Problem J1: Deliv-e-droid

Problem Description
In the game, Deliv-e-droid, a robot droid has to deliver packages while avoiding obstacles. At the end of the game, the final score is calculated based on the following point system:

- Gain 50 points for every package delivered.
- Lose 10 points for every collision with an obstacle.
- Earn a bonus 500 points if the number of packages delivered is greater than the number of collisions with obstacles.

Your job is to determine the final score at the end of a game.

Input Specification
The input will consist of two lines. The first line will contain a non-negative integer P, representing the number of packages delivered. The second line will contain a non-negative integer C, representing the number of collisions with obstacles.

Output Specification
The output will consist of a single integer F, representing the final score.

Sample Input 1
```
5
2
```

Output for Sample Input 1
```
730
```

Explanation of Output for Sample Input 1
There are 5 packages delivered, so $5 \times 50 = 250$ points are gained. There are 2 collisions, so $2 \times 10 = 20$ points are lost. Since $5 > 2$, a bonus 500 points are earned. Therefore, the final score is $250 - 20 + 500 = 730$.

Sample Input 2
```
0
10
```

Output for Sample Input 2
```
-100
```

Explanation of Output for Sample Input 2
There are 0 packages delivered, so $0 \times 50 = 0$ points are gained. There are 10 collisions, so $10 \times 10 = 100$ points are lost. Since $0 \leq 10$, no bonus points are earned. Therefore, the final score is $0 - 100 + 0 = -100$.

La version française figure à la suite de la version anglaise.

Problem J2: Chili Peppers

Problem Description
Ron is cooking chili using an assortment of peppers.

The spiciness of a pepper is measured in Scolville Heat Units (SHU). Ron's chili is currently not spicy at all, but each time Ron adds a pepper, the total spiciness of the chili increases by the SHU value of that pepper.

The SHU values of the peppers available to Ron are shown in the following table:

Pepper Name	Scolville Heat Units
Poblano	1500
Mirasol	6000
Serrano	15500
Cayenne	40000
Thai	75000
Habanero	125000

Your job is to determine the total spiciness of Ron's chili after he has finished adding peppers.

Input Specification
The first line of input will contain a positive integer N, representing the number of peppers Ron adds to his chili. The next N lines will each contain the name of a pepper Ron has added. Each pepper name will exactly match a name that appears in the table above. Note that more than one pepper of the same name can be added.

Output Specification
The output will consist of a positive integer T, representing the total spiciness of Ron's chili.

Sample Input
```
4
Poblano
Cayenne
Thai
Poblano
```

Output for Sample Input
```
118000
```

Explanation of Output for Sample Input
A Poblano pepper has an SHU value of 1500. A Cayenne pepper has an SHU value of 40000. A Thai pepper has an SHU value of 75000. The total spiciness of Ron's chili is therefore $1500 + 40000 + 75000 + 1500 = 118000$.

La version française figure à la suite de la version anglaise.

Problem J3: Special Event

Problem Description
You are trying to schedule a special event on one of five possible days.

Your job is to determine on which day you should schedule the event, so that the largest number of interested people are able to attend.

Input Specification
The first line of input will contain a positive integer N, representing the number of people interested in attending your event. The next N lines will each contain one person's availability using one character for each of Day 1, Day 2, Day 3, Day 4, and Day 5 (in that order). The character `Y` means the person is able to attend and a period (`.`) means the person is not able to attend.

The following table shows how the available 15 marks are distributed:

Marks	Description
6	There will be exactly one day on which every person will be able to attend.
6	There will be exactly one day on which the largest number of people will be able to attend.
3	There might be more than one day on which the largest number of people will be able to attend.

Output Specification
The output will consist of one line listing the day number(s) on which the largest number of interested people are able to attend.

If there is more than one day on which the largest number of people are able to attend, output all of these day numbers in increasing order and separated by commas (without spaces).

Sample Input 1
```
3
YY.Y.
...Y.
.YYY.
```

Output for Sample Input 1
```
4
```

Explanation of Output for Sample Input 1
All three people are able to attend on Day 4, and they are not all available on any other day.

La version française figure à la suite de la version anglaise.

Sample Input 2

```
5
YY..Y
.YY.Y
.Y.Y.
.YY.Y
Y...Y
```

Output for Sample Input 2

```
2,5
```

Explanation of Output for Sample Input 2

There is no day on which all five people are able to attend. Four people are able to attend on both Day 2 and Day 5.

La version française figure à la suite de la version anglaise.

Problem J4/S1: Trianglane

Problem Description

Bocchi the Builder just finished constructing her latest project: a laneway consisting of two rows of white equilateral triangular tiles. However, at the last moment, disaster struck! She accidentally spilled black paint on some of the tiles. Now, some of the tiles are wet and the other tiles are dry. Bocchi must place warning tape around the perimeters of all wet areas. Can you help her determine how many metres of tape she needs?

The first triangular tile will point upwards. Each pair of adjacent tiles (that is, tiles that share a common side) will point in opposite directions. Each tile has a side length of 1 metre.

Input Specification

The first line of input will consist of one positive integer C, representing the number of columns.

The next two lines will each consist of C integers separated by spaces. Each integer represents the colour of a tile along the laneway, with 1 indicating that the tile is black (wet) and 0 indicating that the tile is white (dry).

The following table shows how the available 15 marks are distributed:

Marks	Description	Bound
3	The laneway is not very long, black tiles are never adjacent and the second row is fully white.	$C \leq 2\,000$
3	The laneway is not very long, black tiles may be adjacent and the second row is fully white.	$C \leq 2\,000$
5	The laneway is not very long, black tiles may be adjacent and may appear in the second row.	$C \leq 2\,000$
4	The laneway may be very long, black tiles may be adjacent and may appear in the second row.	$C \leq 200\,000$

Output Specification

Output a single integer representing the length of tape Bocchi needs, in metres.

Sample Input 1

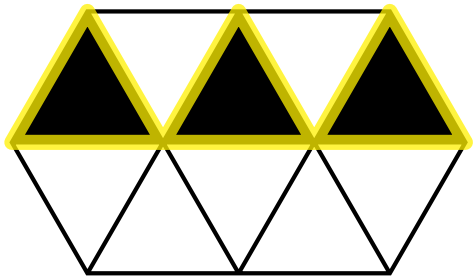
```
5
1 0 1 0 1
0 0 0 0 0
```

Output for Sample Input 1

```
9
```

La version française figure à la suite de la version anglaise.

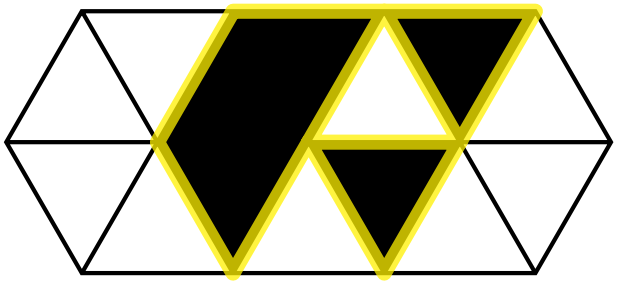
Explanation of Output for Sample Input 1
The tiles are painted as follows, creating three wet areas. Bocchi will need 9 metres of warning tape as shown in yellow.



Sample Input 2
```
7
0 0 1 1 0 1 0
0 0 1 0 1 0 0
```

Output for Sample Input 2
```
11
```

Explanation of Output for Sample Input 2
The tiles are painted as follows, creating three wet areas. Bocchi will need 5 metres of warning tape to surround one area and 3 metres of warning tape to surround each of the other two areas as shown in yellow.



La version française figure à la suite de la version anglaise.

Problem J5: CCC Word Hunt

Problem Description

In the CCC Word Hunt, words are hidden in a grid of letters. The letters of a hidden word always appear in order on horizontal, vertical, or diagonal line segments in one of two ways. One way is for the letters of a word to appear on one line segment. The other way is for the letters of a word to appear on one line segment up to some letter and then on a second line segment that forms a right angle at this letter.

Given a grid of letters and a single word to search for, your job is to determine the number of times that particular word is hidden in the grid.

Input Specification

The first line of input will contain a string of distinct uppercase letters, W, representing the word you are to search for in the grid. The length of W will be at least two. The second line of input will be an integer R ($1 \leq R \leq 100$), where R is the number of rows in the grid. The third line of input will be an integer C ($1 \leq C \leq 100$), where C is the number of columns in the grid.

The remaining input will provide the letters in the grid. It will consist of R lines, where each line contains C uppercase letters separated by single spaces.

The following table shows how the available 15 marks are distributed:

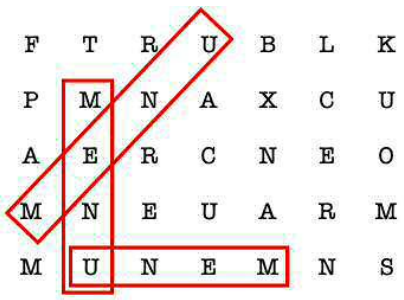
Marks	Word Placement
2	On one horizontal line segment
2	On one horizontal or vertical line segment
2	On one horizontal, vertical, or diagonal line segment
9	On one line segment or two perpendicular line segments

Output Specification

The output will consist of a single non-negative integer H, representing the number of times the word is hidden in the grid.

Sample Input 1

```
MENU
5
7
F T R U B L K
P M N A X C U
A E R C N E O
M N E U A R M
M U N E M N S
```



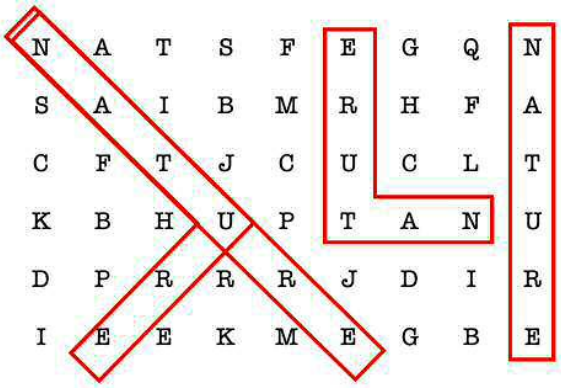
La version française figure à la suite de la version anglaise.

Output for Sample Input 1
```
3
```

Explanation of Output for Sample Input 1
The word `MENU` is hidden three times in the grid. Once horizontally, once vertically, and once diagonally as shown.

Notice that a single letter can be used more than once.

Sample Input 2
```
NATURE
6
9
N A T S F E G Q N
S A I B M R H F A
C F T J C U C L T
K B H U P T A N U
D P R R R J D I R
I E E K M E G B E
```



Output for Sample Input 2
```
4
```

Explanation of Output for Sample Input 2
The word `NATURE` is hidden four times in the grid. Once diagonally, once vertically, and twice on perpendicular line segments.

La version française figure à la suite de la version anglaise.

Problem S2: Symmetric Mountains

Problem Description

Rebecca is a tour guide and is trying to market the Rocky Mountains for her magazine. She recently took a beautiful picture consisting of N mountains where the i-th mountain from the left has a height h_i. She will crop this picture for her magazine, by possibly removing some mountains from the left side of the picture and possibly removing some mountains from the right side of the picture. That is, a crop consists of consecutive mountains starting from the l-th to the r-th mountain where $l \leq r$. To please her magazine readers, Rebecca will try to find the most symmetric crop.

We will measure the *asymmetric value* of a crop as the sum of the absolute difference for every pair of mountains equidistant from the midpoint of the crop. To help understand that definition, note that the absolute value of a number v, written as $|v|$, is the non-negative value of v: for example $|-6| = 6$ and $|14| = 14$. The asymmetric value of a crop is the sum of all $|h_{l+i} - h_{r-i}|$ for $0 \leq i \leq \frac{r-l}{2}$. To put that formula in a different way, we pair up the mountains working from the outside in toward the centre, calculate the absolute difference in height of each of these pairs, and sum them up.

Because Rebecca does not know how wide the picture needs to be, for all possible crop lengths, find the asymmetric value of the most symmetric crop (the crop with the minimum asymmetric value).

Input Specification

The first line consists of an integer N, representing the number of mountains in the picture. The second line consists of N space-separated integers, where the i-th integer from the left represents h_i.

The following table shows how the available 15 marks are distributed:

Marks	Bounds on N	Bounds on h_i	Additional Constraints
5	$1 \leq N \leq 300$	$0 \leq h_i \leq 10^5$	None
5	$1 \leq N \leq 5000$	$0 \leq h_i \leq 10^5$	Height of mountains are in non-decreasing order from left to right.
5	$1 \leq N \leq 5000$	$0 \leq h_i \leq 10^5$	None

Output Specification

Output on one line N space-separated integers, where the i-th integer from the left is the asymmetric value of the most symmetric picture of crops of length i.

Sample Input 1

```
7
3 1 4 1 5 9 2
```

La version française figure à la suite de la version anglaise.

Output for Sample Input 1

```
0 2 0 5 2 10 10
```

Explanation of Output for Sample Input 1

We will show why the fifth value from the left is 2. Let us try to compute all the asymmetric values of crops with length 5.

The height of the mountains in the first crop is $[3, 1, 4, 1, 5]$. The asymmetric value of this crop is $|3 - 5| + |1 - 1| + |4 - 4| = 2$.

The height of the mountains in the second crop is $[1, 4, 1, 5, 9]$. The asymmetric value of this crop is $|1 - 9| + |4 - 5| + |1 - 1| = 9$.

The height of the mountains in the last crop is $[4, 1, 5, 9, 2]$. The asymmetric value of this crop is $|4 - 2| + |1 - 9| + |5 - 5| = 10$.

Hence, the most symmetric crop of length 5 is 2.

Sample Input 2

```
4
1 3 5 6
```

Output for Sample Input 2

```
0 1 3 7
```

Explanation of Output for Sample Input 2

This sample satisfies the second subtask. Note that the only crop of length 4 is $[1, 3, 5, 6]$ which has asymmetric value of $|1 - 6| + |3 - 5| = 7$.

La version française figure à la suite de la version anglaise.

Problem S3: Palindromic Poster

Problem Description
Ryo and Kita are designing a new poster for Kessoku Band. After some furious brainstorming, they came to the conclusion that the poster should come in the form of a 2-D grid of lowercase English letters (i.e. `a` to `z`), with N rows and M columns.

Furthermore, it is known that Ryo and Kita both have peculiar tastes in palindromes. Ryo will only be satisfied with the poster if exactly R of its rows are palindromes, and Kita will only be satisfied with the poster if exactly C of its columns are palindromes. Can you design a poster that will satisfy both Ryo and Kita, or determine that it is impossible to do so?

Note: A string is considered a *palindrome* if it is the same when read forwards and backwards. For example, `kayak` and `bb` are palindromes, whereas `guitar` and `live` are not.

Input Specification
The first and only line of input consists of 4 space-separated integers N, M, R, and C.

The following table shows how the available 15 marks are distributed:

Marks	**Bounds on N**	**Bounds on M**	**Bounds on R**	**Bounds on C**
2 marks	$2 \leq N \leq 2\,000$	$2 \leq M \leq 2\,000$	$R = 1$	$C = 1$
2 marks	$N = 2$	$M = 2$	$0 \leq R \leq N$	$0 \leq C \leq M$
4 marks	$N = 2$	$2 \leq M \leq 2\,000$	$0 \leq R \leq N$	$0 \leq C \leq M$
7 marks	$2 \leq N \leq 2\,000$	$2 \leq M \leq 2\,000$	$0 \leq R \leq N$	$0 \leq C \leq M$

Output Specification
If it is impossible to design a poster that will satisfy both Ryo and Kita, output `IMPOSSIBLE` on a single line.

Otherwise, your output should contain N lines, each consisting of M lowercase English letters, representing your poster design. If there are multiple possible designs, output any of them.

Sample Input 1
```
4 5 1 2
```

Output for Sample Input 1
```
union
radar
badge
anime
```

La version française figure à la suite de la version anglaise.

Explanation of Output for Sample Input 1
In the given design, only the second row (namely `radar`) and the second and third columns (namely `naan` and `iddi`) are palindromes. Since exactly $R = 1$ of the rows and $C = 2$ of the columns are palindromes, this is an acceptable design.

Sample Input 2
```
2 2 2 1
```

Output for Sample Input 2
```
IMPOSSIBLE
```

Explanation of Output for Sample Input 2
In this case, it can be proven that it is impossible to satisfy both Ryo and Kita.

La version française figure à la suite de la version anglaise.

Problem S4: Minimum Cost Roads

Problem Description
As the newly elected mayor of Kitchener, Alanna's first job is to improve the city's road plan.

Kitchener's current road plan can be represented as a collection of N intersections with M roads, where the i-th road has length l_i meters, costs c_i dollars per year to maintain, and connects intersections u_i and v_i. To create a plan, Alanna must select some subset of the M roads to keep and maintain, and that plan's cost is the sum of maintenance costs of all roads in that subset.

To lower the city's annual spending, Alanna would like to minimize the plan's cost. However, the city also requires that she minimizes travel distances between intersections and will reject any plan that does not conform to those rules. Formally, for any pair of intersections (i, j), if there exists a path from i to j taking l meters on the existing road plan, Alanna's plan must also include a path between those intersections that is at most l meters.

Input Specification
The first line contains the integers N and M.

Each of the next M lines contains the integers u_i, v_i, l_i, and c_i, meaning that there currently exists a road from intersection u_i to intersection v_i with length l_i and cost c_i ($1 \leq u_i, v_i \leq N, u_i \neq v_i$).

The following table shows how the available 15 marks are distributed.

Marks	**Bounds on N and M**	**Bounds on l_i**	**Bounds on c_i**	**Additional Constraints**
3 marks	$1 \leq N, M \leq 2\,000$	$l_i = 0$	$1 \leq c_i \leq 10^9$	None
6 marks	$1 \leq N, M \leq 2\,000$	$1 \leq l_i \leq 10^9$	$1 \leq c_i \leq 10^9$	There is at most one road between any unordered pair of intersections.
6 marks	$1 \leq N, M \leq 2\,000$	$0 \leq l_i \leq 10^9$	$1 \leq c_i \leq 10^9$	None

Output Specification
Output one integer, the minimum possible cost of a road plan that meets the requirements.

Sample Input
```
5 7
1 2 15 1
2 4 9 9
5 2 5 6
```

La version française figure à la suite de la version anglaise.

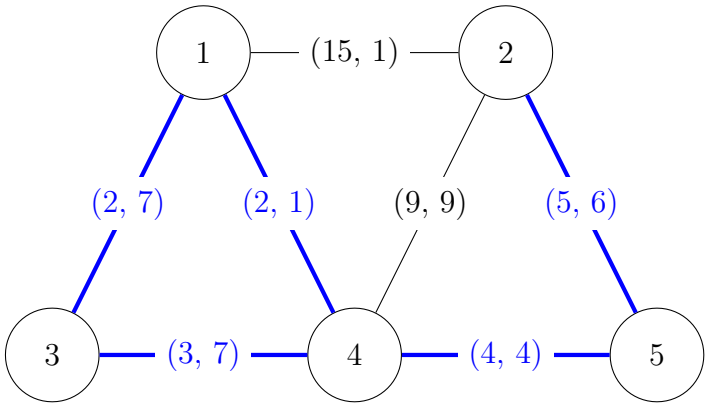
```
4 5 4 4
4 3 3 7
1 3 2 7
1 4 2 1
```

Output for Sample Input

```
25
```

Explanation of Output for Sample Input

Here is a diagram of the intersections along with a valid road plan with minimum cost.



Each edge is labeled with a pair (l, c) denoting that it has length l meters and cost c dollars. Additionally, the roads that are part of the plan are highlighted in blue, with a total cost of $7 + 1 + 6 + 7 + 4 = 25$.

It can be shown that we cannot create a cheaper plan that also respects the city's requirements.

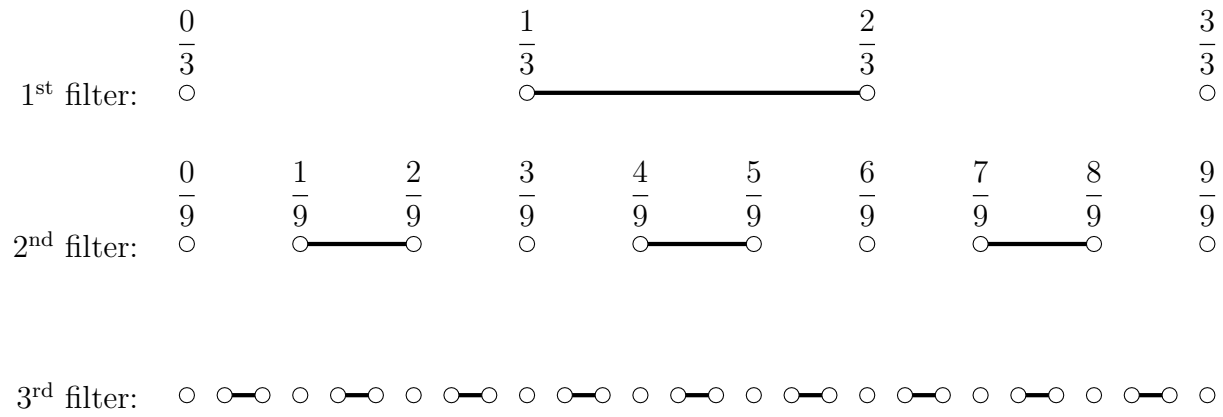
La version française figure à la suite de la version anglaise.

Problem S5: The Filter

Problem Description
Alice, the mathematician, likes to study real numbers that are between 0 and 1. Her favourite tool is the *filter*.

A filter covers part of the number line. When a number reaches a filter, two events can happen. If a number is not covered by the filter, the number will pass through. If a number is covered, the number will be removed.

Alice has infinitely many filters. Her first 3 filters look like this:



In general, the k-th filter can be defined as follows:

- Consider the number line from 0 to 1.
- Split this number line into 3^k equal-sized pieces. There are $3^k + 1$ points and 3^k intervals.
- The k-th filter consists of the 2^{nd} interval, 5^{th} interval, 8^{th} interval, and in general, the $(3i-1)^{\text{th}}$ interval. The points are **not** part of the k-th filter.

Alice has instructions for constructing the *Cantor set*. Start with the number line from 0 to 1. Apply all filters on the number line, and remove the numbers that are covered. The remaining numbers form the Cantor set.

Alice wants to research the Cantor set, and she came to you for help. Given an integer N, Alice would like to know which fractions $\frac{x}{N}$ are in the Cantor set.

Input Specification
The first line contains the integer N.

The following table shows how the available 15 marks are distributed.

La version française figure à la suite de la version anglaise.

Marks	Bounds on N	Additional Constraints
3 marks	$3 \leq N \leq 3^{18}$	N is a power of 3
4 marks	$2 \leq N \leq 10^5$	None
8 marks	$2 \leq N \leq 10^9$	None

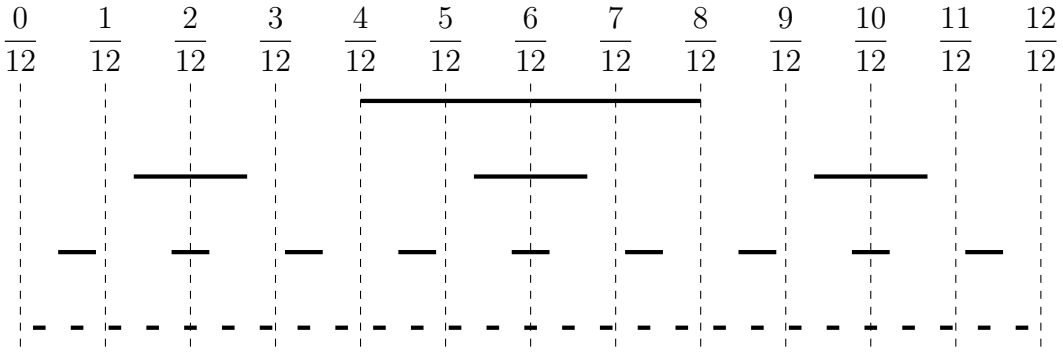
Output Specification
Output all integers x where $0 \leq x \leq N$ and $\frac{x}{N}$ is in the Cantor set.

Output the answers in increasing order. The number of answers will not exceed 10^6.

Sample Input
```
12
```

Output for Sample Input
```
0
1
3
4
8
9
11
12
```

Explanation of Output for Sample Input
Here is a diagram of the fractions and the first 4 filters. In reality, there are infinitely many filters.



$\frac{5}{12}$, $\frac{6}{12}$, and $\frac{7}{12}$ are not in the Cantor set because they were covered by the 1st filter.

Furthermore, $\frac{2}{12}$ and $\frac{10}{12}$ are not in the Cantor set because they were covered by the 2nd filter.

It can be shown that the remaining fractions will pass through all filters.

La version française figure à la suite de la version anglaise.

2023 Canadian Computing
Problem 1

Flip it and Stick it

Time Limit: 1 second

Problem Description
Finn is playing a game of "Flip it and Stick it" which is abbreviated as FiSi. FiSi is a one-player game played on two strings, S and T, of `0`s and `1`s. Finn is allowed to make moves of the following form:

- Select a substring of S and reverse it, gluing the pieces of the string back together in their original order to form the new string S.

For example, Finn may take the string $S =$ `101100`, take the substring `011` starting at index 2 (assuming 1-based string indexing), and create the string $S =$ `111000` in one move.

Finn wins the game if S does **not** contain T as a substring. Your task is to help Finn determine the length of the shortest winning sequence of moves or tell him that the game cannot be won.

Input Specification
The first line of input contains the string S ($1 \leq |S| \leq 200\,000$).

The second line of input contains the string T ($1 \leq |T| \leq 3$).

In the table below, T_1 is the first bit in T, T_2 is the second bit in T, and T_3 is the third bit in T, when reading from left-to-right.

Marks Awarded	**Bounds on T**
1 mark	$\|T\| = 1$
3 marks	$\|T\| = 2, T_1 \neq T_2$
4 marks	$\|T\| = 2$
5 marks	$\|T\| = 3, T_1 \neq T_3$
5 marks	$\|T\| = 3, T_1 \neq T_2$
7 marks	$\|T\| = 3$

Output Specification
Output the minimum number of moves needed or -1 if it is impossible to win the game.

Sample Input 1
```
100110
10
```

Output for Sample Input 1

```
2
```

Explanation of Output for Sample Input 1

Finn starts with the string `100110`. He cannot avoid `10` as a substring in one move, but he can in two moves.

For example, his first move could be to reverse the substring from index 4 to index 6 (`110`) to get `100011`. Then, his second move can be to reverse the substring from index 1 to index 4 (`1000`) to get `000111`, which does not have `10` as a substring.

Sample Input 2

```
000
00
```

Output for Sample Input 2

```
-1
```

Explanation of Output for Sample Input 2

No matter how many moves Finn makes, the string S will always contain T as a substring.

2023 Canadian Computing
Problem 2

Travelling Trader

Time Limit: 2 seconds

Problem Description
A trader would like to make a business of travelling between cities, moving goods from one city to another in exchange for a profit. There are N cities labelled $1, \ldots, N$ and $N-1$ roads. Each road joins two cities and takes one day to traverse. It is possible to reach any city from any other city using these roads.

The i-th city can give a profit of p_i if the trader is currently in that city and chooses to do business in that city, but this profit may only be obtained once. The trader starts by doing business in city 1 and wants to travel along the roads, visiting cities to maximize their total profit. However, the trader's boss will get unhappy and lay off the trader as soon as the trader goes more than K days in a row without increasing their total profit. Note that the trader will take only one day to move between adjacent cities, regardless of whether the trader does business in either city. We would like to know the maximum profit the trader can make under this condition and a route that obtains this profit.

Input Specification
The first line of input contains two space-separated integers N and K.

The next $N-1$ lines of input each contain two space-separated integers u_i and v_i ($1 \leq u_i, v_i \leq N, u_i \neq v_i$), describing a road.

The last line of input contains N integers $p_1, \ldots, p_N$ ($1 \leq p_i \leq 10^9$), the profits given by choosing to do business in the corresponding city.

Marks Awarded	**Bounds on N**	**Bounds on K**
2 marks	$2 \leq N \leq 200\,000$	$K = 1$
7 marks	$2 \leq N \leq 200$	$K = 2$
3 marks	$2 \leq N \leq 2\,000$	$K = 2$
4 marks	$2 \leq N \leq 200\,000$	$K = 2$
4 marks	$2 \leq N \leq 2\,000$	$K = 3$
5 marks	$2 \leq N \leq 200\,000$	$K = 3$

Output Specification
On the first line, output the maximum possible total profit.

On the second line, output M ($1 \leq M \leq N$), the number of cities the trader does business in on an optimal route.

On the third line, output M space-separated integers $x_1, \ldots, x_M$, the cities the trader does business in on an optimal route in order, starting with $x_1 = 1$.

If there are multiple possible correct outputs, any correct output will be accepted.

Sample Input 1

```
4 1
1 2
1 3
2 4
3 1 4 1
```

Output for Sample Input 1

```
7
2
1 3
```

Explanation of Output for Sample Input 1

On day 1, the trader starts by doing business in city 1, making a profit of 3.

On day 2, the trader moves to city 3 and does business there, making a profit of 4.

At this point, the trader cannot reach another city in which they have not done business before getting laid off, so their total profit is 7.

Sample Input 2

```
5 2
1 2
1 3
2 4
2 5
3 1 4 1 5
```

Output for Sample Input 2

```
14
5
1 4 5 2 3
```

Explanation of Output for Sample Input 2

The trader can make a profit in every city by visiting them in the order $1, 2, 4, 2, 5, 2, 1, 3$.

Note that the trader strategically delays doing business in city 2 to ensure they do not go more than 2 days without making a profit.

2023 Canadian Computing
Problem 3

Triangle Collection

Time Limit: 4 seconds

Problem Description

Alice has a collection of sticks. Initially, she has c_ℓ sticks of length ℓ for each $\ell = 1, \ldots, N$.

Alice would like to use her sticks to make some isosceles triangles. An isosceles triangle is made of two sticks of the same length, say ℓ, and a third stick with a length between 1 and $2\ell - 1$ inclusive. Note that the triangles must strictly obey the triangle inequality, and equilateral triangles are okay. Each stick may be used in at most one triangle. Alice would like to know the maximum number of isosceles triangles she can make with her sticks.

There are Q events that change the collection of sticks she has. The i-th event consists of two integers ℓ_i and d_i, representing that the number of sticks of length ℓ_i changes by d_i. Note that d_i may be positive, negative, or even 0, but Alice will never have a negative number or more than 10^9 sticks of each length.

Your task is to determine the maximum number of isosceles triangles Alice can make after each event if she uses her sticks optimally.

Input Specification

The first line of input contains two space-separated integers N and Q.

The second line of input contains N space-separated integers $c_1, c_2, \ldots, c_N$ ($0 \leq c_i \leq 10^9$), representing Alice's initial collection.

The next Q lines of input each contain two space-separated integers ℓ_i and d_i ($1 \leq \ell_i \leq N, -10^9 \leq d_i \leq 10^9$), representing an event.

Initially and after each event, the number of sticks of length ℓ is between 0 and 10^9 for all $\ell = 1, \ldots, N$.

Marks Awarded	**Bounds on N, Q**	**Additional Constraints**
5 marks	$1 \leq N, Q \leq 2\,000$	There are at most 2 000 sticks in total initially and after each event.
5 marks	$1 \leq N, Q \leq 2\,000$	No additional constraints.
5 marks	$1 \leq N, Q \leq 200\,000$	The number of sticks of each length is either 0, 1, or 2 initially and after each event.
5 marks	$1 \leq N, Q \leq 200\,000$	For each event, $\lvert d_i \rvert = 1$.
5 marks	$1 \leq N, Q \leq 200\,000$	No additional constraints.

Output Specification
Output Q lines each containing a single integer, the answer after each event.

Sample Input
```
4 3
3 1 4 1
3 -3
1 6
2 1
```

Output for Sample Input
```
1
3
4
```

Explanation of Output for Sample Input
After the first event, Alice can make a single triangle with sticks of lengths $(1, 1, 1)$.

After the second event, Alice can make 3 triangles with sticks of lengths $(1, 1, 1)$.

After the third event, Alice can make 3 triangles with sticks of lengths $(1, 1, 1)$ and a triangle with sticks of lengths $(2, 2, 3)$.

2023 Canadian Computing

PROBLEM 4

Binaria

Time Limit: 1 second

Problem Description

You have been hired by the Cheap Communication Organization (CCO) to work on a communication breakthrough: sub-message sum (SMS). This revolutionary idea works as follows.

Given a binary string of length N, and some positive integer K with $K \leq N$, the SMS for the string consists of a sequence of $N - K + 1$ sums. The first sum in the sequence is the sum of digits 1 through K, the second sum is the sum of digits 2 through $K + 1$, and so on until the last sum which is the sum of digits $N - K + 1$ through N.

For example, if $K = 4$, the SMS of the binary string `110010` is 2,2,1. This is because $1 + 1 + 0 + 0 = 2$, $1 + 0 + 0 + 1 = 2$, and $0 + 0 + 1 + 0 = 1$.

Since you are a very junior developer, your job is not to find the original binary string from a given SMS, but rather the number of binary strings that could have formed this SMS.

Input Specification

The first line of input contains the two space-separated integers N and K where $1 \leq K \leq N$.

The second line of input contains $N - K + 1$ space-separated integers which is the SMS of at least one binary string.

Marks Awarded	**Bounds on N**	**Additional Bounds on K**
3 marks	$1 \leq N \leq 10$	$K \leq 3$
3 marks	$1 \leq N \leq 10$	None
4 marks	$1 \leq N \leq 1\,000$	$K \leq 10$
4 marks	$1 \leq N \leq 10^6$	$K \leq 20$
4 marks	$1 \leq N \leq 10^6$	$K \leq 3\,000$
7 marks	$1 \leq N \leq 10^6$	None

Output Specification

Output the remainder of T divided by the prime number $10^6 + 3$ where T is the positive integer equal to the total number of possible binary strings that correspond to the given SMS.

Sample Input

```
7 4
3 2 2 2
```

Output for Sample Input

```
3
```

Explanation of Output for Sample Input

The possible strings of length 7 are `1011001`, `1101010`, and `1110011`.

2023 Canadian Computing

PROBLEM 5

Real Mountains

Time Limit: 5 seconds

Problem Description

Thanks to your help with cropping her picture, Rebecca's scenic photo is now featured on the front cover of the newest issue of her magazine. However, it seems that some of her readers still aren't pleased with the picture. In particular, they seem to believe that the mountain in the picture is fake!

For simplicity, we can describe the picture as a sequence of N columns of pixels. In the i-th column, the first h_i pixels from the bottom are of mountains. Her readers will only believe that the picture contains a real mountain if it contains a single (possibly wide) peak. That is, if there exists some index p with $1 \leq p \leq N$ such that $h_1 \leq h_2 \leq \cdots \leq h_p \geq \cdots \geq h_{N-1} \geq h_N$.

Luckily, Rebecca can still pay her editors to modify the picture and reprint the magazine. Unfortunately for her though, the editors have a very peculiar pricing scheme for their work. The only way Rebecca can edit the picture is by sending emails to her editors containing the integers (i, j, k) such that $1 \leq i < j < k \leq N$ and $h_i > h_j < h_k$. The editors will then add an extra pixel of mountains in the j-th column (i.e. increment h_j by 1) for a cost of $h_i + h_j + h_k$ cents. Note that the change in h_j may affect the costs of future edits.

To please her readers, Rebecca would like to edit the picture so that they believe it contains a real mountain. Can you tell her the minimum cost required to do so?

Input Specification

The first line of input contains an integer N.

The second line of input contains N space-separated integers $h_1, h_2, \ldots, h_N$.

Marks Awarded	**Bounds on N**	**Bounds and constraints on h_i**
3 marks	$3 \leq N \leq 5\,000$	$1 \leq h_i \leq 100$; $h_1 \geq h_2 \geq \cdots \geq h_p \leq \cdots \leq h_{N-1} \leq h_N$ for some p, $1 \leq p \leq N$
3 marks	$3 \leq N \leq 5\,000$	$1 \leq h_i \leq 100$
3 marks	$3 \leq N \leq 5\,000$	$1 \leq h_i \leq 10^6$
3 marks	$3 \leq N \leq 5\,000$	$1 \leq h_i \leq 10^9$
4 marks	$3 \leq N \leq 10^6$	$1 \leq h_i \leq 100$
5 marks	$3 \leq N \leq 10^6$	$1 \leq h_i \leq 10^6$
4 marks	$3 \leq N \leq 10^6$	$1 \leq h_i \leq 10^9$

Output Specification
Output the remainder of T divided by the prime number $10^6 + 3$ where T is the minimum cost (in cents) that Rebecca would need to incur in order to please her readers.

Sample Input
```
8
3 2 4 5 4 1 2 1
```

Output for Sample Input
```
14
```

Explanation of Output for Sample Input
Rebecca can send two emails, the first containing the integers $(2, 6, 7)$ and the second containing the integers $(1, 2, 5)$. The first email costs 5 cents and increases h_6 by 1, while the second email costs 9 cents and increases h_2 by 1.

The h_i values in the final picture will be $[3, 3, 4, 5, 4, 2, 2, 1]$. Her readers will believe this final picture contains a real mountain.

2023 Canadian Computing

PROBLEM 6

Line Town

Time Limit: 2 seconds

Problem Description

The N residents of Line Town have arranged themselves in a line. Initially, the residents have happiness values of $h_1, h_2, \ldots, h_N$ from left to right along the line.

Since you are the mayor of Line Town, you are implementing the third pillar of your plan entitled "Community, Candy, and Organization" (CCO). As such, you have taken the mayoral power to swap the resident's locations. In one swap, you may tell two *adjacent* residents to swap their positions in the line. However, this swap will cause both residents to negate their happiness values.

You would like to perform some swaps so that the residents' happiness values are in non-decreasing order from left to right in the line. Determine whether this is possible, and if so, the minimum number of swaps needed.

Input Specification

The first line of input contains a single integer N.

The next line of input contains N integers $h_1, \ldots, h_N$ ($-10^9 \leq h_i \leq 10^9$), the happiness values of the residents from left to right.

Marks Awarded	Bounds on N	Bounds on h_i
3 marks	$1 \leq N \leq 2\,000$	$\lvert h_i \rvert = 1$ for all i
3 marks	$1 \leq N \leq 500\,000$	$\lvert h_i \rvert = 1$ for all i
3 marks	$1 \leq N \leq 2\,000$	$\lvert h_i \rvert \leq 1$ for all i
4 marks	$1 \leq N \leq 500\,000$	$\lvert h_i \rvert \leq 1$ for all i
4 marks	$1 \leq N \leq 2\,000$	$\lvert h_i \rvert \neq \lvert h_j \rvert$ for all $i \neq j$
3 marks	$1 \leq N \leq 500\,000$	$\lvert h_i \rvert \neq \lvert h_j \rvert$ for all $i \neq j$
2 marks	$1 \leq N \leq 2\,000$	No additional constraints.
3 marks	$1 \leq N \leq 500\,000$	No additional constraints.

Output Specification

On a single line, output the minimum number of swaps, or -1 if the task is impossible.

Sample Input 1

```
6
-2 7 -1 -8 2 8
```

Output for Sample Input 1

```
3
```

Explanation of Output for Sample Input 1

It is possible to perform 3 swaps as follows:

1. Swap the 2nd and 3rd resident so that the line becomes $[-2, 1, -7, -8, 2, 8]$.
2. Swap the 4th and 5th resident so that the line becomes $[-2, 1, -7, -2, 8, 8]$.
3. Swap the 3rd and 4th resident so that the line becomes $[-2, 1, 2, 7, 8, 8]$.

The residents are now arranged in non-decreasing order of happiness values as required. No non-decreasing arrangement can be obtained with less than 3 swaps.

Sample Input 2

```
4
1 -1 1 -1
```

Output for Sample Input 2

```
-1
```

Explanation of Output for Sample Input 2

There is no sequence of swaps that will place residents in non-decreasing order of happiness values.

Marks	Bounds on N	Additional Constraints
3 marks	$3 \leq N \leq 3^{18}$	N is a power of 3
4 marks	$2 \leq N \leq 10^5$	None
8 marks	$2 \leq N \leq 10^9$	None

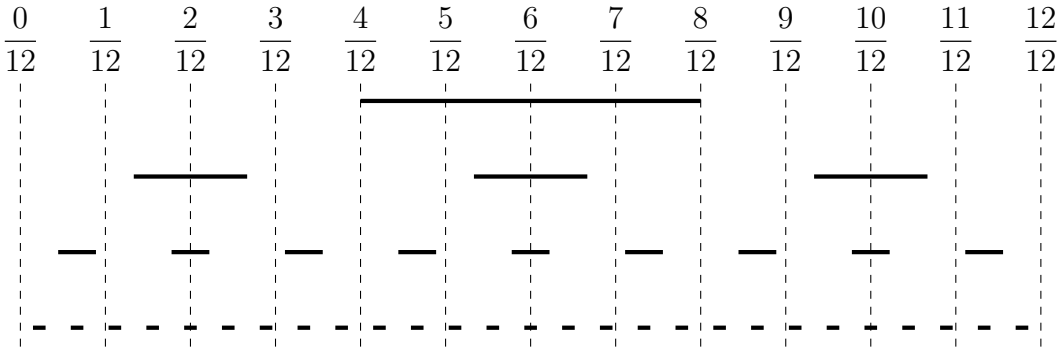
Output Specification
Output all integers x where $0 \leq x \leq N$ and $\frac{x}{N}$ is in the Cantor set.

Output the answers in increasing order. The number of answers will not exceed 10^6.

Sample Input
```
12
```

Output for Sample Input
```
0
1
3
4
8
9
11
12
```

Explanation of Output for Sample Input
Here is a diagram of the fractions and the first 4 filters. In reality, there are infinitely many filters.



$\frac{5}{12}$, $\frac{6}{12}$, and $\frac{7}{12}$ are not in the Cantor set because they were covered by the 1st filter.

Furthermore, $\frac{2}{12}$ and $\frac{10}{12}$ are not in the Cantor set because they were covered by the 2nd filter.

It can be shown that the remaining fractions will pass through all filters.

La version française figure à la suite de la version anglaise.

J1 Deliv-e-droid

A small calculation involving the input values is needed to calculate the number of points gained based on the number of packages delivered and the number of points lost based on the number of collisions with obstacles. An if statement is also needed to possibly apply bonus points.

J2 Chili Peppers

This problem can be solved by using a variable to accumulate the total spiciness of Ron's chili. This value can be updated for each pepper name read from the input using a loop which iterates N times. One way to perform this update in the body of the loop is to use an if statement with a test for each possible pepper name. An alternative approach is to effectively place the table given in the problem statement in memory and then use the table to look up the SHU value given the name of a pepper. Different programming languages provide different ways of storing this table.

J3 Special Event

Each of the subtasks for this problem requires us to keep a count for each day of the number of people able to attend the event on that day. These can be kept in five separate variables because there are only five days. However, it is cleaner and probably less error-prone to store these counts in a list or array.

For the first subtask, we can simply search for a day with a count of N . We are told that there will only be one such day.

For the second subtask, we have to calculate the maximum of these five counts and then find the day corresponding to this maximum value. We are told there will only be one such day.

For the third and final subtask, we need to handle "ties". That is, there could be more than one day with the same maximum count value. It is a bit tricky to output the right answer in this case. It is important to avoid printing a comma after the final day number.

J4 Trianglane

To help Bocchi determine how much tape she needs, we need to do some careful counting.

For the first subtask, we can count B, the number of black triangles, which is the number of times 1 appears in the input. The correct answer will be $3 \times B$. Note that we can ignore the second line of input which is also true for the second subtask. However, for this second subtask, there could be adjacent black triangles and Bocchi does not need to place warning tape on the common side in these cases. Notice that each of these common sides contributes two to the value of $3 \times B$. This means we obtain the correct final answer by subtracting two from $3 \times B$ for each pair of adjacent black triangles.

The same principle applies to the third subtask but we also have to look for black triangles that are adjacent but in different rows. These can only occur at even-numbered positions (assuming we start counting at

zero).

It is possible to solve this problem without making the observation that you can subtract from 3 × B. This approach involves looking for all the places where warning tape needs to be placed. An algorithm is needed which includes sides of triangles on the perimeter of the pathway as well as sides between adjacent triangles which are not the same colour.

Regardless of which approach is taken, many solutions to the first three subtasks will be quite efficient. However, the fourth subtask is meant to ensure this by disallowing overly slow solutions. It specifically requires that not too many passes are made over either row. More formally, only a constant number of passes of each row is allowed. Solutions that miss this requirement are probably too slow because they use a built-in function within a loop which itself loops through a row of triangles.

J5 CCC Word Hunt

The intended challenge of the final problem in this year's competition was different in nature than the last few years. Here, systematically considering each possible placement of the word in the grid will earn full marks if implemented correctly. In fact, it is not possible to solve this problem more efficiently. What some participants may have found difficult is coding this algorithm or approach so that it works correctly in all cases.

We need to carefully consider all possible starting locations of the word and each possible direction from each of these locations. The number of possibilities increases from subtask to later subtask.

Other than for the first subtask, the grid must be stored in memory. It is a two-dimensional structure so a data structure such as a list of lists (or array of arrays) is needed. It is easy to make errors "around the edge of the grid". Care must be taken to only access valid locations within the data structure of choice.

For this last subtask, the possibility of perpendicular bends in the hidden word presents a significant additional challenge because the number of possible locations for the hidden word is quite large. This can result in long repetitive code with lots of cases which is very error-prone. Carefully designed functions with well-chosen parameters can help remove lots of the redundancy reducing the likelihood of errors and simplifying the process of debugging if errors do occur.

S1 Trianglane

To help Bocchi determine how much tape she needs, we need to do some careful counting.

For the first subtask, we can count B, the number of black triangles, which is the number of times 1 appears in the input. The correct answer will be 3 × B. Note that we can ignore the second line of input which is also true for the second subtask. However, for this second subtask, there could be adjacent black triangles and Bocchi does not need to place warning tape on the common side in these cases. Notice that each of these common sides contributes two to the value of 3 × B. This means we obtain the correct final answer by subtracting two from 3 × B for each pair of adjacent black triangles.

The same principle applies to the third subtask but we also have to look for black triangles that are adjacent but in different rows. These can only occur at even-numbered positions (assuming we start counting at zero).

It is possible to solve this problem without making the observation that you can subtract from 3 × B. This approach involves looking for all the places where warning tape needs to be placed. An algorithm is needed which includes sides of triangles on the perimeter of the pathway as well as sides between adjacent triangles which are not the same colour.

Regardless of which approach is taken, many solutions to the first three subtasks will be quite efficient. However, the fourth subtask is meant to ensure this by disallowing overly slow solutions. It specifically requires that not too many passes are made over either row. More formally, only a constant number of passes of each row is allowed. Solutions that miss this requirement are probably too slow because they use a built-in function within a loop which itself loops through a row of triangles.

S2 Symmetric Mountains

To solve this problem, we will find the asymmetric value for each crop and compute the minimum asymmetric value for each crop's corresponding length. For every subtask, we find different ways to compute the asymmetric value of each crop.

Subtask 1

Considering iterating over all crops. A crop is defined by its left endpoint and right endpoint. Hence, we can use a nested for loop to first fix the left endpoint and then fix the right endpoint. There are other ways to loop through crops, such as looping through its left endpoint and its length and vice versa. To find the asymmetric value of a crop, we can have an accumulator and sum up the absolute difference for every such pair by looping over i defined in the problem statement. There are $N \cdot$

2

$(N - 1)/2 = N/2 - N/2$ such crops,

and we need about N calculations on average to to compute the asymmetric value of each crop; hence, it runs in time that is roughly cubic in the number of mountains.

Subtask 2

For this subtask, we note that if we were to find the asymmetric value of each crop, then we have to do it faster than roughly N time. Now consider a crop with a left endpoint of l and a right endpoint of r. We will take a closer look at the formula to compute the asymmetric value:

$|h_l - h_r| + |h_{l+1} - h_{r-1}| + |h_{l+2} - h_{r-2}| + \ldots + |h_{l+t} - h_{r-t}|$

where $t = \frac{r-l}{2}$. Note that because the array is non-decreasing in this subtask, then it follows that for mountains with indices $i \leq j$, we have that $|h_i - h_j| = h_j - h_i$. Hence the above formula is equivalent to

$(h_r - h_l) + (h_{r-1} - h_{l+1}) + (h_{r-2} - h_{l+2}) + \ldots + (h_{r-t} - h_{l+t})$.

Let $m = \frac{l+r}{2}$, which represents the midpoint of the mountains (it may not be an integer, but that does not matter). Notice that for every term whose index is greater than m, it is being added, and for every term whose index is less than m it is being subtracted. We can then rearrange the formula to become

$h_r + h_{r-1} + h_{r-2} + \ldots + h_{r-t} - (h_l + h_{l+1} + h_{l+2} + \ldots + h_{l+t})$.

Note that we have simplified this problem into static range sum queries. We can compute them quickly by building a prefix sum array of the mountains' heights (i.e., an array where the kth element is the sum of the first k elements of h) before iterating through all the crops. Note that to calculate the asymmetric crop, in this case, it will be constant time (i.e., independent of N), so this runs in quadratic time in the number of mountains.

Subtask 3

We will now let [l, r] represent a crop where $l \leq r$ are the left and right endpoints, respectively. Now for this subtask, let us take a closer look at the formula for the crop [l, r]:

$|h_l - h_r| + (|h_{l+1} - h_{r-1}| + |h_{l+2} - h_{r-2}| + \ldots + |h_{l+t} - h_{r-t}|)$.

Notice that the terms in brackets are the asymmetric value of the crop [l + 1, r − 1]. If we have a way to iterate through the crops so that crop [l + 1, r − 1] comes right before [l, r], we can add $|h_l - h_r|$ directly to the accumulator to get the asymmetric value of crop [l, r]. This motivates us to cleverly iterate over the crops by fixing the midpoint and expanding outwards while maintaining an accumulator for each midpoint.

Note that the midpoint will not be an actual mountain for crops with even length. This runs in time which is the square of the number of mountains. As we move from crop [l + 1, r − 1] to [l, r], we only add a term to the accumulator. Note that there exist dynamic programming solutions, but these were not necessary to receive full marks.

S3 Palindromic Poster

Subtask 1

For this subtask, since R = 1 and C = 1, we can try and make only the first row and column a palindrome. One way to do this is to fill the first row and column with the letter a and fill the rest of the grid with the letter b.

Subtask 2

Because there are so few cases, it suffices to solve this subtask on paper and hardcode the answers. To reduce the amount of casework, notice that if we have a solution for (R, C), we can get a solution for (C, R) by flipping the grid.

La version fran¸caisefigure`alasuitedelaversionanglaise.

Another possible approach is to write a brute force since there are only four important possibilities for each cell.

Subtask 3

This subtask is intended to aid students in thinking about the problem.

Subtask 4

The solutions to subtasks 1 and 2 should give some intuition here. From subtask 1, we propose the following general solution:

Fill the first R rows and the first C columns with the letter a and fill the rest of the grid with the letter b.

We notice that this fails when R = 0, C = 0, R = N or C = M . We can handle these in pairs since we can flip the grid.

If R = 0, we can fill the first M −1 columns with the letter a, and the last column with the letter b, and then increment the last M − C characters of the last row. For instance, for the input 4 4 0 2 we can output:

aaab

aaab

aaab

aabc

If R = N , every row must be a palindrome. Because of this, starting from a full grid of a characters, we can

easily reduce the number of palindromic columns by two at a time by making matching columns of the first row b characters. For instance, for the input 4 4 4 2, we can output:

baab

aaaa

aaaa

aaaa

However, this fails if C is not the same parity as M , that is, if we cannot get C by subtracting 2 an integer number of times from M .

In this case, it depends: if M is odd, we can use the middle column. For instance, for the input 5 5 5 2:

babab

aaaaa

aaaaa

aaaaa

aaaaa

However, the input 4 4 4 1 is impossible since there is no middle column.

We can handle C = 0 and C = M symmetrically by flipping the inputs, processing, and then flipping the output.

S4 Minimum Cost Roads

Subtask 1

This subtask can be solved by running an MST (Minimum Spanning Tree) algorithm that only considers

La version fran¸caisefigure`alasuitedelaversionanglaise.

the cost of each edge. Note that we may have multiple components, so our final answer will be the sum of costs of the MST of each component. Luckily, an algorithm like Kruskal's handles this problem without any additional complexity.

Subtask 2

The constraints of this subtask allow us to focus on an invariant that is very useful in the long run. Consider a single edge e = (a, b, l, c). We get the following cases:

If e is the unique shortest path from a to b, we definitely want to take e;

If there exists a path from a to b with length l < l, we definitely don't want to take e;

If there exists another path from a to b with length l = l (but no paths that are shorter), we still don't want to take e. However, this time the proof is much more complex:

Suppose p is a path from a to b consisting of edges {p1, p2, . . . , pk} such that

$$\sum_{i=1}^{k-1} \text{length}(p_i) = l$$

Now, suppose that i, pi is the unique shortest path between its endpoints a(pi) and b(pi). If this is not the case for some i, then we can find another path from a(pi) to b(pi) with length exactly length(pi) and replace pi with that path (if that path had length < length(pi) then we would have a path from a to b with length < l, which is a contradiction). Observe that this process can only be repeated a finite number of times.

In the end, we have a path from a to b that has length l, does not take the edge e, and every edge in that path is the unique shortest path between its endpoints. We'll call the edges on this path q1, . . . , qm.

Now consider some optimal answer that contains the edge e. Clearly, q1, . . . , qm must be in the answer. However, this means that we can remove e as any time we need to traverse from a to b, we can take the edges q1, . . . , qm instead. This violates the optimality of the answer and is thus a contradiction.

This results in a very simple solution for this subtask: we loop over every edge e = (a, b, l, c) and check whether it's the unique shortest path between a and b. One way to do this is to first run Dijkstra's algorithm n times to find the shortest path between every pair of nodes (and store it in a 2-D array dist).

Then, for each edge we check if

$$l < \min_{1 \le i \le n, i \checkmark = a, i \checkmark = b} (\text{dist}[a][i] + \text{dist}[i][b])$$

Remark: Notice that we don't even care about costs at all! In fact, the original problem didn't have costs (li = ci), but we thought it was an interesting extension :)

Subtask 3

Unfortunately, we lose the guarantees that allow our solution from subtask 2 to work. Fortunately, we can restore these guarantees with some clever reductions.

First, let's look at the graph formed by all of the 0-length edges. If any pair of nodes (a, b) are connected

in this graph, then our answer must contain a path of length 0 between a and b. Thus, we must choose a

La version fran¸caisefigure`alasuitedelaversionanglaise.

subset of the 0-length edges that has the same connectivity properties as the original graph (i.e. if (a, b) are connected by 0-length edges in the original graph, then they must be connected by 0-length edges in our answer). Trying to do so while also minimizing cost is analogous to finding the minimum spanning forest of the graph, which is the same as our solution in subtask 1.

Unfortunately, the 0-length edges are still part of our graph. In order to remove them for good, we must make the following observation: we can treat each component in our spanning forest as a single node as we're able to travel between any two nodes in that component with a path of length 0. This motivates us to compress our initial graph by these components to remove all 0-length edges.

Note: Be careful, the compressed graph may also contain self-edges so make sure to ignore them.

Next, to remove multiedges notice that we only ever want at most one edge between any pair of nodes. Thus, we take the one with lowest length, breaking ties by cost.

Finally, we run our solution from subtask 2 and sum the answer we obtained from that solution with the one from computing the minimum spanning forest.

Alternative Solution

There is also an alternative solution much closer to Kruskal's algorithm for computing MST: we loop over the edges in order of length, breaking ties by cost and adding it to the graph if it improves the shortest path between its endpoints. This solution also has the added benefit of not worrying about 0-length edges or multi-edges separately.

S5 The Filter

Subtask 1

The answers for N = 3 are 0, 1, 2, 3.

k k+1

Suppose the answers for N = 3 are a1, a2, . . . , aM . The answers for the first third of N = 3 are

a1, a2, . . . , aM

The Cantor set has copies of the same structure. In particular, the first third of the Cantor set is identical

k+1

to the last third. The remaining answers for N = 3 are

2 ×

k k k

3 + a1, 2 × 3 + a2, . . . , 2 × 3 + aM

These observations are enough to generate the answer for any power of 3.

Subtask 2

34676 th

Note that is covered by the 49 filter, but not any earlier filter. This is the maximum record for

51169

5

$N \leq 10$. A naive approach is expected to fail on N = 51169.

This subtask requires deeper observations about Alice's filters.

Property 1 (Filters are symmetric). If r is covered by the k-th filter, then 1 − r is covered by the k-th filter. Likewise, if the k-th filter does not cover r, then 1 − r is not covered by the k-th filter.

Property 2 (The k-th filter and (k + 1)-th filter are related). Let k be a positive integer, and let

1

$0 \leq r \leq$. If r is covered by the (k + 1)-th filter, then 3r is covered by the k-th filter. Likewise, if r is not

3

covered by the (k + 1)-th filter, then 3r is not covered by the k-th filter.

La version fran¸caisefigure`alasuitedelaversionanglaise.

The proof of these 2 properties is left as an exercise. From these properties, it becomes natural to define the function f (r) = 3 × min(r, 1 − r). Here are a few theorems about f .

Theorem 1. If r is not covered by any filter, then f (r) is not covered by any filter.

Proof. Apply Property 1 and Property 2.

Theorem 2. Suppose r is covered by the k-th filter. Then either k = 1, or f (r) is covered by the (k − 1)-th filter.

Proof. Apply Property 1 and Property 2.

Here is the approach to solving subtask 2:

x

Let r = . Construct this sequence: r, f (r), f (f (r)), f (f (f (r))), . . .

N

If r is in the Cantor set, then by Theorem 1, every term in the sequence is in the Cantor set. Since

1

every term is a multiple of , the sequence will enter a cycle.

N

If r is not in the Cantor set, then r is covered by a filter. By Theorem 2, a term in the sequence will

be covered by the first filter.

There are 2 outcomes. Either the sequence will enter a cycle, or a term will be covered by the first filter.

The algorithm needs to handle both outcomes. The intended time complexity is O(N) or slightly slower.

Alternatively, it is enough to ignore cycle detection and construct the first 49 terms. In the worst case (i.e.

34676

r =), the 49-th term is covered by the first filter.

51169

Subtask 3

222534799 th

Note that is covered by the 130 filter, but not any earlier filter. The author believes this is the

867579680

9

maximum record for N ≤ 10 .

6

Firstly, use the first 18 filters to remove most of the numbers. After this step, less than 10 numbers remain.

Next, apply the algorithm from subtask 2 on each of the remaining numbers. It may be a good idea to use

memoization to improve time complexity. There are other tricks to improve performance.

log 2 log 2

The intended time complexity is O(N

3

log N) because there are O(N

3

) numbers to consider, and each

number takes roughly O(log N) to consider.