2026 Team Formation Test Day 1

Task Overview

ID	Name	Time Limit	Memory Limit	Subtasks
T2611	Gem Collection	1.000 s	1024 MB	7 + 8 + 15 + 46 + 24
T2612	Mini Metro II	1.000 s	1024 MB	5 + 11 + 10 + 22 + 27 + 25
T2613	Triple Keys	3.000 s	1024 MB	3 + 4 + 14 + 11 + 23 + 27 + 18

Notice:

- All tasks are divided into subtasks. You need to pass all test cases in a subtask to get points.
- There is an attachment package that you can download from the contest system, containing sample graders, sample implementations, example test cases, and compile and run scripts.
- When testing your programs with the sample grader, your input should match the format and constraints from the task statement, otherwise, unspecified behaviors may occur.
- In sample grader inputs, every two consecutive tokens on a line are separated by a single space, unless another format is explicitly specified.
- The task statements specify signatures using generic type names such as `void`, `bool`, `string`, `int`, `int64`, `pair`, `int[]` (array) and `int[][]` (array of array).
- In C++, the graders use appropriate data types or implementations, as listed below:

void	**bool**	**string**	**int**	**int64**
`void`	`bool`	`std::string`	`int`	`long long`

pair(int, bool)	**int[]**	**int[][]**	**length of array a**
`std::pair<int, bool>`	`std::vector<int>`	`std::vector<std::vector<int>>`	`a.size()`

T2611 - GEM COLLECTION

Time Limit: 1.000 s / Memory Limit: 1024 MB

In the ancient times, the Hackerland is once a kingdom. In the Hackerland, N ($3 \leq N \leq 100$) is the magic number, which everything depends on N.

The king of the Hackerland is a gem enthusiast with a huge gem collection. There are N types of gems, labelled from type 0 to type $N-1$. The king also has N gems per type, so in total, there are N^2 gems.

As a world-known wizard, you are now invited to perform in front of the king! The king takes out all the N^2 gems and labels them arbitrarily from gem 0 to gem N^2-1. Gem i ($0 \leq i < N^2$) has the type $A[i]$ where $0 \leq A[i] < N$, which is unknown to you. Then, the gems will be wrapped with metal foil, indistinguishable to the bare eye. Your target is to identify all the gems of the same type to prove your ability as a wizard.

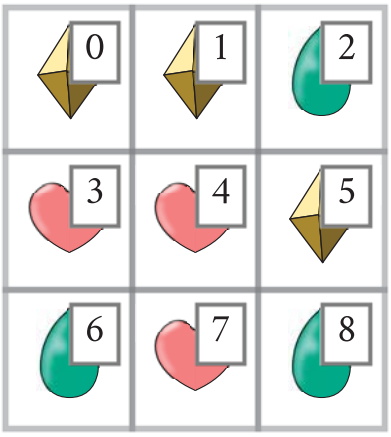


Figure 1: Example gem collection of $N = 3$.

As part of your wizard ability, you can perform the following operation any number of times: pick **exactly** N gems, feel the interactions between the gems to measure the *purity* among the N gems, which is the number of unordered pairs among the N gems that share the same type.

Unfortunately, even with your magic, it is hard to identify all the gems quickly! You decide to spend some time to invent some tricks to identify the gem types using as few measurements as possible, since each measurement costs you some mana, and you don't want the king to feel bored as well.

IMPLEMENTATION DETAILS

You should implement the procedure `identify_gems`:

```
void identify_gems(int N, int T)
```

- This procedure will be called at most 5 times per run. You should implement your strategy here.
- N: The number of different gems.
- T: The subtask number (1, 2, 3, 4 or 5).

Your procedure can make use of the following two procedures:

```
int measure(int[] I)
```

- This procedure counts the number of unordered gem pairs that are of the same type within the N provided gems.
- I: A set of N **distinct** indices, each between 0 and N^2-1 (inclusive).
- The function returns an integer, the number of pairs (x, y) such that gem $I[x]$ and gem $I[y]$ are of the same type with $0 \leq x < y < N$.
- You may only call this function **at most 128000 times** in a single call of `identify_gems`.

```
void report(int[] I)
```

- This procedure should be called exactly N times in a single call of `identify_gems`.
- Each call should report the set of N gems of the same type.
- I: A set of N **distinct** indices, each between 0 and N^2-1 (inclusive).
- Each gem type should be reported exactly once, in any order.

Important Notice

In this task, the grader is **NOT** adaptive. That means that A is fixed before the procedure `identify_gems` is called and does not depend on your queries.

Remember that there may be **multiple calls** to `identify_gems` **within the same run**. Contestants need to pay attention to the impact of the remaining data of the previous call on the current call.

EXAMPLE

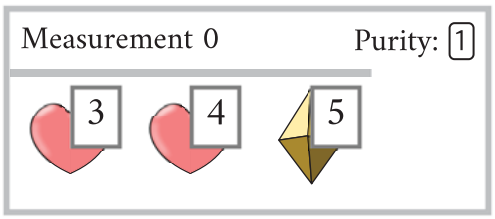
Suppose $N = 3$, $A = [0, 0, 2, 1, 1, 0, 2, 1, 2]$, and the subtask number is $T = 5$. You may refer to figure 1 for visualisation. The procedure `identify_gems` is called as follows:

```
identify_gems(3, 5)
```

To measure the purity among gem 3, gem 4 and gem 5, you can call

```
measure([3, 4, 5])
```

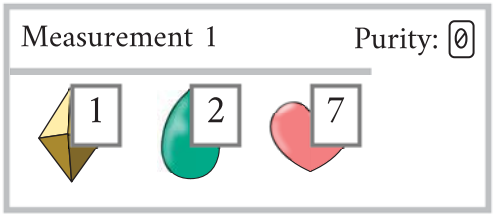
The gem types are 1, 1 and 0. There is one unordered pair of gems of same type, so the procedure returns 1.



Then, to measure the purity among gem 1, gem 2 and gem 7, you can call

```
measure([1, 2, 7])
```

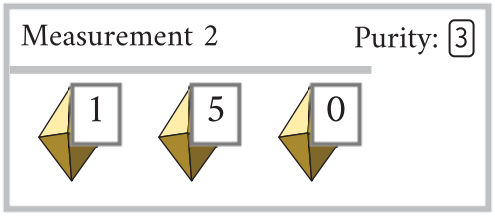
The gem types are 0, 2 and 1. There is no unordered pair of gems of same type, so the procedure returns 0.



Then, to measure the purity among gem 1, gem 5 and gem 0, you can call

```
measure([1, 5, 0])
```

The gem types are 0, 0 and 0. There are three unordered pair of gems of same type, so the procedure returns 3.



Suppose you deduce that

- Gems $0, 1, 5$ are of the same type;
- Gems $3, 4, 7$ are of the same type;
- Gems $2, 6, 8$ are of the same type.

Then you should call `report([0, 1, 5])`, `report([3, 4, 7])` and `report([2, 6, 8])` to report the answers. Note that the order within the same group, nor the order of the groups matters; for example, you can make the calls `report([4, 3, 7])`, `report([8, 6, 2])` and `report([5, 1, 0])`, which is also accepted.

SAMPLE GRADER

The sample grader reads input in the following format:

- The first line consists of two integers, N and T.
- The next lines consists of N^2 integers, $A[0], A[1], \ldots, A[N^2 - 1]$.

For the run, if your program is judged as incorrect, the type of the error will be outputted (`Wrong Answer: Reason`).

Otherwise, the program outputs `Correct` followed by the value of Q, the number of times `measure` is called.

Please note that the sample grader does not check whether the input is valid or not. Unexpected behaviour might occur if the input is invalid.

SAMPLE TESTS

	Input	Output
1	3 5 0 0 2 1 1 0 2 1 2	Correct: Q = 3

SUBTASKS

For all cases:
$3 \leq N \leq 100$
$0 \leq A[i] < N$ for all $0 \leq i < N^2$
Each of $0, 1, \ldots, N-1$ appears exactly N times in A

	Points	Constraints
1	7	$N \leq 40$ $A[i] = i$ for all $0 \leq i < N$
2	8	$N \leq 40$ $A[i] = 0$ for all $0 \leq i < N$
3	15	$N \leq 40$
4	46	$A[i] = 0$ for all $0 \leq i < N$
5	24	No additional constraints

SCORING

For each test case, you will receive zero score if your program does not exit correctly, violates any rules mentioned in the section "Implementation", makes more than 128000 calls to `measure` in a single call to `identify_gems`, or does not report all the gem types correctly.

Otherwise, you will receive full score in subtask 1, 2 and 3, while partial scoring applies for subtask 4 and 5. Suppose the maximum number of times `measure` called over all calls to `identify_gems` is Q.

In subtask 4, your score is calculated according to the following formula:

$$score = \begin{cases} 46 & \text{if } Q \leq 27000 \\ 34 & \text{if } 27000 < Q \leq 32000 \\ 18 & \text{otherwise.} \end{cases}$$

In subtask 5, your score is calculated according to the following formula:

$$score = \begin{cases} 24 & \text{if } Q \leq 27000 \\ 24 \times \dfrac{27000}{Q} & \text{otherwise.} \end{cases}$$

Your score of each subtask is the minimum of the scores over all test cases in the subtask.

T2612 - MINI METRO II

Time Limit: 1.000 s / Memory Limit: 1024 MB

Robo is once again addicted to playing video games. This year, he has become particularly obsessed with the game "Mini Metro".

Robo's best friend, Bob, has already set up a circular metro line consisting of N stations, conveniently numbered from 0 to $N-1$ in clockwise order. The metro travels continuously in a clockwise direction. It stops at every station and takes 1 minute to travel between adjacent stations. More formally, if the metro stops at station i at minute t, then at minute $t+1$, the metro stops at station $i+1$ if $i < N-1$, or station 0 if $i = N-1$.

Each station must be classified as exactly one of two types: a triangular station or a square station. **Station 0 must be a triangular station, and station $N-1$ must be a square station.** You may freely classify the remaining $N-2$ stations as either of the two types.

Of course, a metro network is meaningless without passengers. Each passenger has a target type of station, either triangular or square. A passenger starting from station i will travel to the first station j in clockwise order such that the type of station j matches their target type. The travel time for a passenger is the time the metro takes to get from station i to station j. More precisely:

- If station i already matches the target type of the passenger, then the passenger is considered to have reached its destination immediately, resulting in a travel time of 0 minutes.
- Otherwise, they board the metro and alight at the first station that matches its target type.

Robo knows that for every station i (where $0 \leq i \leq N-1$), there are $T[i]$ passengers who wish to go to a triangular station, and $S[i]$ passengers who wish to go to a square station. Robo's game score is defined as the total travel time of all passengers. Help Robo **minimise** his game score so he can reach the global leaderboard and brag to his friends!

IMPLEMENTATION DETAILS

You should implement the following procedure:

```
int64 minimum_cost(int N, int[] T, int[] S)
```

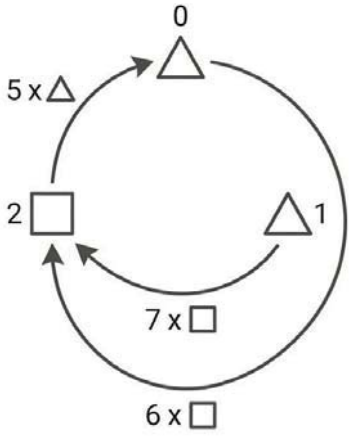
- This procedure is called exactly once per run.
- N: The number of stations on the metro line.
- T: An array of size N, representing the number of passengers who wish to go to a triangular station from each station.
- S: An array of size N, representing the number of passengers who wish to go to a square station from each station.
- The procedure should return an integer, the minimum total travel time of all passengers, in minutes.

EXAMPLE

Consider the following call:

```
minimum_cost(3, [3, 6, 5], [6, 7, 3])
```

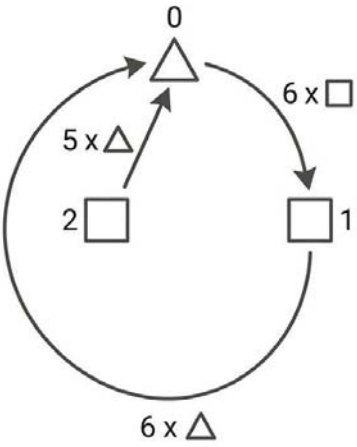
Recall that station 0 must be a triangular station, and station $N-1=2$ must be a square station. If we classify station 1 as a triangular station, then the metro network can be illustrated as follows:



- For passengers starting at station 0 (a triangular station),
  - $T[0]=3$ passengers have reached a triangular station already, and so do not board the metro.
  - $S[0]=6$ passengers board the metro and alight at station 2.
- For passengers starting at station 1 (a triangular station),
  - $T[1]=6$ passengers do not board the metro.
  - $S[1]=7$ passengers board and alight at station 2.
- For passengers starting at station 2 (a square station),
  - $T[2]=5$ passengers board and alight at station 0.
  - $S[2]=3$ have reached a square station and do not board the metro.

So, the total travel time of all passengers is $6\times(2-0)+7\times(2-1)+5\times(3-2)=24$.

On the other hand, if we classify station 1 as a square station, then the metro network can be illustrated as follows:



- For passengers starting at station 0 (a triangular station),
  - $T[0]=3$ passengers do not board the metro.
  - $S[0]=6$ passengers board and alight at station 1.
- For passengers starting at station 1 (a square station),
  - $T[1]=6$ passengers board and alight at station 0.
  - $S[1]=7$ passengers do not board the metro.
- For passengers starting at station 2 (a square station),
  - $T[2]=5$ passengers board and alight at station 0.
  - $S[2]=3$ have reached a square station and do not board the metro.

So, the total travel time of all passengers is $6\times(1-0)+6\times(3-1)+5\times(3-2)=23$.

Therefore, the minimum possible total travel time of all passengers is 23 and so the procedure should return `23`.

SAMPLE GRADER

The sample grader reads input in the following format:

- The first line contains a single integer, N.
- The second line contains N integers, $T[0],T[1],\ldots,T[N-1]$.
- The third line contains N integers, $S[0],S[1],\ldots,S[N-1]$.

The sample grader outputs a single integer, the returned integer of the procedure.

SAMPLE TESTS

	Input	Output
1	3 3 6 5 6 7 3	23
2	6 3 1 4 1 5 9 2 6 5 3 5 8	23

SUBTASKS

For all cases:
$2 \le N \le 5 \times 10^5$
$0 \le T[i], S[i] \le 10^6$ for all $0 \le i \le N-1$

	Points	Constraints
1	5	$N = 3$
2	11	$T[i] = S[i]$ for all $0 \le i \le N-1$
3	10	$N \le 500$ $T[0] = S[0] = T[N-1] = S[N-1] = 0$ $\min(T[i], S[i]) \le 2$ and $\max(T[i], S[i]) = 10^6$ for all $1 \le i \le N-2$
4	22	$N \le 500$ $T[i], S[i] \le 500$ for all $0 \le i \le N-1$
5	27	$N \le 3000$
6	25	No additional constraints

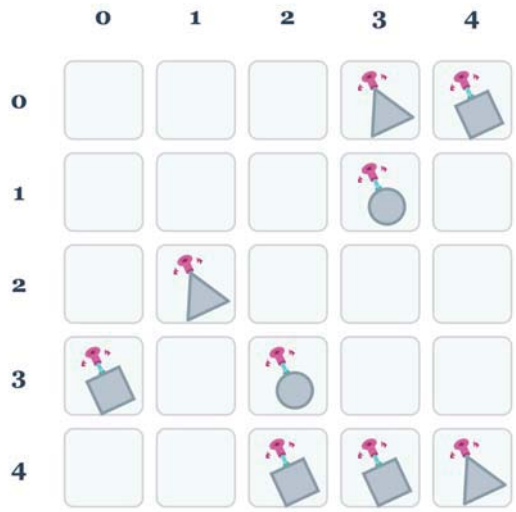
T2613 - TRIPLE KEYS

Time Limit: 3.000 s / Memory Limit: 1024 MB

After incorrectly trusting your friend about investing in HKOI Coin, you lose all your wealth and properties. Heavily indebted, you have no choice but to join *Fishy Game*, yet another gamble where stakes are unimaginably high.

You have made it pretty far to the fourth game. This game is called *Triple Keys*, where you are trapped in a maze, which can be represented as an $N \times N$ grid. The rows and columns of the grid are numbered from $0, 1, \ldots, N-1$. For $0 \le i, j \le N-1$, denote (i, j) as the cell on row i and column j. Each cell in this grid has either a Square key, a Circle key, a Triangle key or nothing. For $0 \le i, j \le N-1$, we let:

- $A[i][j] =$ `S` if cell (i, j) has a Square key,
- $A[i][j] =$ `C` if cell (i, j) has a Circle key,
- $A[i][j] =$ `T` if cell (i, j) has a Triangle key,
- $A[i][j] =$ `.` if cell (i, j) has nothing.



Visualization of a grid with $N = 5$. Here, $A[0][4] =$ `S`, because cell $(0, 4)$ has a Square key. $A[2][2] =$ `.`, because cell $(2, 2)$ has nothing.

You can only travel between adjacent cells in this maze. Two cells $(i_1, j_1), (i_2, j_2)$ (where $0 \le i_1, j_1, i_2, j_2 \le N-1$) are considered adjacent if and only if $|i_1 - i_2| + |j_1 - j_2| = 1$. It takes 1 second to travel between a pair of adjacent cells.

In order to clear the game, you must start from your starting cell, visit three cells that contain a Square key, a Circle key and a Triangle key respectively (therefore obtaining each of the three keys at least once), and return to your starting cell. You can take the three keys in any order.

Since you engaged in some improper trading with the manager of *Fishy Game*, you are given the structure of the maze. Being paranoid about whether you can clear the next game fast enough (after all, something bad might happen to you if you lose!), you imagine Q hypothetical scenarios. For the k-th scenario ($1 \le k \le Q$), you wonder: if you start at $(R[k], C[k])$ (where $0 \le R[k], C[k] \le N-1$), what is the minimum number of seconds needed to clear the game? Note that the Q scenarios are independent. That is, the keys return to where they were originally at the start of each scenario.

IMPLEMENTATION DETAILS

You should implement the following procedures:

```
void triple_keys(int N, string[] A)
```

- This procedure is called exactly once, before any calls to `minimum_time`.
- N: The size of the grid.
- A: A length-N string array, where for $0 \leq i, j \leq N-1$, the $(j+1)$-th character of the $(i+1)$-th string (that is, $A[i][j]$) denotes the contents inside the cell (i, j) of the grid.

```
int minimum_time(int R, int C)
```

- This procedure is called Q times.
- R: The row index of the starting cell.
- C: The column index of the starting cell.
- The procedure should return the minimum number of seconds needed to clear the game, if you start at (R, C).

EXAMPLE

Consider the following call:

```
triple_keys(5, ["...TS", "...C.", ".T...", "S.C..", "..SST"])
```

This call represents initialization of a $N = 5$ grid, which corresponds exactly to the figure in the statement.

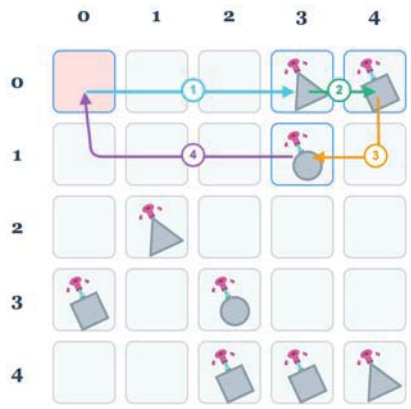
Consider the following query:

```
minimum_time(0, 0)
```

The optimal path, illustrated in the figure below, is as follows:

- Go from $(0, 0)$ to $(0, 3)$, obtaining a Triangle key. This takes 3 seconds.
- Then, go to $(0, 4)$, obtaining a Square key. This takes 1 second.
- Then, go to $(1, 3)$, obtaining a Circle key. This takes 2 seconds.
- Finally, go back to $(0, 0)$ with all three keys obtained. This takes 4 seconds.

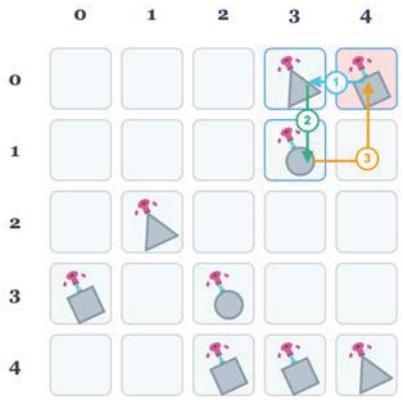
This path takes a total of $3 + 1 + 2 + 4 = 10$ seconds.



Consider the following query:

```
minimum_time(0, 4)
```

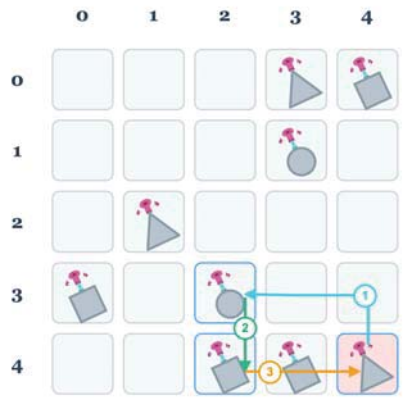
The optimal path, illustrated in the figure below, obtains a Square key at $(0, 4)$ itself, then a Triangle key at $(0, 3)$, then a Circle key at $(1, 3)$, and finally goes back to $(0, 4)$. This path takes a total of $0 + 1 + 1 + 2 = 4$ seconds.



Consider the following query:

```
minimum_time(4, 4)
```

The optimal path, illustrated in the figure below, obtains a Triangle key at $(4, 4)$ itself, then a Circle key at $(3, 2)$, then a Square key at $(4, 2)$, and finally goes back to $(4, 4)$. This path takes a total of $0 + 3 + 1 + 2 = 6$ seconds.



SAMPLE GRADER

The sample grader reads input in the following format:

- The first line contains a single integer N.
- The following N lines each contain a string of N characters. For $0 \leq i \leq N - 1$, the $(i + 1)$-th line represents $A[i][0], A[i][1], \ldots, A[i][N - 1]$.
- The following line contains a single integer Q.
- For $1 \leq k \leq Q$, the k-th of the following Q lines each contain two integers $R[k]$ and $C[k]$.

The sample grader outputs Q integers: the answer for each of the Q scenarios in order.

SAMPLE TESTS

	Input	Output
1	5 ...TS ...C. .T... S.C.. ..SST 3 0 0 0 4 4 4	10 4 6
2	4 .... ..C. .T.. S... 3 2 1 0 0 2 2	8 10 8

SUBTASKS

For all cases:
$2 \leq N \leq 1700$
$1 \leq Q \leq 10^6$
$A[i][j]$ is one of S, C, T or . for $0 \leq i, j \leq N-1$
$0 \leq R[k], C[k] \leq N-1$ for $1 \leq k \leq Q$
There exists at least one key of each shape in the grid

	Points	Constraints
1	3	$Q \leq 2500, N \leq 50$ For $1 \leq k \leq Q$, • $1 \leq R[k], C[k] \leq N-2$ • $A[R[k]][C[k]] =$ S • $A[R[k]+1][C[k]] = A[R[k]][C[k]+1] = A[R[k]-1][C[k]] = A[R[k]][C[k]-1] =$ T
2	4	$Q \leq 2500, N \leq 50$ There exist integers R', C' where $0 \leq R', C' \leq N-2$, satisfying the following conditions: • $A[R'][C'] =$ S, $A[R'][C'+1] =$ C, $A[R'+1][C'] =$ T • All other cells in the grid are empty
3	14	$N \leq 50$
4	11	$N \leq 200$
5	23	$N \leq 500$
6	27	$N \leq 1000$
7	18	No additional constraints