Singapore Informatics League 2026

Task Statements (Final Contest)

15 August 2026

Contents

Task 1: `6 ate 8`

Authored by: shoryu386

Statement

You are given a sequence of numbers a of length n, where the i-th number is $a[i]$.

In a world where 6 is afraid of 7 (because 7 ate 9), any number x can eat the number $x+2$ if $x+2$ is **directly to the right** of x in any sequence of numbers. Formally, a number $a[i]$ in the sequence can eat another number $a[j]$ if and only if $j = i+1$ and $a[j] = a[i]+2$. When a number is eaten, the remaining numbers in the sequence close the gap left by the eaten number.

Determine the minimum number of numbers that can remain if the order of eating is chosen optimally.

In this problem, there are tc independent test cases. Let $\text{ans}[i]$ represent the answer to the i-th test case. Output $\text{ans}[1] \cdot 1 + \text{ans}[2] \cdot 2 + \cdots + \text{ans}[tc] \cdot tc$.

Input Format

The first line of input contains one integer tc, the number of test cases.

The first line of each test case contains one integer n.

The second line of each test case contains n space-separated integers $a[1], a[2], \ldots, a[n]$.

Output Format

Output $\text{ans}[1] \cdot 1 + \text{ans}[2] \cdot 2 + \cdots + \text{ans}[tc] \cdot tc$.

Scoring

For all subtasks:

- $1 \leq tc \leq 10$
- $1 \leq n \leq 10^6$
- $1 \leq a[i] \leq 10^9$ for all $1 \leq i \leq n$
- $1 \leq \sum n \leq 5 \cdot 10^6$ across all test cases.

Subtask 1: $a[1] \leq a[2] \leq \cdots \leq a[n]$

Subtask 2: $n \leq 1000$ and $a[i] \leq 1000$ for all $1 \leq i \leq n$

Subtask 3: No additional constraints

Example

Input	Output
4 3 2 4 6 6 1 3 5 5 2 2 4 1 2 3 4 9 12 6 8 10 10 7 6 8 8	35

Note

There are four test cases in the example.

In the second test case, the original sequence is $a = [1,3,5,5,2,2]$.

One optimal order of eating is as follows:

- Number $a[2]$ (3) eats $a[3]$ (5), then $a = [1,3,5,2,2]$.
- Number $a[2]$ (3) eats $a[3]$ (5), then $a = [1,3,2,2]$.
- Number $a[1]$ (1) eats $a[2]$ (3), then $a = [1,2,2]$.

3 numbers remain in the sequence. It can be shown that this is the minimum possible.

In the third test case, none of the numbers are able to eat. Hence the sequence can only remain unchanged and the answer is 4.

It can be shown that the answers for the first and fourth test cases are 1 and 4 respectively. Hence, the correct output for the example is $1 \cdot 1 + 3 \cdot 2 + 4 \cdot 3 + 4 \cdot 4 = 35$.

Task 2: `Dog Walk`

Authored by: shoryu386

Statement

You live on a 2-dimensional grid of height h and width w. The rows of the grid are numbered 1 to h from top to bottom, and the columns of the grid are numbered 1 to w from left to right. We refer to the cell located at row r and column c of the grid as cell (r, c).

Some cells of the grid are blocked. A cell is either `'.'` (free) or `'#'` (blocked), and you may only occupy free cells.

The grid is given as h rows of w characters each, where the i-th row describes the cells $(i, 1)$, $(i, 2)$, $\ldots$, (i, w).

You are walking your dog, and you and your dog always occupy the same cell. Initially, both you and your dog are on the cell (s_x, s_y) (row s_x, column s_y), and you wish to reach the goal cell (e_x, e_y) (row e_x, column e_y).

You may take up to n turns. In one turn, you may move to an adjacent cell up, down, left, or right, or not move at all.

On the i-th turn, the dog will refuse to move in the direction $d[i]$, where $d[i] \in \{$`'U'`, `'D'`, `'L'`, `'R'`$\}$, representing the directions up, down, left, and right respectively. Since you and your dog always occupy the same cell, you cannot move in the direction $d[i]$ on the i-th turn.

If it is not possible to reach the goal cell within n turns, the answer is -1. Otherwise, the answer is the minimum number of turns to reach the goal cell.

In this problem, there are tc independent test cases. For each test case, let $\text{ans}[i]$ represent the answer to the i-th test case. Output $\text{ans}[1] \cdot 1 + \text{ans}[2] \cdot 2 + \cdots + \text{ans}[tc] \cdot tc$.

Input Format

The first line of input contains one integer tc, the number of test cases.

The first line of each test case contains three space-separated integers h, w, and n.

The second line of each test case contains four space-separated integers s_x, s_y, e_x, and e_y.

The third line of each test case contains a string of n characters, $d[1], d[2], \ldots, d[n]$.

The following h lines of each test case each contain a string of w characters, representing the cells of the grid.

Output Format

Output $\text{ans}[1] \cdot 1 + \text{ans}[2] \cdot 2 + \cdots + \text{ans}[tc] \cdot tc$.

Scoring

For all subtasks:

- $1 \leq tc \leq 60$
- $1 \leq n, h \cdot w \leq 10^6$ for each test case
- $1 \leq \sum(h \cdot w) \leq 10^7$
- $1 \leq s_x, e_x \leq h$ for each test case
- $1 \leq s_y, e_y \leq w$ for each test case
- $d[i] \in \{\texttt{'U'}, \texttt{'D'}, \texttt{'L'}, \texttt{'R'}\}$ for all $1 \leq i \leq n$
- Neither the starting cell nor the goal cell is blocked.

Subtask 1: The grid contains no blocked cells

Subtask 2: $n, h \cdot w \leq 1000$ for each test case, and $\sum(h \cdot w) \leq 20\,000$

Subtask 3: No additional constraints

$\sum(h \cdot w)$ denotes the sum of $(h \cdot w)$ over all test cases in a subtask.

Examples

Input	Output
2 3 5 8 3 1 1 5 RURRRLUL .... 1 1 1 1 1 1 1 R .	8

Input	Output
5 5 7 10 5 3 1 4 UUUUUUUUUU ###.### #.....# #..##.# #.....# ##.#### 5 7 10 5 3 1 4 LLLLLLLLLL ###.### #.....# #..##.# #.....# ##.#### 5 7 10 5 3 1 4 LUUURRRRRR ###.### #.....# #..##.# #.....# ##.#### 5 7 10 5 3 1 4 LUUUURRRRR ###.### #.....# #..##.# #.....# ##.#### 5 7 10 5 3 1 4 LUUUULLLLL ###.### #.....# #..##.# #.....# ##.####	121

Note

There are two test cases in the first example and five test cases in the second example.

In each example, the answer path is written as a string where the characters `'U'`, `'D'`, `'L'`, and `'R'` represent moving up, down, left, and right respectively, and `'S'` represents not moving.

In the first test case of the first example, you can take the path `URUSSRRR`, using 8 turns.

In the second test case of the first example, you do not need to move, thus using 0 turns.

Hence, the correct output for the first example is $8 \cdot 1 + 0 \cdot 2 = 8$.

The answers for the test cases in the second example are -1, 5, 9, 10, and 9 in order.

The paths taken are (no valid path), `UUURU`, `URRRUULLU`, `URRRSUULLU`, and `USSSSUURU` in order.

Note that for test cases with answer -1, the term -1 multiplied by the test case number must still be included in the final output.

Hence, the correct output for the second example is $-1 \cdot 1 + 5 \cdot 2 + 9 \cdot 3 + 10 \cdot 4 + 9 \cdot 5 = 121$.

Task 3: `Gremlin Nob`

Authored by: shoryu386

Statement

You are fighting a Gremlin Nob in the game Slay the Spire!

You have n attack cards, the i-th card dealing $a[i]$ damage to the Gremlin Nob. The Gremlin Nob has h health points, and it dies when its health is reduced to 0 or less.

You take the first turn, and turns alternate between you and the Gremlin Nob.

On its turn, the Gremlin Nob deals x damage to you, where x is initially 0.

When you play an attack card of $a[i]$ damage, the Gremlin Nob loses $a[i]$ health, and x increases by $a[i]$. Then, **if it is not dead** after your attack, you take x damage. The card is then discarded and cannot be played again.

What is the minimum total damage you can take to kill it?

In this problem, there are tc independent test cases. Let $\text{ans}[i]$ represent the answer for the i-th test case, i.e., the minimum possible total damage. Output $\text{ans}[1] \cdot 1 + \text{ans}[2] \cdot 2 + \cdots + \text{ans}[tc] \cdot tc$.

Input Format

The first line of input contains one integer tc, the number of test cases.

The first line of each test case contains two space-separated integers n and h.

The second line of each test case contains n space-separated integers $a[1], a[2], \ldots, a[n]$.

Output Format

Output $\text{ans}[1] \cdot 1 + \text{ans}[2] \cdot 2 + \cdots + \text{ans}[tc] \cdot tc$.

Scoring

For all subtasks:

- $1 \leq tc \leq 10$
- $1 \leq n \leq 45$
- $1 \leq h \leq \sum a[i]$
- $1 \leq a[i] \leq 10^9$ for all $1 \leq i \leq n$

Subtask 1: $1 \leq a[i] \leq 3$ for all $1 \leq i \leq n$

Subtask 2: $1 \leq n \leq 20$

Subtask 3: No additional constraints

Example

Input	Output
2 5 6 1 2 3 4 5 10 53 8 5 8 4 5 7 10 3 1 10	267

Note

There are two test cases in the example.

In the first test case, it is optimal to use the first attack and last attack, in that order.

This results in the following sequence of events:

- You deal 1 damage to the Gremlin Nob, and x increases from 0 to 1.
- The Gremlin Nob deals $x = 1$ damage to you.
- You deal 5 damage to the Gremlin Nob, and x increases from 1 to 6.
- Since the Gremlin Nob's health has been reduced to 0 or less, it dies, and the total damage you take is 1.

In the second test case, it is optimal to use the attacks with indices $[2,5,6,1,3,7,10]$ (1-indexed) in that order. Other optimal arrangements of attacks may exist.

Hence, the correct output for the example is $1 \cdot 1 + 133 \cdot 2 = 267$.

Task 4: `Reveille`

Authored by: TheRaptor

Statement

It is reveille, but the soldiers in Platoon Major Jayden's surprisingly enormous platoon are still sound asleep. Rather than waking every soldier himself, Jayden relies on a helpful fact: once a soldier wakes up, they will immediately wake the bunkmate to their left or right.

There is a row of n bunks in the platoon's barracks, numbered 1 to n. Each bunk is occupied by exactly one soldier. Every soldier is represented by either the character `'L'` or the character `'R'`, indicating which direction they will turn to wake another soldier.

Initially, every soldier is asleep. Jayden may choose some soldiers and wake them up manually.

When the soldier in bunk i wakes up:

- If they are represented by `'L'`, they will wake up the soldier in bunk $i-1$, if such a bunk exists.
- If they are represented by `'R'`, they will wake up the soldier in bunk $i+1$, if such a bunk exists.

A soldier can only be woken up once.

Determine the minimum number of soldiers Jayden must wake up manually so that every soldier is eventually awake for reveille.

In this problem, there are tc independent test cases. For each test case, let $\text{ans}[i]$ represent the answer to the i-th test case. Output $\text{ans}[1] \cdot 1 + \text{ans}[2] \cdot 2 + \cdots + \text{ans}[tc] \cdot tc$ modulo $10^9 + 7$.

Input Format

The first line of input contains one integer tc, the number of test cases.

The first line of each test case contains one integer n.

The second line of each test case contains a string s of length n, consisting only of the characters `'L'` and `'R'`. The i-th character of s represents the soldier in bunk i.

Output Format

Output $\text{ans}[1] \cdot 1 + \text{ans}[2] \cdot 2 + \cdots + \text{ans}[tc] \cdot tc$ modulo $10^9 + 7$.

Scoring

For all subtasks:

- $1 \leq n \leq 100\,000$
- The sum of all values of n does not exceed 10^6.

Subtask 1: $tc = 4$ and in each test case, there is exactly one index i ($1 \leq i < n$) such that $s[i] \neq s[i+1]$

Subtask 2: $tc = 5000$ and $n \leq 14$

Subtask 3: $tc = 9$

Example

Input	Output
3 2 LR 2 RL 16 RRRRRRRRRRRLLLLL	10

Note

There are three test cases in the example.

Note that the example does not satisfy the constraints of any subtask.

In the first test case, the string is `LR`.

The first person cannot wake anyone. The second person cannot wake anyone.

Both people must be woken up manually, so $\text{ans}[1] = 2$.

In the second test case, the string is `RL`.

Waking either person manually will eventually wake the other person.

Therefore, $\text{ans}[2] = 1$.

In the third test case, the string is `RRRRRRRRRRRLLLLL`.

You can wake up the leftmost person and the rightmost person. It can be shown that waking up only one person is not sufficient.

Therefore, $\text{ans}[3] = 2$.

Hence, the correct output for the example is $2 \cdot 1 + 1 \cdot 2 + 2 \cdot 3 = 10$.

Task 5: `Slimes`

Authored by: TheRaptor

Statement

Alice and Bob keep n slimes in a row, numbered from 1 to n from left to right. The i-th slime has size $s[i]$.

A slime may eat one of the slimes currently adjacent to it, provided that its own size is greater than or equal to the size of the slime it eats. When it does, its size becomes the sum of the two sizes, the eaten slime disappears, and the slimes close the gap so that they again form a single continuous row. The eater's number does not change in this process.

Over the next $n-1$ seconds, the slimes eat one another according to a fixed schedule: at second t, slime $x[t]$ eats slime $y[t]$.

Let moment 0 be the arrangement before any time has passed, and moment t the arrangement after the first t seconds have passed. There are therefore n moments $0, 1, \ldots, n-1$, and only one slime is left at moment $n-1$. A slime is alive at moment t if it has not been eaten during the first t seconds.

Alice and Bob each want to know what could have happened instead.

To settle such a question, they can simulate freezing the row at some moment, picking one living slime, and letting that slime eat as much as it can, in whatever order it likes, while always obeying the rule above. These simulations are hypothetical, and thus do not affect the sizes and arrangement of the slimes in the actual schedule or in other simulations.

Alice's question ($q = 1$): for each slime i, let $a[i]$ be the number of moments at which slime i is alive and could eat every other slime alive at that moment. A moment at which slime i is the only slime alive does count towards $a[i]$, since there is nothing left for it to eat. Alice wants $a[1] \cdot 1 + a[2] \cdot 2 + \cdots + a[n] \cdot n$.

Bob's question ($q = 2$): for every moment, and for every slime alive at that moment, determine the largest number of other slimes that it could eat. Bob wants the sum of these numbers over all n moments and all slimes alive at each moment.

The two answers can be very large, so every sum above, and the sum below, is taken modulo 998 244 353.

In this problem, there are tc independent test cases. Let $\text{ans}[j]$ represent the answer for the j-th test case, i.e. Alice's answer if $q = 1$, and Bob's answer if $q = 2$.

Output $\text{ans}[1] \cdot 1 + \text{ans}[2] \cdot 2 + \cdots + \text{ans}[tc] \cdot tc$ modulo 998 244 353.

Input Format

The first line of input contains one integer tc, the number of test cases.

The first line of each test case contains two space-separated integers n and q, the number of slimes and the question to answer.

The second line of each test case contains n space-separated integers $s[1], s[2], \ldots, s[n]$, the sizes of the slimes.

The following $n-1$ lines of each test case each contain two space-separated integers. The t-th of these lines contains $x[t]$ and $y[t]$, meaning that at second t, slime $x[t]$ eats slime $y[t]$.

Output Format

Output $\text{ans}[1] \cdot 1 + \text{ans}[2] \cdot 2 + \cdots + \text{ans}[tc] \cdot tc$ modulo 998 244 353.

Scoring

For all subtasks:

- $1 \leq tc \leq 100$
- $1 \leq q \leq 2$
- $1 \leq s[i] \leq 10^9$ for all $1 \leq i \leq n$
- $1 \leq x[t], y[t] \leq n$, $x[t] \neq y[t]$ for all $1 \leq t \leq n-1$
- At second t, slimes $x[t]$ and $y[t]$ are both alive, every slime between them has already been eaten, and the size of $x[t]$ at that second is greater than or equal to the size of $y[t]$, for all $1 \leq t \leq n-1$.

Subtask 1: $1 \leq n \leq 50000$ and $1 \leq \sum n \leq 170000$

Subtask 2: $q = 1$, $1 \leq n \leq 4 \cdot 10^6$, and $1 \leq \sum n \leq 8.1 \cdot 10^6$

Subtask 3: $q = 2$, $1 \leq n \leq 4 \cdot 10^6$, and $1 \leq \sum n \leq 8.1 \cdot 10^6$

$\sum n$ denotes the sum of n over all test cases in a subtask.

Example

Input	Output
2 3 1 2 1 3 1 2 1 3 3 2 2 1 3 1 2 1 3	21

Note

There are two test cases in the example. Both describe the same three slimes, of sizes 2, 1, 3, following the same schedule: at second 1, slime 1 eats slime 2, and at second 2, slime 1 eats slime 3. The second meal is legal even though both slimes have size 3 at that point, because the eater's size only needs to be greater than or equal to the size of the slime it eats.

There are 3 moments.

At moment 0, the row is 2, 1, 3.

- Slime 1 has size 2, so it can eat slime 2 and reach size 3, and then eat slime 3 since $3 \geq 3$. It could eat every other living slime.
- Slime 2 has size 1 and is smaller than both of its neighbours, so it can eat nothing at all.
- Slime 3 has size 3, so it can eat slime 2 and reach size 4, and then eat slime 1. It could eat every other living slime.

At moment 1, the row is slime 1 and slime 3, both of size 3. Each could eat the other, so both could eat every other living slime.

At moment 2, only slime 1 is left. For Alice's question, this moment counts for slime 1, since there is nothing left for it to eat.

For Alice ($q = 1$), slime 1 qualifies at all three moments, slime 2 at none, and slime 3 at moments 0 and 1. So $a[1] = 3$, $a[2] = 0$, $a[3] = 2$, and the answer to the first test case is $3 \cdot 1 + 0 \cdot 2 + 2 \cdot 3 = 9$.

For Bob ($q = 2$), moment 0 contributes $2 + 0 + 2 = 4$, moment 1 contributes $1 + 1 = 2$, and moment 2 contributes 0. So the answer to the second test case is $4 + 2 + 0 = 6$.

Hence, the correct output for the example is $9 \cdot 1 + 6 \cdot 2 = 21$.

Task 6: `Snowwabbits`

Authored by: penguin133

Statement

It is snowing in Bunnyland! Whiterabbit wants to build a snowman! To do so, he needs to stack n wabbits in a tower.

Whiterabbit has access to n wabbits, the i-th of which has size $a[i]$. For the snowman to be proper, the following conditions must be satisfied:

- The largest wabbit must be at the bottom.
- For every wabbit stacked on top of another wabbit, its size must be either the same as, or exactly one smaller than, the size of the wabbit below it.
- The top wabbit must be of size 1.

However, Whiterabbit may not be able to make a proper snowman using all n wabbits with their current sizes. This is where you, a magician, can help Whiterabbit!

In one spell cast, you can magically change the size of any wabbit to any positive integer. After some spell casts, Whiterabbit should be able to stack all the wabbits on top of each other to form a snowman.

What is the minimum number of spell casts you need to help Whiterabbit achieve his goal of building the snowman?

In this problem, there are tc independent test cases. Let $\text{ans}[i]$ represent the answer for the i-th test case, i.e. the minimum number of spell casts needed in the i-th test case. Output $\text{ans}[1] + \text{ans}[2] + \cdots + \text{ans}[tc]$.

Input Format

The first line of input contains one integer tc, the number of test cases.

The first line of each test case contains one integer n.

The second line of each test case contains n space-separated integers $a[1], a[2], \ldots, a[n]$.

Output Format

Output a single integer, $\text{ans}[1] + \text{ans}[2] + \cdots + \text{ans}[tc]$.

Scoring

For all subtasks:

- $1 \leq tc \leq 5$
- $1 \leq n \leq 1\,000\,000$
- $1 \leq a[i] \leq 10^9$

Subtask 1: $a[i] = 2 \cdot i$ for all $1 \leq i \leq n$

Subtask 2: All $a[i]$ are distinct

Subtask 3: No additional constraints

Example

Input	Output
4 4 2 1 2 3 5 10 20 30 40 50 5 4 4 4 4 4 5 5 2 2 1 4	9

Note

There are four test cases in the example.

In the first test case, Whiterabbit has 4 wabbits with sizes 2, 1, 2, and 3. The snowman can already be formed with wabbits of sizes 1, 2, 2, and 3 from top to bottom. This requires 0 spell casts.

In the second test case, Whiterabbit has 5 wabbits with sizes 10, 20, 30, 40, and 50. We must cast a spell on each of the 5 wabbits to change their sizes. This requires 5 spell casts.

In the third test case, Whiterabbit has 5 wabbits of size 4. A valid snowman requires 3 spell casts on 3 of the wabbits to change them into sizes 1, 2, and 3. A snowman can then be formed with sizes 1, 2, 3, 4, and 4. This requires 3 spell casts.

In the fourth test case, Whiterabbit has 5 wabbits with sizes 5, 2, 2, 1, and 4. Since there is no wabbit of size 3, a snowman cannot be formed with the current sizes. We can fix this with a single spell cast by changing the 5 into a 3 (forming 1, 2, 2, 3, and 4), or by changing one of the 2s into a 3 (forming 1, 2, 3, 4, and 5). This requires 1 spell cast.

Hence, the correct output for the example is $0+5+3+1=9$.

Task 7: Spring Sunlight

Authored by: kai824

Statement

Spring has arrived in Bunnyland, and Whiterabbit's factory is busy producing Easter eggs to deliver throughout the land!

The production line has produced n eggs, with the i-th being assigned a quality rating of $a[i]$. Before shipping them out, Whiterabbit must divide the eggs into k non-empty batches, with each batch consisting of a contiguous segment of the production line and every egg belonging to exactly one batch.

We define the median quality of a batch containing s eggs to be the m-th largest quality rating in the batch (counting duplicates), where m is the smallest integer such that $2 \cdot m \geq s$.

The overall quality of the shipment is defined as the smallest median quality among the k batches. With a quality inspection just around the corner, Whiterabbit wants to partition the eggs so that this overall quality is as high as possible. Help him find the maximum possible overall quality.

In this problem, there are tc independent test cases. Let $\text{ans}[i]$ represent the answer for the i-th test case, i.e. the maximum possible overall quality. Output $\text{ans}[1] \cdot 1 + \text{ans}[2] \cdot 2 + \cdots + \text{ans}[tc] \cdot tc$.

Input Format

The first line of input contains one integer tc, the number of test cases.

The first line of each test case contains two space-separated integers n and k.

The second line of each test case contains n space-separated integers $a[1], a[2], \ldots, a[n]$.

Output Format

Output $\text{ans}[1] \cdot 1 + \text{ans}[2] \cdot 2 + \cdots + \text{ans}[tc] \cdot tc$.

Scoring

For all subtasks:

- $1 \leq tc \leq 10$
- $1 \leq k \leq n \leq 10^6$
- $1 \leq \sum n \leq 5 \cdot 10^6$ across all test cases
- $1 \leq a[i] \leq 10^9$ for all $1 \leq i \leq n$

Subtask 1: $k = 2$

Subtask 2: $1 \leq k \leq n \leq 500$

Subtask 3: No additional constraints

Example

Input	Output
2 5 2 1 5 2 3 4 7 3 2 3 4 4 5 5 4	11

Note

There are two test cases in the example.

In the first test case, $n = 5$, $k = 2$, and $a = [1,5,2,3,4]$. Splitting a into $[1,5]$ with median 5 and $[2,3,4]$ with median 3 gives an overall quality of 3. It can be shown that no partition achieves a higher overall quality.

In the second test case, $n = 7$, $k = 3$, and $a = [2,3,4,4,5,5,4]$. Splitting a into $[2,3,4,4,5]$ with median 4, $[5]$ with median 5, and $[4]$ with median 4 gives an overall quality of 4. It can be shown that no partition achieves a higher overall quality.

Hence, the correct output for the example is $3 \cdot 1 + 4 \cdot 2 = 11$.

Task 8: `StringStringString`

Authored by: shoryu386

Statement

Shor the Duck really really really really ... really likes playing with strings today. He has called two friends over, Duck A and Duck B, to help him play with strings.

Shor the Duck has fixed a positive integer K.

Shor the Duck has come up with a game; this is how it goes:

- Duck A is given a string x and Duck B is given a string y.
- Duck A concatenates K copies of x together to form a string X.
- Duck B forms a string Y by concatenating the maximum number of copies of y such that Y is a subsequence of X.
- The score for this game is the length of Y.

Shor the Duck presents Duck A with a string S and Duck B with a string T. Both S and T consist of lowercase English letters.

Let S_m denote the prefix of S consisting of its first m letters, and let T_m denote the prefix of T consisting of its first m letters.

Duck A wonders: what is the score of the game if x is S_m and y is T, for each $1 \leq m \leq |S|$? ($|S|$ denotes the length of S.) Duck A wants to find the total score of these games; call this value total_A.

Duck B wonders: what is the score of the game if x is S and y is T_m, for each $1 \leq m \leq |T|$? ($|T|$ denotes the length of T.) Duck B wants to find the total score of these games; call this value total_B.

Shor the Duck wants to know the value of $\text{total}_A + \text{total}_B$. Could you help him?

More formally, if $\text{score}(x, y)$ denotes the score of the game, find the value of $(\text{score}(S_1, T) + \text{score}(S_2, T) + \cdots + \text{score}(S_{|S|}, T)) + (\text{score}(S, T_1) + \text{score}(S, T_2) + \cdots + \text{score}(S, T_{|T|}))$.

In this problem, there are tc independent test cases. Let $\text{ans}[i]$ represent the answer for the i-th test case, i.e. the value of $\text{total}_A + \text{total}_B$ in the i-th test case. Output $\text{ans}[1] + \text{ans}[2] + \cdots + \text{ans}[tc]$.

Input Format

The first line of input contains one integer tc, the number of test cases.

The first line of each test case contains one integer K.

The second line of each test case contains one string S.

The third line of each test case contains one string T.

Output Format

Output $\text{ans}[1] + \text{ans}[2] + \cdots + \text{ans}[tc]$.

Scoring

For all subtasks:

- $1 \leq tc \leq 10$
- $1 \leq K \leq 10^9$
- $1 \leq |S|, |T| \leq 5000$, where $|S|$ and $|T|$ represent the lengths of S and T respectively.

Subtask 1: $1 \leq K \leq 500$ and $1 \leq |S|, |T| \leq 500$

Subtask 2: $1 \leq |S|, |T| \leq 500$

Subtask 3: No additional constraints

Example

Input	Output
5 2 ab ba 5 abc cd 10 aab aba 3 abac abca 4 shortheducklikesplayinglongstringgames sil	549

Note

There are five test cases in the example.

In the first test case:

- For Duck A:
  - When $x =$ `a` and $y =$ `ba`, $X =$ `aa` and the longest possible Y is the empty string.
  - When $x =$ `ab` and $y =$ `ba`, $X =$ `abab` and the longest possible Y is `ba`.
- For Duck B:
  - When $x =$ `ab` and $y =$ `b`, $X =$ `abab` and the longest possible Y is `bb`.
  - When $x =$ `ab` and $y =$ `ba`, $X =$ `abab` and the longest possible Y is `ba`.

Hence, $\text{total}_A + \text{total}_B = 0 + 2 + 2 + 2 = 6$.

In the second test case:

- For Duck A:
  - When $x =$ `a` and $y =$ `cd`, $X =$ `aaaaa` and the longest possible Y is the empty string.
  - When $x =$ `ab` and $y =$ `cd`, $X =$ `abababababab` and the longest possible Y is the empty string.
  - When $x =$ `abc` and $y =$ `cd`, $X =$ `abcabcabcabcabc` and the longest possible Y is the empty string.
- For Duck B:
  - When $x =$ `abc` and $y =$ `c`, $X =$ `abcabcabcabcabc` and the longest possible Y is `ccccc`.
  - When $x =$ `abc` and $y =$ `cd`, $X =$ `abcabcabcabcabc` and the longest possible Y is the empty string.

Hence, $\text{total}_A + \text{total}_B = 0 + 0 + 0 + 5 + 0 = 5$.

In the third test case, it can be shown that $\text{total}_A + \text{total}_B = 94$.

In the fourth test case, it can be shown that $\text{total}_A + \text{total}_B = 29$.

In the fifth test case, it can be shown that $\text{total}_A + \text{total}_B = 415$.

Hence, the correct output for the example is $6 + 5 + 94 + 29 + 415 = 549$.

Task 9: Topsy Turvy

Authored by: penguin133

Statement

Alice and Bob each have n minions. Alice's i-th minion has $a[i]$ attack power and $b[i]$ health points. Bob's i-th minion has $c[i]$ attack power and $d[i]$ health points. To prove that her minions are more powerful, Alice decides to attack Bob's minions. She orders each of her minions to attack one of Bob's minions. Each of Bob's minions can be attacked by at most one of Alice's minions. Alice wins if all of Bob's minions are killed. Minion x can kill minion y if minion x's attack power is greater than or equal to minion y's health points.

To guarantee her success, Alice approaches a witch for help. The witch can swap the attack power and health points of **any** minion (one of Alice's or one of Bob's) for a cost of 1 per swap. Find the minimum cost v she has to pay to ensure she wins. If she cannot win, then $v = -1$.

In this problem, there are tc independent test cases. For each test case, let ans$[i]$ represent the answer to the i-th test case, i.e. the value of v. Output $\text{ans}[1] \cdot 1 + \text{ans}[2] \cdot 2 + \cdots + \text{ans}[tc] \cdot tc$.

Input Format

The first line of input contains one integer tc, the number of test cases.

The first line of each test case contains one integer n.

The second line of each test case contains n space-separated integers $a[1], a[2], \ldots, a[n]$.

The third line of each test case contains n space-separated integers $b[1], b[2], \ldots, b[n]$.

The fourth line of each test case contains n space-separated integers $c[1], c[2], \ldots, c[n]$.

The fifth line of each test case contains n space-separated integers $d[1], d[2], \ldots, d[n]$.

Output Format

Output $\text{ans}[1] \cdot 1 + \text{ans}[2] \cdot 2 + \cdots + \text{ans}[tc] \cdot tc$.

Scoring

For all subtasks:

- $1 \leq tc \leq 10$
- $1 \leq n \leq 10^6$
- $1 \leq a[i], b[i], c[i], d[i] \leq 10^9$ for all $1 \leq i \leq n$

Subtask 1: $n \leq 1000$

Subtask 2: $c[i] = d[i]$ for all $1 \leq i \leq n$

Subtask 3: No additional constraints

Example

Input	Output
3 3 2 5 7 1 5 7 2 4 7 2 4 7 3 1 4 8 6 4 8 10 10 10 2 5 7 3 1 5 9 4 5 9 2 4 9 2 6 9	8

Note

There are three test cases in the example.

In the first test case, Alice's attack powers are $[2,5,7]$ and Bob's health points are $[2,4,7]$. Alice can assign her first, second, and third minions to attack Bob's first, second, and third minions respectively. Hence, Alice can already kill all of Bob's minions without swaps, so the minimum cost is 0.

In the second test case, Alice initially has attack powers $[1,4,8]$, while Bob's minions have health points $[2,5,7]$. Without any swaps, Alice's minions cannot kill all three of Bob's minions.

Alice can swap the attack power and health points of her first minion. Its attack power changes from 1 to 6, so Alice's attack powers become $[6,4,8]$, allowing her minions to kill all of Bob's minions. Therefore, the minimum cost for this test case is 1.

In the third test case, it can be shown that the minimum cost is 2.

Hence, the correct output for the example is $0 \cdot 1 + 1 \cdot 2 + 2 \cdot 3 = 8$.

Task 10: `Xiaoyang Ranking`

Authored by: Xiaoyang

Statement

There are n xiaoyangs, numbered from 1 to n, and m zanes, numbered from 1 to m.

Every zane submits his own ranking of all n xiaoyangs, ordered from most xiao to least xiao, so each zane gives every xiaoyang a rank between 1 and n. Your job is to publish a single final ranking of all n xiaoyangs.

Of course, the zanes will complain about every xiaoyang you move away from where they put him. Given a zane and a xiaoyang, that zane's offence for that xiaoyang is the absolute difference between the rank he gave that xiaoyang and the rank you gave it. The outrage of your final ranking is the largest offence over all pairs of a zane and a xiaoyang.

Your task is to find a final ranking whose outrage k is as small as possible. If several final rankings achieve the minimum possible outrage, you must choose the lexicographically smallest one.

Formally, the ranking of the i-th zane is given as a permutation $q[i][1], q[i][2], \ldots, q[i][n]$ of the integers 1 to n, meaning that the i-th zane places xiaoyang $q[i][j]$ at rank j. Let $s[i][x]$ be the rank that the i-th zane gives to xiaoyang x, that is, $s[i][q[i][j]] = j$. Your final ranking is also a permutation $p[1], p[2], \ldots, p[n]$ of the integers 1 to n, where $p[t]$ is the xiaoyang you place at rank t, and its outrage is

$$k = \max_{1 \le i \le m} \max_{1 \le t \le n} \left| t - s[i][p[t]] \right|.$$

Here, rankings are compared as the sequences $p[1], p[2], \ldots, p[n]$ of xiaoyang numbers written in order of rank. A ranking p is lexicographically smaller than a ranking p' if, at the first rank t where they differ, $p[t] < p'[t]$. Equivalently, among all rankings achieving the minimum k, first make the xiaoyang at rank 1 as small as possible, then among those make the xiaoyang at rank 2 as small as possible, and so on.

In this problem, there are tc independent test cases.

Each test case additionally contains an integer mode, which tells you what to report for that test case.

- If mode $= 1$, you only have to report the minimum possible value of k.
- If mode $= 2$, you have to report $p[1] \cdot 1 + p[2] \cdot 2 + \cdots + p[n] \cdot n$, where p is the lexicographically smallest ranking achieving the minimum possible k.

For each test case, let $\text{ans}[i]$ represent the answer to the i-th test case, as described above. Output $\text{ans}[1] \cdot 1 + \text{ans}[2] \cdot 2 + \cdots + \text{ans}[tc] \cdot tc$.

Input Format

The first line of input contains one integer tc, the number of test cases.

The first line of each test case contains three space-separated integers n, m, and mode.

The following m lines of each test case each contain n space-separated integers. The i-th of these lines contains $q[i][1], q[i][2], \ldots, q[i][n]$, the ranking submitted by the i-th zane, from most xiao to least xiao.

Output Format

Output $\text{ans}[1] \cdot 1 + \text{ans}[2] \cdot 2 + \cdots + \text{ans}[tc] \cdot tc$.

It is guaranteed that this value fits into a signed 64-bit integer type.

Scoring

For all subtasks:

- $1 \leq tc \leq 15$
- $1 \leq m \leq 10$
- mode $\in \{1, 2\}$
- $1 \leq q[i][j] \leq n$ for all $1 \leq i \leq m$ and $1 \leq j \leq n$
- $q[i][1], q[i][2], \ldots, q[i][n]$ are pairwise distinct for all $1 \leq i \leq m$

Subtask 1: mode $= 1$ in every test case and $1 \leq n \leq 1\,000\,000$

Subtask 2: mode $= 2$ in every test case and $1 \leq n \leq 500$

Subtask 3: mode $= 2$ in every test case and $1 \leq n \leq 1\,000\,000$

Examples

Input	Output
1 4 2 1 2 1 3 4 2 1 4 3	1

Input	Output
1 6 3 2 2 1 3 5 4 6 2 1 5 3 6 4 2 1 3 5 6 4	90

Note

In the first example, there is one test case, and it has mode $= 1$, so only the minimum possible outrage is reported.

Both zanes agree that xiaoyang 2 is the most xiao and xiaoyang 1 the second most, but they disagree about the last two ranks: the first zane places xiaoyang 3 at rank 3 and xiaoyang 4 at rank 4, while the second zane places them the other way round. Xiaoyang 3 therefore receives rank 3 from one zane and rank 4 from the other, so wherever you place him, one of the two zanes is offended by at least 1, and $k \geq 1$. The ranking $p = [2,1,3,4]$ achieves $k = 1$, so $\text{ans}[1] = 1$ and the output is $1 \cdot 1 = 1$.

In the second example, there is one test case, and it has mode $= 2$, so the lexicographically smallest optimal ranking is reported as the sum $p[1] \cdot 1 + p[2] \cdot 2 + \cdots + p[n] \cdot n$.

All three zanes agree that xiaoyang 2 is the most xiao and xiaoyang 1 the second most. The remaining xiaoyangs receive the following ranks from the three zanes: $3,4,3$ for xiaoyang 3, $5,6,6$ for xiaoyang 4, $4,3,4$ for xiaoyang 5, and $6,5,5$ for xiaoyang 6. Since xiaoyang 3 is given both rank 3 and rank 4, some zane must be offended by at least 1 wherever he is placed, so $k \geq 1$.

The ranking $p = [1,2,3,5,4,6]$ achieves $k = 1$, because every xiaoyang in it sits at most one rank away from every rank he receives. It can be shown that it is also the lexicographically smallest ranking achieving $k = 1$.

The reported value is $1 \cdot 1 + 2 \cdot 2 + 3 \cdot 3 + 5 \cdot 4 + 4 \cdot 5 + 6 \cdot 6 = 90$, and the output is $90 \cdot 1 = 90$.

Optimisation Task: `Civilisation`

Authored by: shoryu386

Statement

You have a grid of size $n \times n$, where n is odd.

The rows of the grid are numbered 1 to n from top to bottom, and the columns of the grid are numbered 1 to n from left to right. We refer to the tile located at row row and column col of the grid as tile (row, col).

The centre tile of the grid is the starting point of your civilisation. The centre tile is $(\frac{n+1}{2}, \frac{n+1}{2})$.

m tiles on the grid are mines. Each mine is either a stone mine or a gold mine. The i-th mine is positioned on tile $(\text{row}[i], \text{col}[i])$. The i-th mine is a stone mine if $b[i] = 0$, or it is a gold mine if $b[i] = 1$.

While active, mine i produces $a[i]$ of its resource at the end of each turn. No two mines share a tile, and no mine lies on the central tile.

You begin with x stone and 0 gold. Building one road tile costs r stone. There are t turns.

The following occurs every turn:

- You may build up to one road on one empty tile, paying r stone to do so; you may build only if you currently hold at least r stone, and a tile is empty if it is not the centre tile, not a mine, and does not already have a road.
- After the building step, mines are considered active this turn if there is a valid path connecting the centre tile and this mine tile. A valid path must consist of tiles that are adjacent, and each tile must either be the centre tile, a road tile, or a mine tile.
- For the i-th mine, if it is active, it provides $a[i]$ units of either stone or gold, depending on whether it's a stone mine or a gold mine.

Here, we define two tiles as adjacent if and only if they share a side. Note that due to the mine activity rule above, a mine adjacent to another already active mine will also become active.

Maximise the amount of gold you have at the end of turn t.

Visualiser

We highly recommend using the visualiser, which you can open from this page. It replays your output on any of the subtask inputs and scores it exactly as the grader does, so the gold it reports is the raw score you will receive.

Input Format

The first line of input contains five space-separated integers n, m, t, x, and r.

The next m lines of input each contain four space-separated integers. The i-th of these lines contains row$[i]$, col$[i]$, $a[i]$, and $b[i]$.

Output Format

Output a sequence of t lines, representing the action on each turn.

The i-th line should contain two space-separated integers depending on the action on the i-th turn as follows:

- If you do not build a road that turn, it should contain `-1 -1`.
- If you build a road that turn, it should contain the integers row and col, where row and col are the coordinates of the tile the road is built on.

An action that is not a legal build — a tile outside the grid, the centre tile, a mine tile, a tile that already has a road, or any build while you hold less than r stone — is ignored, and that turn is spent without building anything.

Your output must however contain an action for every one of the t turns. A submission with fewer than t actions, or containing anything that is not an integer, is rejected outright and scores nothing.

Scoring

Your raw score is the amount of gold you hold at the end of turn t.

You should aim to produce the best possible construction; better outputs receive higher scores. You will still receive partial scores for sub-optimal constructions.

Raw scores are converted to normalised scores by comparison against all other teams: for each subtask, the team with the highest raw score receives the full 15 points, and teams with lower raw scores receive progressively fewer points.

Specifically, your team's normalised score for each subtask is computed according to the following formula.

$$10 \cdot \left(1 - \frac{\text{rank} - 1}{N}\right)^{1.25} + 5 \cdot \left(1 - \left(1 - \frac{\text{score}}{\text{best}}\right)^{0.4}\right)$$

N	The total number of teams, regardless of whether they attempted the subtask ($N = 32$)
rank	Your team's rank for that subtask by raw score (from 1 to N). Teams with equal raw scores share the best (smallest) rank of the tied group. Teams that did not attempt the subtask are also ranked.
score	Your team's raw score for that subtask
best	The highest raw score for that subtask across all teams

As a special case, a team that does not attempt a subtask (i.e. makes no valid submission for it) always receives 0 points for that subtask.

For all subtasks:

- $1 \leq \text{row}[i], \text{col}[i] \leq n$
- $b[i]$ is either 0 or 1

Subtask	n	m	t	x	r	Centre tile
1	11	10	40	40	10	$(6,6)$
2	35	24	130	1000	100	$(18,18)$
3	65	76	350	25500	1000	$(33,33)$

Example

Input	Output
9 4 15 30 10 7 7 8 0 5 2 6 1 1 2 8 1 1 1 1 1	5 6 5 7 6 7 -1 -1 5 4 5 3 4 2 3 2 -1 -1 2 2 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1

Note

The grid is 9×9, so the centre tile is $(5,5)$. There are 15 turns, you start with 30 stone, and each road costs 10 stone.

The first three roads at $(5,6)$, $(5,7)$ and $(6,7)$ spend all 30 of the starting stone and reach the stone mine at $(7,7)$, which then produces 8 stone per turn. Turn 4 waits to afford the next road.

Roads at $(5,4)$ and $(5,3)$ then run west, activating the gold mine at $(5,2)$ on turn 6; it produces 6 gold per turn for the remaining 10 turns, or 60 gold. Roads at $(4,2)$, $(3,2)$ and $(2,2)$ continue north and reach the gold mine at $(1,2)$ on turn 10, worth 8 gold per turn over 6 turns, or 48 gold. The mine at $(1,1)$ is adjacent to it, so it activates on the same turn and adds 1 gold per turn, or 6 gold.

Together the three gold mines yield $60+48+6=114$ gold at the end of turn 15, so this output scores 114.