宫水三叶的刷题日记

数位 DP

Author : 宫水三叶
Date : 2021/10/07
QQ Group: 703311589
WeChat : oaoaya

> **🍭 更多精彩内容，欢迎关注：公众号 / Github / LeetCode / 知乎 **

噔噔噔噔，这是公众号「宫水三叶的刷题日记」的原创专题「数位 DP」合集。

本合集更新时间为 2021-10-07，大概每 2-4 周会集中更新一次。关注公众号，后台回复「数位 DP」即可获取最新下载链接。

💡**下面介绍使用本合集的最佳使用实践：**

学习算法：

1. 打开在线目录（Github 版 & Gitee 版）；
2. 从侧边栏的类别目录找到「数位 DP」；
3. 按照「推荐指数」从大到小进行刷题，「推荐指数」相同，则按照「难度」从易到难进行刷题‘
4. 拿到题号之后，回到本合集进行检索。

维持熟练度：

1. 按照本合集「从上往下」进行刷题。

学习过程中遇到任何困难，欢迎加入「每日一题打卡 QQ 群：703311589」进行交流 🍭🍭🍭

> **🍭 更多精彩内容，欢迎关注：公众号 / Github / LeetCode / 知乎 **

题目描述

这是 LeetCode 上的 **600. 不含连续1的非负整数** ，难度为 **困难**。

Tag : 「数位 DP」

给定一个正整数 n，找出小于或等于 n 的非负整数中，其二进制表示不包含**连续的1**的个数。

示例 1:

输入: 5

输出: 5

解释:
下面是带有相应二进制表示的非负整数<= 5：
0 : 0
1 : 1
2 : 10
3 : 11
4 : 100
5 : 101
其中，只有整数3违反规则（有两个连续的1），其他5个满足规则。

说明: 1 <= n <= 109

数位 DP

这是一道典型的「数位 DP」题。

对于「数位 DP」题，都存在「询问 $[a,b]$（a 和 b 均为正整数，且 $a<b$）区间内符合条件的数值个数为多少」的一般形式，通常我们需要实现一个查询 $[0,x]$ 有多少合法数值的函数 `int dp(int x)`，然后应用「容斥原理」求解出 $[a,b]$ 的个数：$dp(b)-dp(a-1)$。

对于本题，虽然只需要求解 $[0,n]$ 范围内数的个数，但其实拓展到求 $[a,b]$ 区间个数的也不会增加难度。

具体的，对于「数位 DP」问题通常是「从高位到低位」的分情况讨论。

不失一般性的考虑数值 n 的某一位 cur 是如何被处理的：

1. 如果当前位 $cur=1$ 的话，由于我们需要满足「小于等于 n」的要求，因此如果该位填 0 的话，后面的低位填什么都是满足要求的，因此我们期望能够查表得出「长度为 $i+1$，且二进制位置 i 数值为 0 时」有多少合法数值，将其累加到答案中；与此同时，我们需要确保当前位选 1 是合法的，即我们需要记录上一位 $prev$ 是什么，确保 cur 和 $prev$ 不同时为 1。
2. 如果当前位 $cur=0$ 的话，我们只能选 0，并决策下一位。

当出现「当前位无法填入 cur」或者「决策到最低位」时，则完成了所有合法答案的统计。

至于流程 1 中的查表操作，我们可以使用 `static` 预处理出 `f` 数组，定义 $f[i][j]$ 为考虑二进制长度为 i，且最高位为 j（0 or 1）时的合法数个数（值不超过）。

PS. 值不超过的含义代表了不仅仅统计高位为 j 的情况。例如 $f[4][1]$ 代表长度为 4，最高为 1，其包含了 1xxx 和 0xxx 的合法数的个数。

> 注意：为了防止重复计数问题，我们在不失一般性的计算 $f[i][0]$ 和 $f[i][1]$ 时，既能采用诸如 $f[i][cur]+=f[i-1][prev]$ 的 “后向查找依赖” 的方式进行转移，也能采用 $f[i+1][cur]+=f[i][prev]$ “前向主动更新” 的方式进行转移。

不失一般性的考虑 $f[i][0]$ 和 $f[i][1]$ 能够更新哪些状态：

- 如果期望当前位填 0 的话，需要统计所有满足 $(0...)_2$ 形式的合法数值，当前位的低一位只能填 1（填 0 会出现重复计数，即需要忽略前导零的数值），此时有：$f[i+1][0]=f[i][1]$；
- 如果期望当前位填 1 的话，需要统计所有满足 $(1...)_2$ 和 $(0...)_2$ 形式的合法数值：
    - $(1...)_2$ 时，当前位的低一位只能填 0；此时有：$f[i+1][1]+=f[i][0]$；
    - $(0...)_2$ 时，当前位的低一位只能填 1；此时有：$f[i+1][1]+=f[i][1]$。

代码：

```Java
class Solution {
    static int N = 50;
    // f[i][j] 为考虑二进制长度为 i，而且最高位为 j（0 or 1）时的合法数个数（值不超过）
    // 如 f[2][1] 代表二进制长度为 2，且（值不超过）最高位为 1 的合法数的个数：10、01、00
    static int[][] f = new int[N][2];
    static {
        f[1][0] = 1; f[1][1] = 2;
        for (int i = 1; i < N - 1; i++) {
            f[i + 1][0] = f[i][1];
            f[i + 1][1] = f[i][0] + f[i][1];
        }
    }
    int getLen(int n) {
        for (int i = 31; i >= 0; i--) {
            if (((n >> i) & 1) == 1) return i;
        }
        return 0;
    }
    public int findIntegers(int n) {
        int len = getLen(n);
        int ans = 0, prev = 0;
        for (int i = len; i >= 0; i--) {
            // 当前位是 0 还是 1
            int cur = ((n >> i) & 1);
            // 如果当前位是 1，那么填 0 的话，后面随便填都符合，将方案数累加
            if (cur == 1) ans += f[i + 1][0];
            // 出现连续位为 1，分支结束，方案数被计算完
            if (prev == 1 && cur == 1) break;
            prev = cur;
            if (i == 0) ans++;
        }
        return ans;
    }
}
```

- 时间复杂度：由于我们预处理 `f` 数组的操作使用了 `static` 修饰（在跑样例数据前已经预处理完，且预处理结果被所有样例数据所共享），因此访问 `f` 数组是 $O(1)$ 的查表操作；统计答案的复杂度与二进制长度相关，复杂度为 $O(\log n)$。整体复杂度为 $O(\log n)$
- 空间复杂度：令 C 为预处理数值的大小，固定为 $50 * 2$，复杂度为 $O(C)$

💡**更新 Tips：本专题更新时间为 2021-10-07，大概每 2-4 周 集中更新一次。**

最新专题合集资料下载，可关注公众号「宫水三叶的刷题日记」，回台回复「数位 DP」获取下载链接。

觉得专题不错，可以请作者吃糖 🍭🍭🍭 ：

"给作者手机充个电"

YOLO 的赞赏码