guruárth

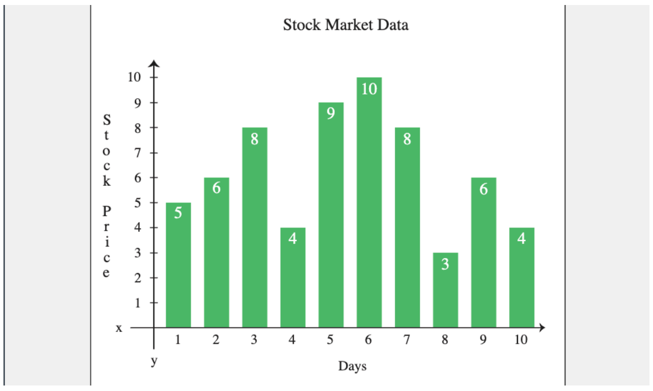
guruárth **Challenge #5**

Problem #1: Stock market prediction
Score points: 100

In this prediction game, the first player gives the second player some stock market data for some consecutive days. The data contains a company's stock price on each day. The rules for the game are:

- Player *1* will tell player *2* a day number
- player *2* has to find the nearest day on which stock price is smaller than the given day
- If there are two results, then player *2* finds the day number which is smaller
- if no such day exists, then the answer is *-1.*

For example, the image below shows the stock market data for *10* consecutive days. The horizontal axis represents the day number, starting at 1, and the vertical axis represents the stock price on that day.



Example

n = 10

stockData = [5, 6, 8, 4, 9, 10, 8, 3, 6, 4]

queries = [6, 5, 4]

On day 6, the stock price is 10. Both 9 and 8 are lower prices one day away. Choose 9 (day 5) because it is before day 6.
On day 5, the stock price is 9. 4 is the closest lower price on day 4.
On day 4, the stock price is 4. The only lower price is on day 8.
The return array is [5, 4, 8].

Function Description

Complete the *predictAnswer* function.

predictAnswer has *2* parameters:
int stockData[n]: the value of each *stockData[i]* is the stock price on the *i+1th* day (where *0 ≤ i < n*).
int queries[q]: the value of each element *queries[j]*, is the day number given in the query (where *0 ≤ j < q*).
Return
int[q]: the value at each index *i* is the answer to *queries[i]*
Constraints

- $1 \le n \le 10^5$
- $1 \le stockData[i] \le 10^9$
- $1 \le q \le 10^5$
- $1 \le queries[j] \le 10^9$

The first line contains an integer, *n*, denoting the number of elements in *stockData*.
Each line i^{th} of the *n* subsequent lines contains an integer, *stockData[i],* the stock price on the *i+1th* day.
Next line contains an integer, *q*, the number of elements in *queries*.
Each line j^{th} of the *q* subsequent lines contains an integer, *queries[j],* the day number of the *jth* query.

Sample Input 0

```
STDIN    Function
-----    --------
10  →   stockData[] size n = 10
5   →   stockData = [5, 6, 8, 4, 9, 10, 8, 3, 6, 4]
6
8
```

4

9

10

8

3

6

4

3 → queries[] size q = 3

3 → queries = [3, 1, 8]

1

8

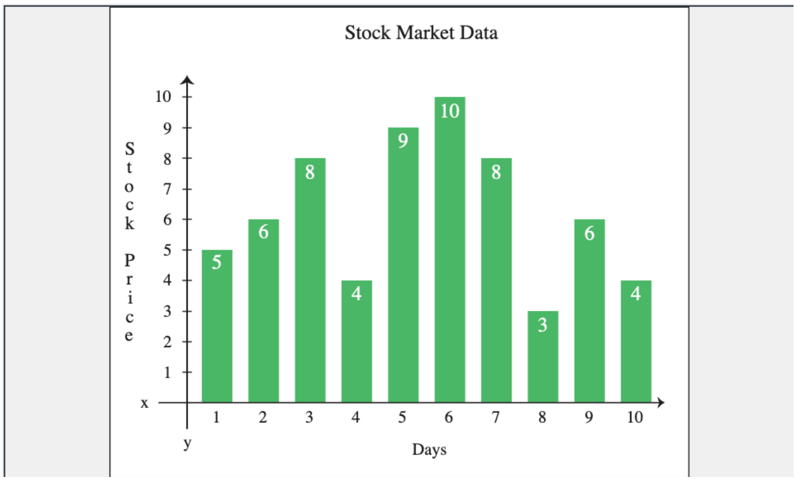
Sample Output 0

2

4

-1

Explanation 0



- If the day number is *3,* both days 2 and 4 are smaller. Choose the earlier day, day 2.
- If the day number is *1,* day 4 is the closest day with a smaller price.
- If the day number is *8,* there is no day where the price is less than 3. The answer is -1.
- The return array is [2, 4, -1]

Problem #2: Turnstile
Score points: 100

A university has exactly one turnstile. It can be used either as an exit or an entrance. Unfortunately, sometimes many people want to pass through the turnstile and their directions can be different. The i^{th} person comes to the turnstile at *time[i]* and wants to either exit the university if *direction[i] = 1* or enter the university if *direction[i] = 0*. People form *2* queues, one to exit and one to enter. They are ordered by the time when they came to the turnstile and, if the times are equal, by their indices.
If some person wants to enter the university and another person wants to leave the university at the same moment, there are three cases:

- If in the previous second the turnstile was not used (maybe it was used before, but not at the previous second), then the person who wants to leave goes first.
- If in the previous second the turnstile was used as an exit, then the person who wants to leave goes first.
- If in the previous second the turnstile was used as an entrance, then the person who wants to enter goes first.

Passing through the turnstile takes *1* second.
For each person, find the time when they will pass through the turnstile.
Function Description
Complete the function *getTimes*.
getTimes has the following parameters:
int time[n]: an array of *n* integers where the value at index *i* is the time in seconds when the i^{th} person will come to the turnstile
int direction[n]: an array of *n* integers where the value at index *i* is the direction of the i^{th} person
Returns:
int[n]: an array of *n* integers where the value at index *i* is the time when the i^{th} person will pass the turnstile
Constraints

- $1 \le n \le 10^5$
- $0 \le time[i] \le 109$ for $0 \le i \le n - 1$
- $time[i] \le time[i + 1]$ for $0 \le i \le n - 2$
- $0 \le direction[i] \le 1$ for $0 \le i \le n - 1$

Input Format For Custom Testing
Locked stub code reads input from stdin and passes it to the function.

The first line contains an integer, *n*, the number of persons, the number of *time* values and the number of *direction* values.
Each of the next *n* lines contains an integer *time[i]*.
The next line contains *n*.
Each of the next *n* lines contains an integer *direction[i]*.

Sample Case 0
Sample Input 0

```
STDIN    Function
-----    --------
4   →   time[] size n = 4
0   →   time = [0, 0, 1, 5]
0
1
5
4   →   direction[] size n = 4
0   →   direction = [0, 1, 1, 0]
1
1
0
```

Sample Output 0

```
2
0
1
5
```

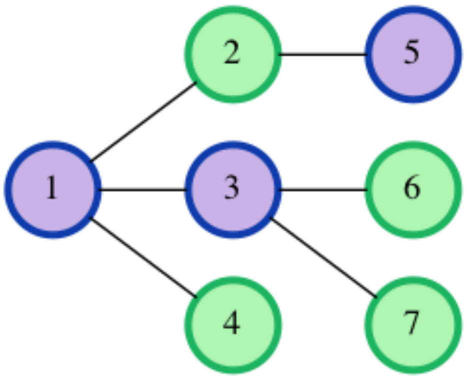
Explanation 0
At time *0*, persons *0* and *1* want to pass through the turnstile. Person *0* wants to enter the university and person *1* wants to leave the university. The turnstile was not used in the previous second, so the priority is on the side of the person *1*.
At time *1*, persons *0* and *2* want to pass through the turnstile. Person *2* wants to leave the university and at the previous second the turnstile was used as an exit, so the person *2* passes through the turnstile.
At time *2*, person *0* passes through the turnstile.
At time *5*, person *3* passes through the turnstile.

Problem #3: Opening Hospitals
Score points: 100

The health care of XYZ country is very poor. There are no hospitals in any of the country's cities. The cities in XYZ are connected by a two-way road network in the form of a tree. Elections are coming up, and the government has decided to open the minimum number of hospitals needed to ensure that all the country's citizens have access to a hospital in their own city or a neighbor city with a direct road connection.

Write a function that receives the number of cities, *city_nodes*, and two arrays, *city_from* and *city_to*, where *city_from[i]* has a bidirectional road connecting it with *city_to[i]* (where *0 <= i < city_nodes - 2*) and returns the minimum number of hospitals needed to provide the desired access for everyone.

For example, there are *city_nodes* = 7 cities and the roads connect cities *city_from = [1, 3, 1, 3, 2, 1]* with cities *city_to = [2, 6, 4, 7, 5, 3]*.
A visual representation is:



One optimal solution is to build hospitals in cities 1, 3, and 5 (colored in blue) since all the other cities have at least one road connecting them to one of these cities, and it can not be solved with fewer hospitals. The final answer is 3.

Function Description
Complete the function *hospital*. The function must return an integer.

hospital has the following parameter:
  city_nodes: an integer
  city_from: an array of integers

city_to: an array of integers

Constraints

- $2 \le city\_nodes \le 10^5$
- $city\_edges = city\_nodes - 1$
- $1 \le city\_from[i], city\_to[i] \le$ city_nodes

Input Format For Custom Testing

The first line contains two space-separated integers that describe *city_nodes* and *city_edges*.
Each line *i* of the *n* subsequent lines (where $0 \le i < n$) contains two space-separated integers, *city_from[i]* and *city_to[i].*

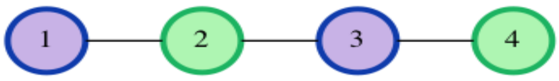
Sample Case 0
Sample Input For Custom Testing

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

Sample Output

```
2
```

Explanation



One solution is to build hospitals in cities 1 and 3. It can not be solved with 1 hospital.