6150. Construct Smallest Number From DI String

My Submissions (/contest/weekly-contest-306/problems/construct-smallest-number-from-di-string/submissions/)

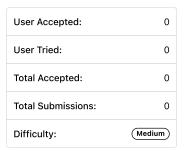
Back to Contest (/contest/weekly-contest-306/)

You are given a **0-indexed** string pattern of length n consisting of the characters 'I' meaning **increasing** and 'D' meaning **decreasing**.

A **0-indexed** string num of length n + 1 is created using the following conditions:

- num consists of the digits '1' to '9', where each digit is used at most once.
- If pattern[i] == 'I', then num[i] < num[i + 1].
- If pattern[i] == 'D', then num[i] > num[i + 1].

Return the lexicographically **smallest** possible string num that meets the conditions.



ر (/problems)

ladder-ii/)

Example 1:

```
Input: pattern = "IIIDIDDD"
Output: "123549876"
Explanation:
At indices 0, 1, 2, and 4 we must have that num[i] < num[i+1].
At indices 3, 5, 6, and 7 we must have that num[i] > num[i+1].
Some possible values of num are "245639871", "135749862", and "123849765".
It can be proven that "123549876" is the smallest possible num that meets the conditions.
Note that "123414321" is not possible because the digit '1' is used more than once.
```

Example 2:

```
Input: pattern = "DDD"
Output: "4321"
Explanation:
Some possible values of num are "9876", "7321", and "8742".
It can be proven that "4321" is the smallest possible num that meets the conditions.
```

Constraints:

- 1 <= pattern.length <= 8
- pattern consists of only the letters 'I' and 'D'.

```
JavaScript
                                                                                                                             C
 1 let p, res, n;
    const smallestNumber = (pattern) => {
 2 ▼
 3
        p = pattern;
        n = p.length;
 5
        res = '';
 6
        dfs([]);
 7
        return res;
 8
    };
 9
10 ▼
    const dfs = (cur) => {
11
        if (cur.length > n + 1) return;
12 •
        if (cur.length == n + 1) {
            let t = cur.join("")
13
            if (res.length == 0) {
14 ▼
15
                 res = t;
            } else {
16 •
17
                 if (t < res) res = t;
18
19
        for (let i = '1'; i <= '9'; i++) {
20 •
21
             cur.push(i);
22
             if (ok(cur)) dfs(cur);
23
             cur.pop();
24
        }
```

```
25
     };
26
27 \cdot const ok = (a) => {
            let u = new Set(a);
28
29
            if (u.size != a.length) return false;
           for (let i = 1; i < a.length; i++) {
    let mark = a[i] > a[i - 1] ? 'I' : 'D';
    if (mark != p[i - 1]) return false;
30 •
31
32
33
            }
34
            return true;
35
      };
```

 $\ \square$ Custom Testcase

Use Example Testcases

Run (Submit

Submission Result: Accepted (/submissions/detail/773091591/) ?

More Details > (/submissions/detail/773091591/)

Share your acceptance!

Copyright © 2022 LeetCode

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

United States (/region)