01、

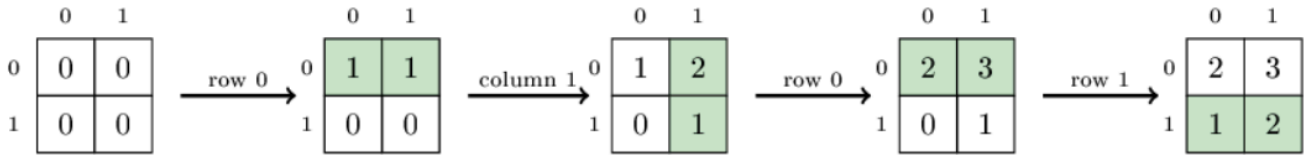
A two-dimensional array of integers of size NxN is given. Both rows and columns of the array are numbered from 0 to N−1. All of the values inside the array are initially zero.

A sequence of M operations is performed on the array. This sequence is described by an array of integers A and a string S. The K-th operation (for K within the range 0..M−1) performed on the array may be of one of two types:

- increase all of the values in the **row** number A[K] by 1, if S[K] is equal to *'R'*;
- increase all of the values in the **column** number A[K] by 1, if S[K] is equal to *'C'*.

Find the highest value in the array after processing the whole sequence of operations.

For example, given N = 2, A = [0, 1, 0, 1] and S = "RCRR", the subsequent states of the array will be as follows:



The answer is the highest value in the array, which is 3.

The answer is the highest value in the array, which is 3.

Write a function:

```
class Solution { public int solution(int N, int[] A, String
S); }
```

that, given an integer N, an array A of M integers and a string S of length M, returns the highest value in the array after processing all of the operations.

Examples:

1. Given N = 2, A = [0, 1, 0, 1] and S = "RCRR", the function should return 3, as explained above.

2. Given N = 10, A = [2, 4, 7, 8, 5, 2, 5, 8, 7] and S = "RCRCRCRCR", the function should return 4.

3. Given N = 1, A = [0, 0] and S = "CR", the function should return 2. There is only one cell in the array and it is incremented twice.

Assume that:

- N and M are integers within the range [1..100];
- each element of array A is an integer within the range [0..N−1];
- string S consists only of the characters "R" and/or "C";
- string S and array A have the same length, equal to M.

In your solution, focus on **correctness**. The performance of your solution will not be the focus of the assessment.

02、

A company is planning N projects, numbered from 0 to N−1. Completing the K-th project will bring value V[K] to the company. For some projects there may be additional requirements - the L-th requirement states that before starting project B[L], project A[L] should be completed. There are M such requirements.

The company has enough resources for **at most two** projects to be completed. If two projects are chosen, they will be completed one by one (in sequential order) and the total value they bring to the company is the sum of their individual values. What is the highest value that a valid choice of projects can bring to the company?

Write a function:

```
class Solution { public int solution(int[] V, int[] A, int[]
B); }
```

that, given array V of N integers and two arrays A and B of M integers each, returns the maximum value that the company may gain by completing at most two possible projects.

Examples:

1. Given V = [−3, 5, 7, 2, 3], A = [3, 1] and B = [2, 4], the function should return 9. This can be achieved by completing project 3 (with value 2) first and then project 2 (with value 7).

2. Given V = [1, 1, 5], A = [0, 1] and B = [2, 2], the function should return 2.

03、

3. Given V = [5, 6, 6, 7, −10], A = [0, 0, 0, 1, 2, 3] and B = [1, 2, 3, 3, 1, 2], the function should return 5. The projects that are possible to be completed are 0 and 4. As project 4 would bring negative value to the company, it is optimal to do only project 0. The structure of dependencies of projects 1, 2 and 3 form a cycle, so none of them can be completed in a valid choice of projects.

Write an **efficient** algorithm for the following assumptions:

- N is an integer within the range [1..100,000];
- M is an integer within the range [0..100,000];
- each element of array V is an integer within the range [−1,000,000,000..1,000,000,000];
- each element of arrays A and B is an integer within the range [0..N−1];
- a project may not require itself to be completed (A[K] ≠ B[K]);
- projects' requirements do not repeat.

In the city there are N districts (numbered from 0 to N−1) connected with M streets. The connections are described by two arrays, A and B, both of length M. A pair (A[K], B[K]) marks a street between districts A[K] and B[K] (for K from 0 to M−1). There are also L hospitals whose locations are described by an array H. The J-th hospital is placed in district H[J] (for J from 0 to L−1).

If an ambulance is needed in a district, one is sent from the hospital from which it will arrive in the shortest time. The ambulance arrives by the shortest possible route; passing one street takes it exactly 1 minute.

Potentially, there might be a patient in need in any district of the city. What is the longest time required to reach any possible patient with an ambulance?

Write a function

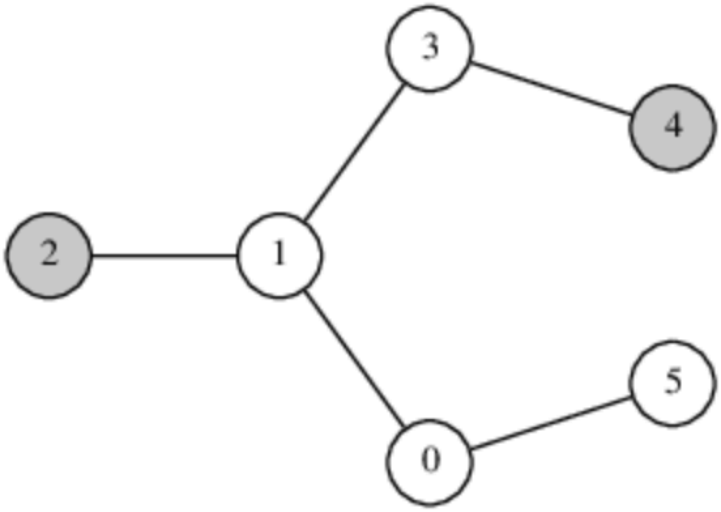
```
class Solution { public int solution(int N, int[] A, int[] B,
int[] H); }
```

that, given an integer N, arrays A and B describing streets in the city and an array H giving locations of hospitals, returns an integer − the longest driving time (in minutes) between a district and its closest hospital. If some district cannot be reached with an ambulance, the function should return −1.

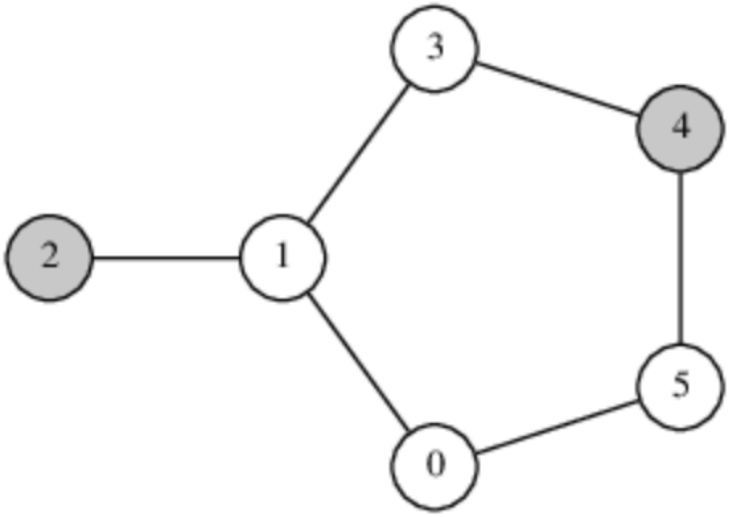
Examples:

1. Given N = 6, A = [0, 1, 1, 3, 0], B = [1, 2, 3, 4, 5], and H = [2, 4], your function should return 3. District number 5 has the longest waiting time. The ambulance will arrive from the hospital in district 2 in three minutes via route 2 → 1 → 0 → 5.

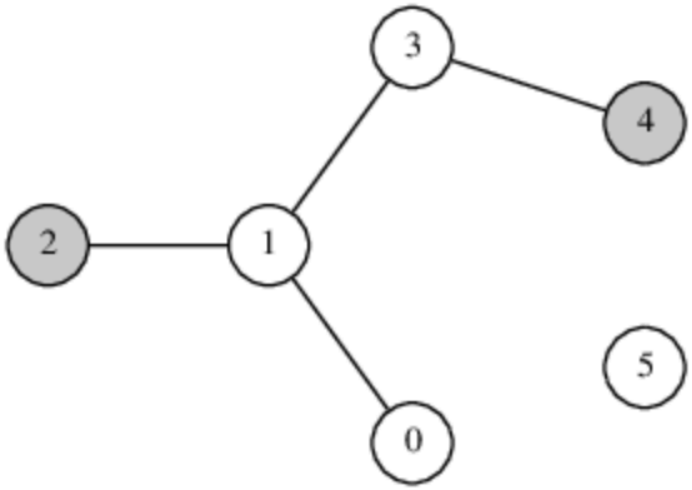
The city is shown in the diagram below. Districts with hospitals are shaded in gray.



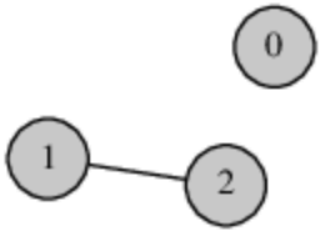
2. Given N = 6, A = [0, 1, 1, 3, 0, 4], B = [1, 2, 3, 4, 5, 5], and H = [2, 4], your function should return 2. Now the district with the longest ambulance arrival time is district 0. Every other district can be reached by an ambulance in at most 1 minute.



3. Given N = 6, A = [0, 1, 1, 3], B = [1, 2, 3, 4], and H = [2, 4], your function should return −1. District 5 is not connected with any district with a hospital.



4. Given N = 3, A = [1], B = [2], and H = [0, 1, 2], your function should return 0. There is a hospital in every district.



Write an **efficient** algorithm for the following assumptions:

Write an **efficient** algorithm for the following assumptions:

- N is an integer within the range [1..100,000];
- M is an integer within the range [0..100,000];
- L is an integer within the range [1..N];
- the elements of H are all distinct;
- each element of arrays A, B and H is an integer within the range [0..N−1];
- every street goes between two different districts;
- there are no multiple streets between two districts.