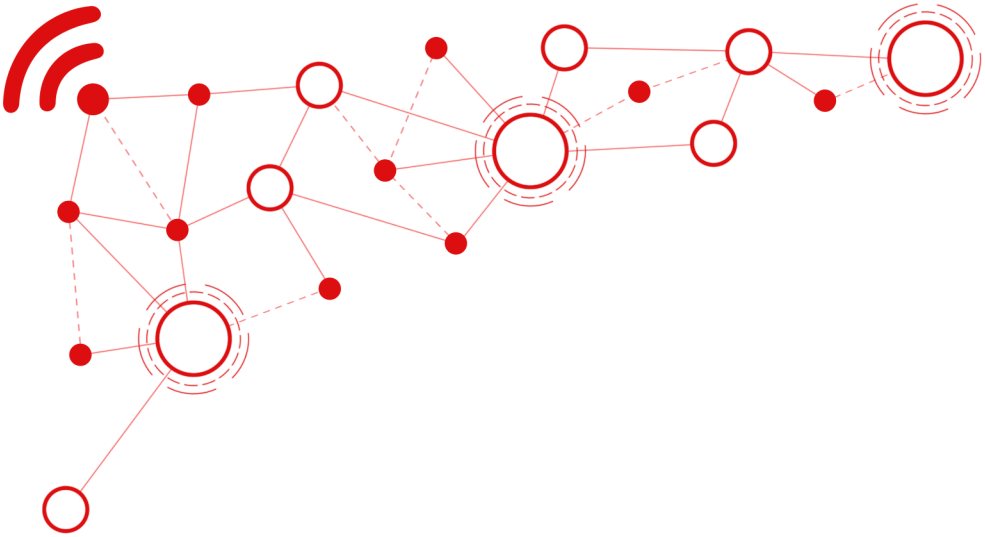
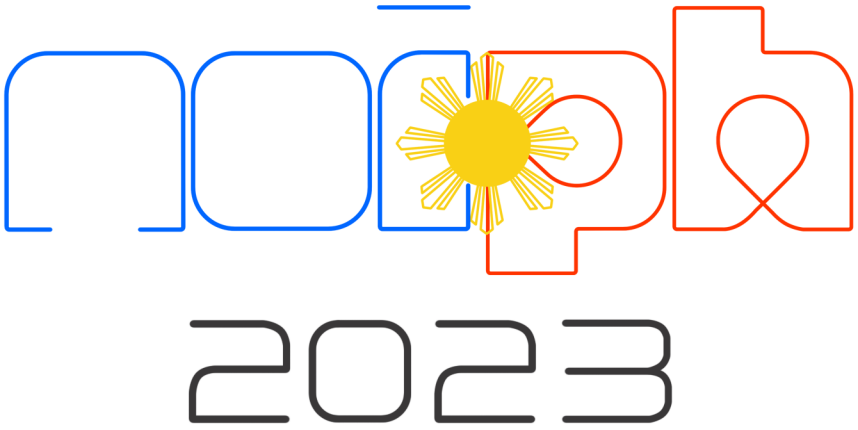
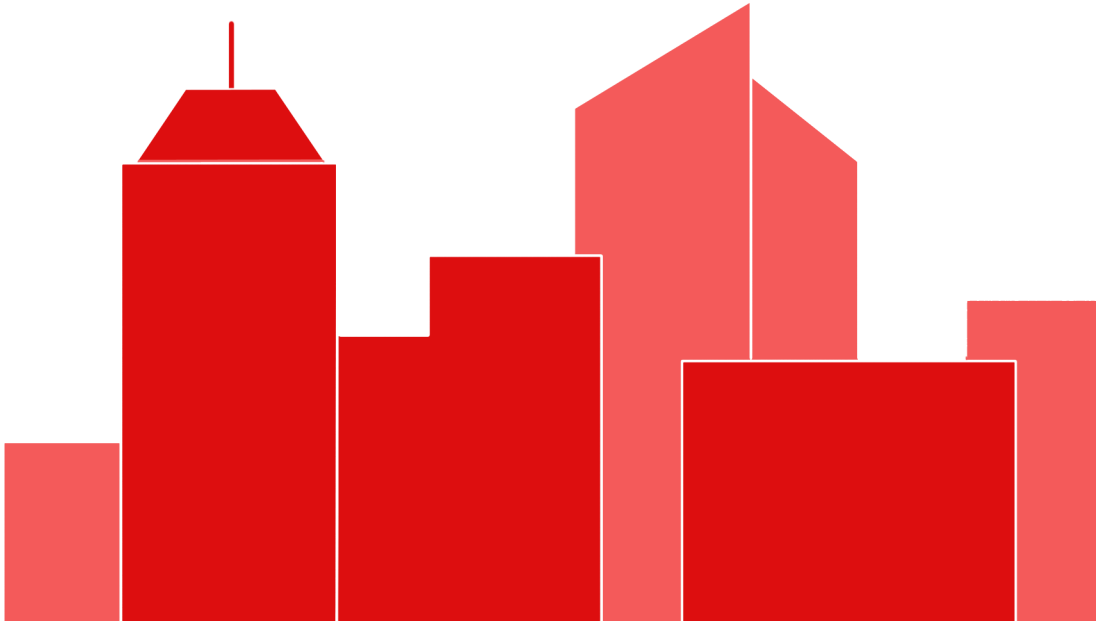




National Olympiad in Informatics

Finals Round 1



Contents

Notes

- Many problems have large input file sizes, so use fast I/O. For example:
  - In C/C++, use `scanf` and `printf`.
  - In Python, use `sys.stdin.readline()`
- On interactive problems, make sure to **flush** your output stream after printing.
  - In C++, use `fflush(stdout);` or `cout << endl;`
  - In Python, use `sys.stdout.flush()` or `print(flush=True)`
  - For more details, including for other languages, ask a question/clarification through CMS.
- Good luck and enjoy the problems!

Important! Read the following:

- Your solution will be checked by running it against one or more (usually several) hidden test cases. You will not have access to these cases, but a correct solution is expected to handle them correctly.
- The output checker is **strict**. Follow these guidelines strictly:
  - It is **space sensitive**. Do not output extra leading or trailing spaces. Do not output extra blank lines unless explicitly stated.
  - It is **case sensitive**. So, for example, if the problem asks for the output in lowercase, follow it.
  - Do not print any tabs. (No tabs will be required in the output.)
  - Do not output anything else aside from what's asked for in the Output section. So, do not print things like "`Please enter t`".

  Not following the output format strictly and exactly will likely result in the verdict "*Output isn't correct*".
- Do not read from, or write to, a file. You must read from the standard input and write to the standard output.
- Only include one file when submitting: the source code (.cpp, .py, etc.) and nothing else.
- Only use letters, digits and underscores in your filename. Do not use spaces or other special symbols.
- Many problems have large input file sizes, so use fast I/O. For example:
  - In C/C++, use `scanf` and `printf`.
  - In Python, use `sys.stdin.readline()`

  We recommend learning and using these functions during the Practice Session.
- On interactive problems, make sure to **flush** your output stream after printing.
  - In C++, use `fflush(stdout);` or `cout << endl;`
  - In Python, use `sys.stdout.flush()` or `print(flush=True)`
  - For more details, including for other languages, ask a question/clarification through CMS.
- Good luck and enjoy the contest!

Problem A
JLO Lawsuit Organization

The Justice League of Ormoc has announced that they are funding the construction of a brand new park! After all, they believe that as wealthy and powerful superheroes, they have an obligation to improve society by using their resources and influence in order to proactively push for social reform, rather than only reactively respond to perceived threats against the status quo that benefits the elite.

What?

Anyway, the main plaza of the park can be modeled as a rectangular grid with r rows and c columns. Every square of this grid must be covered by a brick, and it must be covered in its entirety.

We have two types of **bricks** available to us:

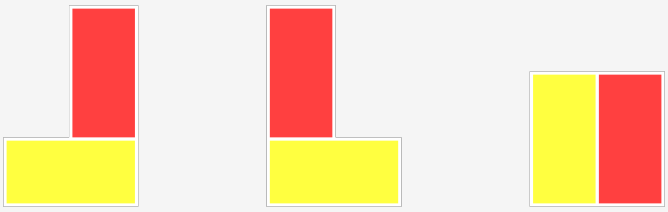
- A 1×2 brick, ethically-sourced from local and sustainable industries.
- A 1×1 luxury brick, each one individually flown over from Switzerland and covered in flavorless gold foil.

For obvious reasons, they would like to minimize the number of luxury bricks that they have to use when covering the plaza. This is made a tad bit more complicated by the fact that they may not break any of the bricks, and also no part of any brick may extend beyond the $r \times c$ rectangle.

Oh, and if that wasn't enough, they also received this missive from the "JLO Lawsuit Organization" of the "JLO Légion Officielle":

To whom it may concern,

Although the Justice League of Ormoc bears the initials of JLO, that does not make you an *official* JLO member. Therefore, you are henceforth prohibited from forming any of the following three figures **or their rotations** anywhere *on the floor* of your plaza.



We call these the `J`, `L` and `O` figures, respectively. Each figure consists of exactly two touching 1×2 bricks.

Failure to comply will result in legal action. Thank you!

Disrespectfully yours,

- J. Lo
- John Locke
- Jake Logan
- Jennifer Lorence
- Jude Low
- Jorge Locas
- John Lonnen
- Jacob Lotner
- Japan Lovers
- Jacquard Loom
- Jolly Longganisa
- Jennifer Love-Hewitt
- Jake Long
- Jaena, Lopez
- Joseph-Louis Logronge
- Joan Lona
- Julia Loberts
- John Loydcruz
- Jueteng Lords
- Jo Logs
- Jumbo Lollipop
- Jackpot Lotto

(the list of signatures goes on for several more pages)

The Justice League of Ormoc doesn't have time for this nonsense, so they delegate this task to you while their Justice Loyers handle the legal shenanigans. What is the minimum number of 1×1 bricks that need to be used in order to completely cover the $r \times c$ plaza, while following the ban on the `J` `L` `O` figures? Better yet,

demonstrate your answer by giving one such tiling.

The Justice League of Ormoc realizes that this problem may seem intimidating, so they remind you that real superheroes ***tackle the subtasks first*** in order to gain insights for solving the full problem (as well as partial points).

Input Format

The first line of input contains an integer t, the number of test cases. The t test cases' descriptions follow.

Each test case is described by a single line containing two space-separated integers r and c.

Output Format

For each test case, output a tiling of the plaza.

More precisely, for each test case, output r lines, each containing a string with c characters, where each character represents a square in the plaza.

- If a square is represented by a period `.` then we interpret that as a 1×1 luxury brick covering that square.
- All other squares should contain uppercase English letters. Each such square should have **exactly one** neighboring square directly to its north, south, east, or west, that is represented by the same letter as it. We interpret this as a 1×2 brick covering both squares.

Your solution will be accepted if

- it satisfies all the formatting specifications described in this section;
- the number of 1×1 bricks used is minimal;
- it contains none of the forbidden shapes.

Constraints and Subtasks

For all subtasks
$1 \leq t \leq 800$ $1 \leq r, c \leq 40$

Subtask	Points	Constraints
1	**17**	$r = 1$
2	**17**	$r = 2$
3	**11**	$r = 3$
4	**11**	$r = 4$
5	**7**	$r = c$
6	**11**	$r < c$
7	**11**	$r > c$
8	**15**	No additional constraints.

Sample I/O

Input 1	Output 1
1 1 2	GG

Input 2	Output 2
1 2 3	II. .OO

Explanation

Sample Input 1 is valid for subtasks 1, 6 and 8, while Sample Input 2 is valid for subtasks 2, 6 and 8. Figure 1 illustrates the sample outputs.

(a) Sample Output 1. There are no luxury bricks.

(b) Sample Output 2. There are two luxury bricks.

Figure 1: Depiction of the sample outputs. The yellow 1×1 bricks are the luxury bricks.

Note that there are multiple valid answers for the test case in Sample Input 2 (aside from just using different uppercase letters). Here are two more:

```
.XX
YY.
```

or

```
X.Y
X.Y
```

These are illustrated in Figure 2.

Figure 2: Depiction of valid alternative outputs for Sample Input 2. Each of these has two luxury bricks, which is the minimum possible for $(r, c) = (2, 3)$.

On the other hand, the three tilings illustrated in Figure 3 are all invalid.

Figure 3: Depiction of *invalid* outputs for Sample Input 2.

Problem B
Residential Evil (Randomizer %)

Residential Evil is a beloved franchise of horror games that forces each player to confront one of humanity's primal fears—gated communities! Watch as house ownership becomes less and less affordable for middle class families! *Spoooooky!*

Gameplay takes place in Umbrella Residences, an upscale village in Racoon City which comprises n **houses** labeled 1 to n, and m two-way **passages** that have a house on each end. You cannot travel between houses except by using these passages.

Some of these passages might be blocked by a sequence of locked **gates**. (Such safety precautions became necessary, after the Racoon Virus.) Gates are unlocked using a **key** of the appropriate type. Each gate and each key are identified with a positive integer ID, and a key can be used to unlock a gate if their IDs match. There is only one key with each ID, but it is possible for many gates to share the same ID. For convenience, if there are k keys in total, then their IDs will be the positive integers from 1 to k.

While a passage is blocked by at least one locked gate, travel along it is impossible. Furthermore, gates must be unlocked *in the correct order*. You cannot unlock a gate along some passage if there is still some *other* locked gate between it and the house you're on. Note that these passages are two-way, so "the correct order" depends on which house you're starting from. See Figure 4.

You may hold at most x keys at any given time. You start the game holding some set of keys (possibly empty) and all other keys must be acquired by exploration.

A house may hold at most y keys at any given time. Each house initially has some set of keys (possibly empty) that you can later collect.

Each house may also (or may not) contain a **backpack** item. Picking one up means that from this point on, you can carry another extra b keys with you while exploring.

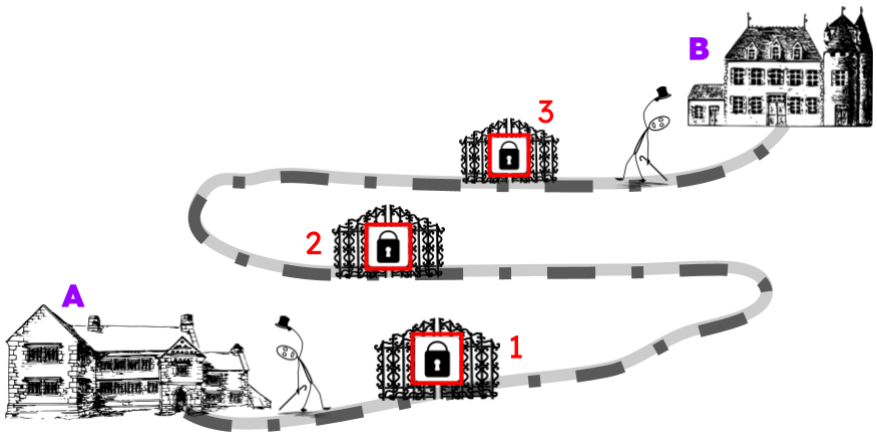


Figure 4: From house A, you can only unlock the gates in the order $1 \to 2 \to 3$. From house B, you can only unlock the gates in the order $3 \to 2 \to 1$.

Formally, here are all of the moves available to you. You can perform any sequence of moves that you like as you explore the game, any number of times, but only *one at a time* (i.e., you can't do two moves "at the same time").

- If the house that you are in has a backpack, then you can **pick it up**, increasing your key-carrying capacity by b (formally, applying the update $x \leftarrow x + b$). Note that picking up multiple backpacks *does* cause their effects to stack.
- If you are holding $< x$ keys, and the house you are in has a key, then you may choose a key from the house and **take** it, adding it to your inventory. You now carry it with you while exploring.
- If the house that you are in has $< y$ keys, then you may choose a key in your inventory and **leave** it on the floor. It will remain there, but you can come back to pick it up later.
- If a locked gate has the same ID number as a key in your inventory, and there are no other locked gates between it and the house you are at, then you may **unlock** that gate. The gate is unlocked forever from now on, but note that this does **not** consume the key (it stays in your inventory and can be used again later).
- If there is a passage between your current house and some other house, and all the gates along this passage (if any) are unlocked, then you can **travel** from your current house to that other house.

Notably, keys are **never** destroyed—only shuffled around between your inventory and the various houses.

Now, in a normal playthrough, it is possible to reach every house through the nat-

ural progression of the game. But suppose you're playing a challenge run with an entirely different configuration of passages, keys and backpacks, and locked gates. Unfortunately, this means it could be possible for some houses to be unreachable, for example if the key to some gate is located at a house blocked off by that gate.

Given the player's starting house s, and the layout of the houses and passages, including the locations of the keys and gates, how many different houses can the player visit?

Input Format

The first line of input contains a single integer t, the number of test cases. Each of the t test cases is described as follows.

The first line of each test case has five space-separated integers n, x, y, b, and s.

This is followed by two lines, describing the backpacks. The first line contains a nonnegative integer p, the number of houses with backpacks. The second line contains p space-separated integers $P_1, P_2, \ldots, P_p$, the houses that initially have a backpack in them. (If $p = 0$, then this second line is empty.)

This is followed by another two lines, describing the keys. The first line contains the nonnegative integer k, the number of keys. Recall that the keys have IDs 1 to k. The second line contains k space-separated integers $K_1, K_2, \ldots, K_k$. (If $k = 0$, then this second line is empty.) The meaning of the K_is are as follows:

- If K_i is 0, then key i starts in your inventory.
- Otherwise, key i is initially contained in house K_i.

Then, the next line contains a single integer m, the number of passages. Then, m lines follow, each describing a passage.

Each line contains some number of space-separated integers. The ith line begins with two positive integers u_i and v_i, the labels of the two houses connected by this passage. Next, a non-negative integer l_i, the number of locked gates along this passage. Finally, a sequence of l_i integers, the IDs of the gates along this passage *in the order you would pass through them* from house u_i to house v_i (approaching this passage from house v_i will naturally have the gates in reverse sequence).

Note that there may be multiple passages that connect the same pair of houses, and there may be passages connecting a house to itself.

Output Format

For each test case, output a single line containing a single integer—the maximum number of different houses that can be visited in this instance of the game.

Note that the answer is always at least 1—since you can, at the very least, visit the house that you start in.

Constraints and Subtasks

Here,

- $\ell = l_1 + l_2 + \ldots + l_m$ be the total number of locked gates in a test case;
- let N, M, K and L, be the respective sums of n, m, k and ℓ across all test cases.

For all subtasks

$1 \le t \le 2 \times 10^4$
$1 \le n$
$0 \le m$
$0 \le k$
$0 \le l_i$
$1 \le x, y, b \le 3$
$1 \le s \le n$
$0 \le p \le n$
$1 \le P_i \le n$
$0 \le K_i \le n$
$P_i \ne P_j$ for $i \ne j$, i.e., the P_is are distinct.
The player character is initially holding at most x keys.
Each gate ID is between 1 and k, inclusive.
Each room initially has at most y keys.

Subtask	Points	Constraints
1	**22**	$s = P_i$ for some i, i.e., the starting room has a backpack. $n \leq 80$, $N \leq 2000$ $m \leq 160$, $M \leq 4000$ $k \leq 8$, $K \leq 200$ $\ell \leq 160$, $L \leq 4000$
2	**15**	$n \leq 80$, $N \leq 2000$ $m \leq 160$, $M \leq 4000$ $k \leq 8$, $K \leq 200$ $\ell \leq 160$, $L \leq 4000$
3	**32**	$n \leq 80$, $N \leq 2000$ $m \leq 160$, $M \leq 4000$ $k \leq 80$, $K \leq 2000$ $\ell \leq 160$, $L \leq 4000$
4	**8**	$n \leq 6000$, $N \leq 24000$ $m \leq 12000$, $M \leq 48000$ $k \leq 6000$, $K \leq 24000$ $\ell \leq 12000$, $L \leq 48000$
5	**23**	$n \leq 250000$, $N \leq 250000$ $m \leq 350000$, $M \leq 350000$ $k \leq 250000$, $K \leq 250000$ $\ell \leq 350000$, $L \leq 350000$

Sample I/O

Input	Output
1 7 1 1 2 3 3 2 6 3 6 1 2 0 7 3 5 7 1 2 3 1 4 2 1 3 1 3 3 2 4 5 6 1 6 3 4 0 4 5 0 5 6 1 4 2 7 6 1 2 3 4 5 6	5

Explanation

In the sample input, there are $n = 7$ houses, $m = 7$ passages, 3 rooms with backpacks, and 6 keys, laid out as shown in Figure 5.

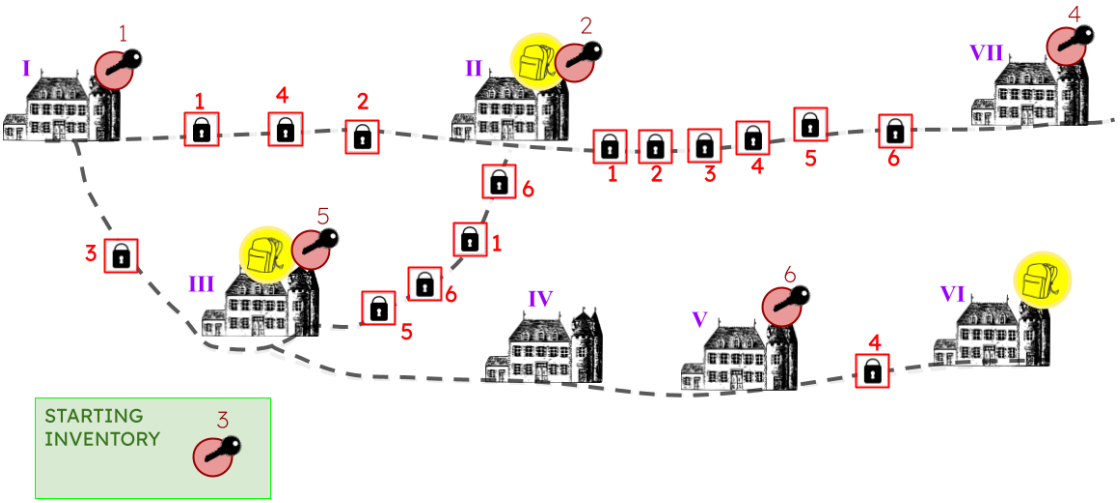


Figure 5: The layout for the sample input. The house numbers are written in roman numerals.

Initially, your player can only carry $x = 1$ key at a time, and each room can only hold at most $y = 1$ key at a time. Finally, every backpack you pick increases your carrying capacity by $b = 2$ extra keys. Your starting location is house $s = 3$.

With the right sequence of moves, you can visit 5 houses—namely houses 1, 2, 3, 4 and 5 (but not necesssarily in that order)—and it can be shown that this is the maximum possible.

Problem C
Ang Dakilang LoomPoints

To increase his chances of getting nice Aguinaldos for Christmas, Apol would perform the Macarena in front of his family members every Christmas.

For those too young to know the Macarena, it is a traditional dance in the 90s wherein "one Macarena" culminates with the dancer simultaneously jumping in the air and facing 90 degrees to their left.

For some unrelated reason, Apol contracted polio and became wheelchair-bound.

Young Apol discovered that this wheelchair was mystical in that it was linked to a set $\mathcal{S}$ of n *distinct* special points in the Cartesian plane, which we call our **loom points**. This magic reveals itself whenever he dances the Macarena.

Whenever he dances *one Macarena*, the wheelchair allows him to choose any loom point that he can currently see *other than his current location*, and then teleport to that point. Apol can see all points that are reachable by walking some (possibly zero) steps forward and some (possibly zero) steps to his left or to his right. (See Figure 6.) Still respecting the traditional Macarena dance, the wheelchair (and hence Apol) would then face 90° *to his left* after each teleportation; note that the wheelchair *always* turns left, regardless of the chosen teleportation point.

For example, suppose Apol is on the point $(6, 9)$ in the Cartesian plane and is facing the positive y direction, and also suppose that the point $(4, 20)$ is in the set $\mathcal{S}$. Apol can see all points whose y-coordinate is greater than or equal to 9.

We can show that he can see $(4, 20)$ from this position, so he can choose to teleport there after one Macarena. We also emphasize the fact that he *cannot* remain on $(6, 9)$ even if it were a loom point, because the point he chooses to teleport to cannot be his current location. After he teleports though, he would be facing the negative x direction, because that's 90° to the left of the positive y direction. (See Figure 7.)

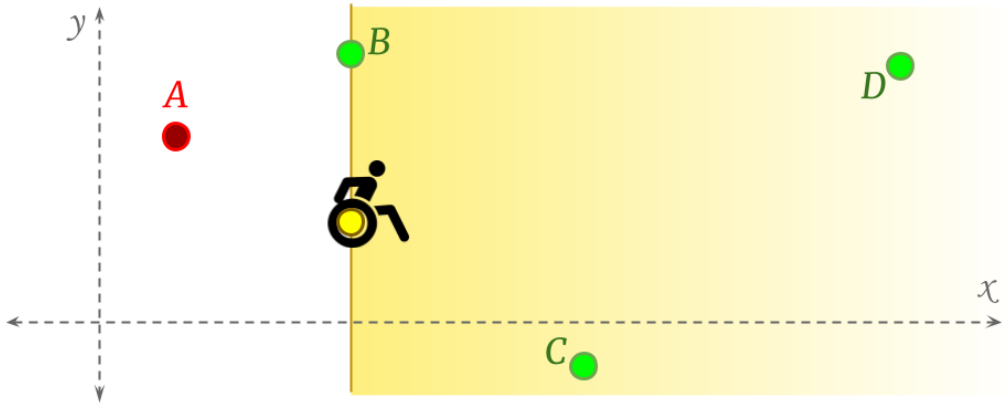


Figure 6: Assuming points A, B, C and D are in $\mathcal{S}$, Apol can teleport from his current location to B, C or D, but not to A.

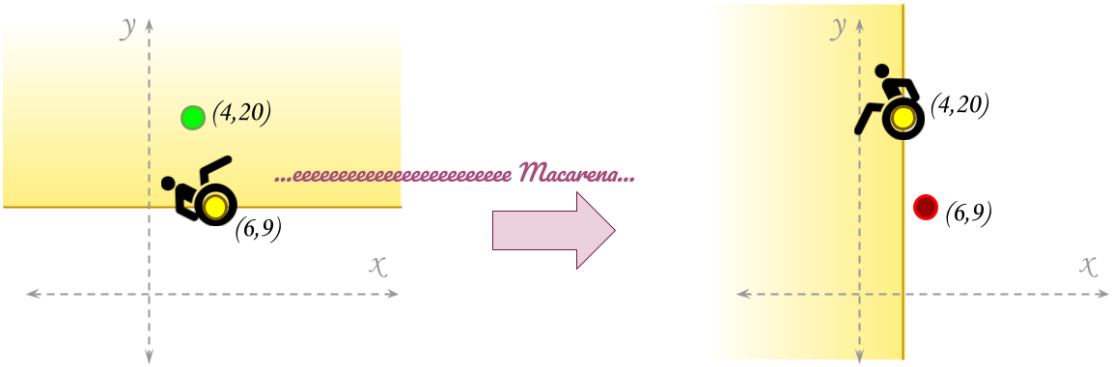


Figure 7: Teleporting from $(6, 9)$ and facing the positive y direction to $(4, 20)$ and facing the negative x direction.

In this problem, we will give you the initial set $\mathcal{S}$ of n distinct loom points and then q **operations** of any of the following two forms:

- **A replacement operation**. You're given two points P and Q where:
  - P is currently in $\mathcal{S}$
  - Q is currently not in $\mathcal{S}$

  Then, apply the following updates to $\mathcal{S}$:
  - Delete P from $\mathcal{S}$
  - Insert Q into $\mathcal{S}$
- **A question**. You're given two points U and V, and a direction d, where:
  - U is any point in the Cartesian plane,
  - V is any point currently in $\mathcal{S}$,

- d is one of the four cardinal directions `N`, `E`, `W`, `S` (more information in the input format).

When you encounter a question, you must output whether or not the point V can be reached by Apol if he:

- starts at the point U, facing the direction d, and
- only does Macarenas and does not move his wheelchair or change the direction he is facing in any other way.

If you say that such a feat is possible, you must then output the minimum number of Macarenas Apol must do in order to arrive at the point V.

Don't forget!

- The set $\mathcal{S}$ changes its members for every replacement operation and has n elements at all times.
- Doing one Macarena must always teleport Apol and his wheelchair to a point **different** from where he does the Macarena.
- After every teleportation, Apol and his wheelchair faces 90 degrees to his left.

Input Format

The first line of input contains two space-separated integers n and q, whose meanings are described in the problem statement.

The next n lines describe the coordinates of the points in $\mathcal{S}$, the loom points. The ith of these lines contains two space-separated integers x_i and y_i, indicating that the point (x_i, y_i) is in $\mathcal{S}$, and thus a loom point.

The next q lines describe the operations. In particular, the ith of these lines describes the ith operation and is of the form:

- "`!` x_P y_P x_Q y_Q" (without the quotes) indicating that the ith operation is a replacement operation with $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$.
- "`?` x_U y_U x_V y_V d" (without the quotes) indicating that the ith operation is a question with $U = (x_U, y_U)$ and $V = (x_V, y_V)$ and direction d, where d is one of:
  - `N` denoting the positive y direction,
  - `E` denoting the positive x direction,
  - `W` denoting the negative x direction, or
  - `S` denoting the negative y direction.

Output Format

For each question, output a single line containing:

- The string "`APPALL`" (without the quotes) if V cannot be reached by Apol, or
- An integer m if V can be reached by Apol and m is the minimum number of Macarenas Apol must do to arrive at V.

Constraints and Subtasks

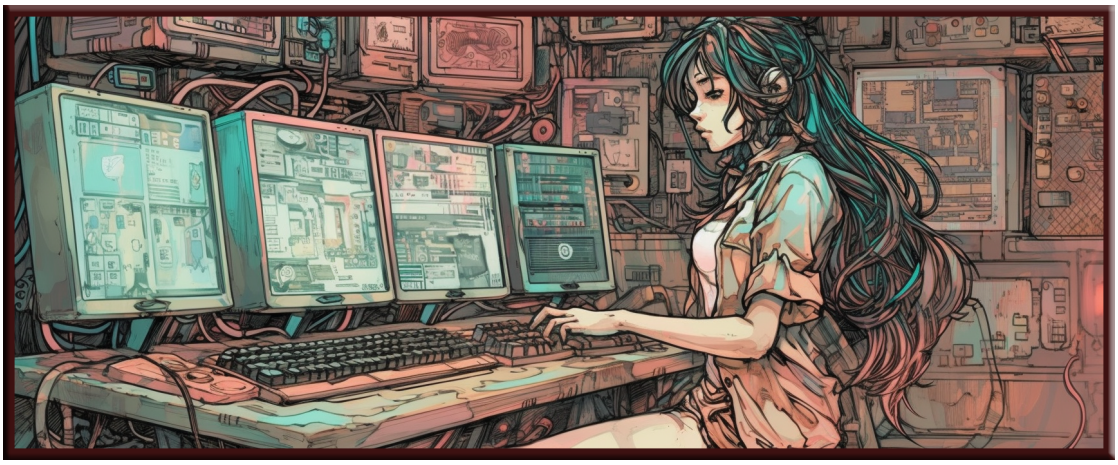
For all subtasks
$n \geq 1$ $q \geq 1$ Every coordinate is an integer in the interval $[1, 10^8]$. For every replacement operation, P is in $\mathcal{S}$ and Q is not in $\mathcal{S}$. For every question, V is in $\mathcal{S}$. The set $\mathcal{S}$ contains n distinct points at all times. d is one of `N`, `E`, `W`, `S`. There is at least one question.

Subtask	Points	Constraints
1	**26**	$n \leq 80$ $q \leq 10^4$ There are no replacement operations.
2	**4**	$n \leq 80$ $q \leq 10^4$
3	**10**	$n \leq 6000$ $q \leq 10^4$ There are no replacement operations.
4	**4**	$n \leq 6000$ $q \leq 10^4$
5	**27**	$n \leq 2 \cdot 10^5$ $q \leq 2 \cdot 10^5$ There are no replacement operations.
6	**29**	$n \leq 2 \cdot 10^5$ $q \leq 2 \cdot 10^5$

Sample I/O

Input	Output
4 5 13 11 8 13 6 8 11 6 ? 10 9 6 8 N ! 13 11 10 14 ! 6 8 8 7 ? 14 10 8 13 S ? 10 6 8 7 S	2 2 APPALL

Problem D
MakaFeeling Ka



> Panalangin ko sa habang buhay...
> ...makafeeling ka, makasama ka,
> ...'yan ang panalangin ko.
>
> – AFO Hiking Society

Maria Clara recently read that empathy is important to strengthen a relationship. Because of this, she wanted to be ready to support her jowa Ibarra when he is sad (😿), have fun with Ibarra when he is happy (😸), and send vegetable emojis to Ibarra when he is in the mood to receive vegetable emojis (🥦).

However, her jowa Ibarra is not very expressive. Desperate for help, Maria Clara goes to the local fortune teller, JudgeGPT4. For a fee, JudgeGPT4 can predict anybody's daily mood for the next n days. You can represent this as a string S of length n consisting only of the characters `h` (for happy), `s` (sad) and `v` (vegetable).

There is one problem though: Maria Clara was poor and JudgeGPT4 would never agree to release the mood string S without a fee.

Luckily for Maria Clara, JudgeGPT4 has an unlimited free trial policy! A **free trial** is described as follows:

> The customer gives JudgeGPT4 a nonempty string T of length at most n. JudgeGPT4 then tells the customer how many times this string occurs in S as a substring so that the customer can later verify that this is right and perhaps become a paying customer later on.

For example, if the string S was `vhvhv` and the customer gives `vhv` as the string T, JudgeGPT4 would reply with 2, because 🌳😸🌳 appears 2 times as a substring in 🌳😸🌳😸🌳:

1. 🌳😸🌳😸🌳
2. 🌳😸🌳😸🌳

Ironically, JudgeGPT4 was not able to predict what happened next! Maria Clara used the free trial an absurdly large number of times and eventually found out the string S without even paying!

Can you do better than Maria Clara?

In this problem, your task is to use JudgeGPT4's free trial as many times as you need to determine the string S. However, the fewer times you use the free trial, the better your score!

Note: Although JudgeGPT4 has an *unlimited* free trial policy, for the purposes of scoring this problem, only answers obtained within the first $q = 200000$ free trials will be considered. Read the Scoring section below for more details.

Interaction

This is an interactive problem.

There will be 10 sets of interactions, each with a different n from the set

$$\{5, 15, 25, 35, 45, 65, 66, 67, 68, 69\}.$$

In all interactions, q is set to 200000.

For each interaction, JudgeGPT4 first prints a line containing the two integers n and q separated by a space. After that, several test cases are run.

For each test case, JudgeGPT4 first uniformly randomly chooses the string S of length n from the alphabet `h`, `s`, `v`. JudgeGPT4 then waits for free trials.

You can use a free trial by printing the line "`TRY` T" (without the quotes), where T is a string of length between 1 and n from the alphabet `h`, `s`, `v`. JudgeGPT4 then responds with a line containing a single integer, the number of times T occurs as a substring of S.

You can answer a test case by printing the line "`ANS` T" (without the quotes), where T is a string of length n. JudgeGPT4 responds with a line containing a single integer c, which is

$$c = \begin{cases} 1 & \text{if } S = T \text{ and } \texttt{TRY} \text{ has been used at most } q \text{ times so far,} \\ 0 & \text{if } S = T \text{ and } \texttt{TRY} \text{ has been used more than } q \text{ times so far,} \\ -1 & \text{if } S \neq T. \end{cases}$$

The next test case starts immediately after this, except if $c = -1$ (see below).

You can choose to end the interaction at any time by printing a line containing the string "`END`" (without the quotes). JudgeGPT4 will print a line containing the integer -1. If JudgeGPT4 receives wrong or incorrectly formatted output, it ends the interaction abruptly by printing -1.

After more than q free trials are used, JudgeGPT4 will still respond to more free trials, but answers after that point won't be counted towards the total score. The only way to gracefully end the interaction is to print `END`.

Finally, once JudgeGPT4 has printed -1 at any point, *it will stop responding*, and your program should exit as well.

Notes:

- Scores are reproducible; for each n, JudgeGPT4 uses the same sequence of randomly-chosen strings S.[1]
- Make sure to **flush** your output stream after printing; otherwise, the judge will not be able to receive your output, and it will wait forever.
  - In C++, use `fflush(stdout);` to flush the stream, or `cout << endl;` to print a newline character and then flush the stream.
  - In Python, use `sys.stdout.flush()` to flush the stream, or use `print(flush=True)` to print a newline character and then flush the stream.
  - For more details, including for other languages, ask a question/clarification through CMS.

Scoring

Each of the 10 interactions is worth up to 10 points and is independent of the others.

For each interaction, if you give an incorrect answer, print invalid output, or fail via one of the other common errors (compilation failed, runtime error, time limit exceeded, etc.), your score will be 0. Otherwise,

- let t be the number of test cases you've answered in your submission within the first 200000 free trials; note that this is also equal to the number of times JudgeGPT4 responded to an `ANS` line with 1;
- let $t_{\max}$ be the maximum t among the author's and the testers' submissions; and
- let $r = \frac{t}{t_{\max}}$.

[1] For each n, JudgeGPT4 uses a fixed seed for its random number generator.

Then your score will be:

$$\text{score} = \begin{cases} 0 & \text{if } r = 0, \\ 0.222 + 2.2\sqrt{r} & \text{if } 0 < r \leq \frac{1}{8}, \\ 1 + 9r^5 & \text{if } \frac{1}{8} < r < 1, \\ 10 & \text{if } r \geq 1, \end{cases}$$

rounded down to three decimal places.

Sample Interaction

Note: The blank lines are only here to illustrate the chronology. The judge doesn't print blank lines. Your solution shouldn't print blank lines either.

JudgeGPT4	Your Program

`5 200000`
 	
`2`
 	
`1`
 	
`0`
 	
`1`
 	
`1`
 	
`1`
 	
`-1`
	
	`TRY vhv`
	
	`TRY hvh`
	
	`TRY ss`
	
	`ANS vhvhv`
	
	`TRY vhsvv`
	
	`ANS vhsvv`
	
	`END`

Explanation

In this interaction, $n = 5$ and $q = 200000$. JudgeGPT4 printed these on the first line, separated by a space.

In the first case, the solution used 3 free trials:

- In the first free trial, the solution program chose $T =$ `vhv`. JudgeGPT4 responded with the integer 2, indicating that `vhv` appears two times as a substring of S.
- In the second free trial, the solution program chose $T =$ `hvh`. JudgeGPT4 responded with the integer 1, indicating that `hvh` appears once as a substring of S.

- In the third free trial, the solution program chose T = `ss`. JudgeGPT4 responded with the integer 0, indicating that `ss` doesn't appear as a substring of S.

It turns out that there's only one string of length 5 satisfying these, and that is the string `vhvhv`, so the solution printed `ANS vhvhv`. JudgeGPT4 responded with the integer 1 indicating that the answer is correct, and that at most 200000 free trials have been used so far, so this correct answer counts towards the total score.

JudgeGPT4 started the next test case right after that.

In the second test case, the program was somehow able to divine the string `vhsvv`; that's why it was able to answer it with just one free trial. (The program could as well have been bold and answered the question straightaway—with no free trial used—but it used one free trial out of politeness to JudgeGPT4.)

Finally, the solution printed `END`, ending the interaction.

For this interaction, there were $t = 2$ test cases answered within the first 200000 free trials. Assuming that $t_{\max} = 16$ (for the purpose of illustration only), we have $r = \frac{2}{16} = \frac{1}{8}$, so

$$\text{score} = 0.222 + 2.2\sqrt{r} = 0.9998174593...,$$

which is 0.999 when rounded down to three decimal places, so the solution would get 0.999 points for this interaction.

Local Testing Tools

An interactive testing tool is provided to aid local testing, downloadable in the CMS interface. Download the following files:

- `judgegpt4.py`. This program acts as JudgeGPT4, as described in the Interaction section above. Note that you can also run this program on its own and interact with it by manually typing "`TRY` T", "`ANS` T" and "`END`".
- `interactive_runner.py`. This makes two programs interact with each other, connecting the output of one program to the input of the other, and vice versa.

Read the comments at the top of these files to learn more.

To run the testing tool and make it interact with your solution, in your terminal, run

```
pypy3 interactive_runner.py  pypy3 judgegpt4.py  --  [COMMAND]
```

where `pypy3` can be replaced by the command you use to invoke Python 3 in your terminal, and `[COMMAND]` is how you invoke your solution.

The testing tool also accepts arguments such as `-n` and `-q`; run the following for more information:

```
pypy3 judgegpt4.py --help
```

If your solution would be marked as Wrong Answer, the testing tool will output an appropriate message. Otherwise, it will output the number of test cases answered within the first q free trials.

Problem E
The Deathly Halo-Halos

In the 90s, the British magical community was experiencing another deadly civil war. The return of the Dark Lord brought about a reign of terror and unprecedented discrimination. His takeover of the ministry was only foiled by a secret resistance group led by the hero of the previous Wizarding World War, and his champion, the Boy-Who-Lived.

Meanwhile in the Philippines, the Filipino magical community was figuring out how to make enchanted food. It's very important! Signora Daza, matriarch of the Daza clan, noticed that magical potions tend to have effects that are more than the sum of their ingredients. She used this principle to magically create *the world's best halo-halo*. It was so good, that eating too much of it at once might cause *death*, as the human mind isn't capable of handling so much bliss at once. For this reason, her recipe was called the *deathly halo-halo*.

You wish to recreate the deathly halo-halo. Signora Daza, in an effort to make the deathly halo-halo more difficult to recreate (for fear of the damage it could cause), split her halo-halo into n different pieces, and distributed these ingredients across her n **restaurants** (with each restaurant getting one piece).

You have finally tracked down the n restaurants, but they are physically quite far apart. Fortunately, you have noted that the restaurants have n special **fireplace connections**, where each fireplace connection connects two restaurants, and allows for two-way travel between them. Each connection can be used as many times as you wish, *but* you have to pay that fireplace's toll fee *each* time you use it. Thankfully, the system has been set up such that it is guaranteed to be able to reach any restaurant from any other restaurant using only these fireplace connections.

Note that n is both the number of restaurants **and** the number of fireplace connections.

Your heist plan involves designing a tour which visits all of Signora Daza's n

restaurants at least once each, *however* once you start, you may only travel between restaurants using the special fireplace connections, and you may only leave when you're done. You are free to choose which restaurant to visit first, and you are also free to choose which restaurant to end on when you make your escape. Among all such possible tours which accomplish this task, find the *minimum total toll fee cost*.

Input Format

The first line of input contains t, the number of test cases. The descriptions of t test cases follow.

The first line of each test case contains a single integer n, the number of restaurants (and also the number of connections). The restaurants are labeled 1 to n.

Then, n lines follow, with each line containing three space-separated integers u, v, and w, meaning that a fireplace connection exists that allows for two-way travel between restaurants u and v, at a cost of w magical pesos per use.

Output Format

For each test case, output a single line containing an integer—the minimum total toll fee cost (in magical pesos) of a tour that visits all restaurants at least once using only the fireplace connections.

Constraints and Subtasks

Here, N is the sum of the ns in a single test file.

For all subtasks
$1 \leq t \leq 250$ $3 \leq n$ $1 \leq u, v \leq n$ $1 \leq w \leq 2 \cdot 10^8$ Every fireplace connection connects two *distinct* restaurants. No two fireplace connections connect the same unordered pair of restaurants. The restaurants are reachable from one another using only fireplace connections.

Subtask	Points	Constraints
1	**18**	$n \leq 9$, $N \leq 200$, $w \leq 10^7$
2	**12**	$n \leq 14$, $N \leq 900$
3	**22**	$n \leq 250$, $N \leq 3000$
4	**16**	$n \leq 3000$, $N \leq 36000$
5	**32**	$n \leq 250000$, $N \leq 500000$

Sample I/O

Input	Output
1 4 3 1 20 2 3 11 2 4 420 3 4 27	69

Explanation

Here's one possible way to visit all $n = 4$ restaurants with the minimum total fee cost of 69 magical pesos:

$$4 \to 3 \to 2 \to 3 \to 1.$$

The total fee cost is $20 + 11 + 11 + 27 = 69$ magical pesos, as expected.