2375. Construct Smallest Number From DI String

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.*

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'`.

2375. Construct Smallest Number From DI String

Overview

We are given a string pattern consisting of the characters `'I'` (increasing) and `'D'`. We need to construct and return in the form of a string the lexicographically smallest number that satisfies certain conditions determined by the pattern.

> The term "lexicographically smallest" refers to the smallest possible sequence of numbers when compared as strings. This means we need to prioritize smaller numbers in the earlier positions when constructing the sequence.

To break down the problem, let's first understand the requirements. The pattern is a string of length `n`, where each character dictates the relationship between consecutive digits in the number. The primary goal is to satisfy the following conditions:

- If `pattern[i] == 'I'`, then the digit at position `i` in the number should be smaller than the digit at position `i + 1`.
- If `pattern[i] == 'D'`, then the digit at position `i` should be larger than the digit at position `i + 1`.

In other words, this means:

- At positions where the pattern has `'I'`, the number must increase.
- At positions where the pattern has `'D'`, the number must decrease.

The resulting number, `num`, has a length of `n + 1` because it includes one more digit than the pattern. Additionally, the digits used in the number must be distinct, ranging from `'1'` to `'9'`, meaning that each digit can appear at most once.

Consider the input pattern `"IIDIDDD"`. One valid number that satisfies this pattern is `"123549876"`. Here's why:

- For the first three `'I'`s, the numbers must increase: `1 < 2 < 3 < 5`.
- At position 3, we hit a `'D'`, so the numbers must decrease: `5 > 4`.
- Then, we have another `'I'` (position 4), so the number at position 4 must be smaller than the one at position 5: `4 < 9`.
- The rest of the pattern requires a decreasing sequence at positions 5, 6, 7 and 8: `9 > 8 > 7 > 6`.

The number `"123549876"` is the smallest possible number that adheres to this pattern. Notably, each digit is used only once, and the number is constructed in lexicographically smallest order.

2375. Construct Smallest Number From DI String

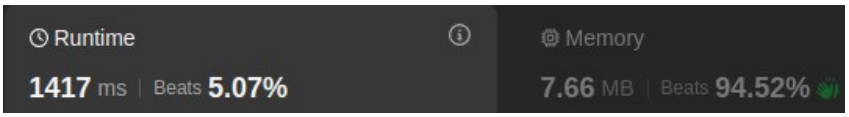
/*

Brute force: Generate all possibilities and check

Time complexity: $O(\frac{9!.n}{(8-n)!})=O(9!.8)$

Space complexity: $O(19+2n)$

*/




```
class Solution {
    public:
        std::string smallestNumber(std::string pattern) {
            int n=pattern.size();

            // Digits to use for the answer
            std::string charset="123456789";

            // Avoid repeated digits in the answer
            std::vector<int> is_used(10,0);

            std::string ans="9999999999",tmp_ans;
```

```
        // Recursive function+backtracking to generate all possibilities
        // for each possible answer, check if is matching the given pattern
        auto generate_all=[&](int k,auto& self){

            // Function to check if a string matches a pattern
            auto is_matching=[&](std::string& s,std::string& p)->bool{
                for(int i=0;i<n;++i){
                    if(p[i]=='I' && s[i]>s[i+1] || p[i]=='D' && s[i]<s[i+1]) return false;
                }
                return true;
            };

            // If the answer string is created
            if(k==0){
                // check if it matches the pattern,
                // if yes, minimize it
                if(is_matching(tmp_ans,pattern)) ans=std::min(ans,tmp_ans);
                return;
            }
            // Try all digits from 1 to 9
            for(int i=0;i<9;++i){
                // if the digit is not used in the answer
                if(!is_used[charset[i]-'0']){
                    is_used[charset[i]-'0']=1; // Mark it as used
                    tmp_ans.push_back(charset[i]); // added to the answer

                    self(k-1,self); // solve for the remaining digits

                    // Backtrack and try other answer
                    is_used[charset[i]-'0']=0;
                    tmp_ans.pop_back();
                }
            }
        };

        // Generate all the strings of size (n+1) using the charest="123456789"
        generate_all(n+1,generate_all);

        return ans;
    }
};
```

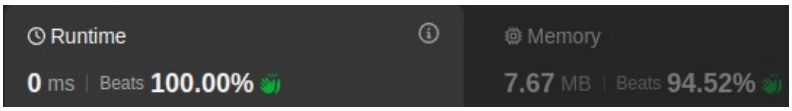
2375. Construct Smallest Number From DI String

/*

Recursion+backtracking: build the answer

Time complexity: $O(n!)$

Space complexity: $O(10+n)$



*/

```
class Solution {
    public:
        std::string smallestNumber(std::string pattern) {
            int n=pattern.size();

            // Avoid repeated digits in the answer
            std::vector<int> is_used(10,0);

            std::string tmp_ans,ans;
```

```
        // Build the answer using backtracking
        auto solve=[&](int i,auto& self)->bool{
            // If the temporary answer is build
            if(tmp_ans.size()==n+1){
                ans=tmp_ans; // take it as a valid answer
                return true;
            }

            // For every digit from 1 to 9
            for(int j=1;j<=9;++j){
                if(is_used[j]) continue;

                if(!tmp_ans.empty() &&
                        (pattern[i-1]=='I' && tmp_ans.back()>=j+'0' ||
                         pattern[i-1]=='D' && tmp_ans.back()<=j+'0')) continue;

                // if the digit is not used in the answer and the answer is matching the pattern

                is_used[j]=1; // Mark it as used

                tmp_ans.push_back(j+'0'); // added to the answer

                // If the digit j is in place
                if(self(i+1,self)) return true; // solve for the remaining digits

                // Otherwise, backtrack and try for an other answer
                is_used[j]=0;
                tmp_ans.pop_back();
            }

            return false;
        };

        solve(0,solve);

        return ans;
    }
};
```

2375. Construct Smallest Number From DI String

```
/*
   Stack: build the answer
   Time complexity: O(n)
   Space complexity: O(n)
*/
class Solution {
   public:
      std::string smallestNumber(std::string pattern) {
         pattern.push_back('I');

         int n=pattern.size();

         std::stack<int> st;

         std::string ans;
         for(int i=0;i<n;++i){
            st.push(i+1);
            if(pattern[i]=='I'){
               while(!st.empty()){
                  ans.push_back(st.top()+'0');
                  st.pop();
               }
            }
         }

         return ans;
      }
};
```