AIPO 2023

March 11, 2023

All Ireland Programming Olympiad

University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Contributors

The AIPO organisers would like to thank the following people.

For leading the question writing:

- Dr Sabin Tabrica
- Andrew Nash
- Marcu Ştefan-Bogdan

Time limits and input/output bounds will become available when the submission system opens.

1 Banking Optimisation

You have been tasked with purchasing a new computer to perform N very important groups of banking transactions every day, one-by-one, in order.

Each day, there is an opening of X seconds, in which money can be transferred to other banks. Each transaction requires a different numbers of operations, given by O_i. Each transaction must be completed one after another in succession.

Find the minimum number of operations per second this computer must be able to perform, to guarantee that the given group of transactions are all completed within X seconds (including exact time X). Assume computers are only available with whole valued ops/second values, so this value must be an integer.

If it is impossible to complete all operations in time, print -1.

Input

- Two integers, N and X
- A single line containing N integers corresponding to O

Output

A single integer number, the minimum operations per second.

Scoring

Your code is tested on many different testcases, grouped into subtasks. To obtain the points for a subtask it is necessary to solve all of the testcases within this subtask.

- **Subtask 1 [1 point]**: All $O[i] = 1$, and $N < X$.
- **Subtask 2 [24 points]**: $N = 100$
- **Subtask 4 [75 points]**: No other specific constraints

Constraints

- $1 \leq X \leq 10^6$
- $1 \leq O[i] \leq 10^5$
- $1 \leq N \leq 10^6$

Sample input

```
3 5
2 3 2
```

Sample Output

```
2
```

Explanation of sample input/output

The first job takes 2/2=1 seconds The second job takes 3/2=1.5 seconds The second job takes 2/2=1 seconds

Total= 1+1.5+1 = 3.5, which is under 5 seconds.

Whereas for 1 op/sec we have 2/1+3/1+2/1 = 7 which is over 5 seconds , and therefore too slow.

2 Similar Pallindromes

You may, or may not have previously encountered the ideas of pallindromes and anagrams.

A string, say s_1 is an anagram of another string s_2, if its letters can be rearranged so that both are identical. E.g., "act" and "cat" are anagrams of one another.

A pallindrome is a string that is spelled the same when read forwards and backwards - e.g. "racecar".

Here, we will also define a *similar* pallidrome. There are a certain set of letters that we define to look *similar* - these are the pairs p and q, b and d, and a and e.

A string is a *similar palindrome* if it is spelled the same forwards and backwards, allowing matches between the above letter pairings. E.g, "aipoqoia", "aipopie", "pq", "qp", "aipoqoie" and "bdbd" are all *similar* pallindromes.

Given some strings, your task is to determine if each is an anagram of either a regular, or a "similar" pallindrome - i.e, can you rearrange the letters to make either a regular, or a "similar" pallindrome.

Input

- One integer T, the number of strings to test.
- Then, T times, we have:
  - A single string on a single line S_i consisting of N_i lowercase alphabetic characters

Output

On T separate lines, one of the following strings:

- Regular
- Similar
- Neither

Depending on whether each of the T strings was an anagram of a regular, or similar pallindrome, or neither.

Scoring

Your code is tested on many different testcases, grouped into subtasks. To obtain the points for a subtask it is necessary to solve all of the testcases within this subtask.

- **Subtask 1 [5 points]**: $N_i \leq 10$.
- **Subtask 2 [15 points]**: Each alphabetic letter features at most once in each S_i.
- **Subtask 3 [25 points]**: Each alphabetic letter in S_i appears at most twice in each S_i.
- **Subtask 4 [55 points]**: No other specific constraints

Constraints

- $1 \leq T \leq 20$.
- $1 \leq N \leq 10^5$.

Example

Sample Input 1

```
4
aipo
aipoqia
eipoqia
aipia
```

Sample Output 1

```
Neither
Similar
Similar
Regular
```

3 Smooth Numbers

A number n is called to be k-smooth if all its prime factors are less or equal with k. This smoothness property means that a number has only small prime divisors (or it does not have large primes as divisor). Smooth numbers are very important in few practical and theoretical applications including here cryptography.

For example, all 2-smooth numbers have only 2 as divisor hence they are powers of 2. While all 3-smooth numbers are all of the form $2^n \cdot 3^m$ for some n, m positive integers.

By convention 1 is k-smooth.

Find how many k-smooth numbers are less or equal to n.

Input

Two integer numbers k, n

Output

One integer representing the number of k-smooth numbers $\leq n$

Constraints

1. $2 \leq n \leq 10000000$
2. $2 \leq k \leq 1000$
3. $k < n$

Sample Input	Sample Output	Explanation
`2 100`	`7`	There are 7 2-smooth numbers smaller or equal to 100: 1, 2, 4, 8, 16, 32, 64

Sample Input 2	Sample Output 2	Explanation
`5 100`	`34`	There are 34 numbers with 2,3 and 5 as prime divisors that are smaller or equal to 100: 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16, 18, 20, 24, 25, 27, 30, 32, 36, 40, 45, 48, 50, 54, 60, 64, 72, 75, 80, 81, 90, 96, 100

4 Panda Reserve

A panda reserve, viewed from above, is rectangular in shape and consists of n identical rows, and in each row there are m identical pens with a square base. The pens are fenced and have doors to all 4 neighbouring pens. The doors are equipped with an access code. They close and open automatically. Through this system, some pens are accessible to the bears, and others are forbidden to them. P pens contain food for the bears.

The bears in the reserve carry a microchip that automatically opens the doors of the pens they can enter and automatically closes the doors of the forbidden pens. A pen is accessible to the bear if the last S digits of the binary representations of the pen code and the k-code on the microchip are complementary. (Example: for S=8, 11101011 and 00010100 are complementary).

In a pen is a teddy bear who has become hungry. The teddy bear can only move parallel to the sides of the rectangle. The transition from one pen to a pen neighbouring with him is done in a second.

Knowing n and m the dimensions of the reservation, the access codes from each of the $n \cdot m$ pens, the coordinates of the P food pens, the coordinates of the pen L and C where the bear is initially located, the k code of his microchip and the number S, determine:

1. The number X of pens that satisfy the property that the last S digits of the binary representation of their code are complementary to the last S digits of the binary representation of the k code carried by the bear, except for the pen in which it is initially located.
2. Minimum number of seconds $Smin$ in which he can reach a food pen as well as the number of food pens np he can reach in this minimum time

Input

On the first line an integer T. This can only take the values 1 or 2 indicating which sub-task should be applied

On the second line 3 integers n, m (representing the size of the reservation) and P (the number of pens containing food).

On the third line 4 integers L, C (the initial coordinates of the bear), k (the code of the bears microchip) and S (the number of significant bytes)

On the next P lines 2 integers denoting the coordinates of the food pens

On the next n lines m integers, separated by spaces, representing the access codes for the doors of the $n \cdot m$ pens of the reserve.

Output

Depending on the value T in the input.

1. if T is 1: only sub-task 1 will be required. In this case the output will be a single integer denoting the total number of pens that the bear can access with the exception of the initial pen.
2. if T is 2: only sub-task 2 will be required. In this case the output will consist of two integers, the minimum number of seconds required for the bear to reach a food pen and the number of food pens the bear can reach within this minimum time. These two integers should be separated by a space

Constraints

- $2 \leq n, m \leq 500$
- $1 \leq S \leq 8$
- $1 \leq P \leq n \cdot m$
- $0 \leq k, codevalues \leq 9999$
- For all tests of the problem there is a solution, that is, the teddy bear can reach at least one of the food pens
- Food can also be found in inaccessible areas

Sample Input

```
1
5 6 4
3 5 1 1
1 2
5 1
2 1
4 3
15 1278 3 1278 1278 1
16 17 18 19 254 20
21 25 26 254 254 254
27 28 29 3 2 254
2 254 4 254 254 254
```

Sample Output

```
19
```

Explanation

T is 1 so **only the first subtask is required**. $k = 1$ and since $s = 1$ only the last binary digit of k needs to be different from the last binary digit of the code in the enclosure.

Sample Input 2

```
2
5 6 4
3 5 1 1
1 2
5 1
2 1
4 3
15 1278 3 1278 1278 1
16 17 18 19 254 20
21 25 26 254 254 254
27 28 29 3 2 254
2 254 4 254 254 254
```

Sample Output 2

```
6 1
```

Explanation

T is 2 so **only the second subtask is required**. If we mark the accessible pens with 1 and the inaccessible ones with 0, we get the following matrix:

```
0 1 0 1 1 0
1 0 1 0 1 1
0 0 1 1 1 1
0 1 0 0 1 1
1 1 1 1 1 1
```

The bear is in the pen at coordinates $(3, 5)$ and can reach only one food pen after 6 seconds. This pen is the one at coordinates $(5, 1)$; the road travelled is; $(3, 5) \rightarrow (4, 5) \rightarrow (5, 5) \rightarrow (5, 4) \rightarrow (5, 3) \rightarrow (5, 2) \rightarrow (5, 1)$

5 Dragons

In a world where vikings can fly with dragons there are N islands. Hiccup, the chief of the Viking tribe on island 1, knows M direct **two-way** flight routes between the islands. For each j between 1 and M, route j joins islands A_j and B_j and has length D_j.

On each island $i, (1 \leq i \leq n)$ there are dragons of species i that can fly without stopping for a maximum distance $Dmaxi$. In other words, the dragons on island i will be able to travel any route $j, (1 \leq j \leq m)$ for which $Dj \leq Dmaxi$, regardless of what other routes they have taken previously. Hiccup wants to get from Island 1 to Island N to save Toothless, his dragon. To get there, he will initially take a species 1 dragon (from island 1). Then, if at some point Hiccup is on an island $i, (1 \leq i \leq n)$ having a dragon of species t with him, he can:

- Fly from island i to another island x with the dragon he has, using a direct route j between islands i and x, of course only if $Dj \leq Dmaxt$
- Exchange the t-species dragon he has for a i-species dragon from that island.

Your task is to:

1. Determine the maximum distance $Dmaxi$ characteristic of a dragon that Hiccup can reach without changing the dragon he originally took from island 1.
2. Determine the minimum distance Hiccup must travel to get from island 1 to island N.

Input

On the first line an integer T. This can only take the values 1 or 2 indicating which sub-task should be applied

On the second line 2 integers n (the number of islands) and m (the number of direct routes between the islands)

On the third line there are N natural numbers, the i-th of which represents the maximum distance $Dmaxi$ that a dragon from island i can fly

On the next M lines there are 3 integers A, B, D describing the direct routes. With the interpretation that there is a **bidirectional** route between islands A and B with a D distance

Output

Depending on the value T in the input.

1. if T is 1: only sub-task 1 will be required. In this case the output will be a single integer representing the maximum distance $Dmaxi$ of a dragon i that Hiccup can reach without changing the dragon he originally took from island 1.
2. if T is 2: only sub-task 2 will be required. In this case the output will consist of a single integer representing the minimum distance that Hiccup must travel to get from island 1 to island N.

Constraints

- $1 \leq N \leq 800$
- $1 \leq M \leq 6000$
- $1 \leq Dmax_i \leq 50000$ for any $1 \leq i \leq N$
- $1 \leq A_j, B_j \leq N$ for any $1 \leq j \leq M$
- $1 \leq D_j \leq 50000$ for any $1 \leq j \leq M$
- there always is a solution for Hiccup to reach island N
- the answer to any of the task will always be an integer smaller than 10^9

Sample Input

```
1
5 6
6 3 13 20 26
1 2 5
1 3 7
1 5 10
2 3 6
3 4 5
3 5 14
```

Sample Output

```
20
```

Explanation

T is 1 so **only the first subtask is required**. There are $N = 5$ islands and $M = 6$ routes between them. Hiccup starts from island 1 with a dragon that can fly a maximum distance of 6. With this dragon he can only reach islands 1, 2, 3 and 4, since to reach island 5 he would have to travel a route of length greater than 6.

The maximum distance that a dragon on islands 1, 2, 3 or 4 can fly is therefore 20 (the dragon on island 4). It is noted that the dragon that can fly a distance of 26 is on island 5 and is inaccessible.

Sample Input 2

```
2
5 6
6 3 13 20 26
1 2 5
1 3 7
1 5 10
2 3 6
3 4 5
3 5 14
```

Sample Output 2

```
28
```

Explanation

T is 2 so **only the second subtask is required**. There are $N = 5$ islands and $M = 6$ routes between them. To travel a minimum distance of 28 between islands 1 and N, Hiccup takes the following steps:

Fly from island 1 to island 2 a distance of 5 with the species 1 dragon. Fly from island 2 to island 3 a distance of 6 with the species 1 dragon. Swap the species 1 dragon for the dragon on island 3, which can fly a maximum distance of 13. Fly from island 3 to island 1 a distance of 7 with the species 3 dragon. Fly from island 1 to island 5 a distance of 10 with the species 3 dragon.

In total he travels a distance of $5 + 6 + 7 + 10 = 28$.

6 Factory Rearrangement

You are working in a factory and are given an $N \times M$ grid of boxes, each of which have a different weight w_{ij}. The only operation you can perform - swapping two adjacent boxes. Boxes can only be swapped if they are adjacent in the same column **or** row, but not both. I.e. we do not allow diagonally adjacent boxes. Since you are the only person in the factory, you want to lift as little weight as possible during the day. Every time you swap two boxes with weights w_{ab} and w_{cd}, we say you have lifted total weight $w_{ab} + w_{cd}$ - the sum of the weights of the boxes you had to lift.

Your task is to figure out how to rearrange the boxes to be in a given target configuration with the minimal total weight lifted.

It is guaranteed that the weights of all the different boxes are unique.

You do not need to return the sequence of operations, just the total weight lifted.

Input

1. A single line, containing integers N and M
2. N lines follow, each containing M space separated integers, corresponding to the weights of boxes in their initial configuration in the factory
3. N more lines follow, each containing M space separated integers, corresponding to the desired configuration of boxes in the factory

Output

A single integer W corresponding to the total weight moved from all swaps to reach the desired configuration. You do **not** need to print the sequence of steps.

Scoring

Your code is tested on many different testcases, grouped into subtasks. To obtain the points for a subtask it is necessary to solve all of the testcases within this subtask.

- **Subtask 1** **[5 points]**: $N = 2, M = 1$.
- **Subtask 2** **[20 points]**: $M = 1, N \leq 3$.
- **Subtask 3** **[35 points]**: $M = 2, N = 2$.
- **Subtask 4** **[40 points]**: No other specific constraints

Constraints

- $1 \leq N \leq 4$.
- $1 \leq M \leq 4$.
- $N \times M < 10$
- $1 \leq S_{ij}, F_{ij} \leq 500$

Example

Sample Input 1

```
2 3
3 19 9
7 14 1
14 7 1
9 19 3
```

Sample Output 1

```
100
```

Explanation of Sample Output 1

The following is an example of a sequence of swaps that achieves the optimal result (100).

Starting from arrangement:

```
3 19 9
7 14 1
```

1. Boxes (7,3) with weight 10

   ```
   7 19 9
   3 14 1
   ```

2. Boxes (14,3) with weight 17

   ```
   7 19 9
   14 3 1
   ```

3. Boxes (1,3) with weight 4

   ```
   7 19 9
   14 1 3
   ```

4. Boxes (19,1) with weight 20

   ```
   7 1 9
   14 19 3
   ```

5. Boxes (9,1) with weight 10

   ```
   7 9 1
   14 19 3
   ```

6. Boxes (9,7) with weight 16

   ```
   9 7 1
   14 19 3
   ```

7. Boxes (9,14) with weight 23

   ```
   14 7 1
   9 19 3
   ```

This achieves the correct arrangements, lifting a total of $10 + 17 + 4 + 20 + 10 + 16 + 23 = 100$ weight