Find Permutation

By now, you are given a **secret signature** consisting of character 'D' and 'I'. 'D' represents a decreasing relationship between two numbers, 'I' represents an increasing relationship between two numbers. And our **secret signature** was constructed by a special integer array, which contains uniquely all the different number from 1 to n (n is the length of the secret signature plus 1). For example, the secret signature "DI" can be constructed by array [2,1,3] or [3,1,2], but won't be constructed by array [3,2,4] or [2,1,3,4], which are both illegal constructing special string that can't represent the "DI" **secret signature**.

On the other hand, now your job is to find the lexicographically smallest permutation of [1, 2, ... n] could refer to the given **secret signature** in the input.

Example 1:

```
Input: "I"
Output: [1,2]
Explanation: [1,2] is the only legal initial spectial string can construct secret s
ignature "I", where the number 1 and 2 construct an increasing relationship.
```

Example 2:

```
Input: "DI"
Output: [2,1,3]
Explanation: Both [2,1,3] and [3,1,2] can construct the secret signature "DI", but
since we want to find the one with the smallest lexicographical permutation, you ne
ed to output [2,1,3]
```

Note:

- The input string will only contain the character 'D' and 'I'.
- The length of input string is a positive integer and will not exceed 10,000

Solution 1

I have used a greedy algorithm:

1. Loop on the input and insert a decreasing numbers when see a 'I'
2. Insert a decreasing numbers to complete the result.

Simple example:
Input: "DIDDID"
0 []
1 [2, 1]
2 [2, 1]
3 [2, 1]
4 [2, 1, 5, 4, 3]
5 [2, 1, 5, 4, 3]
[2, 1, 5, 4, 3, 7, 6]

Then, output is [2, 1, 5, 4, 3, 7, 6]

```
def findPermutation(self, s):
    ret = []
    for i in range(len(s)):
      if s[i] == 'I':
        ret.extend(range(i + 1, len(ret), -1))
    ret.extend(range(len(s) + 1, len(ret), -1))
    return ret
```

Hope that you like my solution. :)

written by lee215 original link here

Solution 2

For example, given `IDIIDD` we start with sorted sequence `1234567`
Then for each `k` continuous `D` starting at index `i` we need to reverse `[i, i+k]` portion of the sorted sequence.

```
IDIIDD
1234567 // sorted
1324765 // answer
```

```
    public int[] findPermutation(String s) {
        int n = s.length(), arr[] = new int[n + 1];
        for (int i = 0; i <= n; i++) arr[i] = i + 1; // sorted
        for (int h = 0; h < n; h++) {
            if (s.charAt(h) == 'D') {
                int l = h;
                while (h < n && s.charAt(h) == 'D') h++;
                reverse(arr, l, h);
            }
        }
        return arr;
    }

    void reverse(int[] arr, int l, int h) {
        while (l < h) {
            arr[l] ^= arr[h];
            arr[h] ^= arr[l];
            arr[l] ^= arr[h];
            l++; h--;
        }
    }
```

written by yuxiangmusic original link here

Solution 3

```
vector<int> findPermutation(string s) {
  vector<int> ret;
  for (int i = 0; i <= s.size(); ++i)
    if (i == s.size() || s[i] == 'I')
      for (int j = i + 1, lenTmp = ret.size(); j > lenTmp; --j)
        ret.push_back(j);
  return ret;
}
```

written by lee215 original link here