29/23, 12.48 AM Count the Number of Infection Sequences - Lecteous Count

plore/) Problems(/problemset/all/) Contest(/contest/)

Discuss(/discuss/)

Interview > Store > 4

(/problems/minimumo 0 time-visiting-all-

points/)



100146. Count the Number of Infection Sequences

My Submissions (/contest/weekly-contest-374/problems/count-the-number-of-infection-sequences/submissions/)

Back to Contest (/contest/weekly-contest-374/)

You are given an integer n and a 0-indexed integer array sick which is sorted in increasing order.

There are $\, n \,$ children standing in a queue with positions $\, 0 \,$ to $\, n \,$ - $\, 1 \,$ assigned to them. The array sick contains the positions of the children who are infected with an infectious disease. An infected child at position $\, i \,$ can spread the disease to either of its immediate neighboring children at positions $\, i \,$ - $\, 1 \,$ and $\, i \,$ + $\, 1 \,$ if they exist and are currently not infected. At most one child who was previously not infected can get infected with the disease in one second.

It can be shown that after a finite number of seconds, all the children in the queue will get infected with the disease. An **infection sequence** is the sequential order of positions in which **all** of the non-infected children get infected with the disease. Return *the total number of possible infection sequences*.



Since the answer may be large, return it modulo $10^9 + 7$.

Note that an infection sequence does not contain positions of children who were already infected with the disease in the beginning.

Example 1:

Input: n = 5, sick = [0,4]

Output: 4

Explanation: Children at positions 1, 2, and 3 are not infected in the beginning. There are 4 possible infection seque – The children at positions 1 and 3 can get infected since their positions are adjacent to the infected children 0 and Now, the child at position 2 is adjacent to the child at position 1 who is infected and the child at position 3 is adj Finally, the child at position 3 gets infected because it is adjacent to children at positions 2 and 4 who are infecte – The children at positions 1 and 3 can get infected because their positions are adjacent to the infected children 0 a Now, the child at position 2 is adjacent to the child at position 1 who is infected and the child at position 3 is adj Finally, the child at position 2 gets infected because it is adjacent to children at positions 1 and 3 who are infecte – The infection sequence is [3,1,2]. The order of infection of disease in the children can be seen as: [0,1,2,3,4] = – The infection sequence is [3,2,1]. The order of infection of disease in the children can be seen as: [0,1,2,3,4] =

Example 2:

```
Input: n = 4, sick = [1]

Output: 3

Explanation: Children at positions 0, 2, and 3 are not infected in the beginning. There are 3 possible infection seque

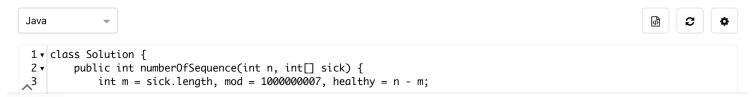
- The infection sequence is [0,2,3]. The order of infection of disease in the children can be seen as: [0,1,2,3] \Rightarrow [0]

- The infection sequence is [2,0,3]. The order of infection of disease in the children can be seen as: [0,1,2,3] \Rightarrow [0]

- The infection sequence is [2,3,0]. The order of infection of disease in the children can be seen as: [0,1,2,3] \Rightarrow [0]
```

Constraints:

- $2 <= n <= 10^5$
- 1 <= sick.length <= n 1
- 0 <= sick[i] <= n 1
- sick is sorted in increasing order.



Start Now

2 Easy Steps

- 1. Click "Start Now"
- 2. Add Data Shield

```
- C.COMD(HCGECHY, H
                                             2 r C V L III
                                                       エコノ,
11
             res %= mod;
12
             healthy -= n - 1 - sick[m - 1];
             for (int i = 1; i < m; i++) {
13 ▼
14
                 int d = sick[i] - sick[i - 1] - 1;
15 ▼
                 if (d > 0) {
                     res *= C.comb(healthy, d);
16
                     res %= mod;
17
18
                     healthy -= d;
19
                     res *= p[d - 1];
20
                     res %= mod;
21
22
23
             return (int) res;
        }
24
25
26 ▼
        long[] buildPowerOf2Array(int n, int mod) {
27
             long[] power = new long[n];
28
             power[0] = 1;
29
             for (int i = 1; i < n; i++) power[i] = power[i - 1] * 2 % mod;
30
             return power;
        }
31
32
        class Combinatorics {
33 ▼
34
             long[] fact, ifact, inv;
35
             int mod;
36
37 ▼
             Combinatorics(int N, int mod) {
38
                 fact = new long[N];
39
                 ifact = new long[N];
40
                 inv = new long[N];
41
                 this.mod = mod;
                 fact[0] = ifact[0] = inv[1] = 1;
42
                 for (int i = 2; i < N; i++) inv[i] = (mod - mod / i) * <math>inv[mod \% i] \% mod;
43
                 for (int i = 1; i < N; i++) {
44 ▼
45
                     fact[i] = fact[i - 1] * i % mod;
                     ifact[i] = ifact[i - 1] * inv[i] % mod;
46
47
                 }
48
             }
49
50 ▼
             long comb(int n, int k) {
51
                 if (n < k \mid l \mid k < 0) return 0;
52
                 return fact[n] * ifact[k] % mod * ifact[n - k] % mod;
53
54
        }
55
    }
```

□ Custom Testcase

Use Example Testcases

Submission Result: Accepted (/submissions/detail/1115498321/)

More Details > (/submissions/detail/1115498321/)

Share your acceptance!

Copyright © 2023 LeetCode

Help Center (/support) | Jobs (/jobs) | Bug Bounty (/bugbounty) | Online Interview (/interview/) | Students (/student) | Terms (/terms) | Privacy Policy (/privacy)

Luited States (/region)