Problem Set

Please check that you have 11 problems that are spanned across 27 pages in total (including Korean translation and this cover page).

A. Balance Scale	(1 + 2 pages)	Korean translation available
B. Bar Magnet	(2 pages)	
C. Container Rearrangement	(2+2 pages)	Korean translation available
D. Flags	(2 pages)	
E. Gift Discount	(1+1 pages)	Korean translation available
F. Islands Tour	(2 pages)	
G. Jar Game	(1 page)	
H. Longest Shortest Paths	(2 pages)	
I. Magic Potion	(2 pages)	
J. Rectangles	(2 pages)	
K. Temporal Graph	(2+2 pages)	Korean translation available

Problem A
Balance Scale
Time Limit: 1 Second

There are N pebbles, numbered from 1 to n. There is a balance scale. We will put these pebbles on the scale according to the following rules.

1. Pebble 1 is put on the left pan and Pebble 2 is put on the right pan.
2. For Pebble $i = 3, \ldots, n$, we do either A or B.
   A. If the scale is in equilibrium, Pebble i is put on the left pan.
   B. Otherwise, Pebble i is put on the lighter side.

After all the pebbles are put on the scale, the balance scale may not be in equilibrium. We will use additional masses for putting the scale in equilibrium. There are seven kinds of masses: 1g, 2g, 5g, 10g, 20g, 50g, and 100g. There is no limit to the number of masses of each kind.

Given the information on pebbles, write a program to output the smallest number of additional masses to put the scale in equilibrium in the end.

Input
Your program is to read from standard input. The input starts with a line containing an integer n ($2 \leq n \leq 10{,}000$), where n is the number of pebbles. The next line contains n integers where the i-th integer represents the weight of Pebble i. Each pebble weighs at least one and the sum of the weights of the pebbles is equal to or smaller than 10,000,000.

Output
Your program is to write to standard output. Print exactly one line. The line should contain the smallest number of additional masses to put the scale in equilibrium after the pebbles are put on the scale as described.

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

Sample Input 1	Output for the Sample Input 1
7 3 1 4 1 5 9 2	2

Sample Input 2	Output for the Sample Input 2
4 2 4 6 4	0

Sample Input 3	Output for the Sample Input 3
5 2 5 3 1 2	1

Problem A

양팔저울

Balance Scale

제한 시간: 1 초

1 부터 n까지 번호가 매겨진 n개의 자갈이 있다. 이 자갈들을 다음 절차에 따라 양팔저울에 올려놓는다.

1. 1 번 자갈을 왼쪽, 2 번 자갈을 오른쪽에 올려놓는다.
2. $i = 3, \ldots, n$번 자갈 각각에 대해서 차례로 다음 과정 중 하나를 수행한다.
   A. 만약 양팔저울이 평형을 이루는 경우, i번 자갈을 왼쪽에 올려 놓는다.
   B. 만약 양팔저울이 평형을 이루지 않는 경우, i번 자갈을 가벼운 쪽에 올려 놓는다.

모든 자갈을 위의 규칙에 따라 올려 놓은 후에도 양팔저울은 평형을 이루지 않을 수 있다. 이 경우 가벼운 쪽에 무게추를 올려서 균형을 맞추려고 한다. 무게추는 1g, 2g, 5g, 10g, 20g, 50g, 100g 7 종류가 있고, 무게추의 개수에는 제한이 없다.

입력 받은 자갈을 위 규칙에 따라 양팔저울에 올렸을 때, 최종적으로 평형을 맞추는데 추가적으로 필요한 무게추의 최소 개수를 구하는 프로그램을 작성하시오.

Input

입력은 표준입력을 사용한다. 첫 번째 줄에 자갈 개수를 나타내는 양의 정수 n $(2 \leq n \leq 10{,}000)$이 주어진다. 다음 줄에 n 개의 수들이 주어지는데, 이들은 번호 순서대로 자갈의 무게이다. 자갈의 무게는 각각 1 이상이며, 모든 자갈의 무게의 총합은 10,000,000 이하이다.

Output

출력은 표준출력을 사용한다. 최종적으로 평형을 맞추는데 추가적으로 필요한 무게추의 최소 개수를 한 줄에 출력한다.

다음은 세 테스트 케이스에 대한 입출력 예이다.

Sample Input 1	**Output for the Sample Input 1**
7 3 1 4 1 5 9 2	2

Sample Input 2	**Output for the Sample Input 2**
4 2 4 6 4	0

Sample Input 3	**Output for the Sample Input 3**
5 2 5 3 1 2	1

Problem B
Bar Magnet
Time Limit: 4 Seconds

Jaehyun makes and sells bar magnets of various lengths and colors. He makes a lot of cube-shaped "basic magnets" of the same size. The basic magnets are colored in one of the 26 colors. See Figure 1. Jaehyun makes various bar magnets according to the customers' request by connecting basic magnets. But connecting basic magnets one by one takes quite a long time. So Jaehyun tried to save time by using a "template bar magnet" consisting of several basic magnets instead of the basic magnets. To simplify the process, Jaehyun decided to use only one type of template bar magnet and make it in bulk in advance. Now he uses the pre-made template bar magnets to produce bar magnets requested by customers. Note that the template bar magnets are polarized and cannot be connected in the opposite direction.

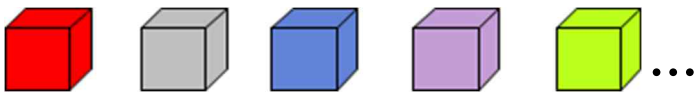


Figure 1 Basic Magnets

There are some rules when Jaehyun uses the template bar magnets to produce requested bar magnets. For convenience, we call the template bar magnet the T-bar.

1. He constructs requested bar magnets from the N pole side to the S pole side, that is, from left to right.
2. Jaehyun spends at most m time for each T-bar where m is the length of T-bar.
3. He checks if the basic magnet of the T-bar matches that of the requested bar magnet one by one from left to right.
4. Comparing each basic magnet takes 0 time.
5. Jaehyun can remove one basic magnet from the T-bar, or he can insert one basic magnet into the T-bar, or he can replace one basic magnet in the T-bar. Each operation (removal, insertion, replacement) takes 1 time. He can repeatedly perform these operations to make them match.
6. If one or more of the last basic magnets of requested bar magnet constructed so far match the front part of the next T-bar, Jaehyun can remove the matched part from the next T-bar. This takes 0 time. So, when Jaehyun tries to use the next T-bar, if he spends some time to remove, or insert, or replace some basic magnets to make the front part of the next T-bar match the last part of requested bar magnet constructed so far, it may reduce the total construction time.
7. When the last basic magnet of the current T-bar is compared and used, another T-bar can be used until the requested bar magnet is made because there are sufficiently many basic magnets and T-bars.

Jaehyun wants to minimize the total time to produce the requested bar magnet using T-bars. Note that it does not matter how many T-bars Jaehyun uses.

For example, assume that the T-bar consists of 6 basic magnets as Figure 2. Assume a customer requests a bar magnet consisting of 17 basic magnets as Figure 3.

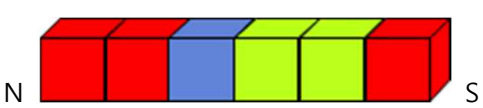


Figure 2 A T-bar consisting of 6 basic magnets

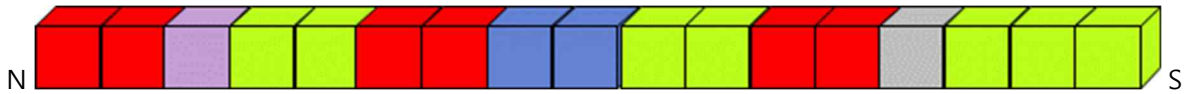


Figure 3 A requested bar magnet by a customer

In this example, the requested bar magnet can be produced using three T-bars. Consider the first T-bar. If the third (blue) basic magnet of the T-bar is replaced with a purple one, since the other basic magnets all match, the length 6 front part of the requested bar magnet can be constructed with 1 time. Next consider the second T-bar. Since the last (red) basic magnet of the bar magnet constructed so far matches the first basic magnet of the T-bar, it can be removed in 0 time. Also, if one blue basic magnet is inserted in the 9th position, the length 12 front part of the requested bar magnet can be constructed with additional 1 time. Now consider the third T-bar. Since the last (red) basic magnet of the bar magnet constructed so far matches the first basic magnet of the T-bar again, it also can be removed in 0 time. And if the 3rd (blue) and the 6th (red) basic magnets of the T-bar are replaced with grey and green one, respectively, the requested bar magnet is constructed with additional 2 time. The total time is 1+1+2 = 4, which is the minimum time to construct the requested bar magnet using the T-bars.

Given a T-bar of length m and a requested bar magnet of length n, write a program to output the minimum time to construct the requested bar magnet using the T-bars. Each of the 26 colors of basic magnets is represented by a different uppercase alphabet. And bar magnets will be given as strings consisting of uppercase alphabets.

Input

Your program is to read from standard input. The input starts with a line containing the T-bar represented by a string consisting of m ($5 \leq m \leq 20$) uppercase alphabets. In the next line, the requested bar magnet is given as a string consisting of n ($10 \leq n \leq 200{,}000$) uppercase alphabets.

Output

Your program is to write to standard output. Print exactly one line. The line should contain the minimum time to construct the requested bar magnet using the T-bars.

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

Sample Input 1	Output for the Sample Input 1
AACEEA AADEEAACCEEAABEEE	4

Sample Input 2	Output for the Sample Input 2
ABABCCC ABABCCABAB	4

Sample Input 3	Output for the Sample Input 3
ABABA ABABABABABABABABABA	0

Sample Input 4	Output for the Sample Input 4
BACBB DAACADCDBA	8

Sample Input 5	Output for the Sample Input 5
ABCDEABCDE ABCDEABCDEX	1

Problem C
Container Rearrangement
Time Limit: 1 Second

There are m containers in the container yard of a harbor. There are n positions on which containers can be stacked on each other, and these positions are arranged in a row. The height of each container is the same, and there is no limit on the number of containers that can be stacked on each position. Therefore, if a_i $(1 \le i \le n)$ denotes the number of containers stacked in the i-th position, $m = \sum_{i=1}^{n} a_i$.

The containers can be stacked without any restriction on the maximum number of stacking, but this is undesirable because it can cause some safety problems. We want to move some of the containers so that all the differences between the heights of stacks of containers at positions are less than or equal to 1. In other words, $|a_i - a_j| \le 1$ should be satisfied for any i, j. At each move, only one container can be moved, and we assume that the cost on the moving distance is negligible.

For example, Figure 1 shows 35 containers stacked on 8 positions corresponding to $m = 35$ and $n = 8$. Figure 2 shows the result of rearrangement to make sure that the height difference among each position is less than or equal to 1, which requires five moves of containers, that is the minimum.

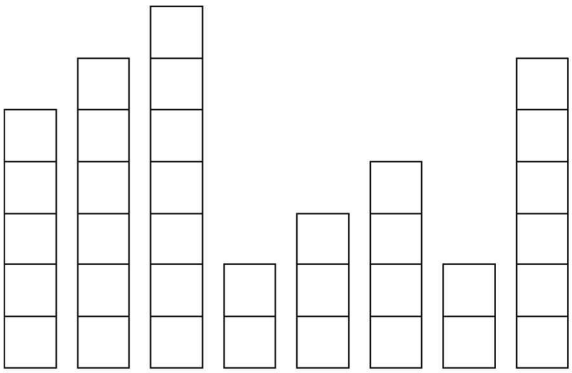


Figure 1. Before rearrangement

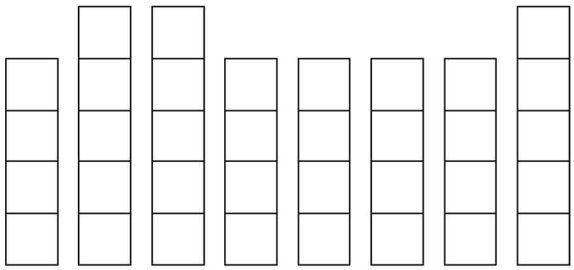


Figure 2. After rearrangement

Given an initial stacking information $a_1, a_2, \ldots, a_n$, write a program whose outputs is the minimum number of container moves necessary to satisfy $|a_i - a_j| \le 1$ for any i, j.

Input
Your program is to read from standard input. The input starts with a line containing one integer n $(1 \le n \le 10^6)$ representing the number of positions on which containers can be stacked. The next line contains n non-negative integers corresponding to the number of stacked containers a_i at the i-th position. The total number of containers m is less than or equal to 2×10^9 $(1 \le m \le 2 \times 10^9)$.

Output

Your program is to write to standard output. Print exactly one line. The line should contain the minimum number of container moves to satisfy the requirement.

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

Sample Input 1	Output for the Sample Input 1
4 1 2 3 3	1

Sample Input 2	Output for the Sample Input 2
8 5 6 7 2 3 4 2 6	5

Problem C

컨테이너 재배치

Container Rearrangement

제한 시간: 1 초

항구의 컨테이너 하치장 바닥에는 컨테이너를 쌓을 수 있는 칸이 일렬로 총 n개가 그려져 있고, 현재 하치장에는 총 m개의 컨테이너가 쌓여 있다. 개별 컨테이너의 높이는 모두 1로 동일하며, 각 칸에 쌓을 수 있는 컨테이너의 개수에는 제한이 없다. 즉, a_i $(1 \le i \le n)$가 현재 i번째 칸에 쌓여 있는 컨테이너의 개수를 나타내면, $m = \sum_{i=1}^{n} a_i$의 관계가 만족된다.

현재와 같이 높이에 아무 제한이 없이 컨테이너가 쌓여 있을 경우 각 칸별로 쌓여있는 컨테이너의 개수의 차이가 심하여 안전상 문제점을 유발할 수 있기 때문에, 일부 컨테이너를 크레인을 이용하여 다른 칸으로 옮겨서 각 칸의 높이 차이가 1 이하가 되도록 재배치하고자 한다. 즉, 임의의 i, j에 대해 $|a_i - a_j| \le 1$ 이 만족되어야 한다. 컨테이너는 한번에 한 개씩만 옮길 수 있고 옮기는 거리에 따른 추가 비용은 없다고 가정한다.

예를 들어 그림 1 과 같이 35 개의 컨테이너가 8 개의 칸에 쌓여 있을 경우 $m = 35, n = 8$에 해당한다. 이를 높이 차이가 1 이하가 되도록 재배치하면 그림 2 와 같은 결과를 얻을 수 있고, 이 경우 5 개의 컨테이너를 옮겨야 한다.

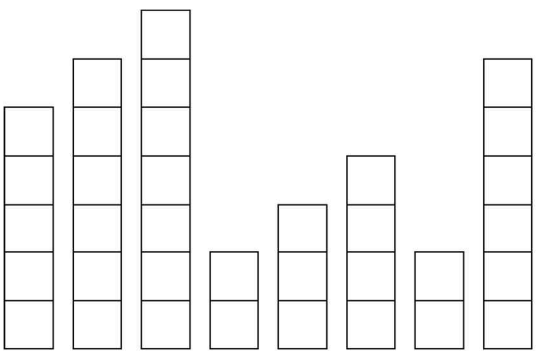


그림 1. 재배치 이전

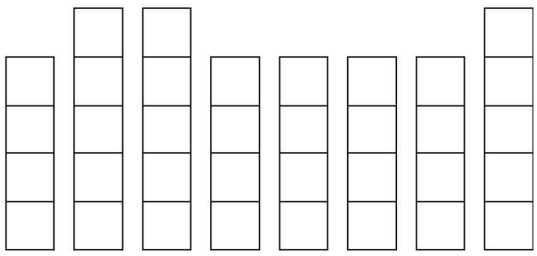


그림 2. 재배치 결과

입력으로 각 칸에 초기에 쌓여 있는 컨테이너의 높이 $a_1, a_2, \ldots, a_n$이 주어질 때, 임의의 i, j에 대해 $|a_i - a_j| \leq 1$ 조건을 만족하기 위해 옮겨야 하는 컨테이너의 최소 개수를 출력하는 프로그램을 작성하시오.

Input

입력은 표준입력을 사용한다. 첫 번째 줄에 컨테이너를 쌓을 수 있는 칸의 개수를 나타내는 양의 정수 n $(1 \leq n \leq 10^6)$이 주어진다. 다음 줄에는 현재 각 칸에 쌓여 있는 컨테이너의 개수 a_i를 나타내는 n 개의 0 이상의 정수들이 주어지고, 컨테이너의 총 개수 m은 $1 \leq m \leq 2 \times 10^9$ 으로 제한한다.

Output

출력은 표준출력을 사용한다. 문제의 조건을 만족하기 위해 옮겨야 하는 컨테이너의 최소 개수를 한 줄에 출력한다.

다음은 두 테스트 케이스에 대한 입출력 예이다.

Sample Input 1	Output for the Sample Input 1
4 1 2 3 3	1

Sample Input 2	Output for the Sample Input 2
8 5 6 7 2 3 4 2 6	5

Problem D
Flags
Time Limit: 1 Second

Consider integer points on the x-axis. Every point with an x-coordinate of 1 or greater is associated with one of the colors: red or black. Specifically, as shown in Figure D.1, the color of the point with the x-coordinate of 1 is red, the next two points are black, the next three points are red, the next four points are black, and so on. In this way, every point with an x-coordinate of 1 or greater is associated with either red or black color.

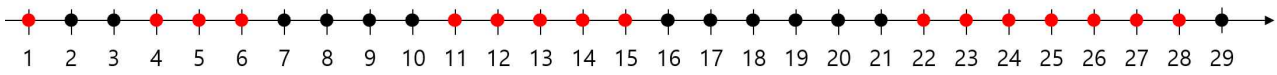


Figure D.1 Colors of points on the x-axis

Let $c(x)$ denote the color of the point with coordinate $x(\geq 1)$. Then $c(x)$ is defined as follows.

$$c(x) = \begin{cases} \text{red}, & \text{if } \sum_{i=0}^{2k} i < x \leq \sum_{i=0}^{2k+1} i \quad \text{for } k = 0,1,2,\ldots \\ \text{black}, & \text{otherwise} \end{cases}$$

There are n flags at given points along the x-axis. For example, Figure D.2 shows a situation where five flags are erected. The x-coordinates of the n flags are denoted as $f_1, f_2, \ldots, f_n$.

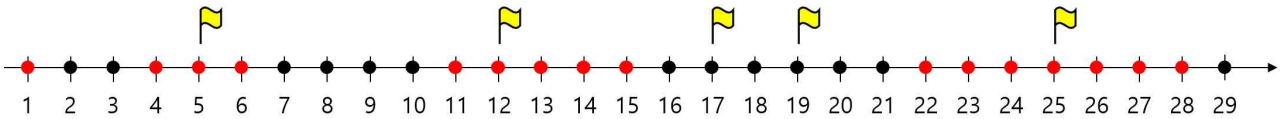


Figure D.2 An example of five flags on the x-axis

Displeased that the colors of points where flags locate are not the same, Bob tries to move all the flags to the right the same distance, say d, so that all the flags locate on the red points. In other words, he wants to find d such that for every $i(1 \leq i \leq n)$, $c(f_i + d)$ is red.

Given information about the locations of n flags, write a program to find the smallest possible value of d.

Input
Your program is to read from standard input. The input starts with a line containing an integer n ($1 \leq n \leq 10^5$), where n is the number of flags. Each of the next n lines contains an x-coordinate in increasing order. The i-th line contains f_i ($1 \leq i \leq n, 1 \leq f_i \leq 10^6$), the x-coordinate of the i-th flag.

Output
Your program is to write to standard output. Print exactly one line. The line should contain the minimum distance d by which all the flags must move to the right so that all the flags are on the red points.

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

Sample Input 1	Output for the Sample Input 1
1 8	3

Sample Input 2	Output for the Sample Input 2
5 4 15 28 60 211	0

Sample Input 3	Output for the Sample Input 3
7 123 129 130 188 189 190 191	23

Problem E
Gift Discount
Time Limit: 1 Second

Given the prices of n gifts, we try to buy the maximum number of gifts with the budget of b. You write a program to find the maximum number of gifts with a budget b you can buy when you can get a half-price discount on up to a gifts. Note that you can only receive a half-price discount at most once per gift.

Input
Your program is to read from standard input. The input starts with a line containing three integers, $n(1 \le n \le 100{,}000)$ representing the number of gifts, $b(1 \le b \le 10^9)$ representing the budget, and $a(0 \le a \le n)$ representing the maximum number of gifts eligible for a half-price discount. The next line contains n integers representing the gift prices. You may assume that all gift prices are between 2 and 10^9 and are even numbers.

Output
Your program is to write to standard output. Print exactly one line. The line should contain the maximum number of gifts that can be purchased.

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

Sample Input 1	Output for the Sample Input 1
6 26 2 4 6 2 10 8 12	5

Sample Input 2	Output for the Sample Input 2
6 23 1 4 6 2 12 8 14	4

Problem E

선물할인

Gift Discount

제한 시간: 1 초

n개의 선물 가격이 주어졌을 때, b의 예산으로 최대로 많은 선물을 사려고 한다. 이때 최대 a개의 선물에 대해서는 반값 할인을 받을 수 있다고 했을 때 최대로 살 수 있는 선물의 수를 구하는 프로그램을 작성하시오. 단, 한 선물에는 최대 한 번만 반값 할인을 받을 수 있다.

Input

입력은 표준입력을 사용한다. 첫 번째 줄에 선물의 개수를 나타내는 양의 정수 n $(1 \le n \le 100{,}000)$, 예산을 나타내는 양의 정수 b $(1 \le b \le 10^9)$, 반값 할인을 받을 수 있는 최대 선물의 수를 나타내는 정수 a $(0 \le a \le n)$가 공백을 사이에 두고 차례로 주어진다. 다음 줄에 n개의 선물 가격이 공백을 사이에 두고 주어진다. 선물 가격은 2 이상 10 억 이하의 값을 갖으며, 항상 짝수로 주어진다.

Output

출력은 표준출력을 사용한다. 조건을 만족하며 최대로 살 수 있는 선물의 수를 출력한다.

다음은 두 테스트 케이스에 대한 입출력 예이다.

Sample Input 1	Output for the Sample Input 1
6 26 2 4 6 2 10 8 12	5

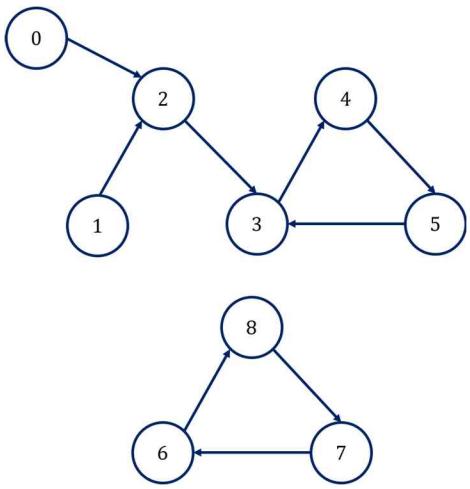
Sample Input 2	Output for the Sample Input 2
6 23 1 4 6 2 12 8 14	4

Problem F
Islands Tour
Time Limit: 1 Second

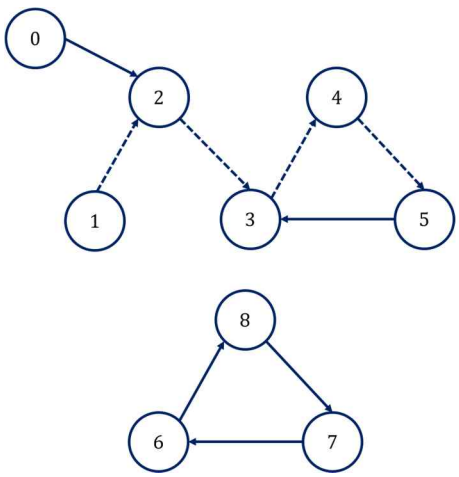
There are beautiful islands connected with zip-lines. A tourist can go from one island to another island sliding through a zip-line that connects the islands. Sliding through a zip-line above sunset sea, a tourist can see breathtaking sceneries of nature with twinkling lights over the sunlit sea waters. These islands are fantastic attractions among tourists. Each island is full of flowers of numerous colors. Travelling from an arbitrary island, a tourist called Optimizer wants to visit as many distinct islands as possible.

The islands are represented as a directed graph $G(V, E)$. A zip-line from an island v to another island w is represented as a directed edge $(v, w) \in E$. We assume that each island has at most one outgoing zip-line, that is, for each vertex $v \in V$, we have at most one outgoing edge.

For example, the figure below shows an example of the islands represented as a directed graph.



The dotted path in the following graph denotes a longest tour that visits as many distinct islands as possible.



Given a directed graph $G(V, E)$ that represents the islands and their connections using zip-lines, write a program to output the maximum number of islands that can be visited by Optimizer. Note that Optimizer can start from an arbitrary island and cannot visit the same island twice or more.

Input

Your program is to read from standard input. The input starts with a line containing two integers, m and n ($0 \leq m \leq n \leq 1{,}000{,}000$), where m is the number of zip-line connections (edges) and n is the number of islands (vertices). The islands are numbered from 0 to $n-1$. In the following m lines, the i-th line contains two integers v_i and w_i that represent a directed edge (v_i, w_i) from v_i to w_i. We assume that each vertex has at most one outgoing edge.

Output

Your program is to write to standard output. Print exactly one line. The line should contain the maximum number of distinct islands that can be visited by riding zip-lines starting from an arbitrary island.

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

Sample Input 1

```
9 9
0 2
2 3
1 2
3 4
4 5
5 3
8 7
7 6
6 8
```

Output for the Sample Input 1

```
5
```

Sample Input 2

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

Output for the Sample Input 2

```
3
```

Sample Input 3

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

Output for the Sample Input 3

```
2
```

Sample Input 4

```
0 4
```

Output for the Sample Input 4

```
1
```

Problem G
Jar Game
Time Limit: 1 Second

Two players **F** (irst) and **S** (econd) play a game with three jars each containing a, b and c pebbles. The game is played according to the following rules:

- Two players take turns one at a time. For each turn, the player chooses a jar and takes some pebbles from the jar.
- **F** starts first, then **S** next. These turns alternate till the game ends.
- The number of pebbles that can be drawn at the k-th turn is k; the number of pebbles taken by **F** at the first turn is 1. So in the next turn, **S** takes 2 pebbles, then at the third turn, **F** takes 3 pebbles, and so on.
- For each turn, the pebbles must be taken out of only one jar.
- At the k-th turn, if the number of pebbles remaining in the chosen jar is less than k, the player should take all the remained pebbles in that jar. If the remained pebbles is greater than k in the chosen jar, then the player is not allowed to take less than k pebbles from the jar.
- If there are no pebbles left in the three jars, then the game is over. The player with more pebbles wins the game when the game is over. So in some cases, a draw may be possible if the number of pebbles two players took is the same.
- We assume that two players **F** and **S** do their best to win.
- Two players always know the exact number of the pebbles remained in three jars. There is no hidden information in this jar game.

Given the number of pebbles in three jars, write a program to find who is the winner or if the draw is possible.

Input
Your program is to read from standard input. The input starts with a line containing three integers, a, b and c ($1 \le a, b, c \le 100$) denoting the number of pebbles in three jars at the beginning.

Output
Your program is to write to standard output. Print exactly one line. The line should contain a capital letter among {**F**,**S**,**D**}. {**F**,**S**} means the winner among two players and **D** denotes a draw when the game ends.

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

Sample Input 1	Output for the Sample Input 1
2 5 3	F

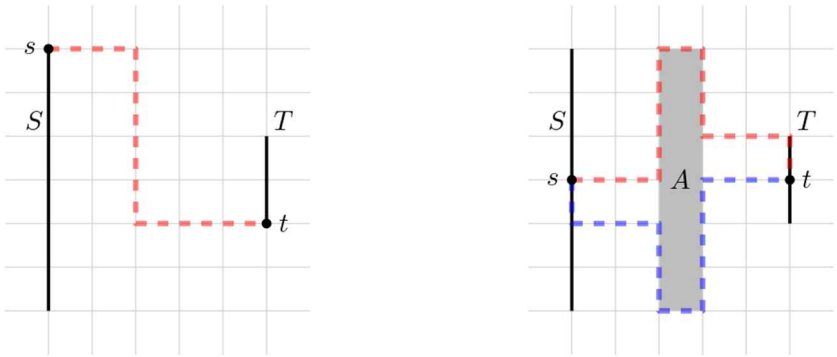
Sample Input 2	Output for the Sample Input 2
4 1 5	D

Sample Input 3	Output for the Sample Input 3
5 3 5	S

Problem H
Longest Shortest Paths
Time Limit: 2 Seconds

Consider n axis-aligned rectangles and two vertical segments S and T in the plane. We assume that all corners of the rectangles and segments are in integer coordinates. We also assume that the rectangles and segments are disjoint each other, that is, no two of them intersect each other and no two of them share a boundary point. For a point p in S and a point q in T, a *path* between p and q is a chain consisting of horizontal or vertical segments that connects p and q and does not intersect the interiors of the rectangles. The length of a path is the sum of the lengths of segments in the path. Thus, a shortest path between p and q is one whose length is the smallest among all paths between p and q.

For every pair of a point in S and a point in T, there is a shortest path between them. Let $d(p, q)$ denote the length of a shortest path between a point p in S and a point q in T. Our goal is to compute the length of the longest path among all shortest paths connecting a point in S and a point in T, that is, $\max_{p \in S} \max_{q \in T} d(p, q)$.



(a) (b)

For example, consider the figures above. Figure (a) shows two vertical segments S and T, and no rectangle in the plane. Every shortest path between a point in S and a point in T has length at most 9. Since $d(s, t) = 9$, we have $\max_{p \in S} \max_{q \in T} d(p, q) = d(s, t) = 9$ for this example.

Figure (b) shows an axis-aligned rectangle A and two vertical segments S and T in the plane. There are two shortest paths between a point s in S and a point t in T, one in red color and one in blue color. Then $d(s, t) = 11$. Observe that every shortest path between a point in S and a point in T has length smaller than or equal to 11. Thus, we have $\max_{p \in S} \max_{q \in T} d(p, q) = d(s, t) = 11$ for this example.

Given n axis-aligned rectangles and two vertical segments S and T that do not intersect each other, write a program to compute the length of the longest path among all shortest paths connecting a point in S and a point in T.

Input

Your program is to read from standard input. The input starts with a line containing six integers. The first three integers represent the x-coordinate and the two y-coordinates of the endpoints of the vertical segment S, and the last three integers represent the x-coordinate and the two y-coordinates of the endpoints of the vertical segment T.

The next line contains an integer n ($0 \leq n \leq 5{,}000$), where n is the number of axis-aligned rectangles. The rectangles are numbered from 1 to n. In the following n lines, the i-th line contains four nonnegative integers. The first two integers represent the x-coordinate and y-coordinate of the top-left corner of the rectangle i, and the last two integers represent the x-coordinate and y-coordinate of the bottom-right corner of the rectangle i.

All the coordinate values of endpoints of S and T, and two corners of the rectangles are nonnegative integers no more than 10^8.

Output

Your program is to write to standard output. Print exactly one line. The line should contain the length of the longest path among all shortest paths between a point a point in S and a point in T.

The following shows sample input and output for three test cases. Sample input 1 corresponds to the case of Figure (a), and sample input 2 corresponds to the case of Figure (b).

Sample Input 1	Output for the Sample Input 1
0 0 6 5 2 4 0	9

Sample Input 2	Output for the Sample Input 2
0 0 6 5 2 4 1 2 6 3 0	11

Sample Input 3	Output for the Sample Input 3
0 10 30 5 10 30 2 2 50 3 12 2 11 3 0	41

Problem I
Magic Potion
Time Limit: 3 Seconds

When making a potion, you first set the intensity of the heat to a certain degree, put ingredients in certain order into the cauldron according to its recipe, and boil the mixture until ready. In the brewing process, it is very important to prevent the heat from fluctuating, because the heat intensity decides the number of ingredients that form an alchemic bond.

Given a sequence X of ingredients in order, when the heat intensity is $i(\geq 1)$, all subsequences of length i in X are alchemic bonds of the corresponding potion; here duplicate bonds are irrelevant. Thus, there can be several different ingredient sequences that produce the same potion if they have the same set of alchemic bonds. On the other hand, different sets of alchemic bonds always produce different potions. Therefore, a potion recipe consists of an intensity of the heat and a sequence of ingredients in order. For instance, the following is a recipe for "the draught of anti-drowsiness".

- Heat intensity $i = 2$
- Ingredient sequence $X = bsm$,

where b denotes coffee bean, s denotes sugar, and m denotes milk.

Then, because the heat intensity is 2, its alchemic bonds are $\{bm, bs, sm\}$. If the heat intensity is 3, then there will be only one alchemic bond bsm . Moreover, if the heat intensity is greater than 3, then there will be no bonds. If someone wants to produce the draught of anti-drowsiness and puts ingredients in bms order under the heat intensity of 2, then the alchemic bonds are $\{bm, bs, ms\}$. Thus, this ingredient sequence does not produce the desired potion since the alchemic bond sm is missing.

Here is another example. For the heat intensity of 2, both ingredient sequences $X = abcabc$ and $Y = bababcab$ produce the same potion since their alchemic bond sets are the same as $\{aa, ab, ac, ba, bb, bc, ca, cb, cc\}$. On the other hand, it is easy to confirm that an ingredient sequence $Z = abbacb$ does not produce the same potion because there are no ca and cc bonds.

The potions department of the ICPC school holds a Potions Olympiad every year. Given two ingredient sequences, participants are asked to find the maximum heat intensity that results in the same potion from two sequences.

Given two ingredient sequences, write a program to output the maximum heat intensity in which both sequences produce the same potion.

Input

Your program is to read from standard input. The input consists of two lines. The first line contains a string X of length m $(1 \leq m \leq 200{,}000)$, and the second line contains a string Y of length n $(1 \leq n \leq 200{,}000)$, where all ingredient characters of X and Y consist of uppercase English letters (from '`A`' to '`Z`'), lowercase English letters (from '`a`' to '`z`') and digits (from '`0`' to '`9`'). Note that uppercase and lowercase letters are different (for example, '`A`' is treated as different from '`a`').

Output

Your program is to write to standard output. Print exactly one line. The line should print the maximum heat intensity i in which the input ingredient sequences X and Y produce the same potion. If it is impossible to make the same potions from X and Y, then the output should be 0 .

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

Sample Input 1	Output for the Sample Input 1
icpc2022bsm bms02icc	0

Sample Input 2	Output for the Sample Input 2
abcabc babcccab	2

Sample Input 3	Output for the Sample Input 3
abccbaabc babbccabaccacb	3

Problem J
Rectangles
Time Limit: 2 Seconds

An axis-parallel rectangle is a rectangle with sides parallel to the x-axis or the y-axis. Also its four vertices are different from each other.

For a set S of points in the plane, an axis-parallel rectangle is called to be *contained in* S if it has, as its vertices, four points belonging to S.

For example, in the left of Figure J.1, a set S of ten points is given in the plane. Then as the right of Figure J.1, there are three axis-parallel rectangles contained in S.

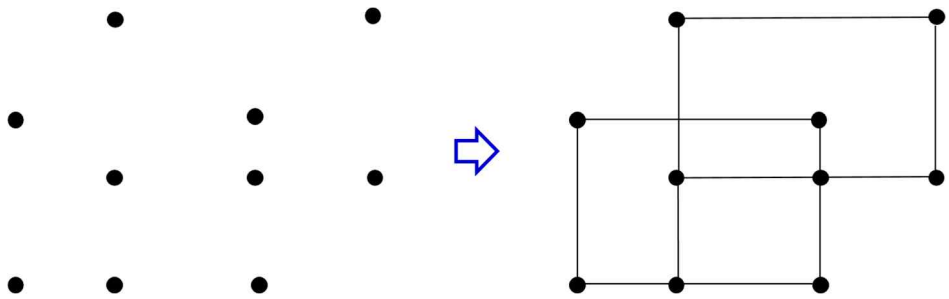


Figure J.1 There are three axis-parallel rectangles contained in the given set of points.

Given a set S of n distinct points in the plane, write a program to output the number of all the axis-parallel rectangles contained in S.

Input
Your program is to read from standard input. The input starts with a line containing an integer n ($1 \leq n \leq 70{,}000$), where n is the number of points given in the plane. In the following n lines, each line contains two integers that represent, respectively, the x-coordinate and the y-coordinate of a point. These coordinate values are between 0 and 10^5 and all the given points are distinct.

Output
Your program is to write to standard output. Print exactly one line. The line should contain the number of axis-parallel rectangles contained in the given point set.

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

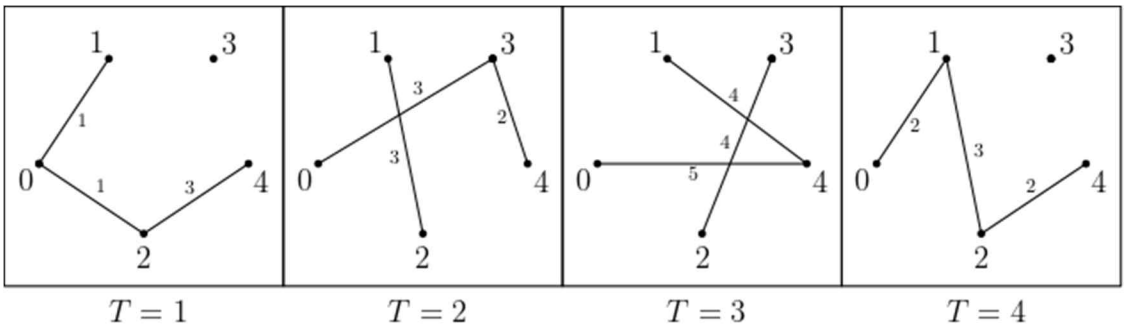
Sample Input 1	Output for the Sample Input 1
4 0 0 0 1 1 0 1 1	1

Sample Input 2	Output for the Sample Input 2
4 0 0 0 1 1 0 1 2	0

Sample Input 3	Output for the Sample Input 3
10 1 1 3 1 6 1 3 3 6 3 8 3 1 4 6 4 3 6 8 6	3

Problem K
Temporal Graph
Time Limit: 2 Seconds

A *temporal graph* is a data structure that expresses relationships that change over time. The vertex set V of a temporal graph does not change over time, and when the number of vertices is $n \geq 1$, V is denoted by $\{0, 1, \ldots, n-1\}$. The timestamp T has a value of positive integers $1, 2, \ldots, t$, and we express the passage of time as the timestamp increases. The edge set E_T is defined for each timestamp T, and the number of edges remains the same. Also, each of the edges has a positive integer weight. The figure below shows an example of a temporal graph with $V = \{0, 1, 2, 3, 4\}$ and $T = 1, 2, 3, 4$.



In a temporal graph, a path from one vertex to another vertex consists of a set of edges that appears in turn according to the passage of time. When constructing a path, at most one edge can be selected for each timestamp, and it is not necessary to select edges from consecutive timestamps. For examples, if we select the three edges $(0, 1)$, $(1, 2)$, and $(2, 4)$ in the temporal graph of the above figure in $T = 1, 2, 4$, respectively, they form a path from vertex 0 to vertex 4. However, if we select the three edges $(0, 2)$, $(2, 3)$, and $(3, 4)$ in $T = 1, 3, 2$, respectively, they cannot form a path from vertex 0 to vertex 4 (because the timestamps considered are not increasing). The length of a path is defined as the sum of the weights of the edges belonging to the path. Therefore, if we select two edges $(0, 2)$ and $(2, 4)$ in $T = 1,\ 4$, respectively, this will be the shortest path from vertex 0 to vertex 4, and the length of the path is $1 + 2 = 3$.

Given a temporal graph and two vertices s and e as the start and the end of the path, respectively, write a program to output the length of the shortest path from s to e.

Input
Your program is to read from standard input. The input starts with a line containing three integers, n, t, and m ($2 \leq n \leq 10{,}000$, $1 \leq t, m \leq 1{,}000$), where n is the number of vertices, t is the range of the timestamp, and m is the number of edges for each timestamp. The following line contains two integers s and e ($0 \leq s \neq e \leq n-1$), where s and e denote the start and end vertex of the path, respectively. The following m lines indicate information on the edges defined when $T = 1$. There is no more than one edge connecting two specific vertices. Each line contains three integers, where the first and second integers represent two end vertices of an edge, and the last integer w ($1 \leq w \leq 10{,}000$) represents the weight of the edge. The following $m \times (t-1)$ lines indicate information on the edges defined when $T = 2, 3, \ldots, t$ in the same way.

Output

Your program is to write to standard output. Print exactly one line. The line should contain the length of the shortest path from s to e. If there is no path from s to e, print `-1`.

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

Sample Input 1	Output for the Sample Input 1
5 4 3 0 4 0 1 1 0 2 1 2 4 3 0 3 3 1 2 3 3 4 2 0 4 5 1 4 4 2 3 4 0 1 2 1 2 3 2 4 2	3

Sample Input 2	Output for the Sample Input 2
5 4 2 0 4 0 1 1 0 2 1 1 2 3 3 4 2 1 2 3 2 3 4 0 1 2 1 2 3	-1

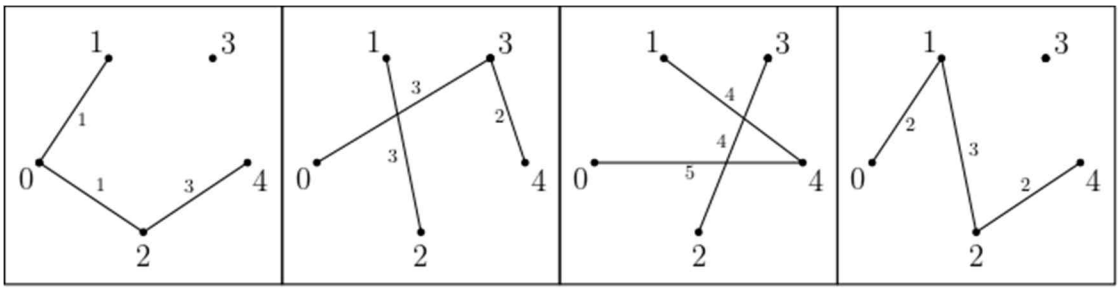
Problem K

템포럴 그래프

Temporal Graph

제한 시간: 2 초

*템포럴 그래프*는 시간의 흐름에 따라 변화하는 관계를 표현하는 자료 구조이다. 템포럴 그래프를 구성하는 정점 집합 V는 시간의 흐름에 따라 변하지 않으며, 정점의 개수가 $n \geq 1$ 이라 할 때 V는 $\{0, 1, \ldots, n-1\}$으로 나타낸다. 시간 표기 T는 양의 정수 $1, 2, \ldots, t$ 의 값을 가지며 시간 표기가 차례로 증가하는 것으로 시간의 흐름을 표현한다. 각 시간 표기 T에서 양의 정수인 가중치를 가지는 간선들의 집합 E_T가 정의되고, E_T에 포함되는 간선의 수는 일정하게 유지된다. 아래 그림은 정점 집합 $V = \{0, 1, 2, 3, 4\}$ 와 시간 표기 $T = 1, 2, 3, 4$ 에서 정의된 템포럴 그래프의 예시이다.



템포럴 그래프의 한 정점에서 다른 정점으로 향하는 경로는 증가하는 시간 표기에 따라 차례로 나타나는 간선들의 집합으로 구성된다. 경로를 구성할 때에는 각 시간 표기에서 최대 한 개의 간선을 선택할 수 있으며, 경로를 구성하는 간선들이 정의되는 시간 표기가 연속할 필요는 없다. 예를 들어, 위 그림의 템포럴 그래프에서 세 간선 $(0,1), (1,2), (2,4)$를 각각 시간 표기 $T = 1, 2, 4$ 에서 선택한다면 이는 정점 0에서 정점 4로 향하는 경로가 된다. 하지만 세 간선 $(0,2), (2,3), (3,4)$ 를 각각 시간 표기 $T = 1, 3, 2$에서 선택한다면 이는 정점 0에서 정점 4로 향하는 경로가 될 수 없다 (왜냐하면, 선택된 시간 표기가 증가하지 않기 때문이다). 경로의 길이는 경로에 포함되는 간선의 가중치의 총 합으로 정의한다. 따라서, 두 간선 $(0,2), (2,4)$ 를 각각 시간 표기 $T = 1, 4$ 에서 선택한다면 이는 정점 0에서 정점 4로 향하는 최단 길이 경로가 되고 경로의 길이는 $1 + 2 = 3$이 된다.

입력으로 템포럴 그래프와 경로의 시작과 끝이 되는 두 정점 s와 e가 주어질 때, s에서 e로 향하는 최단 길이 경로의 길이를 구하는 프로그램을 작성하시오.

Input

입력은 표준입력을 사용한다. 첫 번째 줄에 정점 집합의 크기를 나타내는 양의 정수 n $(2 \leq n \leq 10{,}000)$, 시간 표기의 범위를 나타내는 양의 정수 t $(1 \leq t \leq 1{,}000)$, 매 시간 표기마다 정의되는 간선들의 개수를 나타내는 양의 정수 m $(1 \leq m \leq 1{,}000)$이 차례로 주어진다. 다음 줄에 경로의 시작이 되는 정점을 나타내는 정수 s 와 경로의 끝이 되는 정점을 나타내는 정수 e $(0 \leq s \neq e \leq n-1)$가 차례로 주어진다. 이어지는 m 개의 줄은 시간 표기 $T = 1$에서 정의되는 간선들의 정보를 나타낸다. 특정한 두 정점을 연결하는 간선이 두 개 이상 나타나는 경우는 없다. 각 줄에는 간선이 연결하는 두 정점의 번호와 간선의 가중치를 나타내는 양의 정수 w $(1 \leq w \leq 10{,}000)$ 가 차례로 주어진다. 이어지는 $m \times (t-1)$ 줄은 시간 표기 $T = 2, \ldots, t$ 에서 정의되는 간선들의 정보를 동일한 방식으로 나타낸다.

Output

출력은 표준출력을 사용한다. 정점 s에서 정점 e로 향하는 최단 길이 경로의 길이를 한 줄에 출력하고, 경로가 정의되지 않는 경우에는 `-1` 을 출력한다.

다음은 두 테스트 케이스에 대한 입출력 예이다.

Sample Input 1

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

Output for the Sample Input 1

```
3
```

Sample Input 2

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

Output for the Sample Input 2

```
-1
```