Concours interne de programmation Télécom ParisTech

29 juin 2017

Contents

Problem A: Lush Gears **3**

Problem B: Amateur Radio Network **5**

Problem C: Life on a Torus **7**

Problem D: Grumpy Groundhogs in Gridland **9**

Problem E: Commuting Mathematicians **12**

Problem F: Larger Sport Facility **15**

Problem setters

- Antoine Amarilli
- Bertrand Meyer
- Florian Brandner

Special thanks to Pierre Senellart for setting up and maintaining the judging system, and to Véra Dickman for proofreading.

Barème

Barème pour le cours INF280

Si vous êtes inscrit au cours INF280 en 2016–2017, votre solution pour chaque exercice sera évaluée : elle recevra la note maximale pour cet exercice si elle est acceptée par le juge en ligne, sinon elle recevra une note fractionnaire. Chaque exercice a le même poids dans le barème final.

Attention : vous devez soumettre votre solution, même partielle, sur le juge en ligne pour être corrigé.

Les soumission incorrectes et le temps de soumission n'interviennent pas dans la notation.

Classement

Un classement de l'ensemble des participants sera établi à l'issue du concours. Il est indépendent du cours INF280. Il est basé uniquement sur les exercices pour lesquels votre code est accepté par le juge, et il suit les règles du SWERC.

1. Les participants sont classés en premier lieu par nombre décroissant d'exercices résolus.

2. Pour les participants qui ont résolu le même nombre d'exercices, les candidats sont classés par *temps de départage* croissant. Le temps de départage d'un candidat est la somme, pour chaque problème résolu par ce candidat, du temps entre le début du concours et la soumission de la première solution acceptée pour ce problème par ce candidat. Chaque soumission rejetée pour un problème finalement résolu par le candidat ajoute un malus de 20 minutes au temps de départage du candidat. Les soumissions rejetées sont sans effet si elles concernent un problème que le candidat n'a pas résolu.

Les étudiants seront appelés, dans l'ordre du classement, à représenter Télécom au concours ACM-ICPC SWERC 2017 qui se déroulera les 25 et 26 novembre prochain à Télécom. Les six premiers étudiants éligibles et volontaires seront retenus.

Problem A: Lush Gears

Time limit: 5 seconds

When they are not hiking, climbing or skiing in the backcountry, the Outdoor & Adventure Club of Télécom ParisTech is constantly scavenging for new gear to support their expeditions. Luckily for them, last week, they received a generous donation from a former Télécom student who now manages the famous mountain equipment shop "Indiana Jones's Cave". The manager has awarded them a voucher worth C chococoins that they can spend in her shop. However, the voucher can only be used once, and any money that remains afterwards is irremediably lost!

The Outdoor & Adventure Club has already set up a list of K gear types that they would like to purchase: tents, sleeping bags, stoves, ropes, harnesses, carabiners, ice axes, crampons, ice screws, etc. They also know how many copies of each gear type they would like to purchase. Now, for each gear type, Indiana Jones's Cave has many models: some are quite cheap and others are more expensive. Each model passes the Club's safety requirement, so the Club is fine with buying any model. Hence, the Club must choose, for each gear type, which model they will be purchasing in the requested quantity: to simplify maintenance, they will only choose one single model for each gear type. As they have the choice between models for each gear type, the goal of the Club is to spend their money in a way that minimises the number of chococoins that are lost.

Input

The input consists of several test cases. The first line consists of an integer indicating the number of test cases. Each test case follows. The first line of a test case consists of two integers $0 \leq C \leq 10000$ and $0 \leq K \leq 45$ separated by a single space: C stands for the value of the voucher in chococoins and K stands for the number of different gear types that the Club wants to buy. This is followed by K lines describing each gear type, with each line consisting of the following integers separated by single spaces: the first integer $1 \leq M_i \leq 25$ indicates the number of available models for this gear type, the next M integers $1 \leq P_{i,j} \leq 5000$ indicate the price of each model, and the last integer $0 \leq Q_i \leq 10$ indicates how many copies of this gear type the Club wants to buy.

Output

For each test case in the input, your program should produce one line. If there is no choice of models that will allow the Club to buy the requested number of copies for each gear type, the contents of the line should be `IMPOSSIBLE`. Otherwise, the contents of the line should be a positive integer $d \geq 0$ which describes the number of chococoins that will be lost when the Club buys the requested number of copies for each gear type with a choice of models that minimizes this number. There should be no blank lines in your output.

Sample Input

```
2
20 3
3 3 2 4 2
2 5 10 1
4 1 5 3 5 1
5 2
2 7 3 1
3 2 8 5 2
```

Sample Output

```
1
IMPOSSIBLE
```

Problem B: Amateur Radio Network

Time limit: 10 seconds

The Society for Wireless Emission and Reception Channels (SWERC) is a small community of residents of Hertzville. They love to organize small-talk gatherings on the radio-waves. The SWERC has one official radio frequency which is used by all their members, and all members have very powerful transmitters that can reach all other transmitters of the society so that they can all happily chat together. Whenever they are chatting, each transmitter automatically sets its emission power depending on the distance to the other transmitters used by SWERC members. Specifically, the power of each member's transmitter is proportional to the distance to the furthest transmitter.

Unfortunately for them, the new mayor of Hertzville is not fond of telecommunications. To be precise, the mayor claims that the SWERC's radio communications are a public health hazard, because the transmitters are too powerful. What the mayor is concerned about is the highest power setting of a SWERC transmitter.

In an attempt to negotiate, the SWERC is willing to split up into two groups, each of which would use a separate radio frequency. This would reduce the maximal transmission power, because each group's transmitter would only communicate with the other transmitters of that group. Specifically, the SWERC would decide how to allocate each member to one of the two groups. Within group 1, the transmitters would work as before: each transmitter would set its emission power according to the distance to the furthest transmitter *in group 1*, ensuring that all members of group 1 can still communicate as before. The same is true for group 2. As the SWERC is a very inclusive society, it would never consider isolating any single member. Hence, each group should contain at least two people, so that no member is left alone in a group on their own.

The SWERC will soon be meeting the mayor, and must come up with a concrete proposal. They need to know the smallest distance d such that the SWERC can be split into two groups, each group containing at least two people, such that, for each SWERC member, the maximal distance between that member and a member of their group is at most d. Can you help them?

Input

The input consists of several test cases. The first line consists of an integer indicating the number of test cases. Each test case follows. The first line of a test case consists of a single integer $4 \leq N \leq 700$ indicating the number of members of the SWERC. This is followed by N lines describing each member: each line consists of two integers $-10^3 \leq X_i \leq 10^3$ and $-10^3 \leq Y_i \leq 10^3$ separated by a single space, indicating the x- and y-coordinates of the station of the i-th SWERC member. You may assume that two stations are always in different locations, namely, for all $1 \leq i < j \leq N$, we have $(X_i, Y_i) \neq (X_j, Y_j)$.

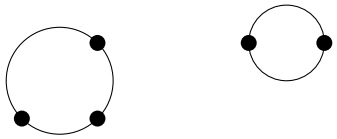


Figure 1: Illustration of the solution of the Sample Input

Output

For each test case in the input, your program should produce one line containing a floating point number d with exactly two digits at the right of the decimal point. The value of d should be the smallest possible value (rounded up) which ensures that there is a way to partition the SWERC members into two groups, each group containing at least two people, such that the Euclidean distance between any two members of the same group is less than or equal to d. There should be no blank lines in your output.

Sample Input

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

Sample Output

```
1.42
```

Problem C: Life on a Torus

Time limit: 5 seconds

Conway's game of life is played on a torus formed of $m \times n$ square cells. An $m \times n$ torus is just an $m \times n$ rectangle where the left and right edges of the rectangle have been glued together, as well as the top and bottom edges. The cells are identified by their coordinates (r, c), indicating the row number and column number. Every cell is either *dead* or *alive*. The *state* of the game is the pattern of live and dead cells on the torus.

Every cell on the torus has exactly 8 neighbors, which are the cells that are horizontally, vertically and diagonally adjacent. In particular, the neighbors of the cells $(1, 1)$ are $(0, 0)$, $(0, 1)$, $(0, 2)$, $(1, 0)$, $(1, 2)$, $(2, 0)$, $(2, 1)$, and $(2, 2)$. The neighbors of the cell $(0, 1)$ are $(m, 0)$, $(m, 1)$, $(m, 2)$, $(0, 0)$, $(0, 2)$, $(2, 0)$, $(2, 1)$, and $(2, 2)$.

At each step in time, the state of the game changes. The state of each cell at step $i + 1$ is determined from the state of its 8 neighbors at step i, according to the following rules:

- If 1 or less of the neighbors of the cell were alive at step i,
  then the cell is dead at step $i + 1$.
- If 2 of the neighbors of the cell were alive at step i,
  then the state of the cell at step $i + 1$ is the same as its state at step i.
- If 3 of the neighbors of the cell were alive at step i,
  then the cell is alive at step $i + 1$.
- If 4 or more of the neighbors of the cell were alive at step i,
  then the cell is dead at step $i + 1$.

The initial state is called the *seed* of the system. For $k \in \mathbb{N}^*$, we say that a state of the game has *period* k if, when the game is at this state, then we get back at exactly the same state k steps later. For instance, the configuration illustrated in the left diagram of Figure 2 has period 2: after one step, we obtain the configuration illustrated in the right diagram, and after one step we obtain again the configuration illustrated in the left diagram.

For a given seed, the *period* of this seed is the smallest $k \in \mathbb{N}^*$ such that the game, starting at this seed, eventually reaches a state with period k. Given a seed, your goal is to compute the period of the system.

Input

The input consists of several test cases. The first line consists of an integer indicating the number of test cases. Each test case follows. The first line of a test case consists of two positive integers $3 \leq m \leq 8$ and $3 \leq n \leq 8$ separated by a single space: m indicates the number of rows of the torus, and n indicates the number of columns. The next m lines represent the seed: each line describes a row and consists of precisely n characters which are either `'x'` for a dead cell or `'o'` for a live cell.

Output

For each test case in the input, your program should produce one line. The contents of this line should be a positive integer $p > 0$ which is the period of the given seed. There should be no blank lines in your output.

Sample Input

```
2
3 4
xxxx
xoxx
xxxx
6 6
xxxxxx
xooxxx
xooxxx
xxxoox
xxxoox
xxxxxx
```

Sample Output

```
1
2
```

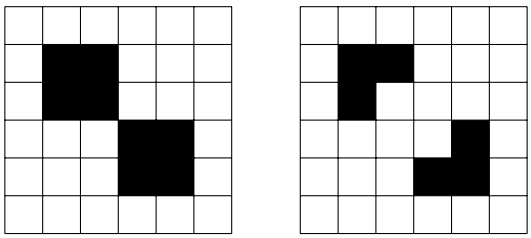


Figure 2: Illustration of the second input from the Sample Input: a seed with period two, and the configuration to which it leads

Problem D: Grumpy Groundhogs in Gridland

Time limit: 10 seconds

The wonderful forest of Gridland is home to the world's cutest population of groundhogs. During the exceptionally hot summer of this year, the groundhogs have merrily frolicked in the blossoming shrubs, and each of them has found their one true love among the other groundhogs.

However, winter has come, and the groundhogs are now all back to their separate burrows for winter. Of course, they are all very grumpy and lonely: they desire nothing more than to be be reunited with their one true love. Fortunately, the groundhogs have requested help from the Subterranean Way Excavation Research Company. As a representative of the company, your task is to build a tunnel network that will allow our enamored groundhogs to pay regular visits to each other. If you can do this, imagine how grateful the Gridland groundhogs would be!

Your preliminary survey has revealed that the Gridland soil consists of square cells labeled by pairs of nonnegative integer coordinates. The first integer represents the *position* on the x-axis, i.e., the distance to the origin (the leftmost point); and the second integer represents the *depth*. Each groundhog lives in a burrow on a cell with depth 0. Your excavation tools can empty any cell of the Gridland soil of depth 1 or more (the burrows themselves do not need to be excavated): the result is called a *tunnel network*. A *path* in the tunnel network is a sequence of excavated cells where each cell is either horizontally or vertically adjacent to the previous one (indeed, as any biologist would tell you, the Gridland groundhog is too fat to move between two diagonally adjacent cells). Your tunnel network should ensure that, for every groundhog couple, there is a path of excavated cells that connects the two burrows of that couple. However, to preserve the privacy of the groundhogs, it is extremely important that there be no path connecting two burrows whose inhabitants do not form a couple!

Your job is to resolve two questions. First, is it even possible to construct a tunnel network satisfying these conditions? Second, if it is possible, what is the maximum depth to which you will need to dig?

Example

In the left diagram of Figure 3, there are four couples of groundhogs: the burrows of the members of each couple are labeled A, B, C, and D. The diagram illustrates a network of tunnels of depth 5, where excavated cells are hatched: in fact, 5 is the lowest possible maximum depth for a suitable tunnel network in this example.

Of course, it can be the case that no suitable tunnel network exists at all! For instance, in the right diagram of Figure 3, there are two groundhog couples A and B. It is not possible to connect the couple members with tunnels that do not intersect, so the poor groundhogs will remain grumpy all winter!

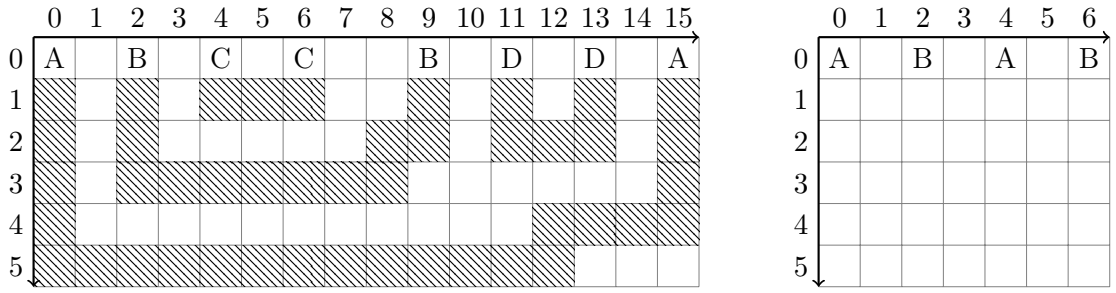


Figure 3: Illustration of the first two inputs from the Sample Input

Input

The input consists of several test cases. The first line consists of an integer indicating the number of test cases. Each test case follows. The first line of a test case consists of a positive integer N indicating the number of groundhog couples, with $1 \leq N \leq 10^6$. This is followed by N lines describing each couple: each line consists of two positive integers $0 \leq a_i < b_i \leq 10^9$ separated by a single space, indicating the position of the burrow of the first and second member of the couple (starting with the leftmost one). There are never two groundhogs living in the same burrow, and there are never two adjacent burrows (i.e., the absolute difference between the position of any two burrows is no smaller than 2).

Output

For each test case in the input, your program should produce one line. If there is no tunnel network that satisfies the conditions, the contents of the line should be `IMPOSSIBLE`. Otherwise, the contents of the line should be a positive integer $d > 0$, which is the smallest possible depth such that there be a tunnel network that satisfies the conditions and where the depth of every excavated cell is no greater than d. There should be no blank lines in your output.

Sample Input

```
4
4
0 15
2 9
4 6
11 13
2
0 4
2 6
1
0 2
```

```
7
0 26
10 16
12 14
20 22
2 24
4 18
6 8
```

Sample Output

```
5
IMPOSSIBLE
1
9
```

Problem E: Commuting Mathematicians

Time limit: 10 seconds

The Association of Commuting Mathematicians (ACM) has a long-standing tradition of debating the latest developments in science, technology, engineering, and mathematics, during subway trips. While on a subway trip, all ACM members strictly abide by the association's first ground rule: "Don't walk and talk." Thus, the debate is interrupted whenever the trip requires a transfer from one subway line to another. ACM members also fit the stereotype of the absent-minded professor, so they always forget the current topic when they are transferring, and open an unrelated new discussion when taking their seats on the new line. Debates are consequently hardly ever conclusive.

ACM members have discussed the problem on the subway, and agreed that they would need a fast and simple way to get around the city using the subway, while minimizing *both* the travel time *and* the number of line transfers. They were about to find an optimal solution for this task, but unfortunately they had to transfer and started talking about radio networks and groundhogs instead... It is thus up to you to solve their annoying problem.

The subway network is a graph consisting of *stations* and *subway lines*. A *station* is identified by a nonnegative integer number, e.g., 42. A *subway line* is a sequence of stops, each stop being a station, with a travel time between each pair of adjacent stations. For instance, in Example 1 of Figure 4, there are three lines: the blue line, the green line, and the orange line. Subway cars follow the line in both directions: travel times in either direction are the same, and the waiting time when transferring from one line to another is neglected. The stops of a line always correspond to different stations, except that lines may be cyclic: in this case, the last stop is the same station as the first stop, and the subway cars follow the line in both directions.

Knowing the details of the subway network, you must find the best travel route for the ACM members from a departure station to a destination station. The first objective of the algorithm is to minimize the travel time. In addition, among all routes having the minimal travel time, the algorithm should also minimize the number of transfers from one line to another. Note that it is never possible (or useful) to transfer from one line to itself.

Input

The input consists of several test cases. The first line contains an integer indicating the number of test cases. Each test case follows. The first line of a test case consists of two positive integers $0 < N \leq 1000$ and $0 < L \leq 50$ separated by a single space: N indicates the number of stations and L indicates the number of subway lines. This is followed by L lines describing each subway line, with each line consisting of the following integers separated by single spaces: the first integer $1 < K_i \leq N + 1$ indicates the number of stops on the line, and the next $2 \cdot K_i - 1$ integers $S_{i,1}, T_{i,1\leftrightarrow 2}, S_{i,2}, T_{i,2\leftrightarrow 3}, \ldots, S_{i,K_i-1}, T_{i,(K_i-1)\leftrightarrow K_i}, S_{i,K_i}$ specify the stops and the travel time between the stops. Specifically, for $1 \leq j \leq K_i$, the integer $0 \leq S_{i,j} < N$ describes the station at the i-th stop and, for $1 \leq j < K_i$, the integer

$0 < T_{i,j\leftrightarrow(j+1)} \leq 60$ describes the travel time in minutes between the stops j and $j+1$ on this line. Cyclic lines may only form a single cycle, i.e., the first station of one line is also the last station of that line. All non-extremal stations have to be different, meaning that $S_p \neq S_q$ for any $1 \leq p < q \leq K_i$ except possibly when $p = 1$ and $q = K_i$. The test case ends with a line that consists of two integers $0 \leq F < N$ and $0 \leq D < N$ separated by a single space: F indicates the departure station and D indicates the destination station. We always have $F \neq D$, and a guarantee that there is a path between the stations F and D.

Output

For each test case in the input, your program should produce one line consisting of two integers separated by a single space. The first integer should be the smallest number of minutes of a path that goes from the departure station to the destination station. The second integer should be the smallest number of transfers of a path that goes from the departure station to the destination station in the smallest number of minutes. There should be no blank lines in your output.

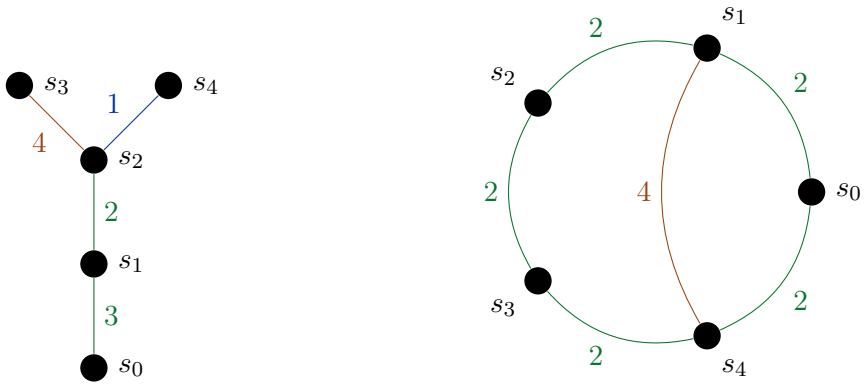
Example

The sample input below specifies two test cases. The subway network of each test case is illustrated by Figure 4. The first example consists of three subway lines. The green line stops at stations 0, 1, and 2, with travel times of 3 and 2 minutes respectively. The orange line connects 2 and 3 with a travel time of 4 minutes. The blue line connects 2 and 4 with a travel time of 1 minute. The ACM members would like to go from station 0 to station 4. The best possible path takes 6 minutes and requires one change, from the green line to the blue line.

The second example consists of two lines. The green line forms a circle connecting all stations in the network and taking 2 minutes to go between any two consecutive stations. The orange line directly connects stations 1 to 4, with a travel time of 4 minutes. The mathematicians want to go from station 1 to 4. There are two fastest routes, each of which has a travel time of 4 minutes: to go via the green line, or by the orange line. Further, neither of these two options requires a line change.

Sample Input

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



(a) Example 1 (b) Example 2

Figure 4: Subway networks of the sample inputs (see below).

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

Sample Output

```
6 1
4 0
```

Problem F: Larger Sport Facility

Time limit: 5 seconds

In a lot of places in the world, elite universities come in pairs and their students like to challenge each other every year. In England, Oxford and Cambridge are famous for *The Boat Race*, an annual rowing race that opposes them. In Switzerland, students from Zürich and Lausanne battle in a famous annual ski challenge.

TelecomParisTech and *Eurecom*, two famous French schools, are also planning to organize a multi-sport tournament. They have already agreed on the choice of sports and the rules of the game, but the only point of disagreement is on where the contest should be hosted. Indeed, every school has been bragging for years about the wonderful sport facilities that they have.

At last, it was agreed that the competition would be hosted at the school which has the larger rectangular sports field. The only thing left is to determine which school this is: given the size of the fields, determine which school has the field with the larger surface.

Input

The input consists of several test cases. The first line contains an integer indicating the number of test cases. Each test case follows. Each test case consists of a single line containing 4 integers $1 \leq l_t$, w_t, l_e, $w_e \leq 10^9$ separated by single spaces: l_t and w_t represent the length and width of the sports field of TelecomParisTech, and l_e and w_e represent the length and width of the sports field at Eurecom.

Output

For each test case in the input, your program should produce one line. The contents of the line should be the name of the school that has the facility with the larger area: `TelecomParisTech` or `Eurecom`. In case of a tie, the contents of the line should be `Tie`. There should be no blank lines in your output.

Sample Input

```
2
3 2 4 2
536874368 268792221 598 1204
```

Sample Output

```
Eurecom
TelecomParisTech
```