Contest 1 : EV

A. Coloring the Blocks

1 second, 256 megabytes

Problem Description

There are *n* blocks placed in a row. Each block must be covered with one of the three colors available, but no two adjacent blocks can be the same color. The cost of coloring each block varies and is given in an array. Given the cost of using each color on each block, determine the minimum cost to color all of the blocks.

Example

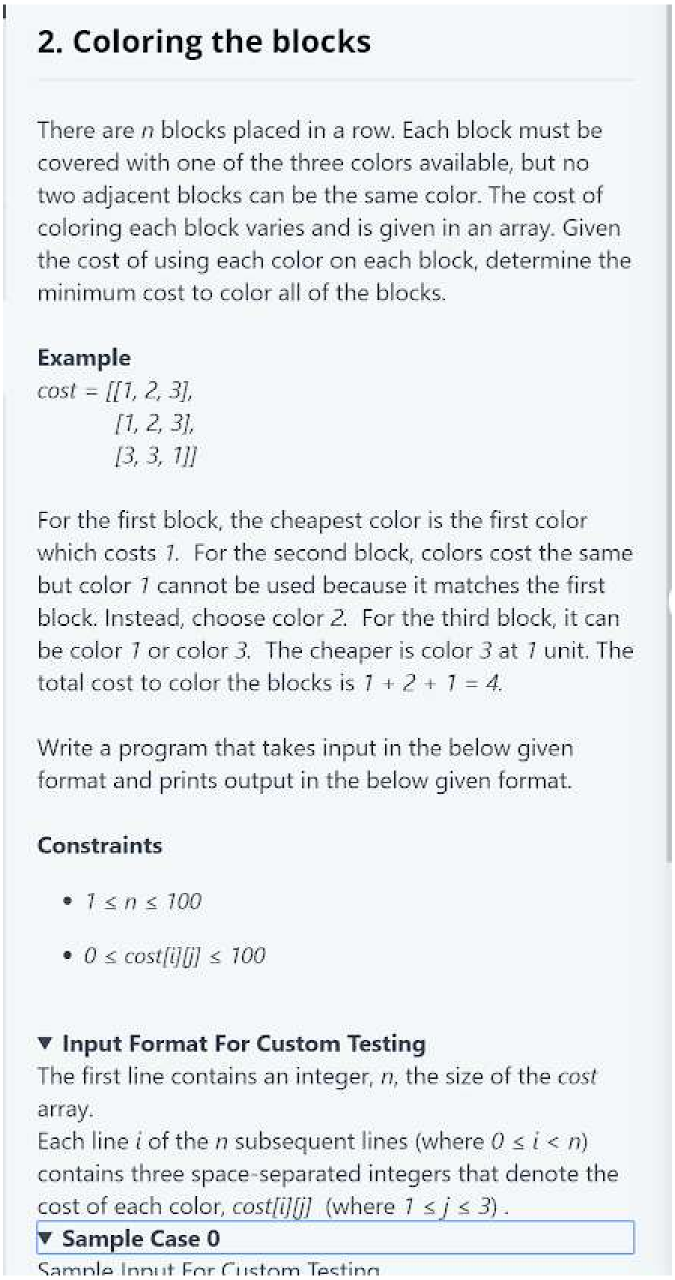
cost $= [[1, 2, 3], [1, 2, 3], [3, 3, 1]]$

For the first block, the cheapest color is the first color which *costs* 1 *unit*. For the second block, colors cost the same but *color 1* cannot be used because it matches the first block. Instead, choose *color 2*. For the third block, it can be *color 1* or *color 3*. The cheaper is *color 3* at 1 *unit*. The total cost to color the ***blocks*** is $1 + 2 + 1 = 4$.

Write a program that takes input in the below-given format and prints output in the below-given format.

Constraints

- $1 \leq n \leq 100$
- $0 \leq cost[i][j] \leq 100$



2. Coloring the blocks

There are *n* blocks placed in a row. Each block must be covered with one of the three colors available, but no two adjacent blocks can be the same color. The cost of coloring each block varies and is given in an array. Given the cost of using each color on each block, determine the minimum cost to color all of the blocks.

Example

cost = [[1, 2, 3],
[1, 2, 3],
[3, 3, 1]]

For the first block, the cheapest color is the first color which costs *1*. For the second block, colors cost the same but color *1* cannot be used because it matches the first block. Instead, choose color *2*. For the third block, it can be color *1* or color *3*. The cheaper is color *3* at *1* unit. The total cost to color the blocks is *1 + 2 + 1 = 4*.

Write a program that takes input in the below given format and prints output in the below given format.

Constraints

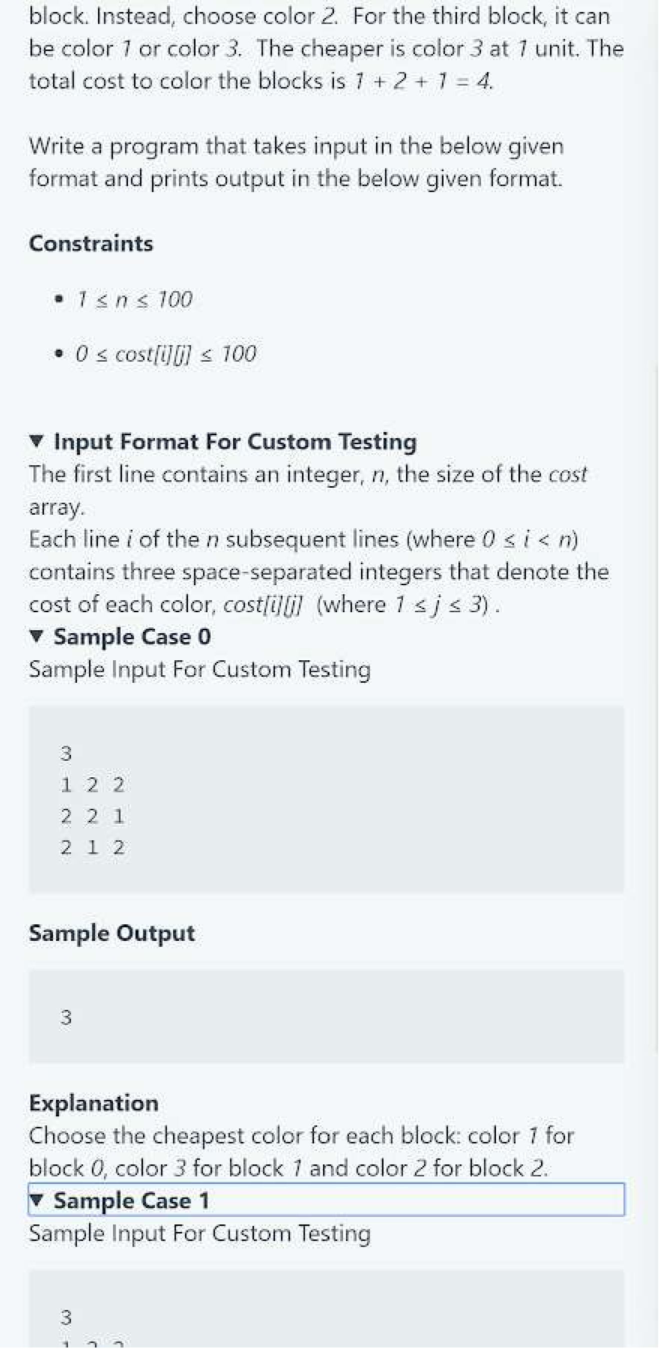
- *1 ≤ n ≤ 100*
- *0 ≤ cost[i][j] ≤ 100*

▼ Input Format For Custom Testing

The first line contains an integer, *n*, the size of the *cost* array.
Each line *i* of the *n* subsequent lines (where *0 ≤ i < n*) contains three space-separated integers that denote the cost of each color, *cost[i][j]* (where *1 ≤ j ≤ 3*) .

▼ Sample Case 0

Sample Input For Custom Testing



block. Instead, choose color *2*. For the third block, it can be color *1* or color *3*. The cheaper is color *3* at *1* unit. The total cost to color the blocks is *1 + 2 + 1 = 4*.

Write a program that takes input in the below given format and prints output in the below given format.

Constraints

- *1 ≤ n ≤ 100*
- *0 ≤ cost[i][j] ≤ 100*

▼ Input Format For Custom Testing

The first line contains an integer, *n*, the size of the *cost* array.
Each line *i* of the *n* subsequent lines (where *0 ≤ i < n*) contains three space-separated integers that denote the cost of each color, *cost[i][j]* (where *1 ≤ j ≤ 3*) .

▼ Sample Case 0

Sample Input For Custom Testing

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

Sample Output

```
3
```

Explanation

Choose the cheapest color for each block: color *1* for block *0*, color *3* for block *1* and color *2* for block *2*.

▼ Sample Case 1

Sample Input For Custom Testing

```
3
```

Input

The first line contains an integer n, the size of the cost array.

Each line i of the n subsequent lines (where $0 \leq i < n$) contains three space-separated integers that denote the cost of each color, $cost[i][j]$, (where $1 \leq j \leq 3$).

Output

Print the minimum cost to color the blocks.

input

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

output

```
3
```

input

```
3
3 5 6
1 4 5
2 3 2
```

output

```
8
```

For the first sample test case, choose the cheapest color for each block. Color 1 for block 0, Color 3 for block 1, Color 2 for block 2.

B. Scatter Palindromes

1 second, 256 megabytes

Problem Description

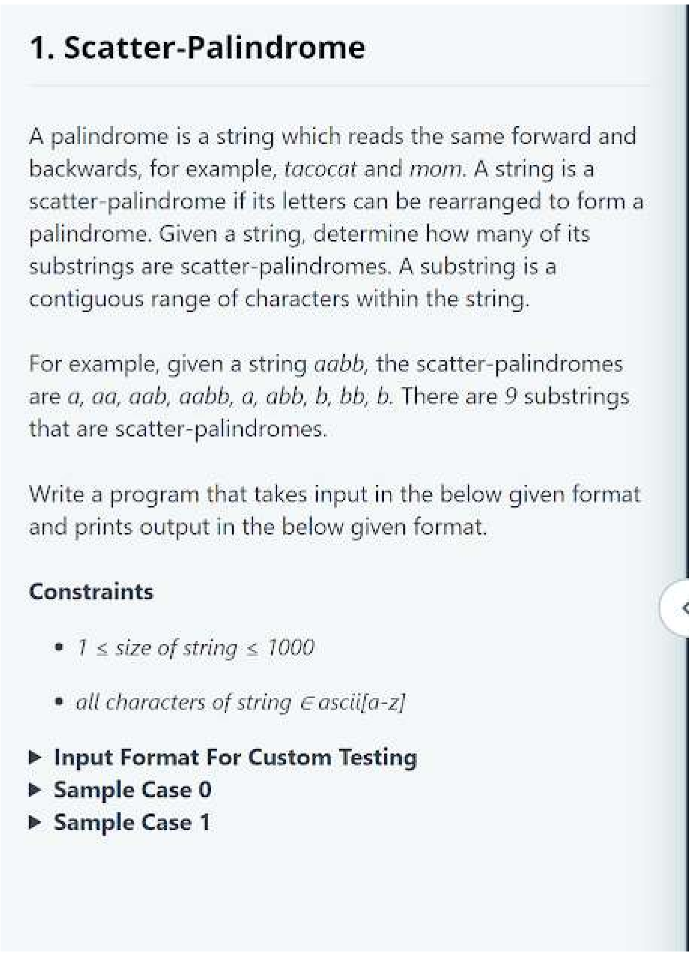
A palindrome is a string which reads the same forward and backwards, for example, ***tacocat*** and ***mom***. A string is a scatter-palindrome if its letters can be rearranged to form a palindrome. Given a string, determine how many of its substrings are **scatter-palindrome**. A **substring** is a contiguous range of characters within the string.

For example, given a string $aabb$, the **scatter-palindromes** are $a, aa, aab, aabb, a, abb, b, bb, b$. There are 9 substrings that are **scatter-palindromes**.

Write a program that takes input in the below given format and prints output in the below given format.

Constraints

- $1 \leq size\ of\ string \leq 1000$
- The string contains lower-case alphabets.



1. Scatter-Palindrome

A palindrome is a string which reads the same forward and backwards, for example, *tacocat* and *mom*. A string is a scatter-palindrome if its letters can be rearranged to form a palindrome. Given a string, determine how many of its substrings are scatter-palindromes. A substring is a contiguous range of characters within the string.

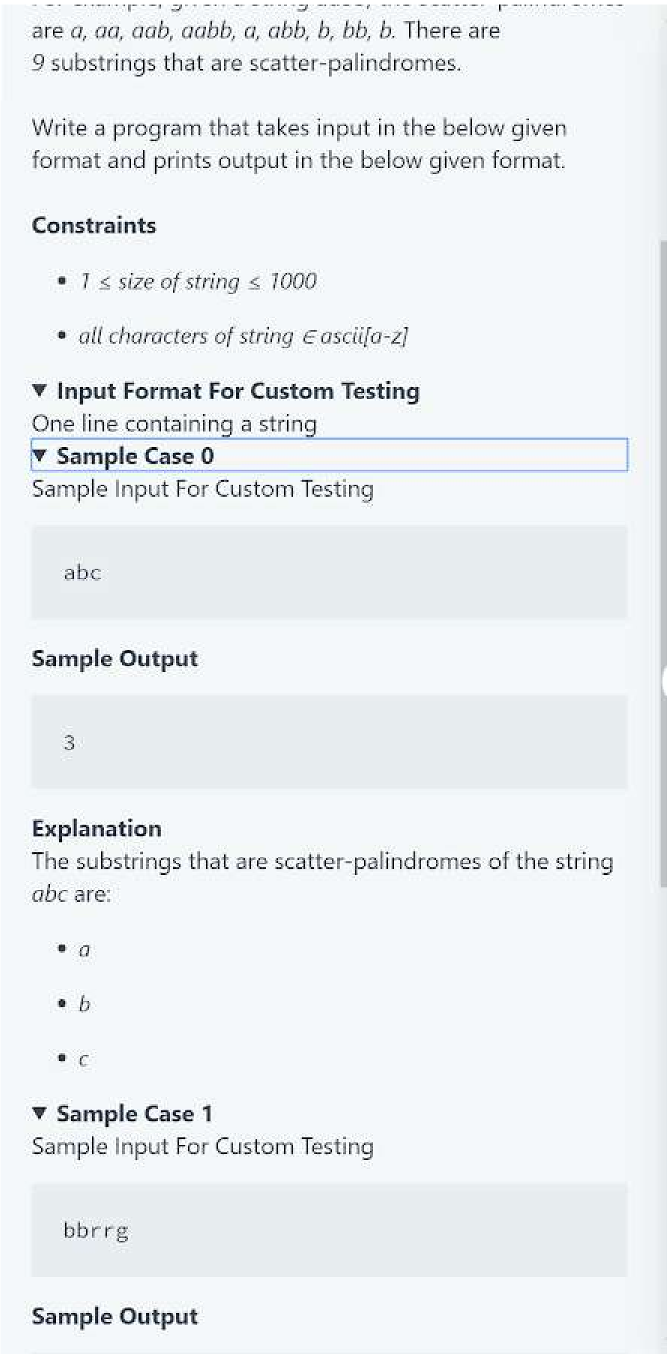
For example, given a string *aabb*, the scatter-palindromes are *a*, *aa*, *aab*, *aabb*, *a*, *abb*, *b*, *bb*, *b*. There are *9* substrings that are scatter-palindromes.

Write a program that takes input in the below given format and prints output in the below given format.

Constraints

- *1 ≤ size of string ≤ 1000*
- *all characters of string ∈ ascii[a-z]*

▶ Input Format For Custom Testing
▶ Sample Case 0
▶ Sample Case 1



are *a, aa, aab, aabb, a, abb, b, bb, b*. There are *9* substrings that are scatter-palindromes.

Write a program that takes input in the below given format and prints output in the below given format.

Constraints

- *1 ≤ size of string ≤ 1000*
- *all characters of string ∈ ascii[a-z]*

▼ **Input Format For Custom Testing**

One line containing a string

▼ **Sample Case 0**

Sample Input For Custom Testing

```
abc
```

Sample Output

```
3
```

Explanation

The substrings that are scatter-palindromes of the string *abc* are:

- *a*
- *b*
- *c*

▼ **Sample Case 1**

Sample Input For Custom Testing

```
bbrrg
```

Sample Output

Input

The first and only line contains the string

Output

Print the number of scatter-palindromes.

input
abc
output
3

input
bbrrg
output
12

In the first sample case, the substrings which are scatter-palindromes are a, b, c

C. Arbitrary Shopping

1 second, 256 megabytes

Problem Description

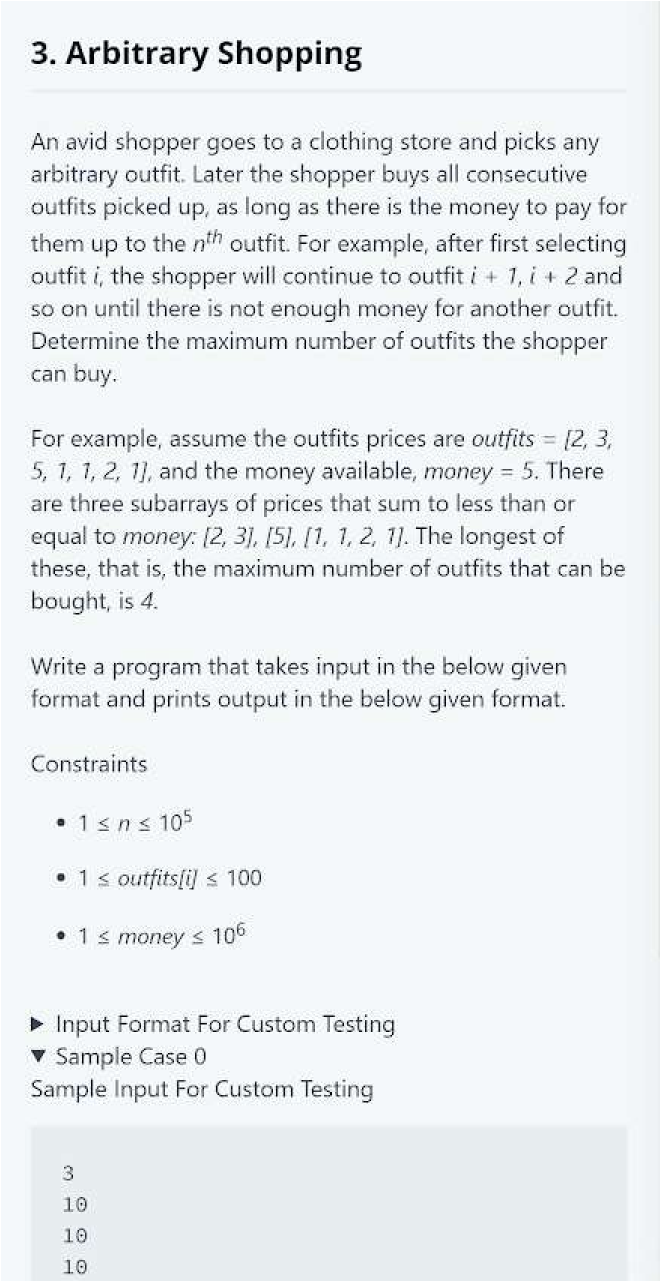
An avid shopper goes to a clothing store and picks any arbitrary outfit. Later the shopper buys all consecutive outfits picked up, as long as there is the money to pay for them up to the n^{th} outfit. For example, after first selecting outfit i, the shopper will continue to outfit $i + 1$, $i + 2$ and so on until there is not enough money for another outfit. Determine the maximum number of outfits the shopper can buy.

For example, assume the outfits prices are ***outfits*** = $[2, 3, 5, 1, 1, 2, 1]$, and the money available, ***money*** = 5. There are **three subarrays** of prices that sum to less than or equal to ***money***:$[2, 3]$, $[5]$, $[1, 1, 2, 1]$. The longest of these, that is, the maximum number of outfits that can be bought, is 4.

Write a program that takes input in the below-given format and prints output in the below-given format.

Constraints

- $1 \leq n \leq 10^5$
- $1 \leq outfits[i] \leq 100$
- $1 \leq money \leq 10^6$



3. Arbitrary Shopping

An avid shopper goes to a clothing store and picks any arbitrary outfit. Later the shopper buys all consecutive outfits picked up, as long as there is the money to pay for them up to the *n^th* outfit. For example, after first selecting outfit *i*, the shopper will continue to outfit *i + 1, i + 2* and so on until there is not enough money for another outfit. Determine the maximum number of outfits the shopper can buy.

For example, assume the outfits prices are *outfits = [2, 3, 5, 1, 1, 2, 1]*, and the money available, *money = 5*. There are three subarrays of prices that sum to less than or equal to *money: [2, 3], [5], [1, 1, 2, 1]*. The longest of these, that is, the maximum number of outfits that can be bought, is *4*.

Write a program that takes input in the below given format and prints output in the below given format.

Constraints

- 1 ≤ *n* ≤ 10^5
- 1 ≤ *outfits[i]* ≤ 100
- 1 ≤ *money* ≤ 10^6

▶ Input Format For Custom Testing

▼ Sample Case 0

Sample Input For Custom Testing

```
3
10
10
10
```

Input

The first line contains an integer n.

The next n lines contain the elements of the $outfit$ array.

The last line contains an integer $money$.

Output

Print the maximum number of outfits that you can buy.

input

```
3
10
10
10
20
```

output

```
2
```

D. Angry Animals

1 second, 256 megabytes

Problem Description

Pi's father, Danny, runs the Hackerville Zoo. He is moving to Rookieville and wants to take all of the zoo animals with him via ship. He is confused about how to arrange them because a few of the species cannot be kept together in the same cabin.

There are n animals placed in a **straight line**. Each animal is identified by a unique number from 1 to n in order. There are m pairs ***(a[i], b[i])*** which imply that animals $a[i]$ and $b[i]$ are **enemies** and should not be kept in the same cabin. Pi is good at solving problems and he came up with following challenge: **Count the number of different groups that do not contain any pair such that they are enemies.** A ***group*** is defined as an ***interval (x,y)*** such that all animals in the range from ***x*** to ***y*** form a group. Determine the ***number of groups*** that can be formed according to Pi's challenge .

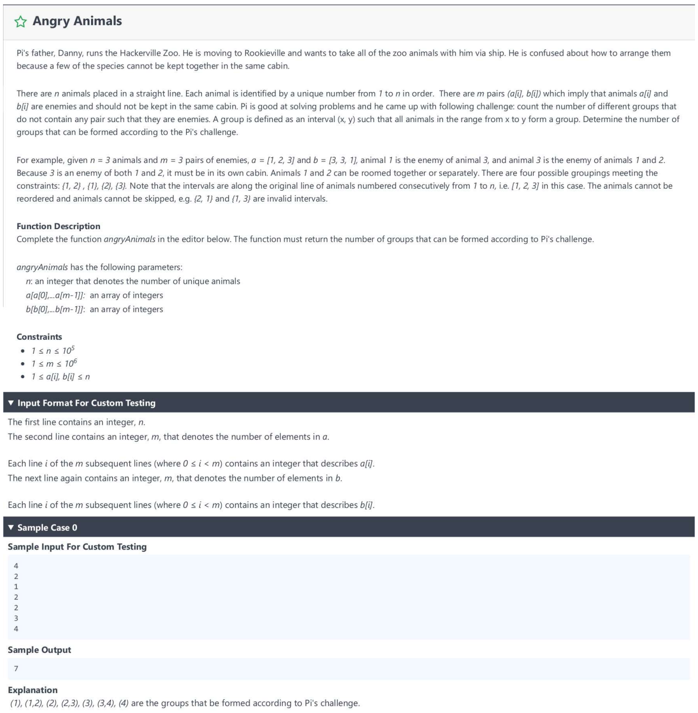
For example, given $n = 3$ animals and $m = 3$ pairs, $a = [1, 2, 3]$ and $b = [3, 3, 1]$, animal 1 is the enemy of animal 3, and animal 3 is the enemy of animal 1 and 2. Because 3 is an enemy of both 1 and 2, it must be in its own cabin.

Animals 1 and 2 can be roomed together or separately. There are four possible groupings meeting the constraints: {1,2} , {1} , {2} , {3}.

Note that the intervals are along the original line of animals numbered consecutively from 1 to n, i.e $\{1, 2, 3\}$ in this case. The animals cannot be reordered and animals cannot be skipped, *e.g* : $\{2, 1\}, \{1, 3\}$ are invalid intervals .

Constraints

- $1 \leq n \leq 10^5$
- $1 \leq m \leq 10^6$
- $1 \leq a[i], b[i] \leq n$



☆ Angry Animals

Pi's father, Danny, runs the Hackerville Zoo. He is moving to Rookieville and wants to take all of the zoo animals with him via ship. He is confused about how to arrange them because a few of the species cannot be kept together in the same cabin.

There are *n* animals placed in a straight line. Each animal is identified by a unique number from *1* to *n* in order. There are *m* pairs *(a[i], b[i])* which imply that animals *a[i]* and *b[i]* are enemies and should not be kept in the same cabin. Pi is good at solving problems and he came up with following challenge: count the number of different groups that do not contain any pair such that they are enemies. A group is defined as an interval (x, y) such that all animals in the range from x to y form a group. Determine the number of groups that can be formed according to the Pi's challenge.

For example, given *n = 3* animals and *m = 3* pairs of enemies, *a = [1, 2, 3]* and *b = [3, 3, 1]*, animal *1* is the enemy of animal *3*, and animal *3* is the enemy of animals *1* and *2*. Because *3* is an enemy of both *1* and *2*, it must be in its own cabin. Animals *1* and *2* can be roomed together or separately. There are four possible groupings meeting the constraints: *{1, 2} , {1}, {2}, {3}*. Note that the intervals are along the original line of animals numbered consecutively from *1* to *n*, i.e. *[1, 2, 3]* in this case. The animals cannot be reordered and animals cannot be skipped, e.g. *{2, 1}* and *{1, 3}* are invalid intervals.

Function Description
Complete the function *angryAnimals* in the editor below. The function must return the number of groups that can be formed according to Pi's challenge.

angryAnimals has the following parameters:
n: an integer that denotes the number of unique animals
a[a[0],...a[m-1]]: an array of integers
b[b[0],...b[m-1]]: an array of integers

Constraints

- $1 \le n \le 10^5$
- $1 \le m \le 10^6$
- $1 \le a[i], b[i] \le n$

▼ Input Format For Custom Testing

The first line contains an integer, *n*.
The second line contains an integer, *m*, that denotes the number of elements in *a*.

Each line *i* of the *m* subsequent lines (where $0 \le i < m$) contains an integer that describes *a[i]*.
The next line again contains an integer, *m*, that denotes the number of elements in *b*.

Each line *i* of the *m* subsequent lines (where $0 \le i < m$) contains an integer that describes *b[i]*.

▼ Sample Case 0

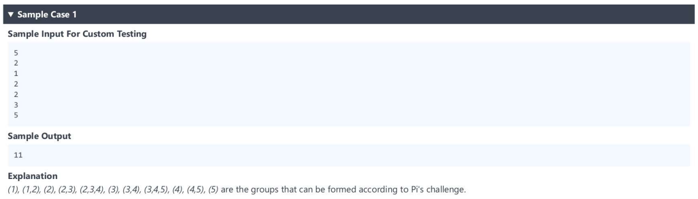
Sample Input For Custom Testing

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

Sample Output

```
7
```

Explanation
(1), (1,2), (2), (2,3), (3), (3,4), (4) are the groups that be formed according to Pi's challenge.



▼ Sample Case 1

Sample Input For Custom Testing

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

Sample Output

```
11
```

Explanation
(1), (1,2), (2), (2,3), (2,3,4), (3), (3,4), (3,4,5), (4), (4,5), (5) are the groups that can be formed according to Pi's challenge.

Input

The first line contains an integer n The second line contains an integer m that denotes the number of elements in a.

Each line i of the m subsequent lines (where $0 \leq i < m$ contains an integer that describes $a[i]$.

The next line again contains an integer m, which denotes the number of elements in b.

Each line i of the m subsequent lines (where $0 \leq i < m$ contains an integer that describes $b[i]$.

Output

Print the number of groups that can be formed according to Pi's challenge.

input

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

output

```
7
```

input

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

output

```
11
```

In the first sample case, $(1), (1, 2), (2), (2, 3), (3), (3, 4), (4)$ are the groups that can be formed according to Pi's challenge.

In the second sample case, $(1), (1, 2), (2), (2, 3), (2, 3, 4), (3), (3, 4), (3, 4, 5), (4), (4, 5), (5)$ are the groups that can be formed according to Pi's challenge.

E. Little Girl and Maximum Sum

1 second, 256 megabytes

The little girl loves the problems on array queries very much.

One day she came across a rather well-known problem: you've got an array of n elements (the elements of the array are indexed starting from 1); also, there are q queries, each one is defined by a pair of integers l_i, r_i $(1 \leq l_i \leq r_i \leq n)$. You need to find for each query the sum of elements of the array with indexes from l_i to r_i, inclusive.

The little girl found the problem rather boring. She decided to reorder the array elements before replying to the queries in a way that makes the sum of query replies maximum possible. Your task is to find the value of this maximum sum.

Input

The first line contains two space-separated integers n $(1 \leq n \leq 2 \cdot 10^5)$ and q $(1 \leq q \leq 2 \cdot 10^5)$ — the number of elements in the array and the number of queries, correspondingly.

The next line contains n space-separated integers a_i $(1 \leq a_i \leq 2 \cdot 10^5)$ — the array elements.

Each of the following q lines contains two space-separated integers l_i and r_i $(1 \leq l_i \leq r_i \leq n)$ — the i-th query.

Output

In a single line print, a single integer — the maximum sum of query replies after the array elements are reordered.

Please, do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

input

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

output

```
25
```

input

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

output

```
33
```