Competitive programming

wu

September 26, 2022

1 Dynamic Programming

1.1 Digit DP

Problem 1.1.1 (LeetCode 788: Rotated Digits). An integer x is a **good** if after rotating each digit individually by 180 degrees, we get a valid number that is different from x. Each digit must be rotated - we cannot choose to leave it alone.

A number is valid if each digit remains a digit after rotation. For example:

- 0, 1, and 8 rotate to themselves,
- 2 and 5 rotate to each other (in this case they are rotated in a different direction, in other words, 2 or 5 gets mirrored)
- 6 and 9 rotate to each other, and
- the rest of the numbers do not rotate to any other number and become invalid.

Given an integer n, return the number of good integers in the range [1, n].

Solution. Given n. Let $f(pos, bound, diff)$ be the number of good numbers satisfying

1. Only consider posth digit and pos starts from left, which means 0th digit is the highest digit. And we assume the first $pos-1$ digits are fixed
2. If digits in $[0, pos-1]$ are first pos digits of n, then $bound$ is `true`

3. If digits in $[0, pos-1]$ has at least one 2/5/6/9, then $diff$ is `true`

Therefore the answer is $f(0, true, false)$, and the transition formula is

$$f(pos, bound, diff) = \sum f(pos+1, bound', diff')$$

- $bound'$ is true iff $bound$ is true and the digit we choose is the posth digit of n
- $diff'$ is true iff $diff$ is true or we chose 2/5/6/9

□

2 Trick and Bit

2.1 Bit operation

Problem 2.1.1 (Leetcode: Missing Two LCCI). You are given an array with all the numbers from 1 to N appearing exactly once, except for two number that is missing. How can you find the missing number in $O(N)$ time and $O(1)$ space?

You can return the missing numbers in any order.

Input	Output
`[1]`	`[2,3]`
`[2,3]`	`[1,4]`

```
nums.length <= 30000
```

Solution. Suppose the missing two numbers are x_1 and x_2, and if we add $1, \ldots, N$ to the end of the array A, then $x = \bigoplus A = x_1 \oplus x_2$.

By `x&-x` we can get the lowest bit of x, assume it's in lth bit. Then we can assume x_1's lth bit is 0, and x_2's lth bit is 1, and we can partition A into A_1 and A_2 by whether the elements' lth bit is 1, then $\bigoplus A_1 = x_1$ and $\bigoplus A_2 = x_2$ □