2008 the 33rd ACM Collegiate Programming Contest Asia Reginal

Taipei Site

November 8, 2008

Problems There are 10 problems in this packet.

Problem Input Input to the problems are through the input files. Input filenames are given in the table below. Each input file may contain one or more test cases. Test cases may be separated by any delimiter as specified in the problem statements.

Problem Output All output should be directed to standard output (screen output).

Time Limit The judges will run each submitted program with certain time limit (given in the table below).

	Problem Name	Input File	Time Limit
Problem A	Dangerous Tunnels	pa.in	2 secs.
Problem B	Fortune Card Game	pb.in	3 secs.
Problem C	Trip Planning	pc.in	1 sec.
Problem D	Road Networks	pd.in	2 secs.
Problem E	Early-Morning Pickup	pe.in	10 secs.
Problem F	Message	pf.in	5 secs.
Problem G	Lost Treasure	pg.in	3 secs.
Problem H	Space Elevator	ph.in	5 secs.
Problem I	Finding the Heaviest Path	pi.in	3 secs.
Problem J	Bonus Adjustment	pj.in	2 secs.

Problem A
Dangerous Tunnels

Input File: *pa.in*
Time Limit: 2 seconds

Problem Description

Somewhere in the world, there are two tribes separated by mountains. The two tribes are named Kulolo and Gulolo, respectively, where Kulolo is at a higher altitude and Gulolo is at a lower altitude. Due to the limitation of geography, Gulolo has fewer resources than Kulolo. In order to transport resources from Kulolo to Gulolo efficiently, several tunnels were built inside the mountains between these two tribes. There are also some rest stations built for people to take a break during the transportation in the tunnels. More specifically, each terminal of a tunnel is either Kulolo, Gulolo, or a rest station.

The structure of those tunnels is not stable. A *dangerous degree* has been estimated for each tunnel, due to its stability, in advance. A tunnel with a higher dangerous degree is considered to be more dangerous; that is, it is more probably to collapse. Kinglolo, the chief of Kulolo, would like to select some paths through the tunnels to Gulolo with little risk. In Kinglolo's opinion, the dangerous degree of a path is equal to the maximum dangerous degree of the tunnels in the path; and the dangerous degree of a set of paths is equal to the maximum dangerous degree of the paths in it. For example, consider Figure 1. The dangerous degrees of P_1, P_2, and P_3 are, respectively, 3, 5, and 6. And, the dangerous degree of $\{P_2, P_3\}$ is 6.

Since all tunnels are narrow, a limited quantity of resources can be transported along a path in one day. Therefore, every day, depending on the amount of resources needed to be transported, a different number, say k, of paths is required. Moreover, to avoid congestion, these k selected paths cannot pass any rest station in common. For example, in Figure 1, P_2 and P_3

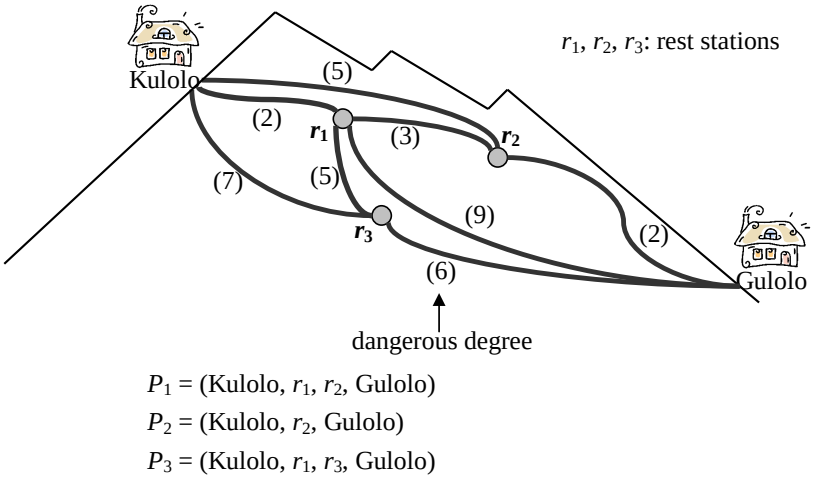


Figure 1: An example.

can be selected at the same time; however, P_1 and P_2 cannot be selected at the same time, since they both pass r_2. Kulolo has a higher altitude than all rest stations while Gulolo has a lower altitude than all rest stations. Kinglolo is a thoughtful chief. It is ensured that the altitudes of the rest stations on each selected path are non-increasing, so that the path is more suitable for transportation. For example, in Figure 1, the path (Kulolo, r_3, r_1, Gulolo) will never be selected, since r_1 is higher than r_3.

People in Kulolo believe that Kinglolo is the most brilliant man in the world, since he always selects a set of k paths that is as little dangerous as possible (i.e., the maximum dangerous degree of the selected paths is minimized). Now, given the data of the constructed tunnels, you are asked to find k paths that Kinglolo may select. In summary, the k selected paths, if exist, should satisfy the following:

1. all paths are from Kulolo to Gulolo,
2. no two paths pass the same rest station,
3. the altitudes of the rest stations on each path are non-increasing, and
4. the maximum dangerous degree of the paths is minimized.

For simplicity, only the maximum dangerous degree of the selected paths should be reported.

Technical Specification

1. The number of rest stations, n: $0 < n \leq 200$.
2. The number of tunnels, t: $t > 0$.
3. The dangerous degree of a tunnel, d: $1 \leq d \leq 100000$.
4. The number of paths which should be selected, k: $1 \leq k \leq 10$.

Input Format

The input consists of multiple test cases. The first line of each case contains a positive integer n ($0 < n \leq 200$) which indicates that there are n rest stations r_1, r_2, ..., r_n. For ease of description, Kulolo and Gulolo are denoted by r_0 and r_{n+1}, respectively. We assume that r_i is higher than r_j for any $0 \leq i < j \leq n + 1$. The second line of each case contains a positive integer t ($t > 0$) that specifies the number of tunnels. Each of the following t lines contains three integers p, q, d ($0 \leq p \leq n + 1$, $0 \leq q \leq n + 1$, $p \neq q$, $1 \leq d \leq 100000$) separated by white space, which indicate there is a tunnel with dangerous degree d connecting r_p and r_q. Then, a line containing a positive integer k ($1 \leq k \leq 10$) is provided, which is the number of paths that should be selected. You can assume that there is at most one tunnel between any two rest stations. The last test case is followed by a line containing a zero.

Output Format

For each test case, print a line containing the test case number (beginning with 1) followed by the maximum dangerous degree of the k paths that Kinglolo may select. If the solution does not exist, print "no solution". Use the format of the sample output.

Sample Input

```
2
4
0 1 3
1 3 12
2 0 10
```

```
2 3 5
1
1
2
0 1 5
1 2 6
2
3
2
0 1 5
3 4 7
1
3
6
0 1 8
0 2 12
0 3 15
3 1 9
3 4 8
2 4 12
2
0
```

Sample Output

```
Case 1: 10
Case 2: no solution
Case 3: no solution
Case 4: 12
```

Problem B
Fortune Card Game

Input file: *pb.in*
Time limit: 3 seconds

Problem Description

A popular card game called "fortune" is getting popular in country X. Fig. 1 shows one of the cards. In each card, a positive integer number (20 in the figure) is listed as the address of the card. A symbol is drawn beside the address. There are five kinds of symbols, which are listed below the card. For convenience, let each symbol be represented by an English letter from 'A'-'E'. The bottom of a card contains another number called "next fortune number."

Figure 1: A sample fortune card and symbols.

In a set of fortune cards, many cards can have same address; that is, address 20 is not limited to appear only in one card. However, there will be

no cards that are identical, i.e., no two cards with same address, symbol, and next fortune number.

The fortune card game is played as follows. A player starts with cards that have address 1. The goal of the game is trying to complete a "spell" that is composed by the symbols. For example, let a spell be "BADDAD". In the first move, the player will look for cards that have address 1 with a star symbol (which matches 'B' in the spell). The next fortune numbers of these cards are the new addresses for the next move. The player can select one card to advance to a new address x. The selected card is then put back to the cards for next move but the fortune number is written down.

Let the example be continued. In the next move, the player needs to look for the cards that have new address x with the cross symbol (which matches the second 'A' in the spell). Again, the player selects one card to advance to another new address. This procedure continues until the spell "BADDAD" is completed. Once the player completes a spell, he wins a score by adding all the next fortune numbers of the selected card, which have been written down.

Given a spell and a set of fortune cards, please output the maximum score that can be played in this card game.

Technical Specification

1. N - the number of test cases, $N <= 10$.
2. C - the number of cards, $C <= 800$.
3. L - the length of a spell, $L <= 150$.

Input Format

Test data begins with an integer N which is the number of test cases. Each test case begins with an integer C, which is the number of cards. Following the number C is C lines of card information. Each card is represented by ($Address\ Symbol\ NextFortuneNumber$). The address and next fortune number are between 1 and 800. The symbols are capital letters from 'A' to 'E'. The last line of a test case is a spell. The spell is a string composed by capital letters from 'A' to 'E'. The length of the spell (L) is less than 150.

Output Format

For each test case, please output the maximum score that can be collected for each test case.

Sample Input

```
2
7
1 A 2
1 A 3
2 A 3
2 B 4
2 B 5
3 A 3
3 B 4
AAAAB
6
1 A 2
1 B 2
1 A 3
1 B 3
2 A 3
2 B 3
AB
```

Sample Output

```
16
5
```

Problem C
Trip Planning

Input File: *pc.in*
Time Limit: 1 second

Problem Description

You are going on a long trip. Initially, you stay at hotel 0. Along the way, there are n hotels. The only place you are allowed to stop are at these hotels. The distance from hotel $i-1$ to hotel i is a_i. You can choose which of the hotels you stop at. You must stop at the final hotel, which is your destination.

You would ideally like to travel 100 kilometers a day. However, this may not be possible. It depends on the spacing of the hotels. There is no limit on the distance you traveled in a day. If you travel x kilometers during a day, the *penalty* for that day is $(x-100)^2$. You want to plan your trip so as to minimize the total penalty—that is, the sum, over all travel days, of the daily penalty. Write a program to determine the optimal sequence of hotels at which to stop.

Input Format

The input file contains a set of test data. Each test data consists of two parts. The first part is the number of hotels n. The second part is a sequence of n integers $a_1,\ a_2,\ \ldots,\ a_n$. Each a_i is the distance between hotel $i-1$ and hotel i. Assume that $0 < a_i < 200$. They may be written in many lines. Assume that $n < 1000$, and $n = 0$ signals the end of the test data.

Output Format

The first line of the output is the minimum penalty p. The second line of the output is the indexes of the hotels to stop at. If the solution is not unique,

print the one with fewer stops. If there are more then one solutions with the same number of stops, print the one which is the lexicographically smallest one. For example (1 2 4) < (1 3 4). Print 30 stops in each line, except the last line which may contain less stops.

Sample Input

```
10
11 48 28 87 35 86 37 83 16 34
20
81 49 50 87 107 20 40 84 60 47 29 30 35 47 108 41 85 106 77 106
0
```

Sample Output

```
p=2271
 0 3 5 7 10

p=4617
 0 1 3 4 6 8 11 14 15 17 18 19 20
```

Problem D
Road Networks

Input file: *pd.in*
Time limit: 2 seconds

Problem Description

There is a road network comprised by M roads and N cities. For convenience, we use $\{1, 2, \ldots, N\}$ to denote the N cities. Each road between two cities i and j, where $1 \leq i, j \leq N$ and $i \neq j$, has two types: One type is *bidirectional*, which allows a citizen to drive a car from i to j (denoted by $i \rightsquigarrow j$) and from j to i (denoted by $j \rightsquigarrow i$). The other type is *unidirectional*, which allows a citizen to drive a car following exactly one direction, either from i to j or from j to i.

We say that City j is *reachable from* City i if one can drive a car from i to j, visiting a sequence of cities $c_1, c_2, \ldots, c_k$ for $k \geq 0$, such that $i \rightsquigarrow c_1 \rightsquigarrow c_2 \rightsquigarrow \cdots \rightsquigarrow c_k \rightsquigarrow j$. (Every city is always reachable from itself.) A *region* is a *maximal* set of cities so that the following *mutually reachable property* holds: for two arbitrary cities i and j, i is reachable from j and j is also reachable from i. The adjective "maximal" means that if we include any other city in the given region, the mutually reachable property cannot be retained. Given a road network, your task is to write a computer program to compute the number of regions in the road network.

Technical Specification

1. We use $\{1, 2, \ldots, N\}$ to denote the N cities.
2. $M \leq 2000$ is a non-negative integer
3. $N \leq 1000$ is a positive integer.

4. If a road between i and j is bidirectional, then we use two order pairs (i, j) and (j, i) to represent it. Otherwise, if a road between i and j is unidirectional from i to j (respectively, j to i), we use (i, j) (respectively, (j, i)) to represent it.

Input Format

The input consists of a number of test cases. The first line of the input file contains an integer indicating the number of test cases to follow. Each test case consists of a road network, which has the following format: the first line of each test case contains two numbers, N and M, separated by a single space. The next M lines contain the description of M roads such that one line contains two cities representing an order pair (i, j). Each line is represented by two positive numbers separated by a single space; the first number representing the former element in the order pair and the second number representing the latter element in the order pair. A 0 at the $(M+2)$th line of each test case (except for the last test case) indicates the end of this test case.

The next test case starts after the previous ending symbol 0. Finally, a -1 signals the end of the whole inputs.

Output Format

The output contains one line for each test case. Each line contains an integer, which is the number of the regions in the given road network.

Sample Input

```
2
3 2
1 2
1 3
0
3 3
1 2
2 3
3 1
```

-1

Sample Output

3
1

Problem E
Early-Morning Pickup

Input File: *pe.in*
Time Limit: 10 seconds

Problem Description

Accelerated Computer Maintenance (ACM) is a company that aims at fixing customers' home computers as soon as possible. While other companies ask the customer to bring her/his computer to the maintenance station by herself/himself, ACM implements a rapid pickup system. After receiving the service request from the customer during work hours, ACM immediately sends out a service representative from a branch office. The representative would then go to the customer's house, pick up the computer, and drop the computer at one of the maintenance stations. ACM's customers are very satisfied with such a rapid service, and thus ACM's revenue keeps increasing.

Seeing the success story of ACM, many other companies are starting to implement their own rapid pickup systems as well, and ACM faces a strong competition from those companies. Thus, ACM decides to improve its rapid pickup system by including early-morning pickup routes. That is, after gathering the service requests from customers during the night, ACM asks some of its service representatives to complete the pickup tasks on their ways to work. To share the loads, each representative only needs to pick up at most one computer. If a representative is selected to stop by a customer's house for the morning pickup task, she/he would start from her/his own house, and orderly go to the customer's house, one of the maintenance stations, and her/his office. Otherwise, she/he directly goes to the office.

To implement such a system, ACM models the city as a network. The representatives' houses, the customers' houses, the maintenance stations, and the branch offices are all parts of the nodes of the network. When two nodes A and B are connected by a road, ACM computes the time needed to travel

from A to B in minutes. ACM, as its name suggests, is a company that highly values efficiency. Thus, it would like to implement a system that directs its representatives to arrive at the office as early as possible. Assume that all representatives leave their houses on `08:00` in the morning. Could you compute the minimum average time that all the representatives would arrive at the office while finishing all the morning pickup tasks?

Technical Specification

1. The number of nodes V is no more than 512; the nodes are labeled by $\{1, 2, \ldots, V\}$; the number of edges E is no more than $\frac{V(V-1)}{2}$.

2. The number of representatives R, the number of customers C, and the number of maintanance stations M are all positive integers. In addition, $C \leq R$.

3. The representatives' houses, the customers' houses, the maintenance stations, and the offices are all different nodes within the network.

4. There is at least one route (linked roads) that connects any of the two maintenance stations (so they can share tools).

5. For each customer's house, there is at least one route that connects it to one of the representative's houses.

6. For each customer's house, there is at least one route that connects it to one of the maintenance stations.

7. For each representative's house, there is at least one route that connects it to her/his office.

8. Every road is bi-directional. That is, a road between A and B can be used to travel both from node A to node B and from node B to node A using the same number of minutes T, which is a positive integer and is no more than 16.

Input Format

The first line of the input file contains an integer indicating the number of test cases to follow. The first line of each test case contains two integers `V`

`E` separated by a space. Each of the following E lines would contain three integers `A B T` separated by spaces, which indicates that nodes A and B are connected by a road with a T-minutes traveling time.

After those E lines, there would be a line with a single integer `R`. Each of the following R lines contains two integers `r d`, where r is the node of the representative's house, and d is the node of the representative's office.

After those R lines, there would be a line with a single integer `C`. The next line contains C integers `c`, each representing the node of a customer's house. In the next line, there is a single integer `M` indicating the number of maintenance stations. Then, the final line contains M integers `m`, each representing the node of a maintenance station.

Output Format

For each test case, output the minimum average time that all the representatives can arrive at the offices while finishing all the pickup tasks. The time is represented by the hour, a colon, and the minute. The hour part would be either `08`, `09`, `10`, or `11`; the minute part is a zero-padded two-digit integer within $\{\texttt{00}, \texttt{01}, \cdots, \texttt{58}, \texttt{59}\}$. The average time would be rounded up to the closest integer minute above. For example, if the average number of minutes is 25.03, it is rounded up to 26.

Sample Input

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

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

Sample Output

```
08:08
08:14
```

Problem F
Message

Input File: *pf.in*
Time Limit: 5 seconds

Problem Description

Astronomers have been looking for living beings in the outer planets. Recently the Hobble telescope has picked up binary (black and white) images from the Zkrets planet (1 light-years away). Each image contains exactly one symbol. Together the sequence of symbols (images) represents a message for aliens from other planets. A team of deciphering experts has found some decoding scheme that assigns an English alphabet to each symbol. The decoding process would be easy if each received image were in perfect condition. Unfortunately, due to long distance transmission, the images received usually contain some noise, meaning that some white pixels are turned into black pixels and vise versa. Furthermore, due to self-rotation of the Hobble telescope, the images, when captured, might be rotated 0, 90, 180 or 270 degrees. To expedite the message decoding process, please write a program to decode the received sequence of images automatically.

Technical Specification

1. The number of images received is n, where $1 \leq n \leq 100$. The size of the image is always scaled to 10x10. Within the image, '1' represents a black pixel (part of the symbol) and '0' represents a white pixel (not part of the symbol).

2. Each received image may contain at most 20% noise. This means that at most 20 pixels may have their gray levels changed (from black to white or from white to black) during transmission. Each image is rotated 0, 90, 180 or 270 degrees in clockwise direction.

3. The number of matching symbols and English Alphabets is m, where $1 \leq m \leq 52$. The matching English Alphabets can be either upper-case or lower-case letters.

Input Format

There are two parts of input, one follows the other. For the first part, it contains the matching English alphabets and image symbols. Thus, the first line contains an integer m, the number of matching alphabets and symbols. The next $11m$ lines define the m sets of matching alphabets and symbols. The first line of each set contains a single unique English letter. The next 10 lines give the matching symbol in a perfect (no noise, no rotation) 10x10 image of '0's and '1's. For the second part, it contains a sequence of received images. The first line contains an integer n, the number of 10x10 images received. The next $10n$ lines present the n images received. Every 10 lines define one image. Each line contains exactly 10 consecutive '0's and '1's.

Output Format

Please print out the corresponding English letters of the received input images in sequence, all on one line. If more than one match is possible for a given input image, output the matching one with the least number of noise pixels. If there is still a tie, output the one that appears first in the input sequence of English alphabets.

Sample Input

Sample Input	Explanations
2	There are 2 deciphered matching codes.
a	The first is coded as letter a.
0000000000	The corresponding image is giving in
0111111110	10 rows of consecutive '0's and '1's.
0111110011	
0111111100	
0000110000	
0000110000	
0001111100	
0011111110	
0000000000	
0000000000	
X	The second image is coded as letter X.
0110000110	The corresponding image is giving in
0110000110	10 rows of consecutive 0s and 1s.
0011001100	
0011001100	
0001111000	
0001111000	
0011001100	
0011001100	
0110000110	
0110000110	
3	There are 3 received images waiting to be decoded.
0110000110	The first received image has 10 rows of consecutive
0110000110	'0's and '1's. It matches with the corresponding image
0011001100	of the coded symbol 'X' perfectly without any noise.
0011001100	The image is either not rotated or perhaps rotated
0001111000	by 180 degrees.
0001111000	
0011001100	
0011001100	
0110000110	
0110000110	

0000000000 1100000011 1111001111 0011111100 0000110000 0000110000 0011111100 1111001111 1100000011 0000000000	The second image has also 10 rows of consecutive '0's and '1's. It also matches with the corresponding image of the coded symbol 'X' perfectly without any noise. However, the image is either rotated by 90 or 270 degrees.
0000000000 0000001110 1110001110 0000110001 0011111110 0011111110 0011001010 0011001010 0010000110 0000000010	The third image has also 10 rows of consecutive '0's and '1's. It contains 12 noise pixels. The image is also rotated by 90 degrees. It matches with the corresponding image of the coded symbol 'a'.

Sample Output

```
XXa
```

Problem G
Lost Treasure

Input File: *pg.in*
Time Limit: 3 seconds

Problem Description

There is an old legend that there is a number of lost ships lay on the seafloor of the Taiwan Straight. Among them, the ship *General II* is one of the most famous ships, and people believe that *General II* was sunk with a tremendous amount of treasures. In the past few decades, tens of groups have tried to raise the treasures from *General II*, but all of the attempts are failed. The reason is simply because those treasures are too fragile to tolerate any impacts in the raising, and unfortunately there are many obstacles scattered inside the ship such that raising a treasure from *General II* becomes an extremely challenging task.

Being a senior underwater-raising analyst of the International Organization of Underwater Archaeology (IOUA), you are now assigned to raise an important treasure from *General II*. Before that, the Underwater Scene Investigation Department (USID) of IOUA will investigate the detail information (e.g., the locations, sizes, and shapes of the obstacles) of *General II*, as well as the information (e.g., location, size, and shape) of the treasure. Then, they will give you a ship map of *General II* that can help you raise the treasure. Figure 1 shows an example of the ship map.

In the ship map, the obstacles are represented by K non-overlapping rectangles that are filled with the grid pattern, and the treasure is composed of R connected rectangles that are colored in black. A destination location for the treasure is also given on the map (filled with the white color). The destination location is an exit of *General II*, and it is very important to move the treasure from its initial location to the destination location without conflicting with any obstacles. Once a treasure is moved to the destination location, it can be raised easily to the sea level through the exit then.

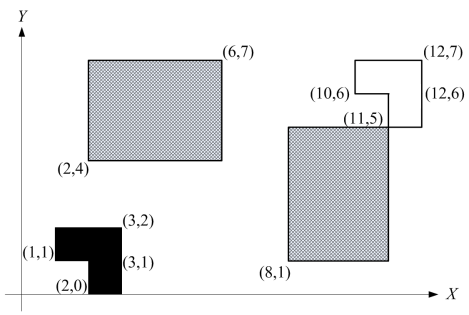


Figure 1: The sample ship map.

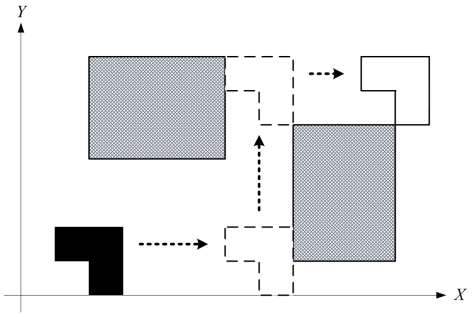


Figure 2: The sample solution.

Note that, the R treasure rectangles must be moved altogether without rotation and/or flipping; moreover, the treasure must be moved only in one of the four directions: upward, downward, leftward, and rightward. The cost of the movement in each direction is:

1. The cost for moving the treasure upward is \$10 per meter.
2. The cost for moving the treasure downward is \$2 per meter.
3. The cost for moving the treasure leftward is \$5 per meter.
4. The cost for moving the treasure rightward is \$5 per meter.

For simplicity, in this problem, we only consider a 2D map that uses the rectangular coordinate system (one meter per unit in both x and y axis), and the origin of the rectangular coordinate system is on the left-bottom corner of the map. In addition, each obstacle/treasure rectangle is described by its left-bottom vertex and its right-top vertex; and the destination of the treasure is given by the position of the left-bottom vertex of the first treasure rectangle. For instance in Figure 1, there are two obstacles: (2,4)-(6,7) and (8,1)-(11,5), and the treasure is a combination of the two rectangles: (1,1)-(3,2) and (2,0)-(3,1). The destination is indicated by (10,6), which is the destined position of the left-bottom vertex of the first treasure rectangle.

Figure 2 illustrates a sample solution of this example. In the solution, the treasure is moved rightward 5 meters first, and then moved upward 5 meters. Finally, it is moved rightward 4 meters to arrive the destination location. Therefore, the total cost for the movement is $5 \times 5 + 5 \times 10 + 4 \times 5 = 95$. Note that, it is likely to have multiple solutions that has the same minimum cost.

Technical Specification

1. M is an integer, and $10 \leq M \leq 1,000,000$.
2. N is an integer, and $10 \leq N \leq 1,000,000$.
3. K is an integer, and $1 \leq K \leq 50$.
4. R is an integer, and $1 \leq R \leq 5$.
5. None of the obstacles are overlapped.
6. All of the obstacle/treasure rectangles are within the ship map.
7. The treasure can not be moved (even partially) outside the map.
8. The initial and destination locations of the treasure are different and not overlapped with any obstacles.
9. The minimum cost for moving the treasure from the initial location to the destination is guaranteed to be less than $2,000,000,000.

Input File Format

The first line of the input file contains an integer indicating the number of test cases to follow. For each test case, the first line contains two positive integers, M and N, representing the width and height of the map (i.e., maximum x and y coordinates) respectively.

The third line is a positive integer K representing the number of obstacle rectangles on the map. In the following K lines, each line contains four non-negative integers: x_1, y_1, x_2, and y_2, where (x_1, y_1) represents the position of the left-bottom vertex of the rectangle, and (x_2, y_2) represents the position of the right-top vertex of the rectangle.

The next line is a positive integer R representing the number of treasure rectangles on the map. In the following R lines, each line contains four positive integers: x_3, y_3, x_4, and y_4, where (x_3, y_3) represents the position of the left-bottom vertex of the rectangle, and (x_4, y_4) represents the position of the right-top vertex of the rectangle.

The last line contains two non-negative integers, x_5 and y_5, which represents the destined position of the left-bottom vertex of the first treasure rectangle.

Output Format

Please output one number in one line for each test case. The number represents the minimum cost to move the treasure from the initial location to the destination location. If there is no solution that can move the treasure to the destination location without conflicting with any obstacles, please output 0.

Sample Input 1

```
1
100 100
2
2 4 6 7
8 1 11 5
2
1 1 3 2
2 0 3 1
10 6
```

Sample Output for the Sample Input 1

```
95
```

Sample Input 2

```
1
12 8
2
2 4 6 7
7 1 10 5
2
1 1 3 2
2 0 3 1
9 6
```

Sample Output for the Sample Input 2

```
0
```

Problem H
Space Elevator

Input File: *ph.in*
Time Limit: 5 seconds

Problem Description

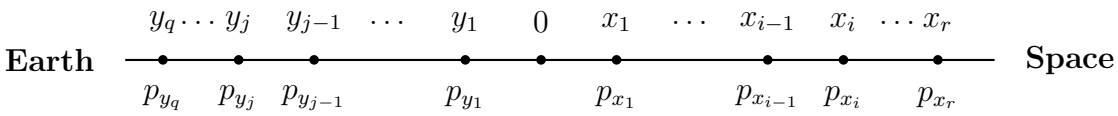
Seen as an engineering undertaking for the opening decades of the 21st century, the space elevator proposal was highlighted during the 2002 Space and Robotics Conferences. In 2108, after a sequence of technology breakthroughs, the dream of traveling from Earth to outer space with an elevator finally comes to reality. A space elevator made of a *carbon nanotubes composite ribbon* anchored to an offshore sea platform stretches to a small counterweight approximately 62,000 miles (100,000 km) into space. Mechanical lifters attached to the ribbon could then climb the ribbon, carrying cargo and humans into space. Carbon nanotubes have the potential to be 100 times stronger than steel and are as flexible as plastic. The strength of carbon nanotubes comes from their unique structure, which resembles soccer balls.

To better understand the concept of a space elevator, think of the game tetherball in which a rope is attached at one end to a pole and at the other to a ball. In this analogy, the rope is the carbon nanotubes composite ribbon, the pole is the Earth and the ball is the counterweight. Now, imagine the ball is placed in perpetual spin around the pole, so fast that it keeps the rope taut. This is the general idea of the space elevator. The counterweight spins around the Earth, keeping the cable straight and allowing the robotic lifters to ride up and down the ribbon.

Air Climber Motor (ACM), a space-infrastucture company, has been among those who support construction of a space elevator. ACM provides various of services to maintain the functionality of the space elevator. A crew of ACM routinely travels along the ribbon up and down at unit speed to serve the tourists and scientists at various stops, which are set up for spectacular sightseeing or scientific activities. ACM makes profits out of serving

the requests at the stops. If the crew serves stop x at time t, then the revenue collected equals $p_x - t$, where we assume the service can be done instantly and ignore the service time.

We use the following diagram to illustrate the space elevator with a horizontal line, where x_i's and y_j's are positive integers (except $x_0 = y_0 = 0$ as the origin) to indicate the locations of stops and p_{x_i}'s, p_{y_j}'s are the profits for the corresponding services at the stops. Initially, the crew starts at the origin. We refer to the stops right of the origin as $x_1, \ldots, x_r$, and those to the left as $y_1, \ldots, y_q$. Both sequences are in increasing order. The goal is



to select stops to serve (i.e., not every client needs to be served) and to find a route for the crew such that the total profits of the served stops minus the latency of the corresponding route is maximized. Since the length of the space elevator is so long and the number of stops is large, you are asked to write a program to help the crew select the stops to serve and maximize the revenue. (You see. This is yet another example that no matter how technology advances, the quest for programmer's help never ends!)

Technical Specification

1. All the numbers are non-negative integers.
2. n: the number of test cases. $n \geq 10$.
3. r, s: the number of requests to the right and left of the origin, respectively. $1 \leq r, s \leq 100$.
4. $1 \leq x_i$, y_j, p_{x_i}, $p_{y_j} \leq 1000$, for all $1 \leq i \leq r$ and $1 \leq j \leq s$.
5. $x_0 = y_0 = 0$, $p_{x_0} = p_{y_0} = 0$.

Input File Format

The first line of the input file contains an integer n indicating the number of test cases to follow. Each test case starts with two positive integers r and q, which are at most 100. In each of the following $r + q$ lines, there are two positive integers, where the first one is the position of a request and the second one indicates the corresponding profit. The first r lines are for x_i's and the following q lines are for y_j's. The range of x_i's, y_j's and the profits is from 1 to 1000. Note that x_i's may not be sorted in the corresponding r lines and likewise for y_j's. Each test case ends with a "0" in a separate line.

Output Format

For each test case, output the maximum profit in a separate line.

Sample Input

```
2
2  1
1  1
2  5
1  4
0
3  3
3  1
1  2
2  3
2  2
3  3
1  1
0
```

Sample Output for the Sample Input

```
Case 1: 4
Case 2: 2
```

Problem I
Finding The Heaviest Path

Input File: *pi.in*
Time Limit: 3 second

Problem Description

Rooted trees are one of the most frequently used data structures. The type of rooted tree with an weight assigned on each node is called a *weighted rooted tree¿* This problem asks you to find a path from the root to a leaf with a given property in a weighted rooted tree where each node has a positive integer weight.

Assume the nodes of the input rooted tree are numbered from 0 to $n-1$, $0 \leq n < 10000$. Note that each node is assigned with a distinct node number. Hence there are a total of n nodes in the input tree. Each node has a weight that is a positive integer within w, $1 \leq w \leq 10000$. Given a node u in the rooted tree T, let T_u be the subtree of T rooted at u. That is, T_u consists of u and the nodes whose ancestor is u. We define $weight(T_u)$ to be the sum of the weights for all the nodes in T_u. The *heaviest path* P_w in a weight rooted tree T is defined as follows.

- The root of T is in P_w.
- If a node u is in P_w and u is not a leaf, then one child v of u is also in P_w, where
  - $weight(T_v) \geq weight(T_s)$ for each child s of u that is not v
  - if $weight(T_v) = weight(T_s)$ for a child s of u that is not v, then the node number of v is larger than the node number of s.

It is cleared that given a weighted rooted tree, its heaviest path is unique. Your task is to find such a path by listing the nodes in the path from the root in sequence.

Technical Specification

1. $0 \leq n \leq 9999$
2. $1 \leq w \leq 10000$

Input File Format

The first line of the input file contains an integer indicating the number of test cases to follow. Each test case consists of $n + 1$ lines. The first line consists of n and the node number of the root. The $(i + 2)$-th line describes the information for the node numbered i which are its node weight, the number of children and the children listed one by one. A node without children is a leaf.

Output Format

For each test case, first output a line with the total weight of the nodes in the heaviest path. Then start a new line to output the node numbers in the heaviest path from top to the bottom starting from the root. There is exactly one space between two numbers. Output exactly 10 numbers in a one line with the remaining less-than-10 numbers, if any, at the last line.

Sample Input 1

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

```
8 0
```

Sample Output for the Sample Input 1

```
12
0 2 4
12
0 2 5
```

Sample Input 2

```
1
13 0
100 1 1
100 1 6
20 3 3 4 5
24 0
25 0
23 0
1 1 7
1 1 8
1 1 9
1 1 10
1 1 11
1 1 12
1 1 2
```

Sample Output for the Sample Input 2

```
252
0 1 6 7 8 9 10 11 12 2
4
```

Problem J
Bonus Adjustment

Input File: *pj.in*
Time Limit: 2 second

Problem Description

A bonus plan is one of the most effective tools to motivate employees, therefore most companies take it seriously to decide the amount of the bonus for each employee. BonBonus is a company striving on automatic bonuses calculation software and serves many international companies. For international companies, each employee has two reporting lines, one for the regional office and the other for the functional head. And each employee's performance is based on the evaluations of the regional office and the functional head. For example, Mr. Doe is working in Taiwan office as a marketing manager, then his bonus is decided by the evaluations of Taiwan office and the marketing division in the head quarter.

Let us skip all the complicated details of the bonus calculation. The final output of the BonBonus program is a two dimensional $m \times n$ table, where each row corresponds to a region and each column corresponds to a functional coding, and the entry in $a_{i,j}$ is the bonus for the employee in the region i and for the function j. As usual, there is no negative bonus. Naturally, the row sums and column sums are the performance for regions and functions. Companies usually announce the performance of regions and functions while they keep the individual's bonus secret.

BonBonus was doing very well until a phone call brought the stormy bad news. A company ran the BonBonus program and happily announced the performance for each region and each function. Later, they found out that the entries in the table were not all integers, it had at most two digits after the decimal point. "Not every country has cents! And we have already announced the performance for regions and functions," shouted at the other

end of the phone line. "Didn't you notice the fractional number problem when you announced them?" asked the BonBonus chief technology officer. "The row sums and column sums are all integers!" replied with the mixture of anger and amusement. "How may I help you?" explored the CTO. "You make all the entries integers! Keep the row sums and column sums intact. And moreover, the difference between the final adjusted bonus and the true bonus must be strictly less than one! And you have 24 hours to fix the problem." That was the last sentence of the phone call. As you had expected, many companies do business in countries without cents. The phone calls rushed in and you, the guru of programming, was called in. "You fix the problem," demanded the CTO. "But, sir. Can it be fixed?" you asked. "If it cannot be fixed, then you have an even bigger problem," said the CTO. You remained silent as if you were pondering. "Fix it and you will get a bonus which will make you beyond happiness." "Okay, sir. I will try my best." You are to write a program to solve the problem above.

Technical Specification

1. Each company has at least 2 regions and functions and at most 50 regions and functions, i.e., $2 \leq n, m \leq 50$.

2. The entries for the original table are greater than or equal to zero and less than or equal to 100 with at most two digits after the decimal point, i.e., $0 \leq a_{i,j} \leq 100$.

3. There will be at most 10 test cases, at most two test cases with inputs in the range from 40 to 50 ($40 \leq n, m \leq 50$), and for the rest of the test cases, the inputs will be less than 20 ($2 \leq m, n \leq 20$).

Input Format

The first line of the input file contains an integer indicating the number of test cases to follow. The first line for each test case contains two integers m, n separated by a space and the following m lines contain n numbers each. And the j^{th} number in the i^{th} line is a_{ij}

Output Format

Output the adjusted bonuses for each case in order. Use a space to separate each column and a line to separate each row. Or output two characters "no" if no such adjustment exists for that case.

Sample Input

```
2
2 2
1.6 1.4
1.4 1.6
2 3
14.29 23.0 21.71
10.71 8.0 15.29
```

Sample Output

```
2 1
1 2
15 23 21
10 8 16
```