Investigating Frog Behaviour on Lily Pad Patterns

Recently, the biologist Ina discovered a new frog species on the lily pads of a pond. She observed the frogs for a while and found them to be very conscious about their personal space because they avoided sharing a lily pad with other frogs. Also, they seemed quite lazy as they did not move often, and if they did, they always jumped to the nearest empty lily pad.

To confirm her hypotheses about the frogs' movement pattern, Ina set up a large number of lily pads in a pool in her laboratory, arranged in a straight line. Since the frogs were attracted to light, she was able to simplify the test setup further by placing a bright light at one end of that line. This way, the frogs would always jump in one direction (towards the light).

Of course, Ina could now place some frogs on the lily pads and sit there all day watching the frogs jump around. But as the frogs move so rarely, it would take ages to gather a sufficient amount of data.

She therefore attached to each frog a tiny device that could log all jumps of that frog. This way, she could put the frogs on the lily pads, leave them alone for a few hours and come back later to collect the data. Unfortunately, the devices had to be so tiny that there was no space for a position tracking system; instead, the devices could only record the times of the jumps.

But if the movement pattern of the frogs is as restricted as Ina thinks, surely the individual movements of the frog can be reconstructed only from the initial positions and the recorded jump time stamps?

Input Format

The input consists of:

- One line with an integer n $(1 \leq n \leq 2 \cdot 10^5)$, the number of frogs.
- One line with n integers $x_1, \ldots, x_n$ $(1 \leq x_i \leq 10^6)$, the number of the lily pad on which the ith frog initially sits. The lily pads are numbered consecutively, starting at 1. It is guaranteed that the initial positions are strictly increasing, i.e. $x_1 < x_2 < \cdots < x_n$.
- One line with an integer q $(1 \leq q \leq 2 \cdot 10^5)$, the number of jumps recorded.
- q lines, each containing an integer i $(1 \leq i \leq n)$, indicating that the ith frog jumped. The jumps are given in chronological order and you may assume that a jumping frog lands before the next jump begins. The frogs always jump to the nearest empty lily pad with a larger number, and you may assume that such a lily pad always exists.

Constraints

$1 \leq n \leq 2 \cdot 10^5$
$1 \leq x_i \leq 10^6$

$x_1 < x_2 < \cdots < x_n$
$1 \leq q \leq 2 \cdot 10^5$
$1 \leq i \leq n$

Output Format

For each jump, output the number of the lily pad the frog lands on.

Sample Input 0

```
5
1 2 3 5 7
3
1
2
4
```

Sample Output 0

```
4
6
8
```

Sample Input 1

```
5
1 2 3 5 7
4
1
1
1
1
```

Sample Output 1

```
4
6
8
9
```

Submissions: 49
Max Score: 100
Difficulty: Medium

Rate This Challenge:
☆☆☆☆☆

More

C

```
#include <stdio.h>
```

```
#define MAX 1200000

int BinarySearch(int A[], int i, int j, int k) {
    int m, result = -1;

    while (i <= j) {
        m = (i + j) >> 1;

        if (A[m] == k) {
            result = m;
            break;
        } else if (k > A[m]) {
            i = m + 1;
        } else {
            j = m - 1;
        }
    }

    if (result == -1)
        result = (-1) * i - 1;

    return result;
}

int main(){
    int n, PosicionRanas[MAX + 1], LiriosLibres[MAX + 1], lirioLibre, contador = 1, q, salto;

    scanf("%d", &n);

    for (int i = 1; i <= n; i++)
        scanf("%d ", &PosicionRanas[i]);

    scanf("%d", &q);

    for (int i = 1; i <= MAX + 1; i++)
    {
        lirioLibre = BinarySearch(PosicionRanas, 1, n + 1, i);

        if (lirioLibre < 0)
        {
            LiriosLibres[contador] = i;
            contador++;
        }

    }

    for (int i = 1; i <= q; i++)
    {
        scanf("%d", &salto);

        int lirioOcupado = PosicionRanas[salto];

        int lirioCercano = BinarySearch(LiriosLibres, 1, contador - 1, lirioOcupado);
        lirioCercano = -1 * lirioCercano - 1;

        int lirioVacio = LiriosLibres[lirioCercano];

        PosicionRanas[salto] = lirioVacio;
        LiriosLibres[lirioCercano] = lirioOcupado;

        printf("%d\n", PosicionRanas[salto]);
    }

    return 0;
}
```

Line: 1 Col: 1

Upload Code as File

Test against custom input

Run Code

Submit Code

Interview Prep | Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy |