2016 ACM ICPC Asia Chung-Li Regional

November 26, 2016

- Problems: There are 13 problems (40 pages in all) in this packet.
- Program Input: Input to the program are through standard input. Program input may contain one or more test cases. Test cases may be separated by any delimiter as specified in the problem statements.
- Program Output: All output should be directed to the standard output (screen output).
- Time Limit: Judges will run each submitted program with certain time limit (given in the table below).

Problem Information

	Problem Name	Time Limit
Problem A	Every Runner is Sometimes Lonely	1 sec.
Problem B	Shooting a Smart Dart	1 sec.
Problem C	Cleaning Robot	1 sec.
Problem D	Maximum Investment Return	1 sec.
Problem E	The Maze Runner	1 sec.
Problem F	University Entrance Examination and Admission	1 sec.
Problem G	The Size of the Smallest Hole	1 sec.
Problem H	Perfect Matchings	1 sec.
Problem I	Graceful Labeling	1 sec.
Problem J	Priceless Treasure	2 sec.
Problem K	Lightbulbs	1 sec.
Problem L	Subsets	3 sec.
Problem M	Surviving Probability	2 sec.

Problem A
Every Runner is Sometimes Lonely
Time Limit: *1 Second*

Soon after 2016 Rio Olympics, marathon coach of ACM team started training his team members for 2020 Tokyo Olympics. An important part of the training program is to let them running at their comfortable speeds around the athletic track. After a few days, the coach observed that every runner is sometimes lonely, meaning that he or she is separated by a "large" distance from all other runners.

The coach's observation may be true. In order to do further study, the coach's observation can be rephrased as follows:

> Suppose there are $n > 1$ runners running on the circular track of circumference c meters. They all started at the same origin and running at different speeds. Then for each runner r, there is at least one time t_r for which that runner is *lonely*, in the sense that he or she is separated by a distance at least c/n meters from every other runner.

In this problem, we are going to verify a simple version of the above observation. Assume that there are two runners A and B. They started running from the same position, called *origin*, on the circular track of length c, at constant speeds v_1 and v_2, respectively. Furthermore, assume that v_1 and v_2 are integers, and $v_1 \neq v_2$. Note that the speeds v_1 and v_2 can be zero or negative. If the speed is positive, then the runner runs counter-clockwise around the track, otherwise he or she runs clockwise.

Write a program to calculate the position d from the origin for runner A such that the two runners are at $c/2$ meters apart. The position d is measured counter-clockwise along the circular track from the origin, and $0 \leq d < c$. Furthermore, A and B may be at $c/2$ meters apart many times. The output of your program should be the first time this condition occurs.

Input File Format

The test data may contain many test cases. Each test case contains three integers c, v_1, and v_2 in a line. The value of c and the absolute values of v_1 and v_2 are all less than 2^{31}. The last test case is followed by a line containing a single 0.

Output Format

The output for each test case is the distance d of the position of A from the origin when the first time A and B are separated $c/2$ meters apart. If the value of d is not an integer, then it must be rational. In this case, print it in q/p format, where p and q are integers, and $\gcd(p, q) = 1$. If the condition never occurs, then print `false`.

Sample Input

```
400 1 2
350 2 5
```

```
400 3 -7
0
```

Output for the Sample Input

```
200
350/3
60
```

Problem B
Shooting a Smart Dart
Time Limit: *1 Second*

In a dart game, before a game begins, players randomly shot many darts onto the dart board. No two darts share a same position. When the game begins, the game rule is that in each round after a player shoots a new dart, say onto position (x, y). The two old darts that are already on the dart board and closest to the new dart at (x, y) are used to help determine the score of the round. Let the positions of the two closest old darts be (x_1, y_1) and (x_2, y_2). The score is $\sqrt{(x-x_1)^2+(y-y_1)^2}+\sqrt{(x-x_2)^2+(y-y_2)^2}$, that is, the sum of Euclidean distances from (x, y) to the two closest old darts. To win the game, you need to minimize the score in each round.

Given the positions of old darts before the game, please compute the minimum score a player can get in the round.

Input Format

The first line of input contains an integer N, which is the number of test cases. For each test case, the first line contains an integer D, $(2 \leq D \leq 5000)$, which is the number of old darts on a dart board. Then for the next D lines, each line gives thd position of an old dart on the dart board, $(x\ y)$, where x and y are integers and separated by a whitespace, $0 \leq x, y \leq 2^{24}$. No two darts have the same position on the dart board.

Output Format

For each test case, please output the minimum score, truncated to the second decimal place. For example, 3.412 and 3.419 both would truncate to 3.41.

Sample Input

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

Output for the Sample Input

```
1.00
1.41
```

Problem C
Cleaning Robot
Time Limit: *1 Second*

A company is producing a cleaning robot that automatically moves around the house and wipes the floor. The robot can record its trajectory (i.e., the track of its movement), and the designer wishes to compute the total area wiped by the robot given its trajectory.

The robot is rectangular, with width w along the x axis and length l along the y axis. It can move only horizontally or vertically, and it cannot rotate. The trajectory of the robot is given by the track of its bottom left (the smallest x- and y- coordinate) corner. For example, consider Fig. 1 (a). The size of the robot is $(w, l) = (5, 3)$, and the robot moves along the track $\{(0, 0), (12, 0), (12, 6), (6, 6), (6, -2)\}$. The total area wiped by the robot is shown in Fig. 1 (b). It is easy to verify that the total area is 124. One possible way to compute the total area is $3 \times 6 + 9 \times 11 + 5 \times 2 - 3 \times 1 = 124$, which is illustrated in Fig. 2.

In this problem, you are asked to compute the total area wiped by the robot given its size and trajectory.

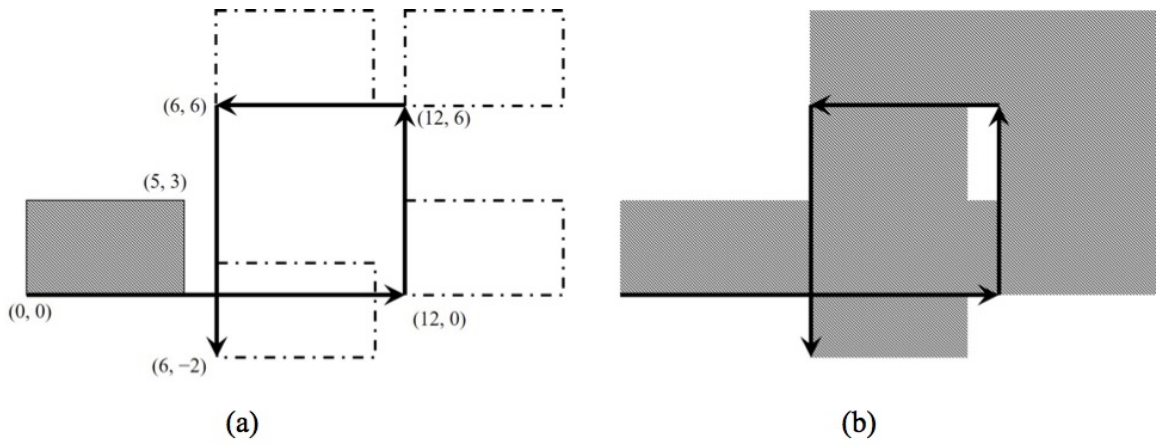


Figure 1. An example.

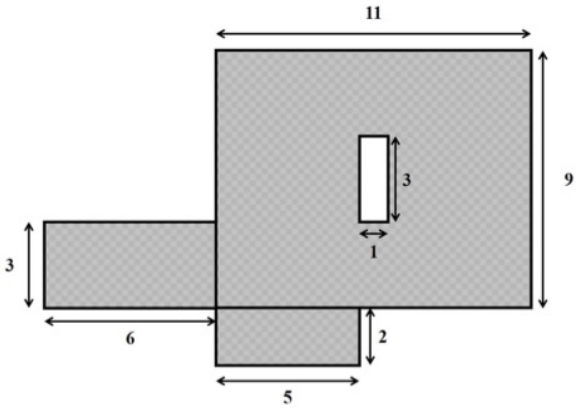


Figure 2.

Technical Specification

- The number of test cases is at most 10.

- The width w and length l are integers between 1 and 100.
- The number of points on the trajectory, n, is an integer between 2 and 500.
- The coordinate of the points x, y are integers between -10000 and 10000.

Input Format

There are at most 10 cases. The first line of each test case gives the three integers w, l, n, where $1 \leq w, l \leq 100$ and $2 \leq n \leq 500$. Then, n lines describing the trajectory are given. The ith line, $1 \leq i \leq n$, gives the x- and y-coordinates of the ith point on the trajectory, where x, y are integers and $-10000 \leq x, y \leq 10000$.

The input is terminated by a line containing three zeros, which should not be processed.

Output Format

For each case, output the total area wiped by the robot.

Sample Input

```
5 3 5
0 0
12 0
12 6
6 6
6 -2
2 3 5
3 -5
3 -25
3 -18
8 -18
8 -25
0 0 0
```

Output for the Sample Input

```
124
75
```

Problem D
Maximum Investment Return
Time Limit: *1 Second*

Joe works in a financial institute as an intern. His job is help making market prediction. The first exercise from his mentor is to learn when to buy and sell a stock to maximize the profit. Consider the historical stock prices of n-consecutive days of a stock, where the days are numbered $1, 2, \ldots, n$. Let $p[i]$ be the price of each share of the stock on day i. Given an integer k, Joe wants to find m pairs of days

$$(b_1, s_1), (b_2, s_2), \ldots, (b_m, s_m)$$

for some $m \leq k$ and $1 \leq b_1 < s_1 < b_2 < s_2 < \cdots < b_m < s_m \leq n$. For each pair (b_t, s_t), it means on day b_t Joe would buy 1000 shares of the stock and sell it on day s_t. Thus the profit would be simply

$$1000 \sum_{t=1}^{m} (p[s_t] - p[b_t]).$$

Given $k \leq n/2$ write a program to find the $m \leq k$ pairs of integers (b_t, s_t) to maximize the profit.

Technical Specification

- The number of test case is at most 20.
- n is the number of days, which is at most 1000.
- Each stock share price is at most 1000.

Input Format

The first line of the input gives the number of test cases, T (≤ 20). T test cases follow. Each test case consists of two lines, where the first has two positive integers n ($3 \leq n \leq 1000$) and k ($k \leq n/2$). n indicates the number of days. The second line has n positive integer $p[1], \ldots, p[n]$, indicating the stock share price of each day. Each $p[i]$ is at most 1000.

Output Format

For each test case, output the maximum profit.

Sample Input

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

```
1 1 1
5 2
1 2 3 1 5
```

Output for the Sample Input

```
2000
0
6000
```

Problem E
The Maze Runner
Time Limit: *1 Second*

One day, a space traveller wakes up and cannot remember his name or what had happened. He finds himself in the middle of an ever-changing maze. After wandering around the maze, he realizes there are other people just like him wandering around not remember anything. We call those people as *mazerunners*. The space traveller feels glad that there are others, but he quickly realized a cruel fact: there are limited resources in the maze, and he needs to get to certain one to ensure survival. He has to try snatching resource(s) before the others do. In other words, the key for survival depends on the speed for which he can reach the resource locations. By exploring the maze several times, he finds that even though the maze is constantly changing, these changes seem to follow certain rules. He organizes these rules as follows and draw the structure of the maze as in Figure 1:

- The maze consists of many corridors and intersections. Each corridor is a road that connects (runs between) two intersections.
- The size of the maze is measured in terms of the number of intersections. The basic maze contains 8 intersections. The maze may grow or shrink recursively. Each time the maze grows, the number of intersections will double. Hence the maze size may be 16, then 32, and so on.
- Each intersection in the maze is assigned a unique binary string with minimal number of bits as address. For example, the addresses of the intersections in an 8-intersection maze are 000, 001, 010, 011, 100, 101, 110, 111; and the addresses for a 16-intersection maze are 0000, 0001, 0010, ..., 1111. So a 32-intersection maze would require addresses of 5 bits long and so on and so forth.
- All intersections in a maze are classified as either belonging to the $M1$ section or the $M0$ section. An intersection belongs to the $M1$ section if its address begins with an 1-bit. If an intersection's address starts with a 0-bit then it belongs to the $M0$ section. Sections $M1$ and $M0$ each has a magic number S and T, respectively.
- Corridors between two intersections in the maze are defined by the following rules.
  - If two intersections have addresses that only differ on the first bit (*i.e.* all the other bits are the same), then the two intersections are connected by a corridor. For example in Figure 1, among the many corridors, there is a corridor between intersections 0000 and 1000 (bit 1 is differnt, but bits 2 through 4 are the same). Also there is a corridor between intersections 01011 and 11011.
  - For any two intersections in the $M1$ section, if their addresses only differ in a single bit among bits 2 through bit $S+1$, then the two intersections are connected by a corridor. For example in Figure 1 (top left), if $S = 1$, then intersections 1011 and 1111 are connected by a corridor because bit 2 through bit $1+1 = 2$ only has one bit that is differnt. In Figure 1 (bottom), if $S = 2$, then intersections 11001 and 11101 are connected by a corridor because bit 2 through bit $2 + 1 = 3$ only differ in one bit (*i.e.* bit 3).

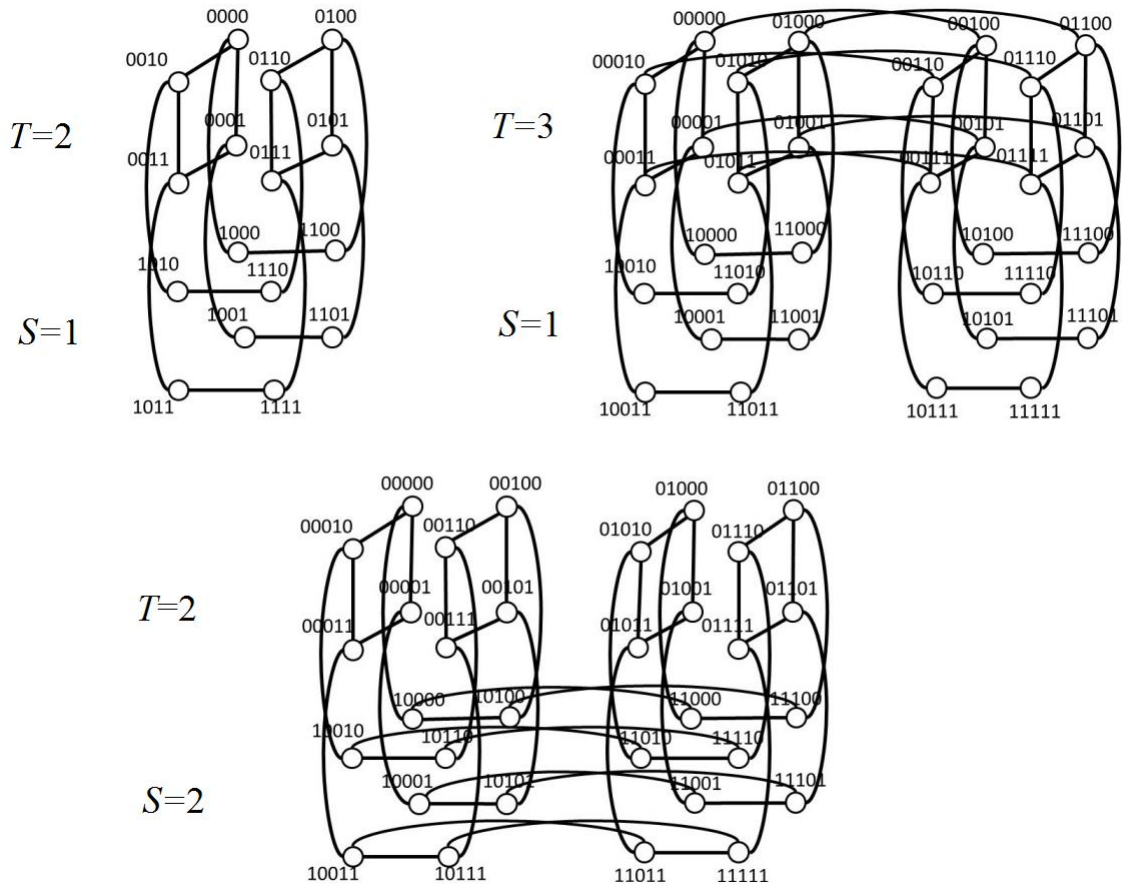


Figure 1: The Maze Runner

– For any two intersections in the $M0$ section, if their addresses only differ in a single bit among bits $S+2$ through bit $S+T+1$, then the two intersections are connected by a corridor. For example in Figure 1 (top left), if $S=1$, and $T=2$, then intersections 0010 and 0011 are connected by a corridor because bit 3 through bit $1+2+1=4$ only has one bit that is differnt. In Figure 1 (bottom), if $S=2$ and $T=2$, then intersections 01100 and 01101 are connected by a corridor because bit 4 through bit $2+2+1=5$ only differ in one bit (*i.e.* bit 5).

Given a maze and the locations (*i.e.* intersections in the maze) of a list of runners and resources, find the shortest path for each runner to reach the designated recources.

Technical Specification

- The magic number for $M1$ and $M0$ sections are S and T, respectively, where $1 \leq S, T \leq 200$. Note the maze must be a 2^{S+T+1}-intersection maze.
- There are k runners in the maze, $1 \leq k \leq 5$.
- The m^{th} intersections along a shortest path, $1 \leq m \leq 200$.

Input Format

The input contains several test cases. For each test case, the first line consists of the two magic numbers S and T. The second line contains a positive integer k. The next k lines, each has two binary strings and an integer m, all separated by whitespaces. The two binary strings are the locations of a runner and the designated resource for that runner. The number m is used for the output. The zero value of S and T indicate the end of test cases and should not be processed.

Output Format

For each test case, output k lines. On line i, first output the address of the m_{th} intersection along the shortest path for the i^{th} runner to reach its designated resource. If the shortest path contains less than m intersections, then output "`-1`". If there are more than one shortest path, chose the one with smallest lexical order. Then output the number of intersections along this shortest path (counting the starting and ending intersections). Output 10 dashes ("`----------`") on a single line after each test case.

Sample Input

```
2 2
3
00000 11111 2
10101 00100 3
01100 00100 4
1 3
2
00111 10010 5
01101 11101 10
1 2
3
1000 0100 2
1111 0000 5
1101 0010 3
0 0
```

Output for the Sample Input

```
00001 6
00100 3
00100 4
----------
-1 4
-1 2
```

```
----------
1100 3
0000 5
0001 5
----------
```

Problem F
University Entrance Examination and Admission
Time Limit: *1 Second*

The senior high school students of Utopia State must take *College Entrance Standardized Test* and apply to be admitted to deparment of their choice.

The standardized test that students must take consists of 5 subjects, namely Language, Mathematics, Science, History, and Geography. Each department places different emphasis on the 5 subjects, so weight are assigned for the scores on each of the 5 subjects. Thus, each department can rank the students based on the sum of weighted students' standardized test scores.

Due to resource constraints, there is a enrollment limit for each department. The admission procedure requires each student to fill out his/her department *preference list* for enrollment. The admissions office then decide on a *admission standard* by announcing the *admission score* for each department.

A student S is *eligible* for admission to department D if his/her sum of weighted test scores (according to the weights assigned by department D) is higher than or equal to the announced admission score for department D. Furthermore, the student S is *admitted* to the department D if the following two condition holds

1. The department D is on the preference list of student S.
2. The student S is eligible for D, but not eligible for all other departments listed before department D on his/her perference list.

An admission standard is *feasible* if

1. The admission of each student satisfied previous conditions.
2. The number of students admitted to each department does not exceed its enrollment limit. The *only* exception is that the department is requested to admit all students whose weighted scores exactly equal to the the announced admission score for that department.

Deciding admission standard presents problems for the applicants as well as the departments. As an example, let two departments, D_1 and D_2, both have their enrollment limit set to 1; and two students, say S_1 and S_2, both are applying for admission to departments D_1 and D_2. Furthermore, student S_1 prefers D_1 over D_2, while S_2 prefers D_2 over D_1.

Suppose the sum of weighted test scores of S_1 is lower than S_2's sum of weighted test scores under department D_1's weighting assignment; and the sum of weighted test scores of S_1 is higher than S_2's sum of weighted test scores under department D_2's weighting assignment. It is possible for the admission office set an admission standard such that only S_2 qualifies for department D_1 and only student S_1 qualifies for department D_2. Such an admission standard might be preferred by departments, so it is called the *department-preference standard*.

On the other hand, admission office could lower the admission standards of both departments so that both students are qualified. Thus, according to their preferences, student S_1 is admitted to department D_1, and student S_2 admitted to department D_2. Such an assignment is surely preferred by both students, so it is called the *student-preference standard*.

This suggests the following principle: other things being equal, students should receive consideration over deparments' consideration. That is, the student-preference admission standard will be used for admission.

Please implement the above student-preference admission standard and admit students based on their preference lists and test performances.

Technical Specification

1. D is the number of departments, $1 \leq D \leq 200$.
2. S is the number of students, $1 \leq S \leq 12000$.
3. P is the length of the preference list, $1 \leq P \leq 80$.
4. q is the enrollment limit of a department, $1 \leq q \leq 90$.
5. w_1, w_2, w_3, w_4, w_5 are weights for the test subjects, $0 \leq w_1, w_2, w_3, w_4, w_5 \leq 500$.
6. t_1, t_2, t_3, t_4, t_5 are student test scores, $0 \leq t_1, t_2, t_3, t_4, t_5 \leq 100$.

Input Format

For each test instance, the first line contains three integers D, S, and P, separated by blank spaces.

For the next D lines, line i contains six integers $q, w_1, w_2, w_3, w_4, w_5$ meaning department i has enrollment limit of q and has assigned test score weights of w_1, w_2, w_3, w_4 and w_5.

For the next S lines, line i contains five integers t_1, t_2, t_3, t_4, t_5, indicating the test scores of student i.

And for the final S lines, line i contains exactly P integers, $\langle d_1, d_2, \ldots, d_p \rangle$ indicating the preference list for student i. d_k is the department id and $1 \leq d_k \leq D$.

Note that input may contain more than one instances. The last instance is followed by a line containing a single 0.

Output Format

The outputs for each test instance is a single line with four integers:

1. The total number of students admitted.
2. The total number of students admitted to the first department on their preference list.
3. The total number of departments exceeding their enrollment limit due to weighted score ties.
4. The total number of departments that have admitted fewer number of students than their enrollment limit.

Sample Input

```
2 2 2
1 200 100 100 100 100
1 100 200 100 100 100
80 90 80 80 80
90 80 80 80 80
1 2
2 1
3 8 2
2 100 100 100 100 100
2 150 100 125 100 100
3 100 150 100 150 150
10 10 10 15 20
20 25 40 40 44
30 33 50 50 50
40 52 61 60 60
50 61 70 70 70
60 70 73 74 80
70 73 75 78 90
80 92 80 82 92
1 2
1 2
1 2
1 2
1 2
1 2
3 1
2 3
0
```

Output for the Sample Input

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

Problem G
The Size of the Smallest Hole
Time Limit: *1 Second*

A cycle in an undirected graph consists of a sequence of different vertices except the starting vertex and the ending vertex, with each two consecutive vertices in the sequence being adjacent to each other in the graph. The length of a cycle is the number of edges in the cycle. A hole in a graph is a cycle of length greater than or equal to 3 such that no two vertices of the cycle are connected by an edge that does not belong to the cycle. The size of a hole is the number of edges in the hole. You are asked to determine the size of the smallest hole of a undirected graph.

Technical Specification

- The number of vertices in any given undirected graph is n, $n \leq 100$.
- There are no parallel edges nor loops in any given graphs.

Input Format

In each problem instance, the first line contains the number of vertices, n, in a graph. The following n lines contain the adjacent list of the graph. That is, line i contains the adjacent vertices of vertex i. There is a whitespace between any two numbers in each line. There may be more than one problem instance. A line containing a single 0 indicates the end of test cases.

Output Format

For each test case, output the smallest hole size of the graph in a line. If there is no hole in the graph, then output 0 in a line.

Sample Input

```
5
2 4
1 3 4
2 4 5
1 2 3 5
3 4
4
2
1 3
2 4
3
8
```

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

Output for the Sample Input

```
3
0
5
```

Problem H
Perfect Matchings
Time Limit: *1 Second*

Given a graph G = (V,E), a matching in G is a set of pairwise non-adjacent edges; that is, no two edges share a common vertex. A perfect matching is a matching which matches all vertices of the graph. That is, every vertex of the graph is incident to exactly one edge of the matching. Finally, a bipartite graph G, is a graph whose vertices can be divided into two disjoint sets U and V such that every edge connects a vertex in U to one in V. In this task, please determine whether the number of perfect matchings of a bipartite graph is old or even. Note that counting the number of perfect matchings on a bipartite graph has been proven to be *#P-complete*. Thus, it would be better to avoid actually counting the number of perfect matchings.

Technical Specification

- The number of bipartite graphs is n, $n \leq 20$.
- Maximum number of vertices in a bipartite graph is v, $v \leq 20$ and v is an even number.
- Maximum number of edges in a bipartite graph is e, $e \leq 46$.

Input Format

The first line consists of a number n, which is the number of bipartite graphs to follow. The first line of each bipartite graph G contains two integers v and e, separated by a blank space. The v vertices are naturally numbered from 1, ..., v. One partition of the bipartite graph G contains vertices $1, \ldots, v/2$, while the other partition contains vertices $v/2+1, \ldots, v$. The next e lines each contains two integers v_i and v_j, separated by a blank space, indicating an edge going from vertex v_i to vertex v_j, where v_i and v_j belong to distinct partitions in the bipartite graph G.

Output Format

Please output 0 if there is an odd number of perfect matchings on the input graph, and output 1 otherwise.

Sample Input

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

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

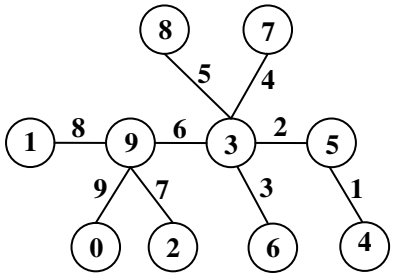
Output for the Sample Input

```
0
1
```

Problem I
Graceful Labeling
Time Limit: *1 Second*

In an ideal wireless network, each communication link is assigned with a unique channel number (a positive integer) so that interference can be minimized. A simple scheme to assign channel number for every link in the whole network is as follows. Given a network, each station in the network is first assigned a distinct integer as station ID. Every communication link in the network is then assigned a channel number that is equal to the absolute value of the difference between the station IDs of the two stations linked by this communication link.

In the graph representation of a wireless network below, each vertex represents a station and each edge indicates a communication link. Station IDs of 0 to 9 is assigned to the 10 vertices in the graph. The link's channel number can then be computed as described above. For example, the link between vertex with Station ID 3 and vertex with Station ID 7 has a channel number of $4(= 7 - 3)$. Note that in this particular network, all channel numbers are distinct.



A graph $G = (V, E)$ is called a *graceful graph* if the nodes in V can be numbered with distinct integers from 0 to $n - 1$ (where $n = |V|$) and the edges in E can be numbered with distinct integers from 1 to m (where $m = |E|$), such that the edge number equals the absolute difference of the two end vertex numbers. Such labeling of a graceful graph is called *graceful labeling* of graph G.

The channel assignment problem in a wireless network described above can be viewed as finding a graceful labeling for a given network (graph). In order for a graph to be graceful, G must be without loops or multiple edges. Unfortunately, not every graph has a graceful labeling. Graceful labelings are known to exist for some types of trees including caterpillar graphs. A caterpillar is a tree in which a single path (the spine) is incident to (or contains) every edge. That is, the vertices not on the spine of a caterpillar are leaves. The graph shown above is a caterpillar with a possible graceful labeling. Please write a program to find a graceful labeling of the given caterpillar.

Technical Specification

1. n, $2 \leq n \leq 30$, is the number of vertices in G.
2. m, is the number of edges in G.

Input Format

The first line of each test case contains an integer n, denoting the number of vertices in G. Assume the vertices are $v_1, v_2, ..., v_n$. The second line of a test case contains an integer m, denoting the number of edges in G. The next m lines each contains a pair of integers i and j (separated by a space), indicating there is an edge (communication link) between v_i and v_j. The last test case will be followed by a line containing a single 0.

Output Format

For each test case, output a single line of n distinct integers, (ranging from 0 to $n-1$, denoting the assigned station ID of $v_1, v_2, ..., v_n$, respectively, in a graceful labeling of the graph G. If there are more than one graceful labelings, output only one (any one) of them.

Sample Input

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

Output for the Sample Input

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

Problem J
Priceless Treasure
Time Limit: *2 Seconds*

Toward the end of Ming Dynasty, political corruptions and famine led to many revolutions. In January of 1644, Zicheng Li (a.k.a King Chuang), one of the leader of military troops, captured some land and founded Dashun kingdom. And then in April of 1644, he captured the capital of Ming Dynasty (Yanjing). At first, soldiers have good discipline. But after just a short while, soldiers began looting around. According the historians, some 37 million silver ingots and 10 million gold ingots were taken from the palace treasury. In today's market, those silver and gold ingots would have cost over US$800,000,000,000 dollars.

But in June 1644, Sangui Wu, and Qing army overthrown Dashun Kindom by defeating King Chuang at Yanjing. When fleeding Yanjing, King Chuang rounded up the treasures in an attempt to take the treasures with him to Xian. Unfortunately, before reaching Xian, Qing army had caught up with King Chuang. King Chuang decide to hide the treasures in mountains and rivers. King Chuang encoded the hiding place with a western proverb book and a sequence of numbers as key to the message. He then handed the proverb book to his four bodyguards and gave them four modified sequences of numbers (modified from the key). King Chuang told the bodyguards that Qing was hard to defeat now. But given time, when corruptions abound, our descendants should recover the hidden treasures and find Dashun Kingdom again.

Descendants of the four bodyguards aggregated together wanting to find the hidden treasures. A part of western proverb book and parts of the 4 sequences are given as follows:

Western proverbs:

```
the blood of the soldiers makes the glory of the general
yellow gold has its price learning is priceless
a letter from home is a priceless treasure
learning is a treasure that will follow its owner everywhere
all are not thieves that dogs bark at
...
...
...
```

Part of 4 sequences of numbers.

```
1 8 9 3 8 2 8 ...
1 5 9 8 8 2 8 ...
1 8 9 8 2 7 8 ...
6 1 9 8 8 2 8 ...
```

One of the bodyguard descents is a computer scientist. He had figured out that the longest common subsequence (LCS) of the 4 sequences is a vital cue to the hidden message. He found two LCS from the partial sequences of numbers given above, namely 1 8 8 2 8 $\cdots$ and 1 9 8 2 8 $\cdots$.

For 1 8 8 2 8, the following message can be constructed:
The 1^{st} word of the 1^{st} proverb is "the".
The 8^{th} word of the 2^{nd} proverb is "priceless".
The 8^{th} word of the 3^{rd} proverb is "treasure".

The 2^{nd} word of the 4^{th} proverb is "is".
The 8^{th} word of the 5^{th} proverb is "at".
The above gives the first 5 words of the hidden code.

But for 1 9 8 2 8 $\cdots$, it was found that there is no 9^{th} word in the 2^{nd} proverb. So 1 9 8 2 8 $\cdots$ cannot be a proper code and is discarded.

The descendants are very excited about this discovery. They now are eager to discover the entire message in order to find the hidden treasures. Please help descendants of the four bodyguards decipher the hidden message.

Technical Specification

1. M, $1 \leq M \leq 32$, is the number of proverbs.
2. The number of words in each proverb is at most 25.
3. Each proverb contains only lower alphabets and spaces.
4. N, $1 \leq N \leq 10$, is the number of test cases, each test case contains 4 sequences of numbers.
5. For each set of 4 sequences, there are at most 7 longest common subsequences.

Input Format

The first line is a number M indicating the number of proverbs. The next M lines contains the M proverbs, one proverb per line. Following the proverbs is a line containing a single number N indicating the number of test cases (number of sets of 4 sequences) to follow. For each test case, there are 4 lines of sequence of numbers, separated by space. Each sequence contains at most ~~M~~ 32 numbers ~~(positive integers less than 12)~~.

Output Format

For each test case (set of 4 sequences), output the number(=L) of possible original messages at first line. Then in the following L lines, output the possible original messages in lexicographical ordering. (A possible original message is formatted as a sequence of words separated by a single whitespace. Take a possible original message as a string of ASCII characters, and sort them in lexical graphic ordering.)

Sample Input

```
10
the blood of the soldiers makes the glory of the general
yellow gold has its price learning is priceless
a letter from home is a priceless treasure
learning is a treasure that will follow its owner everywhere
all are not thieves that dogs bark at
```

```
there is no wool so white but a dyer can make it black
one master in the house is enough
bees that have honey in their mouths have stings in their tails
a good fame is better than a good face
you can not burn the candle at both end
2
1 8 9 3 8 2 8 6 5 4
1 5 9 8 8 2 8 6 5
1 8 9 8 2 7 8 6 5 4
6 1 9 8 8 2 8 6 5
1 4 8 2 5 8 6 6 3 3 7 7 10 11
9 9 3 3 7 7 4 1 8 2 8 5 10
6 6 9 9 1 8 4 2 5 8 7 7 10 11
6 6 8 4 1 2 8 5 9 9 3 3 10
```

Output for the Sample Input

```
1
the priceless treasure is at white house
5
glory gold is everywhere
glory gold treasure everywhere
the gold is everywhere
the gold treasure everywhere
the priceless is everywhere
```

Problem K
Lightbulbs
Time Limit: *1 Second*

ACM amusement park has a new underground maze game. In the maze, there are many interconnecting tunnels. For safety reason, lightbulbs are placed so that there is exactly one lightbulb at each end of a tunnel.

There are n lightbulbs, numbered from 0 to $n-1$, and m tunnels, each connecting two distinct lightbulbs.

Suppose lightbulb i consumes w_i watts of power when it is on and zero watt when it is off. In a tunnel connecting lightbulbs i and j, the degree of visibility is

- $w_i + w_j$ if both lightbulbs i and j are on,
- w_i if lightbulb i is on and lightbulb j is off,
- w_j if lightbulb i is off and lightbulb j is on, and
- 0 if both lightbulbs i and j are off.

There do not exist two distinct tunnels connecting the same (unordered) pair of lightbulbs.

A tunnel connecting lightbulbs i and j is said to be underlit if its degree of visibility is less than its visibility threshold, $t_{i,j}$. For safety concerns, the number of underlit tunnels should not exceed a certain threshold, B. To save power, the total power consumption of the lightbulbs must be at most $\lceil \log_2 n \rceil$ watts.

In summary, we require that

- the number of underlit tunnels be at most B, and
- the sum of the power consumptions of the lightbulbs be no greater than $\lceil \log_2 n \rceil$ watts.

Please determine if it is possible to turn on a subset of the lightbulbs to simultaneously meet the two requirements above. Output the necessary information as specified in the Output Format section.

Technical Specification

1. $B \in \{0, 1\}$.
2. $n \leq 500$ is a positive integer.
3. $t_{i,j} \in \{1, 2, 3\}$ for all i, $j \in \{0, 1, \ldots, n-1\}$ such that there is a tunnel connecting lightbulbs i and j.
4. $w_i \in \{1, 2, 3\}$ for all $i \in \{0, 1, \ldots, n-1\}$.

Input Format

The first line contains an integer indicating the number of test cases to follow. For each test case, the first three lines each contains an integer: n, m and B, respectively. The fourth line contains n integers, namely w_0, w_1, ..., w_{n-1}. Each of the next m lines contains three integers, namely i, j and $t_{i,j}$, indicating tunnel connecting lightbulbs i and j has visibility threshold of $t_{i,j}$. All consecutive numbers in a line are separted by a blank space.

Output Format

For each test case, please output the minimum total power consumption (in watts) of the lightbulbs subject to (1) the number of underlit tunnels being at most B and (2) the total power consumption of the lightbulbs being at most $\lceil \log_2 n \rceil$ watts. In case it is impossible to meet these two requirements simultaneously, please output "no".

Sample Input

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

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

Output for the Sample Input

```
no
3
no
3
```

Problem L
Subsets
Time Limit: *3 Seconds*

Sets are important in mathematics. Almost any thing can be put into a set, as long as the universe of the elements is determined. For a universe Σ of n elements, there are totally 2^n different subsets of Σ. These subsets, again, can be collected as a set, which is often called a *family*. It is not difficult to see that there are 2^{2^n} different ways to collect them. For example, let $\Sigma = \{1, 2, 3\}$. Then, $S = \{\{1\}, \{2, 3\}\}$ is a family of subsets of Σ having 2 subsets. Similarly, $T = \{\{\}, \{1\}, \{1, 2, 3\}\}$ is a family having 3 subsets.

In the following, we consider basic operations to manipulate families of subsets. Let A and B be any families of subsets of Σ and x be an element of Σ.

- `Make`(x_1, x_2, ..., x_k). Build the family of subsets $\{\{x_1, x_2, \ldots, x_k\}\}$ where $x_i \in \Sigma$ for $1 \leq i \leq k$. For example, `Make`$(1, 3) = \{\{1, 3\}\}$, which is a family having the subset $\{1, 3\}$ as its only element.
- `Union`(A, B). Build the family of subsets belonging to A or B.
- `Intersect`(A, B). Build the family of subsets belonging to both A and B.
- `Diff`(A, B). Build the family of subsets belonging to A but not to B.
- `Put`(A, x). Put x into all subsets in A. It x is already in this subset, leave it unchanged; otherwise add x into it.
- `Rem`(A, x). Remove x from all subsets in A. If x is in this subset, remove it; otherwise leave this subset unchanged.
- `Card(`A`)`. Calculate the number of subsets in A.

For example, let $S = \{\{1\}, \{2, 3\}\}$ and $T = \{\{\}, \{1\}, \{1, 2, 3\}\}$. Then

- `Make`()=$\{\{\}\}$;
- `Union`(S,T)=$\{\{\}, \{1\}, \{2, 3\}, \{1, 2, 3\}\}$;
- `Intersect`(S,T)=$\{\{1\}\}$;
- `Diff`(S, T)=$\{\{2, 3\}\}$;
- `Put`(S,2)=$\{\{1, 2\}, \{2, 3\}\}$;
- `Rem`(T, 1)=$\{\{\}, \{2, 3\}\}$; and
- `Card`(T)=3.

Notice that in a set (or family) identical elements are combined as one element. Please write a program to perform a series of basic subset operations specified in the test data.

Technical Specification

1. There is only one test case and there are roughly 500 lines that you need to simulate.
2. The hardest family of subsets is built from about 260 commands.
3. The universe is fixed to be $\Sigma = \{1, 2, 3, \ldots, 50\}$.

Input Format

There is exactly one command per line. Each line in the test data is *implicitily* associated with a line number, which starts from 1. We use this line number as a reference to the family of subsets that is built after applying the corresponding operation. The syntax of each basic operation is as follows.

- `Make` $x_1\ x_2\ \ldots\ x_k$
- `Union` $\#i\ \#j$
- `Intersect` $\#i\ \#j$
- `Diff` $\#i\ \#j$
- `Put` $\#i\ x$
- `Rem` $\#i\ x$
- `Card` $\#i$

Here, i and j are line numbers before the current execution line and # is a fixed leading symbol for identifying a line number. Integers $x, x_1, \ldots, x_k \in \Sigma$. When `Card` is encountered, output its value in a line. There is a single 0 in the last line of the test data, denoting the end of input.

Output Format

Please output the value after each execution of operation `Card` in a single line.

Sample Input

```
Make 1
Make 2 3
Union #1 #2
Make
Make 1
Make 1 2 3
Union #4 #5
Union #6 #7
Card #3
Card #8
Union #3 #8
```

```
Card #11
Rem #8 1
Card #13
0
```

Output for the Sample Input

```
2
3
4
2
```

Problem M
Surviving Probability
Time Limit: *2 Seconds*

A soldier, Raymond, is asked to carry out some military mission at the X region. This region has landmines underground. There are several *military bases* (*bases* for brevity) in the region, some of which are connected by roads such that any two bases can reach each other via at least one road. In addition, any two bases are connected by at most one road, and each road is bidirectional. For convenience, we can view the bases and roads as a graph $G = (V, E)$ in which bases are represented by vertices in V and roads are represented by edges in E. A base is *odd* if there are odd number of roads incident to it; otherwise, it is an *even base*. Each edge (u, v) is associated with a value $r(u, v) = \frac{1}{2^i}$, where $i \geq 0$ is a non-negative integer, which represents the *surviving probability* of passing through road (u, v). Note that $0 < r(u, v) \leq 1$. We assume that the probabilities associated with edges are independent. A *walk* is a list $v_0 \to v_1 \to \cdots \to v_{k-1} \to v_k$ such that, for $1 \leq i \leq k$, vertices v_{i-1} and v_i are adjacent. Note that a walk may contain repeated vertices and edges.

Given two odd bases S and T ($S \neq T$), Raymond would like to find a most reliable S-T *walk*, starting from S and ending at T, that maximizes the surviving probability such that all the roads are passed through at least once (i.e., the product of the surviving probabilities of the edges in the walk is maximum among the other S-T walks). We call such a walk a most reliable S-T walk, and call the product of the surviving probabilities of the edges in this walk the *maximum surviving probability*.

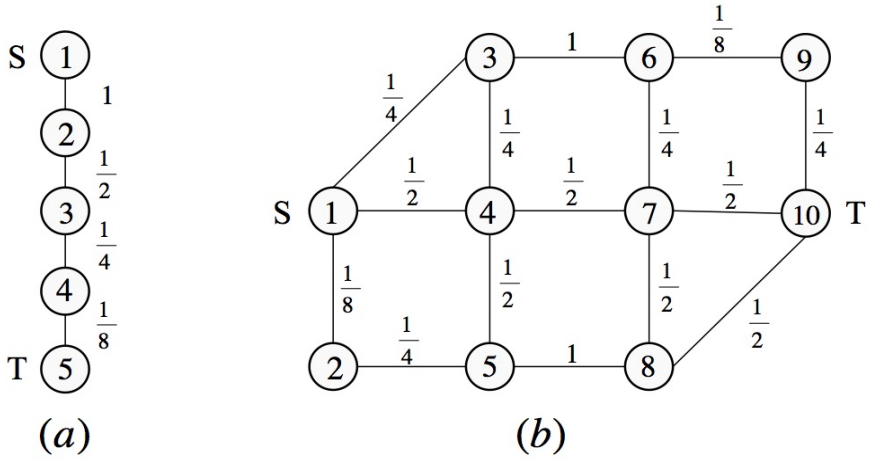


Figure 1: Two examples

For example, as shown in Figure 1(a), "$S = 1 \to 2 \to 3 \to 4 \to 5 = T$" is the only most reliable S-T walk. The maximum surviving probability is $1 \times \frac{1}{2} \times \frac{1}{4} \times \frac{1}{8} = \frac{1}{64} = \frac{1}{2^6}$. In Figure 1(b), there are many desired S-T walks. For example, "$S = 1 \to 3 \to 4 \to 1 \to 2 \to 5 \to 4 \to 7 \to 6 \to 3 \to 4 \to 5 \to 8 \to 7 \to 10 \to 9 \to 6 \to 7 \to 8 \to 10 = T$" and "$S = 1 \to 3 \to 6 \to 9 \to 10 \to 8 \to 5 \to 2 \to 1 \to 4 \to 5 \to 8 \to 7 \to 6 \to 3 \to 4 \to 7 \to 10 = T$" are two S-T walks such that all the roads are passed through at least once. The former is not a most reliable S-T walk (with surviving probability $\frac{1}{2^{28}}$), but the latter one is a most reliable S-T walk with the maximum surviving probability $\frac{1}{2^{22}}$.

Please write a computer program to compute the maximum surviving probability $\frac{1}{2^j}$ of a most reliable S-T walk that passes through all the edges at least once, and output the

value j that is the exponent of the base 2 in the denominator of the maximum surviving probability.

Technical Specification

- Graph $G = (V, E)$ is connected.
- $2 \leq |V| \leq 100$.
- $1 \leq |E| \leq \frac{|V|(|V|-1)}{2}$.
- For each edge (u, v), $r(u, v) = \frac{1}{2^i}$, where $0 \leq i \leq 10000$.

Input Format

The first line of the input contains an integer L ($L \leq 11$) indicating the number of test cases to follow. Each test case consists of a graph $G = (V, E)$, which has the following format: the first line contains four numbers, $n(= |V|)$, $m(= |E|)$, S, and T such that any two consecutive numbers are separated by a single space. The next m lines contain the description of m edges and their surviving probabilities such that each line contains two endpoints of an edge and the exponent of the base 2 in the denominator of the surviving probability corresponding to this edge (any two consecutive numbers are separated by a single space).

Output Format

The output contains one line for each test case. Each line contains the exponent j of the denominator of the maximum surviving probability $\frac{1}{2^j}$ of a most reliable S-T walk.

Sample Input

```
2
5 4 1 5
1 2 0
2 3 1
3 4 2
4 5 3
10 15 1 10
1 2 3
1 3 2
1 4 1
2 5 2
3 4 2
3 6 0
4 5 1
```

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

Output for the Sample Input

```
6
22
```