2nd Malaysian Computing Olympiad 2014

20 April 2014

9.30am - 12.30pm

Task	`passport`	`random`	`roads`	`swaps`	`fish`
Name	Passport Checking	Random	Binary Roads	Swaps	Fish
Time limit	1s	1s	3s	1s	1s
Memory limit	64MB	64MB	64MB	64MB	64MB
Points	20+80=100	10+20+70=100	15+85=100	20+80=100	10+20+20+50 =100

input: standard input (*stdin*)

output: standard output (*stdout*)

Passport Checking - proposed by Ong Shien Jin

This task is inspired by the recent MH370 stolen passport incident.

The Interpol database contains a list of **N** stolen passport numbers. We have another list of **M** passport numbers to check if they are stolen or not.

Input

- The first line of the input contains an integer **N**.
- The next **N** lines contains the list of **N** stolen passport numbers, one passport number per line.
- The next line of the input contains an integer **M**.
- The next **M** lines contains the list of **M** passport numbers, one passport number per line.

Output

Output the total number of stolen passports in the list of **M** passports.

Constraints

- Time Limit: 1s
- Memory Limit: 64MB
- $1 \leq \mathbf{N} \leq 100{,}000$
- $1 \leq \mathbf{M} \leq 100{,}000$
- Passport numbers are alphanumeric (combination of alphabetic and numeric characters) and are of length at most 15

Subtasks

Subtask 1 (20 points): $\mathbf{N} \leq 1{,}000$ and $\mathbf{M} \leq 1{,}000$

Subtask 2 (80 points): $\mathbf{N} \leq 100{,}000$ and $\mathbf{M} \leq 100{,}000$

Sample Input 1

```
10
I220232074
D327045452
L261687170
H720628600
T181830847
I108428623
J316167657
M520837168
H364436380
J531555744
5
U133207838
D327045452
S374500074
U882158042
H364436380
```

Sample Output 1

```
2
```

Both `D327045452` and `H364436380` are stolen passports.

Random - proposed by Lim Yun Kai

Mr. Random's job is to supply random numbers for people who comes to him. Unfortunately, generating a massive amount of random numbers is very hard. So, Mr. Random has decided to get a random number generator to help him.

The random number generator can generate a few million random numbers under 1 second. But Mr. Random has noticed that some of the number generated by the random number generator does not seem random to him so he need to filter all the numbers generated.

For a number to be *random* it must satisfy both the following conditions:

1. The number can contain at most 3 consecutive repeated digit.
2. All factors of the number (except 1) must be greater than **K**.

For example, if **K** = 2, then `111223344551`, `111333555777` and `111` are all *random*, while `122221` (digit 2 is consecutively repeated 4 times) and `1234` (2 is a factor of 1234, which is not greater than **K**) are both not *random*.

Your task is to write a program to help Mr. Random to determine whether a number is *random* or not.

Input
The first line of input contains two positive integers, **N** and **K**.
The next **N** lines contain positive integers, one on each line.

Output
Output **N** lines. Each line should be `YES` if the corresponding number is a *random* number or `NO` if the number is not a *random* number.

Constraints

- Time Limit: 1s
- Memory Limit: 64MB
- $1 \leq \mathbf{N} \leq 10{,}000$
- $1 < \mathbf{K} < 2^{31}$
- The **N** positive integers are all less than 2^{32}. (Use *unsigned int* for storing the positive integers.)

Subtasks

Subtask 1 (10 points): The **N** positive integers are all ≤ 10,000

Subtask 2 (20 points): The **N** positive integers are all ≤ 100,000,000

Subtask 3 (70 points): The **N** positive integers are all $< 2^{32}$

Sample Input 1

```
5 5
3
10
7
7777
121
```

Sample Output 1

```
NO
NO
YES
NO
YES
```

In the example above, **K** = 5.

- 3 is not random because 3 is a factor that is not greater than **K**.
- 10 is not random because 2 and 5 are factors that are not greater than **K**.
- 7 is random because the factor 7 is greater than **K** and contains only 1 consecutive repeated digit.
- 7777 is not random because it contains 4 consecutive repeated digits.
- 121 is random because the factors 11 and 121 are both greater than **K** and contains only 1 consecutive repeated digit.

Binary Roads - proposed by Ranald Lam

Ali is out on a vacation and intends to visit a specific destination. However, he has trouble getting there in the shortest possible time and needs your help!

For simplicity, the place can be treated as a graph with **N** nodes (places) and **E** edges (roads). Nodes will be labelled from *0* to ***N****-1* and all edges are bi-directional. It can be assumed that it will take 1 hour to travel any edge and all nodes and edges can be visited more than once.

Ali will start of at node *0* and his destination is node ***N****-1.* This seems like an easy problem, but things are much more complicated due to binary roads.

Unlike normal roads, all of the **E** edges are binary roads. They either contain a value of *0* or *1*. Similarly, Ali will also have a value of *0* or *1*. When Ali has a value of *0*, he can only travel on edges with a value of *0*. When Ali has a value of *1*, he can only travel on those edges with a value of *1*. However, after travelling on a single edge, the value of Ali will be flipped. If it was originally *0*, it will become *1*. If it was originally *1*, it will become *0*. It will then flip again after he travels on another edge.

Still, Ali wants to know the shortest possible time to reach his destination (in hours). Remember, Ali can choose to start with value *0* or *1* .

Input

The first line of input will contain 2 integers, **N** and **E**.

The next **E** line of input will contain 3 integers: **A**, **B**, and **V**. This means that there is a bi-directional edge from node **A** to node **B** with a value of **V**. It is guaranteed that there will not be more than one edge connecting the *same* two nodes with the *same* value.

Output

Output a single integer, the shortest possible time for Ali to reach his destination (in hours). If it is not possible for Ali to reach his destination, output -1 instead.

Constraints

- Time Limit: 3s
- Memory Limit: 64MB
- $1 \leq \mathbf{N} \leq 200{,}000$
- $0 \leq \mathbf{E} \leq 1{,}000{,}000$
- It is guaranteed that $\mathbf{A} \neq \mathbf{B}$, $\mathbf{0} \leq \mathbf{A}, \mathbf{B} < \mathbf{N}$, $\mathbf{V} =$ *0* or *1* for all **E** lines describing the edges.

Subtasks

Subtask 1 (15 points): **N** ≤ 200,000, **E** ≤ 1,000,000. However, roads come in pairs. For every edge provided, it is guaranteed that there will be another edge with a different value connecting the same two nodes. In other words, the two nodes would have 2 edges with value 0 and 1 connecting them. There will be no two nodes that are connected via only a single edge only.

Subtask 2 (85 points): **N** ≤ 200,000, **E** ≤ 1,000,000. There are no restrictions and there can be 2 edges connecting the same nodes with different values. Furthermore, it is also possible that two nodes are connected via a single edge only (with value 0 or 1).

Sample Input 1

```
5 10
0 1 0
0 1 1
1 2 0
1 2 1
2 3 0
2 3 1
3 1 0
3 1 1
1 4 0
1 4 1
```

Sample Output 1

```
2
```

Sample input 1 follows the restrictions of Subtask 1. Notice that all edges come in pairs (0 1 0 and 0 1 1), (1 2 0 and 1 2 1), etc

The fastest way for Ali is to move from node 0 -> 1 -> 4. This will traverse 2 edges and take 2 hours.

Sample Input 2

```
5 5
0 1 1
1 2 0
3 1 0
3 2 1
1 4 1
```

Sample Output 2

```
5
```

Ali must either move in the order of node 0 -> 1 -> 2 -> 3 -> 1 -> 4 or 0 -> 1 -> 3 -> 2 -> 1 -> 4.

To do so, Ali must start out with a value of 1 at node 0. He will then travel along the edge (0 1 1) to node 1. After reaching node 1, he would have a value of 0. He then travels along the edge (1 2 0) as he is unable to travel on the edge (1 4 1) since the value of the road does not match his own value. After reaching node 2, he would have a value of 1 and he can travel along the edge (3 2 1) to reach node 3 with a value of 0. Subsequently, he can travel back to node 1 alone edge (3 1 0) and he would have reached node 1 with a value of 1. Lastly, he can travel the edge (1 4 1) to his destination, node 4.

Sample Input 3

```
3 2
0 1 1
0 1 0
```

Sample Output 3

```
-1
```

There is no way for Ali to reach node 2 from node 0.

Swaps - proposed by How Si Wei

You have two arrays A and B, each contains numbers 1, 2, ... , N, though not necessarily in this order.

In each operation, you can swap any two numbers in A.

Your task is to determine the least number of operations needed to transform A to B.

Input

- The first line of the input contains an integer **N**, representing the size of arrays A and B.
- The second line contains the elements of the array A, where consecutive elements are separated by a single space.
- The third line contains the elements of the array B, where consecutive elements are separated by a single space.

Output

Output the minimum number of operations needed to transform A to B.

Constraints

- Time Limit: 1s
- Memory Limit: 64MB
- $1 \leq \mathbf{N} \leq 1{,}000{,}000$

Subtasks

Subtask 1 (20 points): $\mathbf{N} \leq 1{,}000$

Subtask 2 (80 points): $\mathbf{N} \leq 1{,}000{,}000$

Sample Input 1

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

Sample Output 1

```
3
```

The 3 swap operations to transform [1,4,2,3] into [4,3,1,2] are:

Swap 1, 4
Swap 2, 3
Swap 1, 3

Sample Input 2

```
7
3 6 4 7 1 2 5
4 3 7 6 1 5 2
```

Sample Output 2

```
4
```

The 4 swap operations to transform [3,6,4,7,1,2,5] into [4,3,7,6,1,5,2] are:

Swap 2, 5
Swap 3, 6
Swap 4, 6
Swap 6, 7

Fish - proposed by Ranald Lam

Kuching the Cat enjoys eating fish. However, he accidently bought a large fish and does not want to eat all of it. To solve his problem, he has subdivided the fish into **N** linear segments (head to tail) and has given each segment a *'satisfaction rating'*. These segments are labelled 1 to **N**. The higher the *satisfaction rating*, the more Kuching will enjoy eating this segment of the fish. However, fish only taste nice if eaten as a chunk. One chunk consists of multiple linear segments of fish that are *contiguous*/*consecutive*.

Kuching is very picky and will not enjoy a chunk of fish where the sum of *satisfaction ratings* is *less* than **K** (i.e. he wants the sum to be **≥ K**). He wonders how many ways are there to cut a single chunk out of the fish such that he would enjoy eating. Those segments not part of the chunk will be thrown and not eaten

*To clarify: a chunk must contain *at least 1* linear segment of fish and can only contain up to **N** segments in total (i.e. the entire fish)

Input

The first line of input will contain two 32-bit signed integers, **N** and **K**. Note that **N** will always be positive while **K** can take both positive and non-positive integers.

The next line will be an array containing the *satisfaction rating* of the **N** segments. The i^{th} integer will be the *satisfaction rating* of the i^{th} segment of the fish. Do note that the *satisfaction ratings* will fit into a 32-bit signed integer and can take on positive and non-positive values.

Output

Output a single integer, the number ways there are to cut a single chunk out of the fish such that Kuching the Cat would enjoy eating (i.e. sum of *satisfaction ratings* **≥ K**).

Constraints

- Time Limit: 1s
- Memory Limit: 64MB
- 1 ≤ **N** ≤ 200,000

Subtasks

Subtask 1 (10 points): **N** ≤ 500

Subtask 2 (20 points): **N** ≤ 5,000

Subtask 3 (20 points): **N** ≤ 200,000. All the integers in the array are **≥ 0** (i.e. each segment's satisfactory rating are non-negative). **K** will also be non-negative.

Subtask 4 (50 points): **N** ≤ 200,000

Sample Input 1

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

Sample Output 1

```
6
```

Kuching the Cat has divided the fish into 5 segments. The first segment has *satisfaction rating* 1, 2nd has rating -2, 3rd has rating 3, 4th has rating -4 and the 5th has rating 5.

He will only enjoy chunks of fish which the sum of *satisfaction ratings* is more than or equal to 2.

Hence, the six chunks that he will enjoy are: [1 -2 3], [1 -2 3 -4 5], [-2 3 -4 5], [3], [3 -4 5], [5]

No other possible chunks will have a sum of *satisfaction ratings* **≥ K**.

Do note that [1 3 5] is not a valid chunk as they are not contiguous/consecutive.

Sample Input 2

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

Sample Output 2

```
13
```