题目地址(989. 数组形式的整数加法)

https://leetcode-cn.com/problems/add-to-array-form-of-integer/

入选理由

1. 简单题目，适合大家上手。

2. 之前力扣官方的每日一题，质量比较高

题目描述

对于非负整数 X 而言，X 的数组形式是每位数字按从左到右的顺序形成的数组。例如，如果 X = 1231，那么其数组形式为 [1,2,3,1]。

给定非负整数 X 的数组形式 A，返回整数 X+K 的数组形式。

示例 1：

输入：A = [1,2,0,0], K = 34
输出：[1,2,3,4]
解释：1200 + 34 = 1234
示例 2：

输入：A = [2,7,4], K = 181
输出：[4,5,5]
解释：274 + 181 = 455
示例 3：

输入：A = [2,1,5], K = 806
输出：[1,0,2,1]
解释：215 + 806 = 1021
示例 4：

输入：A = [9,9,9,9,9,9,9,9,9,9], K = 1
输出：[1,0,0,0,0,0,0,0,0,0,0]
解释：9999999999 + 1 = 10000000000

提示：

1 <= A.length <= 10000
0 <= A[i] <= 9
0 <= K <= 10000
如果 A.length > 1，那么 A[0] != 0

难度

- 简单

标签

- 数组

前置知识

- 数组的遍历

思路

如果你没做出来这道题，不妨先试试 66. 加一 ，那道题是这道题的简化版，即 K = 1 的特殊形式。

这道题的思路是 **从低位到高位计算，注意进位和边界处理。** 细节都在代码里。

为了简化判断，我将 carry（进位） 和 K 进行了统一处理，即 carry = carry + K

关键点

- 处理进位

代码

语言支持：Python3，CPP, Java

Python3 Code:

```
class Solution:
    def addToArrayForm(self, A: List[int], K: int) -> List[int]:
        carry = 0
        for i in range(len(A) - 1, -1, -1):
            A[i], carry = (carry + A[i] + K % 10) % 10, (carry + A[i] + K % 10) // 10
            K //= 10
        B = []
        # 如果全部加完还有进位，需要特殊处理。 比如 A = [2], K = 998
        carry = carry + K
        while carry:
            B = [(carry) % 10] + B
            carry //= 10
```

```
        return B + A
```

CPP Code:

```
class Solution {
public:
    vector<int> addToArrayForm(vector<int>& num, int k) {
        vector<int> res;
        int n = num.size();
        for(int i = n - 1; i >= 0; --i){
            int curSum = num[i] + k % 10;
            k /= 10;
            if(curSum >= 10){
                k++;
                curSum -=10;
            }
            res.push_back(curSum);
        }
        while(k > 0){
            res.push_back(k % 10);
            k /= 10;
        }
        reverse(res.begin(), res.end());
        return res;
    }
};
```

Java Code:

```
class Solution {
    public List<Integer> addToArrayForm(int[] num, int k) {
        //用LinkedList不断从头将位数和加入index 0
        List<Integer> res = new LinkedList<>();
        int n = num.length;
        for (int i = n - 1; i >= 0; i--) {
            //从末尾往前扫，加和取余的值
            res.add(0, (num[i] + k) % 10);
            //更新k存进位carry
            k = (num[i] + k) / 10;
        }

        // post-possing: 处理k位数大于num的情况剩下的部分
        // Time = O(log(k))
```

```
        while (k > 0) {
            res.add(0, k % 10);
            k /= 10;
        }

        return res;
    }
}
```

复杂度分析

令 N 为数组长度。

- 时间复杂度：$O(N + \max(0, K - N)^2)$。for 循环的复杂度为 n，while 循环的复杂度为 $O(\max(0, K - N)^2)$，这是因为在数组前方添加元素的复杂度为 $O(数组长度)$。而实际上我们完全可以使用链表来降低复杂度。只不过我们当前是讲解数组的，因此就没有使用。大家可以在学习完链表后使用链表优化。

- 空间复杂度：$O(\max(1, K - N))$

上一页 下一页