the sequence.

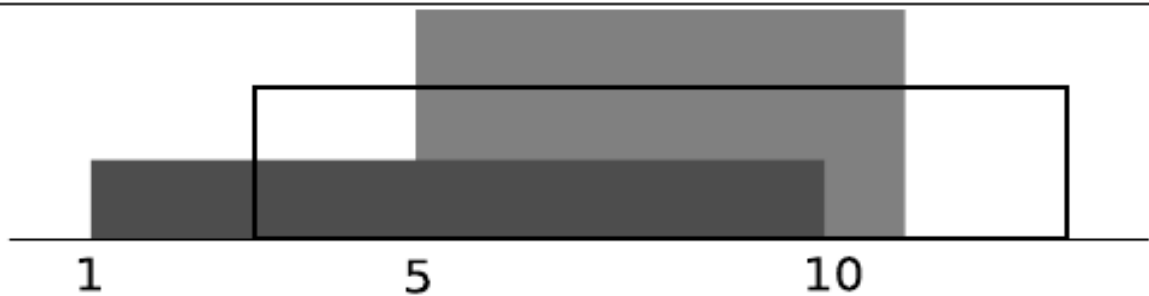
The skyline formed by the first k buildings is the union of the rectangles of the first k buildings (see Figure 4). The overlap of a building, b_i , is defined as the total horizontal length of the parts of b_i , whose height is greater than or equal to the skyline behind it. This is equivalent to the total horizontal length of parts of the skyline behind b_i which has a height that is less than or equal to h_i , where h_i is the height of building b_i . You may assume that initially the skyline has height zero everywhere.

Input

The input consists of a line containing the number c of datasets, followed by c datasets, followed by a line containing the number '0'.

The first line of each dataset consists of a single positive integer, n ($0 < n < 100000$), which is the number of buildings in the proposal. The following n lines of each dataset each contains a description of a single building. The i -th line is a description of building b_i . Each building b_i is described by three positive integers, separated by spaces, namely, l_i , r_i and h_i , where l_i and r_i ($0 < l_i < r_i \leq 100000$) represents the left and right end point of the building and h_i represents the height of the building.

Figure 4 Computing Skyline Overlap



Output

The output consists of one line for each dataset. The c -th line contains one single integer, representing the amount of overlap in the proposal for dataset c . You may assume that the amount of overlap for each dataset is at most 2000000.

Note: In this test case, the overlap of building b_1 , b_2 and b_3 are 6, 4 and 4 respectively. Figure 4 shows how to compute the overlap of building b_3 . The grey area represents the skyline formed by b_1 and b_2 and the black rectangle represents b_3 . As shown in the figure, the length of the skyline covered by b_3 is from position 3 to position 5 and from position 11 to position 13, therefore the overlap of b_3 is 4.

Sample Input

```
1
3
5 11 3
1 10 1
3 13 2
0
```

Sample Output

```
14
```

Singapore 2007-2008



4109 - USHER

Asia - Singapore - 2007/2008

In a large church in New Zealand, after the church service, the priest gives an empty collection box that can hold c dollar coins to the "light-fingered usher". The usher then passes the collection box to a nearby parishioner. When receiving the box a parishioner adds a few dollar coins, then passes it to another nearby parishioner. When the light-fingered usher receives the box, he quietly removes one dollar coin from the box, slips it into his pocket, and passes the box to a nearby parishioner.

The behavior of the usher is given by a set of parishioners to whom he may pass the box. The behavior of a parishioner is given by a set of rules, each consisting of a donation of at least two dollar coins, and another parishioner to whom the box is passed after he places the coins in the box. As soon as the box is full, containing c coins, it is immediately passed to the priest, even when a parishioner cannot finish entering all the coins specified by the chosen rule.

Your problem is to compute the maximal possible gain for the usher, which is achieved when the parishioners always choose a rule that leads to the biggest number of coins in the usher's pocket.

Input

The input consists of a line containing the number c of datasets, followed by c datasets, followed by a line containing the number '0'.

The first line of each dataset contains two numbers b and p , separated by a blank. The number b specifies the capacity of the box, and p the number of parishioners. You can assume that $1 \leq b \leq 1000000$ and $1 \leq p \leq 500$.

The next line specifies the behavior of the usher, represented by integers separated by a blank. The first integer specifies the number q of parishioners, to which the usher may pass the box. The next q integers each represent a parishioner to whom he may pass the box. The parishioners are numbered from 1 to p . Each line i ($1 \leq i \leq p$) of the next p lines of each dataset describes the behavior of parishioner i , and consists of integers separated by a blank. The first integer of the line specifies the number of rules for the parishioner. Each rule is represented by a pair of integers, the first of which specifies the number of coins to give, which must be equal or greater than 2. The second number indicates the parishioner to receive the box next, where the number 0 identifies the usher. When there are two or more rules, the parishioner may choose to apply any of them. You can assume that the number of rules per parishioner is at least 1 and less than or equal to 1000; there may be multiple rules with the same parishioner to receive the box.

Output

The output consists of one line for each dataset. The c -th line contains the maximal number of coins that the usher can obtain for dataset c .

Note: This dataset specifies that the box can hold up to 10 coins, and that there are two parishioners. The usher may pass the box to either one of the two parishioners, as indicated by the second line. Parishioner 1 has two rules, and can choose either one, when the box is passed to him. The first rule says to put down 6 dollar coins and pass the box to the usher (indicated by '0'), and the second rule is put down 4 dollar coins and pass the box to parishioner 2. The last line specifies one single rule for parishioner 2, who must put down 5 dollar

coins, and then pass the box to the usher.

The usher can obtain the maximal amount of 2 dollars by passing the box to parishioner 2, who passes it back to the usher. At that point, there are 5 dollars on the box. After removing one dollar, the box goes with 4 dollars to parishioner 2, and back to the usher, now with 8 dollars. The usher removes another dollar, and the box goes to parishioners 2 with 7 dollars. Parishioner 2 starts to apply his rule, and manages to put three more coins into the box, after which the box is full and goes to the priest. The usher ends up with two dollar coins in his pocket.

Sample Input

```
1
10 2
2 1 2
2 6 0 4 2
1 5 0
0
```

Sample Output

```
2
```

Singapore 2007-2008



4110 - RACING

Asia - Singapore - 2007/2008

Singapore will host a Formula One race in 2008. The race will be held on a 5.067km long street circuit, consisting of 14 left hand turns and 10 right hand turns. In the run up to the F1 race, the number of illegal night street racing activities have been on the rise. Such races consists of several rounds around a designated street circuit.



**Saint Andrew's Road, part of the Formula One circuit
(Kenny Pek, Piccom)**

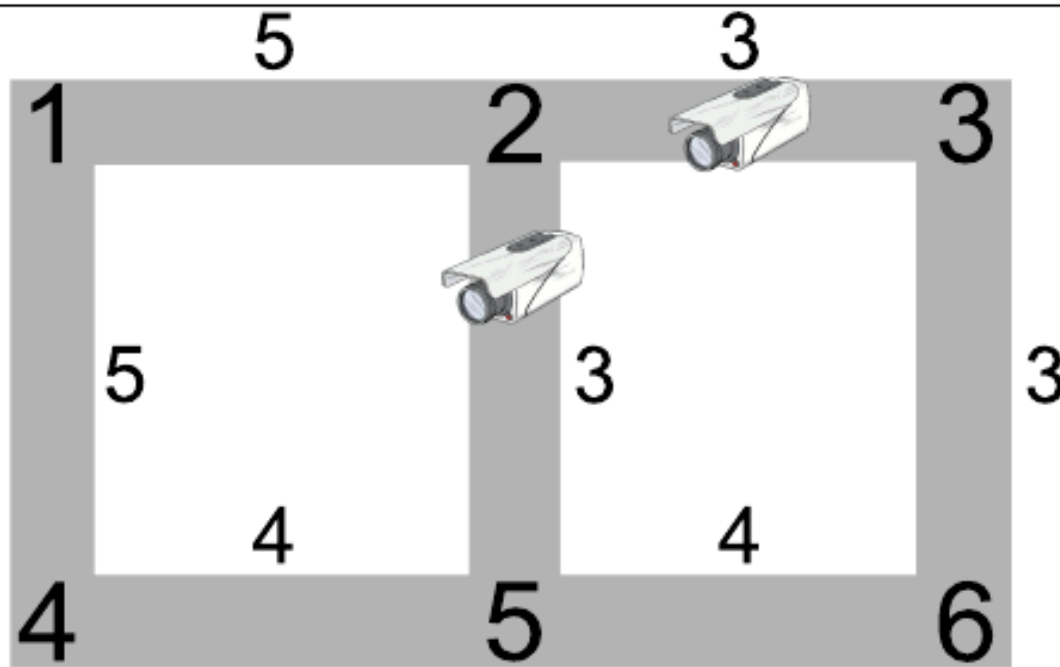
The authorities would like to deploy a new vehicle monitoring system in order to catch these illegal Saint Andrew's Road, part of the Formula One circuit racers. The system consists of a (Kenny Pek, Piccom) number of cameras mounted along various roads. For the system to be effective, there should be at least one camera along each of the possible circuits.

The Singapore road system can be represented as a series of junctions and connecting bidirectional roads (see Figure 5). A possible racing circuit consists of a start junction followed by a path consisting of three or more roads that eventually leads back to the start junction. Each road in a racing circuit can be traversed only in one direction, and only once.

Your task is to write a program that computes the optimal placement of the vehicle-monitoring cameras. You will be provided with a description of a connected road network to be monitored in terms of the roads and junctions. The junctions are identified by the bigger numbers in Figure 5. A camera can be deployed on the

roads (and not the junctions).

Figure 5 Illustration of sample input and one possible optimal placement of cameras



The cost of deploying a camera depends on the road on which it is placed. The smaller numbers by the roads in Figure 5 indicate the cost of deploying a camera on that road. Your job is to select a set of roads that minimizes the total cost of deployment while ensuring that there is at least one camera along every possible racing circuit (i.e. loop in the road network).

Input

The input consists of a line containing the number c of datasets, followed by c datasets, followed by a line containing the number '0'.

The first line of each dataset contains two positive integers, n and m , separated by a blank, which represent the number of junctions and number of roads, respectively. You may assume that $0 < n < 10000$ and $0 < m < 100000$. For simplicity, we label each of the n junctions from 1 to n . The following m lines of each dataset each describes one road. Each line consists of three positive integers which are the labels of two different junctions and the cost of deploying a camera on this road. The cost of deploying a camera is between 1 and 1000.

Output

The output consists of one line for each dataset. The c -th line contains one single number, representing the minimal cost of setting up the vehicle monitoring system such that there is at least one camera along every possible circuit.

Note: This data set depicts the situation shown in Figure 5. The two cameras show where cameras might be placed in order to monitor each circuit at minimal cost. Since each of the cameras have a cost of 3, the total minimal cost is 6.

Sample Input

```
1
6 7
1 2 5
2 3 3
1 4 5
4 5 4
5 6 4
6 3 3
5 2 3
0
```

Sample Output

```
6
```

Singapore 2007-2008