2017 ACM ICPC Asia Hua-Lien Regional

November 25, 2017

- Problems: There are 13 tasks (32 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	Smooth Sequences	1 sec.
Problem B	Threesome	1 sec.
Problem C	Coefficient Computation	1 sec.
Problem D	Network Report	1 sec.
Problem E	Ambush	1 sec.
Problem F	Magic Sequence Pairs	1 sec.
Problem G	Family Gathering at Christmas	1 sec.
Problem H	A Partial Order Relation	1 sec.
Problem I	Tracer Deployment	1 sec.
Problem J	Wheel Connector	4 sec.
Problem K	Assigning Frequencies	2 sec.
Problem L	Finding the Bases	2 sec.
Problem M	Annual Congress of MUD	1 sec.

Problem A
Smooth Sequences

A sequence of n integers is called *smooth*, if the absolute values of the differences between successive numbers are at most d. We call a non-smooth sequence *semi-smooth*, if we can change exactly one number in the sequence to make it smooth. For example, let $n = 4$ and $d = 5$. The sequence 21, 20, 18, 20 is smooth, since each pair of adjacent numbers has a gap smaller than 5. The sequence 21, -8, 20, 20 is semi-smooth, since we can replace -8 with 20 to make it smooth. Write a program to test if a sequence of integers is smooth or semi-smooth.

Technical Specification

1. All integers in the sequence have absolute values bounded by 2^{20}.
2. $1 \leq n, d \leq 1000$.

Input Format

An instance of the problem consists of the size n of the sequence, and the gap d in a line. They are positive integers in the range $[1, 1000]$. Then n integers, separated with space(s), follow in the next line.

Note that the test data file may contain more than one instance. The last instance is followed by a line containing a single 0.

Output Format

For each instance, output "Y", if the sequence is smooth or semi-smooth in a separate line. Otherwise, output "N".

Sample Input

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

Sample Output for the Sample Input

```
Y
N
```

Problem B
Threesome

There is a student association S formed by n college students and every student has a unique ID that is a positive integer. For convenience, we use $\{1, 2, \ldots, n\}$ to represent the n students. Any two students i and j have a *social connectivity*, denoted by $c(i, j)$, between i and j, where $1 \leq c(i, j) \leq 10000$. Three students i, j, k are *best threesome* if the sum of their mutual social connectivities $c(i, j) + c(j, k) + c(i, k)$ is maximum among all the threesomes, i.e., $\max_{i,j,k \in S}\{c(i, j) + c(j, k) + c(i, k)\}$.

Your task is to write a computer program to compute the sum of their mutual social connectivities of a best threesome.

Technical Specification

- $3 \leq n \leq 300$
- Each ID is at least 1 and at most n.
- For any two students i and j, $1 \leq c(i, j) \leq 10000$.

Input File Format

The first line of the input contains an integer, denoting the number of test cases to follow. For each test case, the association $S = \{a_1, a_2, \ldots, a_n\}$ is given with the following format: The first line contains a positive integer n. In the following $\frac{n(n-1)}{2}$ lines, each line contains three integers representing i, j, and $c(i, j)$ such that any two integers are separated by a space.

Output Format

For each test case, output the sum of their mutual social connectivities of best threesome.

Sample Input

```
2
3
1 2 1
1 3 1
2 3 1
4
1 2 2
1 3 2
1 4 1
2 3 2
2 4 1
3 4 1
```

Output for the Sample Input

3
6

Problem C
Coefficient Computation

In mathematics, any of the positive integers that occurs as a coefficient in the binomial theorem is a binomial coefficient. A binomial coefficient is commonly indexed by a pair of integers $n \geq k \geq 0$ and is denoted by $C(n, k)$. It is the coefficient of the x^k term in the polynomial expansion of the binomial power $(1+x)^n$, which is equal to $\frac{n!}{k!(n-k)!}$. Furthermore, in combinatorics, it is the number of ways to choose a subset of k items from a set of n items given that the order of selection does not matter. There are several ways to compute $C(n, k)$. The above formula is factorial formula. And the following formula is the recursive formula,

$$C(n,k) = \begin{cases} C(n-1,k-1) + C(n-1,k), & 1 < k \leq n-1 \\ 1, & k = 0 \text{ or } k = n \\ n, & k = 1 \end{cases}$$

Now, give three integers n, k and d with $n \geq k \geq 0$ and $1 < d < 10$. Please write a program to compute $C(n, k)$ and output the result in base d. That is, the result value should be converted from base 10 to base d.

Technical Specification

1. The number of test cases is no larger than 10.
2. n is an integer and $0 \leq n \leq 300$.
3. k is an integer and $0 \leq k \leq n$.
4. d is an integer and $1 < d < 10$.

Input Format

The first line of the input contains an integer indicating the number of test cases to follow. Each test case contains three integers n, k and d, separated by space on a single line.

Output Format

For each test case, output $C(n, k)$ in base d in a line.

Sample Input

```
3
20 10 3
150 50 8
10 2 7
```

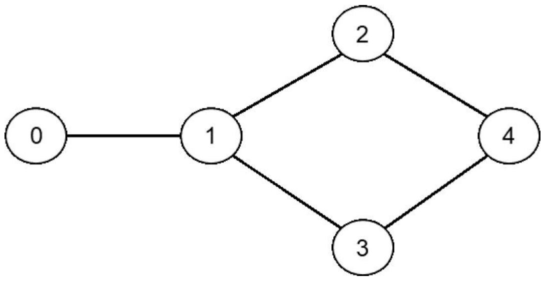
Sample Output for the Sample Input

```
100101102211
354470761204215171415303750634427342737334454
63
```

Problem D
Network Report

For time efficiency, data transmission is often done along the shortest path (with minimum number of hops) between network nodes. In the network graph (undirected and unweighted) below, the length of the shortest path from node 0 to node 2 is 2, and the length of the shortest path from node 0 to node 4 is 3. For a network graph with n nodes, we want to find all-pairs shortest paths, and provide a summary report on how many shortest paths are there for each length (from 1 to the maximal length). That is, this summary report should contain the total number of shortest paths of length 1, the total number of shortest paths of length 2, and so on and so forth.

Note that if multiple shortest paths from node i to node j ($i \neq j$) exist, they should collectively count as one. For example, though there are two shortest paths from node 0 to node 4, only one path is counted. Please write a program to output the above summary report from an input connected graph.



Technical Specification

1. $2 \leq n \leq 200$, nodes are numbered as $0, 1, 2, \ldots, n-1$.
2. Any counting number is within the range of integer.

Input Format

The input contains multiple instances of an undirected and unweighted connected graph (not multigraph). For each instance, the first line contains a single integer n. The second line contains a single integer e, denoting the number of edges in the graph. For the next e lines, each line contains two integers i, j, denoting that there is an edge connecting node i and node j. There is a 0 on a single line following the last test instance. There are at most 10 test instances.

Output Format

For each instance, there are several lines of output pending on the input. Each line of output consists of two integer numbers separated by a single space, where the first number is the path length and the second number is the total number of shortest paths of the corresponding length. The output should display the results from length 1 to the maximal length in the graph incrementally.

Sample Input

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

Sample Output for the Sample Input

```
1 6
1 10
2 8
3 2
```

Problem E
Ambush

In all kinds of military warfare, the strategy used is often the key in determining the outcome of the war. The ambush tactic is often used to surprise and to defeat the invading forces with fewer troops than the invading forces. Based on past battles records, it is known that if ambushing with only one third or more troops than the invading enemy's troops, enemy will be defeated. That is if the invading enemy sends 10 units of troops, the enemy can be defeated if ambushed with only 4 units of troops.

Take towns A, B, C, D, and E, as shown in Figure 1, for example. The enemy is located in town A with 10 units of troops while the capital is in town C. Suppose we deploy 2, 3, and 4 units of troops on the roads $\{B, C\}$, $\{C, D\}$, and $\{C, E\}$ for ambushing, respectively, then if the enemy invades through paths $A \to B \to C$ and $A \to D \to C$, each with 5 units of troops (see Figure 2), the captial will still be safe. Yet, if the enemy attacks through path $A \to B \to C$ with all 10 units of troops(see Figure 3), then the capital will be captured. However, we can ensure absolute safty of the capital, regardless of the enemy's attack plan, if we deploy 4 units of troops on routes $\{B, C\}$, $\{C, D\}$, and $\{A, E\}$, respectively (see Figure 4).

Now the enemy is about to invade with the intend to capture our capital. Intelligence gave us the enemy's total troop units and their location. However, we do not know what the enemy's attacking route(s) nor troops sent on each route. To ensure safety of our capital, we need to assign troops to some of the roads to ambush the enemies if they come that way. You have been appointed to be the commander. Given the intelligence information, please assess the minimum total troop units required to ensure absolute safety of the capital.

Technical Specification

1. The given map contains n towns, numbered from 1 to n, where $2 \leq n \leq 1000$.
2. The location of the enemy troops is at town s, $1 \leq s \leq n$; our capital is at town t, $1 \leq t \leq n$.
3. There is at most one road connecting any two towns.
4. The total number of roads is m, and $1 \leq m \leq 10000$.
5. The enemys troops has k units, where $1 \leq k \leq 100$.
6. There is at least one path (sequence of roads) connecting any two towns.
7. When deploying troops, only integeral troop units are allowed.
8. To simplify the process of evaluation, loss of troops on the either side is not considered.

Input Format

The first line of the input contains an integer indicating the number of test cases. For each test case, the first line contains five positive integers n, m, s, t, k, representing the number

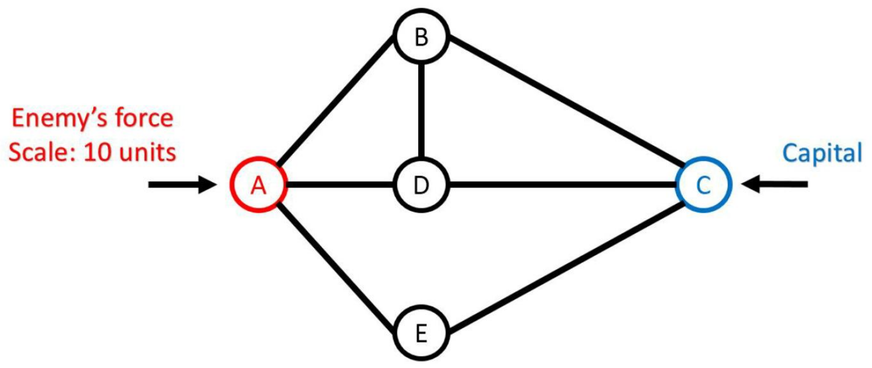


Figure 1: Map of five towns.

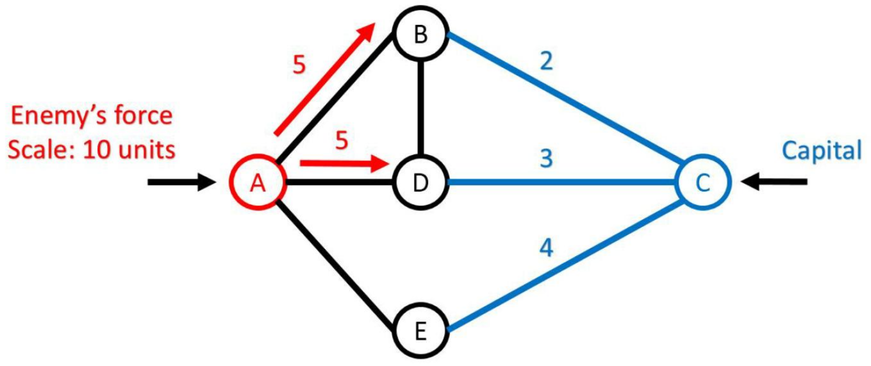


Figure 2: Successful defense. Captial is safe from the given attack.

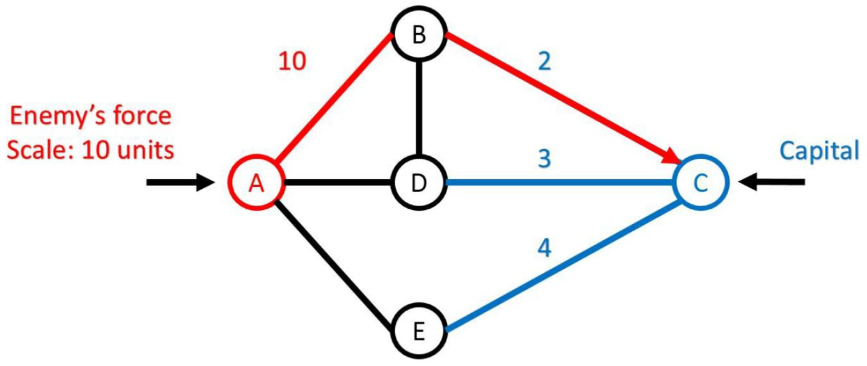


Figure 3: Failed defense. Capital will be captured.

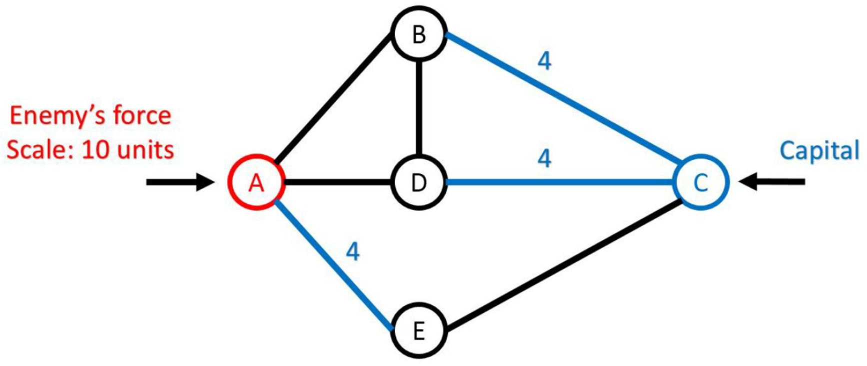


Figure 4: A deployment to ensure absolute safty of the Capital.

of towns, number of roads, enemy troops location, location of the capital and enemy troop units. Each of the following m lines contains two integers i, j indicating there is a road connecting town i and town j.

Output Format

For each test case, output the minimum total ambush troops needed to guarantee absolute safety of the capital.

Sample Input

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

Sample Output for the Sample Input

```
12
```

Problem F
Magic Sequence Pairs

We are particularly concerned with the problem of designing novel sequences encoding large collections of target patterns. There is a need for several distinct sequence designs, each containing a large set of representative substrings. Three distinct objectives of these designs needed to be satisfied:

1. *Minimum Length* – The sequence designs must be as short as possible.
2. *Variability* – The sequence designs are expected to contain as much varieties as possible, especially for the local short patterns.
3. *Orthogonality* – The final goal is that each pair of our designed sequences shall avoid common (or reverse-complement) sequences to minimize the chances of overlapping representatives.

A ternary string is a sequence of 0s, 1s, and 2s. The term k-mer refers to all the possible substrings of length k that are contained in a string. As an example, it is readily checked that the following string

00112202

contains 7 different 2-mers, namely, 00, 01, 11, 12, 22, 20, 02.

It is easily verified that the maximum number of possible k-mers in ternary strings is 3^k. A *magic sequence* (of order k) is a *minimum* length ternary string containing all k-mers exactly once; note that the length of a magic sequence is exactly $3^k + k - 1$. Here we deal with the third criteria: orthogonality. In particular, we define two magic sequences as *orthogonal* if they contain no $(k+1)$-mer in common – even though they both *must* contain *all* k-mers in common. These two orthogonal magic sequences is a *magic sequence pair* (of order k). For example, the following is a magic sequence pair of order 2.

0011220210
0022110120

each contains all 3^2 possible doublets (2-mers) exactly once, but together they do not share any 3-mers in common. Furthermore, the following is a magic sequence pair of order 3.

00010021011022202012111221200
00020012022011101021222112100

each contain all 3^3 possible triplets (3-mers) exactly once, but have no 4-mers in common.

Technical Specification

The base length (order) k in the setting of all k-mers in the magic sequences is less than 8.

Input Format

An instance of the problem consists of a single line with a single integer k, of a magic sequence pair (of order k). There may be more than one test instances. The last instance is followed by a line containing a single 0.

Output Format

The outputs for each test case consists of two parts. The first line print out the input base length (order) k on a single line. The second part prints out a magic sequence pair, one sequence per line.

Sample Input

```
1
3
2
0
```

Sample Output for the Sample Input

```
1
012
021
3
0001002101102220201211 1221200
000200120220111010212221 12100
2
0011220210
0022110120
```

Problem G
Family Gathering at Christmas

Every year, Alice's family has a gathering at Christmas, at a family member's place. The members would like to choose a host so that every member can reach the host's place without spending too much time, where the time for traveling depends on the geographical distance between the two places and the way of transportation. For example, going to a place on foot usually takes longer time than by bicycle. Since a host has to spend much effort to get everything ready, the family decides not to have the gathering at one's place so often. For members who are not the host, they have to prepare a dish for the party. However, all of them are too busy to cook on their own so they always go to the central city to buy a dish right before the party.

Alice suggests a way for her family to determine the gathering place. First, she associates each member with a weight, which quantifies the way of transportation. Then, the time needed for each non-host member m to reach the host is

$$w_m \cdot (d_m + d_h),$$

where w_m is the weight of the member, d_m is the distance from the member's place to the central city, and d_h is the distance from the central city to the host's place. Then she associates each member's place with a key, which is the longest time needed for a non-host member to reach the place. To decide the host, she picks a small number k, and choose a place with the kth smallest key. Please develop an efficient algorithm to help Alice find the kth smallest key.

Technical Specification

1. There is an integer n, $1 < n \leq 40000$, denoting the number of family members.
2. Members are numbered from 1 to n.
3. There are n integers, $w_1, \ldots, w_n$, where w_i is the weight of member i. For $i \in \{1, \ldots, n\}$, $1 \leq w_i \leq 50000$.
4. There are n integers, $d_1, \ldots, d_n$, where d_i is the distance from member i's place to the central city. For $i \in \{1, \ldots, n\}$, $1 \leq d_i \leq 50000$.
5. For $i, j \in \{1, 2, \ldots, n\}$, $w_i(d_i + d_j) < 2 \times 10^9$.
6. k is an integer satisfying $1 < k < n$.

Input Format

The first line of the input is the number of instances to be tested. There are at most 10 instances. An instance consists of 3 lines. The first line contains two integers, n and k. The second line contains n integers, $w_1, \ldots, w_n$, and the third line contains $d_1, \ldots, d_n$. Two consecutive integers in a line are separated by a space.

Output Format

The output for each instance is an integer, which is the kth smallest key.

Sample Input

```
2
3 2
5 3 4
3 8 5
4 2
6 2 8 2
10 18 12 4
```

Sample Output for the Sample Input

```
40
132
```

Problem H
A Partial Order Relation

In mathematics, a partial order set formalizes and generalizes the intuitive concept of an ordering, sequencing, or arrangement of elements of a set. It is a binary relation indicating that, for certain pairs of elements in a set, one of the elements precedes the other in the ordering but not every pair is comparable. For example, the divisors of 120 form a partial order set. Let the binary relation $\prec$ be divisibility relation. So, $120 \prec 24$ and $120 \prec 40$ but there is no such relation between 24 and 40. $\prec$ is a transitive relation such that $120 \prec 40$ and $40 \prec 8$ imply $120 \prec 8$.

Let's define a more restrictive binary relation called greatest divisibility relation $\preceq$. Given a number a. Let $div(a)$ be a set of all divisors of a minus a. Then we define $a \preceq b$ if b is a divisor of a but b cannot divide any other numbers in $div(a)$. For example, $div(30) = \{1, 2, 3, 5, 6, 10, 15\}$. $30 \preceq 6$ because 6 cannot be used to divide other elements in the set.

Given a number, you can construct $\preceq$ relation among the its divisors as in Fig. 1 for 120. The depth of the graph from root node (120) to the terminal node (always 1) is 5 and the number of edges are 28. Given an integer, please compute the number of edges in its $\preceq$ relation graph.

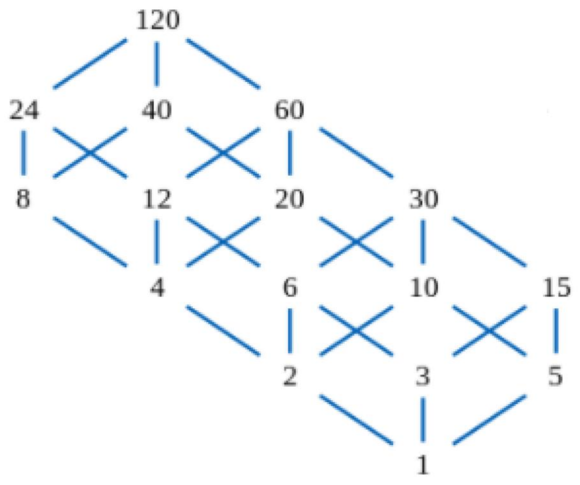


Figure 1: A partial order between the divisors of 120.

Input Format

The input data begins with a number n ($n <= 100$), which is the number of test cases. Each test case contains only a positive integer r, where $1 < r < 2^{40}$. r is the number to build $\preceq$ relation.

Output Format

For each test case, please print the number of edges in its divisibility relation.

Sample Input

```
2
6
120
```

Sample Output for the Sample Input

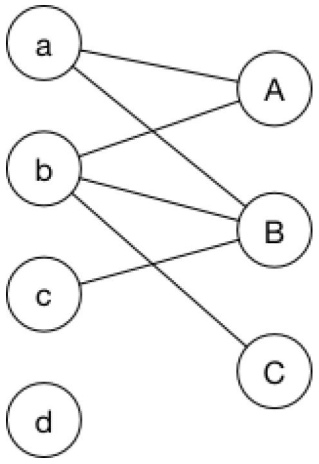
```
4
28
```

Problem I
Tracer Deployment

Distributed denial of service (DDoS) is a rapidly growing network security problem. DDoS attacks always paralyze the network nodes such as servers and occupy the network bandwidth. How to defense DDoS attacks is a challenging issue for network security problem. The key to defense DDoS attacks is to find the attack origins. The log of attack origins can be used as evidence for post-attack investigations. Furthermore, if the attack origins can be located during the attack, the anomaly attack packets can be blocked by cooperative filter nodes distributed over the Internet. Therefore the network bandwidth among the attackers and targets can be prevented from being occupied by the attack packets. The IP traceback technology can be applied to identify the attack origins. It usually relays on enhanced routers and provides assistance in tracing the path traversed by attack traffic and then identify the machines that directly generate attack packets. Here, we refer to the enhance routers which provide tracing service as *tracers*. Evidently, the location of traces will affect the performance of locating the attack origins.

Next, we consider the following tracer deployment problem: How many and where should the tracers be deployed in the network to be effective for locating the attack origins? Notice that the attacker can generate attack packets that appear to have originated from anywhere. It seems impossible to predict the location of attack origins. If the attack packets do not pass through any tracer in the network, the attack origins cannot be located using the IP traceback method.

For example, if tracers are deployed at node A and node C, when an attack initiates at node a, passing through node B, and finally attacks node c, no tracer will detect the above attack. On the other hand, if three tracers are deployed at nodes A, B, and b, then any attack will always pass through at least one of the three tracer nodes, and the attack will be detected.



Please write a program to determine the minimum number of tracers need to be deployed to ensure any attck will pass through (and be detected) at least one tracer. Note that the given network is a bipartite network $G = (S, T, E)$ whose nodes can be partitioned into two subsets S and T so that each communication link (in E) has one end in S and one end in T. Also note that the network may be disconnected or even have isolated nodes.

Input Format

There are at most 10 test cases. The first line of each instance consists of an integer m $(1 \leq m \leq 20)$, where m is the number of nodes (in set S of $G = (S, T, E)$) labeled with $\{0, 1, 2, \ldots, m-1\}$ in the network. The second line of each instance consists of an integer n $(1 \leq n \leq 20)$, where n is the number of nodes (in set T of $G = (S, T, E)$) labeled with $\{0, 1, \ldots, n-1\}$ in the network. The third line of each instance consists of an integer e $(1 \leq e \leq 100)$, where e is the number of communication links in the network. The next e lines, each contains two integers i, j, indicating a communication link exists between nodes i and node j. The last test case will be followed by a line containing a single 0.

Output Format

The output for each instance should contain an integer denoting the minimum number tracers needed for the given network.

Sample Input

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

Sample Output for the Sample Input

```
3
4
```

Problem J
Wheel Connector

A wheel connector has several connecting points on its circumference. Connectors are designed as plugs (male connectors) and receptacles (female connectors). A plug and a receptacle are correctly connected when all the corresponding connecting points are connected in a right way. In order to prevent wheel connectors connected in a wrong way, positions of connecting points are specially designed. Although these positions are specially designed, factory-produced connectors may still contain many positional errors.

A way to find the correctly connected way is to minimize the position error of corresponding connecting points. For example, a wheel connector can be configurated as a sequence of strictly increasing real numbers $[a_0, a_1, a_2, ..., a_{n-1}]$ with $0 \leq a_k < 1$ which means that there are connecting points on angle $2\pi a_k$ of the circumference corresponding to a roughly assigned center of the wheel and a casual assigned base direction. A male and a female connectors are configurated as two strictly increasing sequences as follows: $[a_0, a_1, a_2, ..., a_{n-1}]$ and $[b_0, b_1, b_2, ..., b_{n-1}]$ with roughly assigned centers and casual assigned base directions. When male and female connectors are connected, the connecting points should be mirrored. So for the convience, we level connecting points on male connectors with counter-clockwise direction and female connectors with clockwise direction.

An error function is designed as $\min_{0\leq c\leq n-1, 0\leq d\leq 1} \sum_{0\leq k<n} adf((a_k+d)\%1, b_{(k+c)\%n})$ where c is an integer, d is a real number, % is the remainder function for float number division, and $adf(x, y)$ is the function to calculate smallest position difference between angle $2\pi x$ and angle $2\pi y$. That is, $adf(x, y) = min(abs(x\%1 - y\%1), 1 - abs(x\%1 - y\%1))$ where abs is the function to get absolute value of a real number. For example, $1.732\%1 = 0.732$, $adf(0.3, 0.1) = abs(0.1 - 0.3) = 0.2$, and $adf(0.8, 0.1) = min(abs(0.8 - 0.1), 1 - abs(0.8 - 0.1)) = 0.3$.

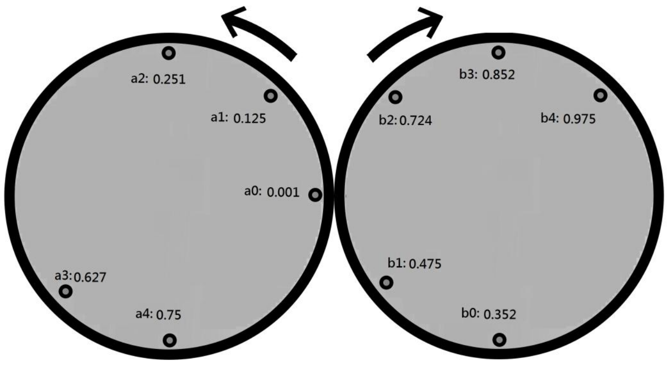


Figure 1: Wheel Connector Example.

In Fig 1, we can correctly connect two wheel connectors, when we find values of c and d such that the minimized error is calculated. For example, two connectors $[0.001, 0.125, 0.251, 0.627, 0.75]$ and $[0.352, 0.475, 0.726, 0.851, 0.976]$ are given. We can find that the error value is minimized as 0.001 when $c = 2$ and $d = 0.725$.

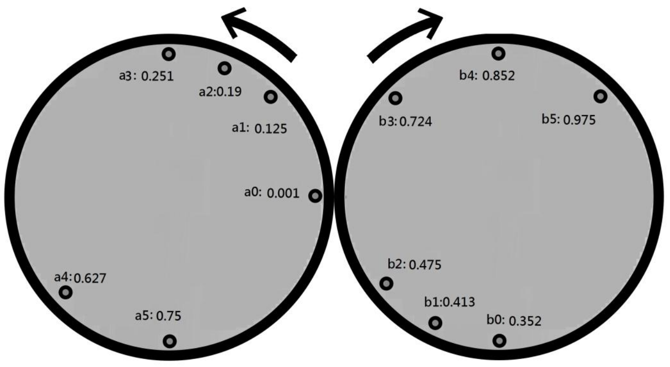


Figure 2: Fault Tolerance Wheel Connector Example.

A fault tolerance wheel connector is designed such that information can transmit on the wire with even several missing connecting points. Two fault tolerance wheel connectors can be configurated as two sequences as follows $[a_0, a_1, a_2, ..., a_{n-1}]$ and $[b_0, b_1, b_2, ..., b_{m-1}]$. A connecting function f is a function from $\{0, 1, 2, ..., n-1\}$ to $\{-1, 0, 1, 2, ..., m-1\}$ such that

1. for all $0 \leq i, j < n, i \neq j, f(i) \neq -1$, we have $f(i) \neq f(j)$ and
2. for all $0 \leq i, j, k < n$, $i < j < k$, $f(i) \neq -1, f(j) \neq -1, f(k) \neq -1$, we have $(f(j) - f(i)) \times (f(k) - f(i)) \times (f(k) - f(j)) > 0$.

In general, a connecting function is a $1-1$ mapping between two set of connecting points with a circular increasing assignment. The condition $f(k) = -1$ means the connecting function does not map a_k to any points. So the condition represents that the corresponding point of a_k is missing.

A connecting function f extends another connecting function g if $\forall_{0 \leq i < n, g(i) \neq -1} g(i) = f(i)$ and $\exists_{0 \leq i < n} f(i)! = g(i)$. A connecting function f is called maximal if there exists no connecting function g such that g extends f. In general, maximal connecting function is a connecting function such that we matches as many mapping pairs as possible based on a connecting function.

An error function for fault tolerance connectors is designed as $\min_{f, 0 \leq d < 1} \sum_{0 \leq k < n, f(k) \neq -1} adf((a_k + d)\%1, b_{f(k)})$ where f is a maximal connecting function and d is a real number.

We can connect two wheel connectors, when we find values of f and d such that the minimized error is calculated.

For example, in Fig 2, two fault tolerance wheel connectors $[0.001, 0.125, 0.19, 0.251, 0.627, 0.75]$ and $[0.352, 0.413, 0.475, 0.724, 0.852, 0.975]$ are given, and we can find that the error value are minimized as 0.005 when $d = 0.725$, and $f(0) = 3, f(1) = 4, f(2) = -1, f(3) = 5, f(4) = 0, f(5) = 2$.

Technical Specification

1. $3 \leq n, m \leq 20$.
2. $0 \leq a_k < 1$ for $0 \leq k < n$. a_k has at most 18 decimal digits after the decimal point.

3. $0 \leq b_k < 1$ for $0 \leq k < m$. b_k has at most 18 decimal digits after the decimal point.

Input Format

The first line of input contains a single integer indicating the number of test cases. Each test case contains two lines of input denoting two fault-tolerance wheel connectors' configurations, respectively. The first line of a test case contains a positive number n, followed by n numbers denoting $a_1, \ldots, a_n$. The second line of a test case contains a positive number m, followed by m numbers denoting $b_1, \ldots, b_m$.

Output Format

For each test case, output the value of minimized position error on a single line.

Sample Input

```
2
5 0.001 0.125 0.251 0.627 0.75
5 0.352 0.475 0.726 0.851 0.976
6 0.001 0.125 0.19 0.251 0.627 0.75
6 0.352 0.413 0.475 0.724 0.852 0.975
```

Sample Output for the Sample Input

```
0.001
0.005
```

Problem K
Assigning Frequencies

Bob wants to assign frequencies to n satellites, numbered from 0 to $n-1$. Two distinct satellites are said to be *adjacent* if, when assigned the same frequency, they interfere with each other. Sensibly, Bob's assignment of frequencies must avoid interference altogether. However, only three frequencies are available to him.

Please determine whether Bob can assign frequencies to all satellites to avoid any interference.

Technical Specification

1. There are at most 85 test cases
2. The number of satellites is n, $n \leq 25$.

Input Format

The first line of the input contains an integer indicating the number of input cases. For each test case, the first line contains an integer n, denoting the number of satellites. The second line contains an integer p, denoting the number of adjacent pairs of satellites. The next p lines each contains two integers i, j, indicating the satellite i may intefer with satelite j when both are assigned the same frequency.

Output Format

For each test case, output "Y" if Bob can assign frequences so there is no integerence. Output "N" otherwise.

Sample Input

```
4
6
6
0 3
1 5
3 2
2 5
0 4
1 0
7
12
6 5
0 3
2 6
```

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

Sample Output for the Sample Input

```
Y
N
Y
N
```

Problem L
Finding the Bases

There are many ways to represent a string. The following syntax

$$(x_1, k_1)(x_2, k_2)\ldots(x_\ell, k_\ell)$$

defines the string $x_1^{k_1}x_2^{k_2}\cdots x_\ell^{k_\ell}$ where x_i is the ith string that has to repeat k_i times. We call this representation a *brief string* because it can represent a very long string by using only relatively small amount of space. For example, $(ab, 2)(a, 4)$ represents *ababaaaa*. If you are given a brief string, certainly you can quickly recover the string that it represents. Conversely, if you are given an ordinary string, you can find many different brief strings that represent it. We are interested in finding the *shortest one.* We define the length of a brief string $(x_1, k_1)(x_2, k_2)\ldots(x_\ell, k_\ell)$ to be $|x_1| + |x_2| + \cdots + |x_\ell|$. That is, we only consider the total length of strings that has to be repeated and ignore all the numbers (as well as the parentheses and commas). The shortest brief string of an ordinary string is called a *basis*. For example, both $(a, 1)(ba, 3)(a, 3)$ and $(ab, 3)(a, 4)$ represent the same string *abababaaaa*. However, only the second one is its basis whose length is 3. In this problem, you need to find the length of a basis of an ordinary string.

Technical Specification

1. The alphabet contains the lowercase English letters.
2. The length of an ordinary string is between 1 and 10000.
3. There are at most 20 test cases.

Input Format

The first line of input contains an integer indicating the number of test cases. For each test case, an ordinary string is given on a single line.

Output Format

Output the length of the basis of the specified ordinary string for each test case.

Sample Input

```
3
aaaaaaaaaa
abcabcabca
abcdab
```

Sample Output for the Sample Input

```
1
4
6
```

Problem M
Annual Congress of MUD

Multiuser dungeon games, also called MUD games, are real-time virtual-world multiplayer games that are usually played online with text-based commands. The players do not meet normally, but every year, there is the *Annual Congress of MUD (ACM)* where MUD lovers all around the world could meet in person.

ACM is so popular that the event each year spans around 20 days. Each day, there will be a special gathering for MUD game designers to introduce their new games to the others. Each player will usually spend a few days on the ACM site, and in-between will be invited in exactly one day to join this special gathering.

This year, ACM is held at your city, and your boss is an organiser, and he wants to find a *best* way to assign the players to these special gatherings (one player to one special gathering within his or her duration of stay), so that the maximum number of players among all gatherings is minimized.

Your boss is an extremely impatient guy. He wants to have a system that can tell the maximum number of players among all gatherings, in any best assignment, after *each* player enters his or her duration of stay. Your task is to help your boss develop such a system.

Technical Specification

1. The number of players joining the congress, N, is an integer with $1 \leq N \leq 10000$.
2. The number of days, D, of the congress is an integer with $3 \leq D \leq 20$.
3. The duration of stay, $[x_i, y_i]$, for the ith player to be keyed in your system, are pairs of integers with $1 \leq x_i \leq y_i \leq D$. This indicates that the ith player will stay from day x_i (inclusive) to day y_i (inclusive) on the ACM site.

Input Format

An instance of the problem consists of $N + 1$ lines. The first line specifies the integers N and D, seperated by a space. In the ith line of the following N lines, it contains two integers x_i and y_i, separated by a space. Note that the test data file may contain more than one instance. The last instance is followed by a line containing a single 0.

Output Format

For each instance, an integer i is *marked* if and only if the maximum number of players in the best assignment increases after the duration of stay $[x_i, y_i]$ of the ith player is keyed in. By default, 1 is always marked. The output of the corresponding instance is the list of all marked integers in ascending order, separated by a space between successive integers, followed by a newline character.

Sample Input

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

Sample Output for the Sample Input

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