



ICPC International Collegiate Programming Contest // 2026-2027

# The 2026 ICPC Asia Seoul Nationalwide Internet Competition



## Practice Problem Set

Please check that you have 3 problems that are spanned across 9 pages in total (including Korean translation and this cover page).

|                         |               |                              |
|-------------------------|---------------|------------------------------|
| A. Best Partition       | (1 + 2 pages) | Korean translation available |
| B. Inverse Look and Say | (1 + 2 pages) | Korean translation available |
| C. Rescue Squad         | (2 pages)     |                              |

## Problem A

### Best Partition

Time Limit: 1 Second

You are given two sequences,  $A$  and  $B$  which are permutations of integers between 1 and  $n$  (inclusive). You want to divide each of them into  $k$  partitions by picking  $k - 1$  positions, both from  $A$  and  $B$ . There is one rule: for each  $i = 1, 2, \dots, k$ , the position of the minimum element within the  $i$ -th partition of  $A$  must be equal to the position of the minimum element within the  $i$ -th partition of  $B$ . For example, assume that  $A = (1, 3, 2, 5, 4)$  and  $B = (5, 4, 3, 2, 1)$ . Then you can divide  $A$  into  $(1), (3, 2), (5, 4)$ , which means that you divide  $B$  into  $(5), (4, 3), (2, 1)$ . It meets the condition mentioned above. Of course, you can divide  $A$  and  $B$  into  $n$  partitions whose length is one. We want to find the partition with the smallest  $k$  satisfying the condition mentioned above and will call it *the best partition*. You can see that the example above is a best partition.

Given  $n$ ,  $A$ , and  $B$ , write a program to find the best partition and output  $k$ .

### Input

Your program is to read from standard input. The input starts with a line containing an integer  $n$  ( $1 \leq n \leq 3,000$ ) where  $n$  is the length of  $A$  and  $B$ . The second line contains  $n$  integers between 1 and  $n$  (inclusive) which describes  $A$ . The third line also contains  $n$  integers between 1 and  $n$  (inclusive) which describes  $B$ . Note that in the second and third lines, no integer appears more than once in the same line.

### Output

Your program is to write to standard output. Print exactly one line. The line should contain the number  $k$  of partitions in the best partition.

The following shows sample input and output for three test cases.

| Sample Input 1              | Output for the Sample Input 1 |
|-----------------------------|-------------------------------|
| 5<br>1 3 2 5 4<br>5 4 3 2 1 | 3                             |
| Sample Input 2              | Output for the Sample Input 2 |
| 5<br>1 2 3 4 5<br>1 5 4 3 2 | 1                             |
| Sample Input 3              | Output for the Sample Input 3 |
| 5<br>1 2 3 4 5<br>5 4 3 2 1 | 5                             |

# Problem A

## 최적의 분할

제한 시간: 1 초

1 이상  $n$  이하 정수들로 이루어진 길이  $n$ 인 순열  $A$ 와  $B$ 가 주어진다.  $A$ 와  $B$ 에서 동일한  $k - 1$ 개의 위치를 골라서 각각  $k$ 개의 조각으로 나누려고 한다. 단, 각  $i = 1, 2, \dots, k$ 에 대해,  $A$ 의  $i$ 번째 조각의 최솟값의 위치와  $B$ 의  $i$ 번째 조각의 최솟값의 위치가 서로 같아야 한다. 예를 들어서,  $A = (1, 3, 2, 5, 4)$  이고  $B = (5, 4, 3, 2, 1)$  라 하자. 만약,  $A$  를  $(1), (3, 2), (5, 4)$  로 나누면,  $B$  는  $(5), (4, 3), (2, 1)$ 로 나누어지며, 위에서 설명한 조건을 만족시킨다. 물론  $A$ 와  $B$ 를 길이가 1인  $n$ 개의 조각들로 나누면 위 조건을 쉽게 만족시킬 수 있다. 따라서 우리는 조건을 만족하면서 조각의 수  $k$ 가 최소가 되도록 나누고자 하며, 이러한 분할을 **최적의 분할**이라고 하자. 위 예시에서  $k = 3$ 인 분할이 최적의 분할이다.

$n$ ,  $A$ ,  $B$ 가 주어질 때, 최적의 분할을 찾고, 그 때의  $k$ 를 출력하는 프로그램을 작성하시오.

### Input

입력은 표준입력을 사용한다. 첫 줄에  $A$ 와  $B$ 의 길이를 나타내는 양의 정수  $n$  ( $1 \leq n \leq 3,000$ )이 주어진다. 두 번째 줄에  $A$ 에 대한 정보가 주어지며, 1 이상  $n$  이하인  $n$ 개의 정수들이 주어진다. 세 번째 줄에  $B$ 에 대한 정보가 주어지며, 1 이상  $n$  이하인  $n$ 개의 정수들이 주어진다. 두 번째 줄과 세 번째 줄에서, 같은 줄에는 같은 정수가 두 번 이상 주어지지 않는다.

### Output

출력은 표준출력을 사용한다. 첫 줄에 최적의 분할의 조각 수를 출력한다.

다음은 세 테스트 경우에 대한 입출력 예이다.

| Sample Input 1              | Output for the Sample Input 1 |
|-----------------------------|-------------------------------|
| 5<br>1 3 2 5 4<br>5 4 3 2 1 | 3                             |
| Sample Input 2              | Output for the Sample Input 2 |
| 5<br>1 2 3 4 5<br>1 5 4 3 2 | 1                             |
| Sample Input 3              | Output for the Sample Input 3 |
| 5<br>1 2 3 4 5<br>5 4 3 2 1 | 5                             |

## Problem B

### Inverse Look-and-Say

Time Limit: 1 Second

Consider the following sequence:  $2 \rightarrow 12 \rightarrow 1112 \rightarrow 3112 \rightarrow 132112 \rightarrow 1113122112 \rightarrow \dots$

The initial value of this sequence is 2, and subsequent values are generated as follows:

- 2 contains “one 2,” so the next term is 12.
- 12 contains “one 1 and one 2,” so the next term is 1112.
- 1112 contains “three 1's and one 2,” and the next term is 3112.
- 3112 contains “one 3, two 1's, and one 2,” the next term is 132112.

This sequence follows the “look-and-say” rule. Let  $f(x)$  denote the number obtained by applying the rule of the sequence to a positive integer  $x > 0$ . Then,  $f(2) = 12$ ,  $f(12) = 1112$ ,  $f(1112) = 3112$ ,  $f(3112) = 132112$ , and so on.

The function  $f(x)$  is defined only when no digit in the positive integer  $x$  appears consecutively more than 9 times. By applying the “look-and-say” rule, we can obtain a number  $n$  such that  $f(x) = n$ . Your task is, given a positive integer  $n$  as input, to find an  $x$  such that  $f(x) = n$ . If  $x$  exists, it is unique. Note that for some positive integers  $n$ , no such  $x$  exists. For example, there is no positive integer  $x$  such that  $f(x) = 311$ . Likewise, there is no positive integer  $x$  such that  $f(x) = 1111$ . In the case of 1111, one might interpret it as “one 1, one 1” and think  $x = 11$ , but  $f(11) = 21$ . Thus, no positive integer can produce 1111 according to the “look-and-say” rule.

### Input

Your program is to read from standard input. The input has a line containing one positive integer  $n$  less than  $10^{1,000}$ .

### Output

Your program is to write to standard output. Print exactly one line with one integer  $x$  such that  $f(x) = n$ . If such  $x$  does not exist, print  $-1$ .

The following shows sample input and output for three test cases.

| Sample Input 1 | Output for the Sample Input 1 |
|----------------|-------------------------------|
| 132112         | 3112                          |
| Sample Input 2 | Output for the Sample Input 2 |
| 331            | -1                            |
| Sample Input 3 | Output for the Sample Input 3 |
| 1111           | -1                            |

## Problem B

### Inverse Look-and-Say

제한 시간: 1 초

다음과 같은 수열을 생각해보자:  $2 \rightarrow 12 \rightarrow 1112 \rightarrow 3112 \rightarrow 132112 \rightarrow 1113122112 \rightarrow \dots$ .

이 수열의 초기값은 2이고, 이후의 수들은 다음과 같이 생성된다.

- 2에는 "1개의 2"가 있으므로, 다음 값은 12이다.
- 12에는 "1개의 1, 1개의 2"가 있으므로 다음 값은 1112이다.
- 1112에는 "3개의 1, 1개의 2"가 있으므로 다음 값은 3112이다.
- 3112에는 "1개의 3, 2개의 1, 1개의 2"가 있으므로, 다음 값은 132112이다.

이 수열은 이러한 "look-and-say" 규칙으로 생성된다. 이 수열에서 다음 항을 계산하는 규칙을 양의 정수  $x > 0$ 에 적용해서 나오는 수를  $f(x)$ 라고 하자. 즉,  $f(2) = 12$ ,  $f(12) = 1112$ ,  $f(1112) = 3112$ ,  $f(3112) = 132112$ 와 같이 주어지는 양의 정수에 대해 이를 십진법으로 읽었을 때 연속되는 숫자의 개수를 보이는 대로 읽어서 만들어지는 수를 의미한다.

양의 정수  $x$ 에 연속해서 나타나는 숫자가 9회를 넘지 않을 때에만  $f(x)$ 가 정의되며, "look-and-say" 규칙을 적용하여  $f(x) = n$ 인  $n$ 을 얻을 수 있다. 여러분이 할 일은 입력으로 양의 정수  $n$ 이 주어질 때,  $f(x) = n$ 인 양의 정수  $x$ 를 구하는 것이다.  $x$ 가 존재한다면 그 값은 유일하다. 주의할 점은 어떤 양의 정수  $n$ 은 그러한  $x$ 를 가지지 않는다는 것이다. 예를 들어,  $f(x) = 311$ 인 양의 정수  $x$ 는 존재하지 않는다. 또한  $f(x) = 1111$ 인 양의 정수  $x$  역시 존재하지 않는다. 1111의 경우 "1개의 1, 1개의 1"로 해석하여  $x$ 가 11이라 생각할 수 있으나,  $f(11) = 21$ 이다. 어떤 양의 정수도 이 "look-and-say" 규칙에 따라 1111을 생성할 수 없다.

#### Input

입력으로 첫 줄에  $10^{1,000}$ 보다 작은 양의 정수  $n$ 이 주어진다.

#### Output

$f(x) = n$  인 양의 정수  $x$ 가 존재한다면 그 값을 출력하고, 그렇지 않다면  $-1$ 을 출력한다.

다음은 세 테스트 경우에 대한 입출력 예이다.

| Sample Input 1 | Output for the Sample Input 1 |
|----------------|-------------------------------|
| 132112         | 3112                          |
| Sample Input 2 | Output for the Sample Input 2 |
| 331            | -1                            |
| Sample Input 3 | Output for the Sample Input 3 |
| 1111           | -1                            |

# Problem C

## Rescue Squad

Time Limit: 1 Second

As a town leader, you must form a rescue squad to help a neighboring town under attack by monsters. There are  $n$  knights in your town, numbered from 1 to  $n$ , and a knight  $i$  ( $1 \leq i \leq n$ ) has a positive integer level  $L_i$ . For efficient monster hunting, the knights who form the rescue squad must be able to trust one another. The trust relationship  $T$  between two knights  $i$  and  $j$  ( $1 \leq i < j \leq n$ ) is defined as follows:

$$\begin{aligned} T(i, j) &= 1, \text{ if } i \text{ and } j \text{ trust each other;} \\ T(i, j) &= 0, \text{ if they do not.} \end{aligned}$$

The rescue squad  $S$  is a set of four distinct knights, and each knight in the squad must have a trust relationship  $T = 1$  with at least two of the other three. The ability of  $S$ ,  $\text{Ability}(S)$ , is defined as the sum of the levels of the four knights in  $S$ .

For example, suppose that  $n = 5$  and the levels of the five knights are  $L_1 = 3, L_2 = 2, L_3 = 3, L_4 = 7$ , and  $L_5 = 1$ . If the trust relationship is defined as  $T(1, 2) = T(2, 3) = T(3, 4) = T(1, 3) = T(1, 4) = T(1, 5) = 1$  and  $T(2, 4) = T(2, 5) = T(3, 5) = T(4, 5) = 0$ , then  $S = \{1, 2, 3, 4\}$  is the only possible rescue squad, and  $\text{Ability}(S)$  is 15.

Given the levels of  $n$  knights and their trust relationships, write a program to find a squad  $S$  for which  $\text{Ability}(S)$  is maximized and output its ability value.

### Input

Your program is to read from standard input. The input starts with a line containing two integers,  $n$  and  $m$  ( $4 \leq n \leq 1,000$ ;  $1 \leq m \leq \min(\frac{n(n-1)}{2}, 25,000)$ ), where  $n$  is the number of knights and  $m$  is the number of pairs of knights  $(i, j)$  such that  $T(i, j) = 1$  and  $i < j$ . In the following  $n$  lines, the  $i$ -th line contains a positive integer that represents the level  $L_i$  ( $1 \leq L_i \leq 10,000$ ) of the knight  $i$ . In the following  $m$  lines, each line contains two integers,  $i$  and  $j$  ( $1 \leq i < j \leq n$ ) that represent a trust relationship  $T(i, j) = 1$ . There are no duplicate entries among the  $m$  lines describing the trust relationships, and for any pair of knights  $i'$  and  $j'$  that do not appear in the input,  $T(i', j') = 0$ .

### Output

Your program is to write to standard output. Print exactly one line. The line should contain the ability  $\text{Ability}(S)$  of a squad  $S$  with the maximum ability among the squads that can be formed. If no squad can be formed, print  $-1$ .

The following shows sample input and output for two test cases.

| Sample Input 1                                                         | Output for the Sample Input 1 |
|------------------------------------------------------------------------|-------------------------------|
| 5 6<br>3<br>2<br>3<br>7<br>1<br>1 2<br>2 3<br>3 4<br>1 3<br>1 4<br>1 5 | 15                            |
| Sample Input 2                                                         | Output for the Sample Input 2 |
| 5 5<br>1<br>2<br>3<br>4<br>5<br>1 2<br>2 3<br>3 4<br>4 5<br>1 5        | -1                            |