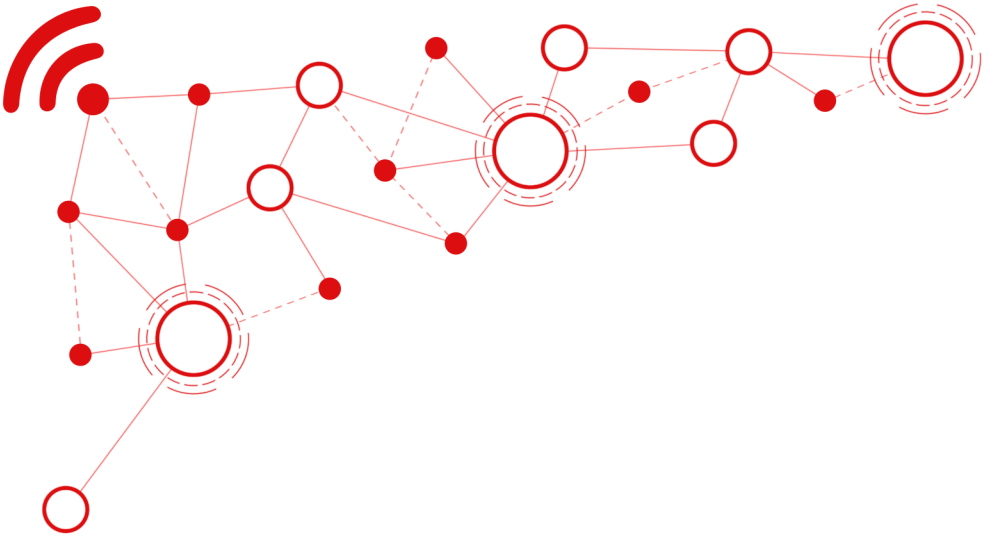
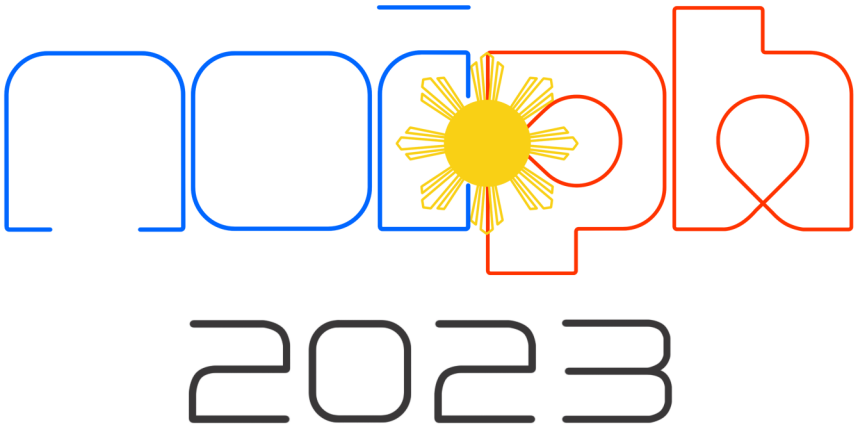
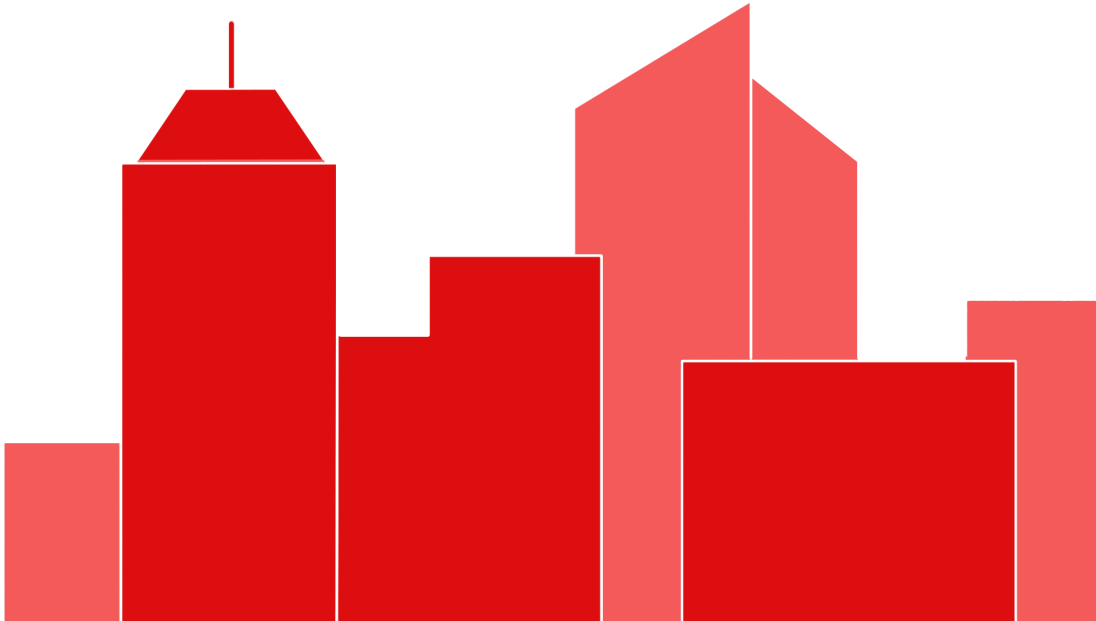




National Olympiad in Informatics

Finals Practice



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()`
- **Practice writing the fast I/O methods above now**, so that you can save some (important) minutes in the real contest later!
- During the practice round, we will answer questions about fast I/O in CMS.
- 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!

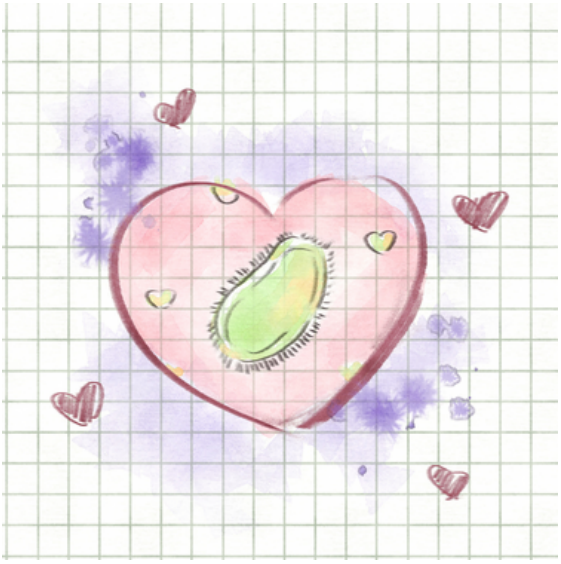
Important! Read the following:

- 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!

Practice Problem A
Decorum Sensing



Alice has been adjusting to her online microbiology class really well! She has a synchronous session with her instructor every week. By far, her favorite part is at the end of the session, when everyone says, "Thank you, sir!" one after another. It gives her a small burst of well-needed wholesome energy for the week. And her teacher friends have confirmed that they really appreciate hearing it as well!

Alice noticed that some students are shier than others. There are n students in her microbiology class. The ith student will say "Thank you, sir!" *only after* at least A_i other students have also said "Thank you, sir!" (if $A_i = 0$, then the ith student says "Thank you, sir!" immediately after class ends). Alice calls this behavior *decorum sensing*, because these students are trying to get a feel for what is appropriate to say or do based on what everyone else is doing.

Given the values of $A_1, A_2, A_3, \ldots, A_n$, find the number of students who will eventually say "Thank you, sir!"

Input Format

The first line of input contains a single integer n.

The second line of input contains n space-separated integers $A_1, A_2, A_3, \ldots, A_n$.

Output Format

Output a single integer, the number of students who will eventually say "Thank you, sir!"

Constraints and Subtasks

For all subtasks
$0 \leq A_i \leq n$

Subtask	Points	Constraints
1	**25**	$1 \leq n \leq 6$
2	**25**	$1 \leq n \leq 100$
3	**25**	$1 \leq n \leq 1000$
4	**25**	$1 \leq n \leq 10^5$

Sample I/O

Input 1	Output 1
10 1 6 1 8 0 3 3 9 8 8	5

Input 2	Output 2
4 1 2 3 4	0

Explanation

In the first sample input, we can show that the 1st, 3rd, 5th, 6th, and 7th students all say, "Thank you, sir!"

- First, the 5th student says, "Thank you, sir!"
- After hearing this, the 1st and 3rd students say, "Thank you, sir!"
- After hearing *them*, the 6th and 7th students say, "Thank you, sir!"
- No further students say anything after that.

In the second sample input, no one will say "Thank you, sir!" How sad.

This problem was borrowed from Abakoda 2021 Long Contest

Practice Problem B

Joker

Cindy thought a fine complement to her athletic prowess would be to learn more fine dexterity and sleight of hand, and what better way to do that than to learn some cool card tricks? The art of *prestidigitation*! *Legerdemain*!

Alice heard about this and excitedly wanted to teach Cindy a special card trick that she had learned. Well, it's actually not really a trick... it's more like a puzzle. A numbers puzzle. But Alice was really excited, so Cindy indulged her.

Cindy combined innumerably many decks, and dealt out n cards face-up on the table. Each card is associated with some numerical value. Two through Ten are each worth the number on them. The "face cards" of Jack, Queen, and King each have a value of 10. For this problem, the Ace *always* has a value of 1. Finally, we have a special card, the **Joker**. The Joker can do anything! It can magically transform into any of the other card types.

Alice and Cindy view the face-up cards that were dealt. Some (possibly none) will be Jokers. Alice gives Cindy some target value m. Cindy must replace each Joker with a non-Joker card, such that the sum of the values of all face-up cards becomes **exactly** equal to m.

Cindy spent so much time practicing the shuffling and the dealing and the flourishes, that she doesn't have any energy left for the computations! Do you think you could help her out?

Input Format

The first line of input contains two space-separated integers n and m, the number of face-up cards and the desired total value.

This is followed by a line containing a string with n characters, encoding the face-up cards. Each character will be one of the following.

- `A` corresponds to an Ace.

- Digits `2` through `9` correspond to the numbers Two through Nine.
- `T` corresponds to a Ten.
- `J`, `Q`, `K` correspond to Jack, Queen, and King, respectively.
- `*` corresponds to a Joker.

Output Format

If the task is possible, output a single line containing the word `YES`; otherwise, output `NO`.

If `YES`, output another line containing a string with n characters. This should be exactly the same string given in the input, except each `*` has been replaced with one of `A23456789TJQK`, such that the total value of all the cards is exactly equal to m. If there are multiple solutions, output any of them.

Constraints and Subtasks

For all subtasks
$1 \leq n \leq 10^5$ $1 \leq m \leq 10^6$

Subtask	Points	Constraints
1	**30**	There are no Jokers.
2	**30**	There is at most 1 Joker.
3	**20**	There are at most 3 Jokers.
4	**20**	No further constraints.

Sample I/O

Input 1	Output 1
5 26 2JAK3	YES 2JAK3

Input 2	Output 2
6 23 3A4A5*	YES 3A4A59

Input 3	Output 3
3 31 ***	NO

Input 4	Output 4
26 27 AAAAAAAAAAAAAAAAAAAAAAAAAA	NO

This problem was borrowed from Abakoda 2021 Long Contest

Practice Problem C
The Road Not Taken

Two roads diverged in a yellow wood.

Robby Frosty and Nascar Wilde find themselves navigating through the Jaundiced Weald, a rite of passage for fledgling poets.

The weald has n **clearings** (labeled 1 to n), with m two-way **roads** that each directly connect two different clearings together. Traveling by these roads is the only way to safely go from clearing to clearing, but thankfully the layout of the weald is such that is it possible to reach any clearing from any other clearing.

Unfortunately, due to the nebulous and metaphorical nature of poetry (and also a thick fog that permeates the forest), **you do not initially know where any of these roads are**. That said, you may perform as many *inspections* as you like: choose two clearings, and you will be told whether a road exists that directly connects those two clearings.

Anyway, despite the two being traveling companions, their preferred paces vary wildly, due to their philosophies in life. To explain them, we first define a **path** from u to v to be a sequence of *distinct* clearings (beginning with u and ending in v) such that a road exists that connects any two consecutive clearings in the sequence (i.e., a "valid travel plan" from u to v). The length of this path is equal to the number of roads traveled (which, we note, is always one less than the number of clearings in the sequence).

- Nascar Wilde has to go fast. If he has a clearing that he wishes to go to, he always travels using the **shortest path** from his current clearing to his destination.
- Robby Frosty is much more chill. If he has a clearing that he wishes to go to, he always travels using the **longest path** from his current clearing to his destination. Note that the longest path is always finite in length because paths, by definition, cannot visit the same clearing more than once.

If there are multiple shortest/longest paths, then they choose one arbitrarily (perhaps they take the one less traveled by, and that makes all the difference). It takes each of them the same amount of time to traverse any road in the weald, so the time it takes for them to travel from one clearing to another is completely determined by the length of their chosen path.

The two of them have asked you to verify whether the Jaundiced Weald satisfies the following property:

> Suppose the two of them start at some clearing u, and they both decide that they wish to go to clearing v (where $u \neq v$). Then, if they leave clearing u at the same time, then they should arrive at clearing v at the same time, and *this should always be true, regardless of the values of u and v.*

Please determine if this condition holds. Also, remember that you don't initially know all the roads in the weald (you are only initially given the values of n and m), so try to solve this problem using as few inspections as possible. The fewer the number of inspections, the better your score!

Now, before you start solving, know that our two heroes are not just poets, but also competent computer scientists—they had taken lessons from the daughter of one of their poetry buddies, and she was a lovely teacher. So, they have some advice which you may or may not find useful, but which they want you to hear.

Nascar Wilde can find shortest paths in a blazingly fast **polynomial time**. He can use the Floyd-Warshall algorithm to determine all pairs of shortest paths in $\mathcal{O}(n^3)$ time with a very low constant factor. Since all roads take the same amount of time to travel through, he could alternatively run a breadth-first search from every possible starting clearing in order to compute the shortest paths in $\mathcal{O}(n(n+m))$, which would be faster than Floyd-Warshall if the weald is sparse.

Robby Frosty, on the other hand, only knows of glacially-slow **exponential-time** algorithms for finding longest paths. We can actually show that the Longest Path problem is NP-Hard by the following chain of reductions:

- The Hamiltonian Path problem can be reduced to Longest Path problem
- The Hamiltonian Circuit problem can be reduced to the Hamiltonian Path problem
- The Minimum Vertex Cover problem can be reduced to the Hamiltonian Circuit problem
- The Maximum Clique problem can be reduced to the Minimum Vertex Cover problem
- 3-Satisfiability can be reduced to the Maximum Clique problem

- 3-Satisfiability is classically known to be NP-Complete by the Cook-Levin Theorem.

Since we don't know whether $P \neq NP$ or not, no polynomial-time algorithms are currently known that can solve the Longest Path problem.

An improvement over the direct $\mathcal{O}(\text{poly}(n) \cdot n!)$ brute force can be made by using dynamic programming with bitmasking to achieve an $\mathcal{O}(\text{poly}(n) \cdot 2^n)$ running time.[1]

Interaction

This is an interactive problem. *This means that your program will be run at the same time as the judge's program, and they will communicate via the standard I/O streams. Your program's output is read as input by the judge's program, and vice versa. Ask questions/clarifications in CMS if you need more details.*

There will be one set of interactions.

First, the judge prints a line containing a single integer t, the number of test cases. After that, several test cases are run.

For each test case, the judge first prints a single line containing two space-separated integers n and m, the number of clearings and roads, respectively. The judge then waits for inspections.

You can inspect a pair of clearings i and j, $1 \leq i, j \leq n$ by printing the line "`INSPECT` i j" (without the quotes). The judge then responds with a line containing a single integer r, which is

$$r = \begin{cases} 1 & \text{if there is a road between clearings } i \text{ and } j; \\ 0 & \text{otherwise.} \end{cases}$$

You can answer a test case by printing a line containing either

- the string "`CONVERGE`" (without the quotes) if the condition above holds, or
- the string "`DIVERGE`" (without the quotes) if the condition doesn't hold.

The judge doesn't respond to this line; it simply proceeds to the next test case. If there are no more test cases, the judge stops responding and exits, and your program should exit as well, otherwise it could keep waiting for input that will never come, and you may get the verdict *Time Limit Exceeded* or *Idleness Limit Exceeded*.

[1] You could also recognize that the Longest Path problem is *fixed-parameter tractable* in the length of the longest path. Supposing that you wish to test for the existence of a path of length d (from u to v), you can use a randomized color-coding algorithm in order to devise a polynomial-time algorithm which succeeds with probability $1/e^d$. The failure rate can be made arbitrarily small by running the randomized algorithm sufficiently many times.

If the judge receives a wrong answer or incorrectly formatted output, it ends the interaction abruptly by printing -1. The judge also stops responding and exits, and your program should exit as well.

Notes:

- 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, or if anything's unclear, ask a question/clarification through CMS.
- The judge is **adaptive**. This means that in some cases, the judge doesn't decide beforehand where the m roads are, but instead may vary them based on the inspections your solution performs. However, it is guaranteed that the judge's responses are *consistent*, that is, after the interaction, you can check that its outputs could have come from a non-adaptive judge that decided beforehand where the m roads were.

Scoring

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 q be the maximum number of inspections your solution performed among all t test cases; and
- let q_{min} be the minimum q among the author's and the testers' submissions. (This represents the best solution obtained by the people who prepared the problem.)

Then your score will be:

$$\text{score} = \begin{cases} 0 & \text{if } 4096 < q, \\ 20 & \text{if } 2023 < q \leq 4096, \\ 20 + \frac{5000}{q - 2q_{\text{min}} + 64} & \text{if } 2q_{\text{min}} < q \leq 2023, \\ 99.5 - \frac{q - q_{\text{min}}}{q_{\text{min}}} & \text{if } q_{\text{min}} < q \leq 2q_{\text{min}}, \\ 100 & \text{if } q \leq q_{\text{min}}, \end{cases}$$

rounded down to three decimal places.

Constraints

Constraints
$t = 143$ $1 \leq n \leq 64$ $0 \leq m \leq 200$ It is possible to go from every clearing to every other clearing using only the roads. There is no road connecting a clearing to itself. There is at most one road connecting a pair of clearings.

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.

Judge	Your Program
1 3 2 0 1 1 0 1 1	 INSPECT 1 3 INSPECT 1 2 INSPECT 3 2 INSPECT 2 2 INSPECT 2 1 INSPECT 3 2 CONVERGE

Explanation

In this interaction, there is only $t = 1$ test case, which is technically disallowed according to the Constraints section, but this is just for illustration purposes.

In the only test case, the judge printed the line `3 2`, indicating that there are $n = 3$ clearings and $m = 2$ roads. The solution program made 6 inspections and discovered that the two roads are between clearings 1 and 2, and also clearings 2 and 3.

With this information, the solution program concluded that the condition holds (see the problem statement), so it printed the line `CONVERGE`. This answer is correct.

Since there's only one test case, the interaction ended after that.

This is clearly not the most optimal way to solve this test case—for one thing, there were redundant inspections, and useless inspections like `INSPECT 2 2`—but it still gets some points.

In this case, we have $q = 6$. Assuming that $q_{\min} = 5$ (for the purpose of illustration only), we have

$$\text{score} = 99.5 - \frac{6-5}{5} = 99.300000...,$$

which is 99.300 when rounded down to three decimal places, so this solution would get 99.300 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:

- `local_tester.py`. This program acts as the judge, as described in the Interaction section above. Note that you can also run this program on its own and interact with it by manually entering the lines that a valid solution is supposed to enter. Just run

  ```
  pypy3 local_tester.py input.txt
  ```

  where `input.txt` is a valid input file (described below).

- `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 local_tester.py input.txt  --  [COMMAND]
```

where

- `input.txt` is a valid input file (described below);
- `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 local_tester.py --help
```

If your solution would be marked as Wrong Answer, the testing tool will output an appropriate message. Otherwise, it will output the value of q (defined above) for the interaction.

The local tester doesn't verify whether your answers (`CONVERGE` or `DIVERGE`) were correct. It also doesn't compute the score for you. (Well, it can't, since it doesn't know q_{min}.)

Local Tester Input Format

This input format is only relevant for the local tester, `local_tester.py`.

The first line must contain a single integer t, the number of test cases. Then the descriptions of t test cases must follow.

The first line of each test case contains two space-separated integers n and m. Then, m lines follow, each containing two space-separated integers i and j with $1 \leq i, j \leq n$, denoting a pair of clearings connected by a road.

Note that the local tester makes very little attempt to validate the input, e.g., to check whether they satisfy the constraints in the Constraints section above. For badly-formatted input, it may crash, but sometimes it may just proceed with the interaction even if the input is technically invalid.

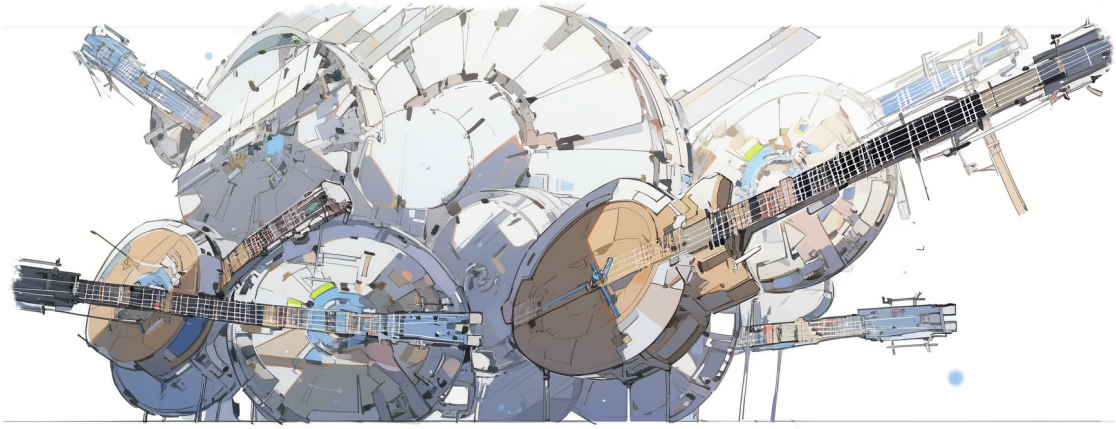
Here's a sample input file. It corresponds to a possible input file used in the sample interaction above.

Local Tester Input

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

Practice Problem D

I used to be a musician, then I took an arrow to the...



It's been a while, and yes, people are still at home. People still struggle to communicate across distant places because of bad connections. Remember Elizabeth? And people need better connection speeds so they can listen to good music and be productive. Definitely not so they could watch random videos online.

Fortunately, a new company has entered this space to provide faster speeds for customers. Don't expect it to be cheap though. While others have tried laying fiber optic cables across the ocean, flashing bits of information through laser beams, or even flying data through hot air balloons, this one's sending data through music! Don't know much about how it works, but this approach is a beat unheard of.

You probably know the company's name by now. It's obviously GuiStarLink.

They called it like that since as it operates, it uses **GuiStar satellites** linked together in a grid of r rows and c columns. These GuiStars are the ones that stream data while playing music.

There's a problem though. Customers have been complaining that those GuiStars are taking up too much of the night sky, blocking their view of their favorite stars!

Now, those GuiStars are equipped with camouflage technology that will make a Helicarrier's tech look cute. But this industry isn't exactly a low-capital business, so sometimes a GuiStar's optical equipment may be turned off, saving costs and making it visible.

But GuiStarLink's CEO, Sgt. Aryus, has heard his customers. Sgt. Aryus is from a race which has horse-like legs and a humanoid form from the waist up. From their tails, they make the strings that the GuiStars use for producing music, as well as for wiring up their bows to shoot their high-tech (or to the uninformed, they may as well be magical) **arrows**. Sgt. Aryus is good at shooting arrows; he

used to do it for less civil purposes, but now utilizes his skills for other purposes like toggling the visibility of the GuiStars.

Due to budget constraints, Sgt. Aryus only has a arrows. Each time an arrow is shot, it will go through either a whole row or a whole column of GuiStars, *flipping the visibility* of each of those GuiStars, that is, turning a visible GuiStar invisible, and an invisible GuiStar visible.

Here's an example. In the following grid, 1 means that the camouflage tech is enabled, so that GuiStar is invisible, while a 0 would mean that it's visible.

```
000
101
110
```

If an arrow is used on the first row, and another arrow is used on the second column, then it would result to:

```
101
111
100
```

Furthermore, aside from being visible or invisible, each GuiStar also has its associated sound value, which relates to its speed and how it streams its data.

So, given that Sgt. Aryus can use at most a arrows, what is the maximum sum of the sound value(s) of all the GuiStars that are invisible that can be obtained, among all possible ways to use at most a arrows?

You must also give one of these ways that achieve this maximum sum.

Input Format

The input starts with a line containing an integer g, the number of GuiStarLink grids. Then g descriptions follow.

Each description starts with a line containing two space-separated integers r and c, the number of rows and columns of that grid. Then r lines follow, each containing a string with c characters that is either a 1 or a 0, describing the rows of the grid.

Another r lines follow. The ith line contains c space-separated integers, the jth of which is the sound value $s_{i,j}$ for the GuiStar in the ith row and jth column.

Finally, a line follows containing an integer a.

Output Format

For each grid, first print a line containing an integer m, the maximum total of sound values of invisible GuiStars. After this, you need to output one possible way to achieve this maximum total.

First, print another line containing an integer f, the number of arrows to fire.

Then f lines shall follow, one for each arrow to use. Each line starts with either '`R`' or '`C`' (without the quotes) depending on whether the arrow was used on a row or a column, respectively. This is followed by a space, and then an integer i representing the index of the row or column at which the arrow was used.

The rows are indexed 1 to r from top to bottom, and the columns are indexed 1 to c from left to right.

If there are multiple valid answers, output any one.

Constraints and Subtasks

For all subtasks
$1 \le g \le 10^4$ $1 \le r, c \le 300$ $1 \le s_{i,j} \le 10^5$ The sum of all rc in a file is at most 10^5

Subtask	Points	Constraints
1	**50**	$1 \le a \le 2$
2	**50**	$1 \le a \le 5$

Sample I/O

Input	Output
2 3 3 000 101 110 7 5 2 4 10 2 1 4 7 1 3 3 000 101 110 7 5 2 4 10 2 1 4 7 2	25 1 R 1 32 2 C 2 R 3

Explanation

In the first case, Sgt. Aryus only has one arrow, so he can only shoot for at most one row or one column. Of all choices, shooting for row 1 results in the following GuiStars (in)visibility:

```
111
101
110
```

Summing up the sound values of all invisible (labeled '1') GuiStars results in the sum 25, and this is the maximum result.

In the second case, the optimal solution is to shoot for column 2 and row 3, resulting in the following GuiStars (in)visibility:

```
010
111
011
```

This results in a sum of 32. Notice that the Guistar in row 3, column 2 remains visible because two arrows passed through it.

This problem was borrowed from Algolympics 2021

Practice Problem E
Señorita

Camila, like many young Filipinos, is a girl in love. She has fallen hopelessly in love with Shawn, her upperclassman in their Programming Club who always helps her debug her code. She wishes that she could pretend she didn't need him, but she knows that he overhears her complaining about her loops not terminating at the right times. "It's true," she lalala-ed. "It should be running."

She doesn't think Shawn even knows her name, since every time they talk, he calls her "señorita." That's perfectly fine with her, since she loves it when he calls her señorita. Thus, to win over his heart, she needs to plan out the perfect outfit for each of the next few days, so she can keep hearing him call her señorita.

She has two stacks of shirts in her cabinet, one with m shirts and the other with n shirts. She is going to wear each of these shirts exactly once over the course of the next $m + n$ days, and each shirt has been assigned a number from 1 to $m + n$ to indicate on what day she wants to wear it.

On day i, Camila wants to wear shirt i from her cabinet. Unfortunately, the shirt that she needs for the day might not be on top, and could be buried under the other shirts. Since her shirts are organized into stacks, Camila only has two operations available to her:

- Camila can take the top shirt of one stack and move it to the top of the other stack. This takes 1 unit of energy.
- If the desired shirt i is on top of either of the stacks, Camila takes it and wears it for the day. This does not cost any energy. The shirt is then thrown into the laundry, and thus permanently removed from the stack (since Camila does not do her own laundry).

She asked her mom if she could arrange her clothes differently, but her mom just replied, "Well, *señorita*, if you have the energy to complain about the way I do the

laundry, maybe you have the energy to do it yourself, then!" Camila decided she doesn't love it as much when it is her mom who calls her señorita. Regardless, her mom is right in that Camila is very lazy, and doesn't want to expend more energy than is necessary in retrieving her shirts.

Given the initial ordering of her shirts, what is the minimum amount of energy that Camila will have to expend in total in order to be able to wear the correct shirt on each of the next $m + n$ days, and have Shawn call her señorita?

Input Format

The input consists of two lines. The first line begins with a single integer m. Then if $m > 0$, this is followed by a space, then m space-separated integers. Similarly, the second line begins with a single integer n. Then, if $n > 0$, this is followed by a space, then n space-separated integers. These represent the two stacks of clothes, with m and n shirts in each stack, respectively, and with the integers themselves representing the day when Camila wishes to wear each shirt. These integers are given in the order the shirts appear in the stack, with the first integer representing the shirt at the **bottom** of that stack, and the last integer of each line representing the shirt at the **top** of that stack.

Each of the integers from 1 to $m + n$ is guaranteed to appear exactly once among the integers in the stacks.

Output Format

Output one line containing the minimum amount of energy that Camila must expend.

Constraints and Subtasks

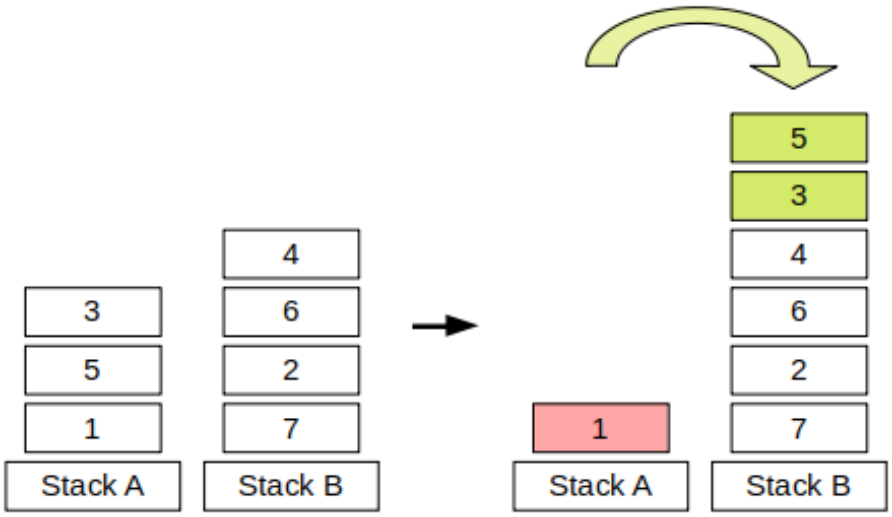
For all subtasks
$0 \leq m, n, m + n \leq 4 \times 10^5$

Subtask	Points	Constraints
1	**30**	$1 \leq m + n \leq 4000$
2	**1**	For each stack, if $i < j$, then shirt i is on top of shirt j.
3	**19**	For each stack, if $i < j$, then shirt i is underneath shirt j.
4	**50**	No further restrictions.

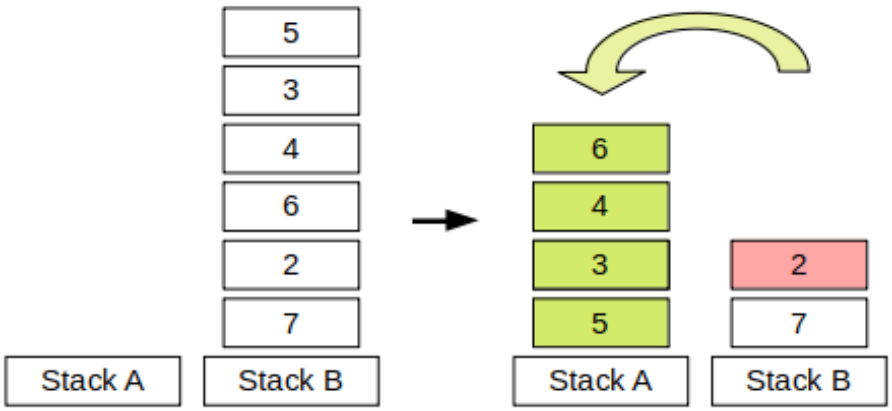
Sample I/O

Input	Output
3 1 5 3 4 7 2 6 4	8

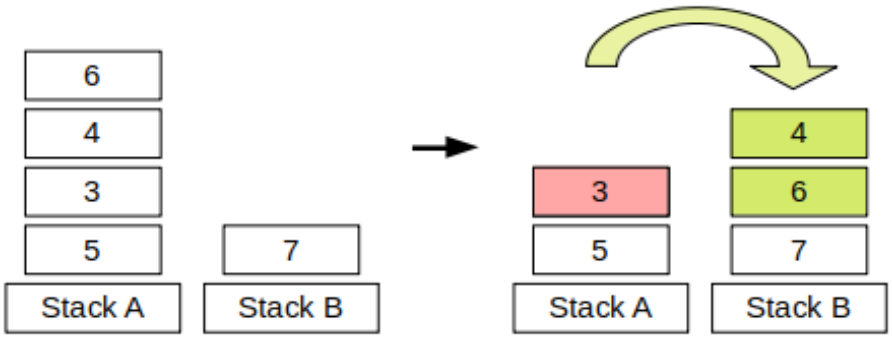
Explanation



First, to retrieve shirt 1, Camila moves shirt 3 and shirt 5 to the other stack, expending 2 units of energy. Then, she can remove shirt 1 from the stacks.



Now, Camila has to retrieve shirt 2. To do this, she moves shirts 5, then 3, then 4, then 6, which are on top of shirt 2, onto the other stack. This costs 4 units of energy. Then, she can remove shirt 2 from the stacks.



Next, Camila has to retrieve shirt 3. To do this, she moves shirt 6 then shirt 4 over to the other stack, expending 2 units of energy. Then, she can remove shirt 3 from the stacks.

From this position, we can show that Camila can retrieve the remainder of her shirts without expending any more energy. We can also show that this series of moves is optimal for minimizing the amount of energy spent. Thus, Camila must spend $2 + 4 + 2 = 8$ units of energy in total.

This problem was borrowed from NOI.PH 2020 Elimination Round