0066-Plus One

Description

Given a non-empty array of digits representing a non-negative integer, plus one to the integer.

The digits are stored such that the most significant digit is at the head of the list, and each element in the array contain a single digit.

You may assume the integer does not contain any leading zero, except the number 0 itself.

Example 1:

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

Explanation: The array represents the integer 123.
Example 2:

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

Explanation: The array represents the integer 4321.

Solution 1

- 第一次解这个问题的时候，想偷懒，把digits数组转为整型，然后加一，再反转为digits数组返回。但这种想法是行不通的，因为case中的数组长度可以很大，完全超过任何整型的精度。换句话说，这个题目假设的场景是超长整型该如何加一的问题。
- 下面的解法思路是遍历一遍数组，判断每一位是否需要发出进位和接收进位。
- 算法的实现基本满足要求，但程序结构并不优雅。
- 时间复杂度：O(N)
- 空间复杂度：O(N)，因为使用了额外的栈

```java
class Solution {
    public int[] plusOne(int[] digits) {
        Stack<Integer> stack = new Stack<>();
        if(digits[digits.length-1] != 9) {
            digits[digits.length-1]++;
            return digits;
        }

        boolean jinwei = true;
        for(int i = digits.length - 1; i >= 0; i--) {
            int d = digits[i];
            if(jinwei) {
                d++;
            }
            if(d == 10) {
                jinwei = true;
                stack.push(0);
            } else {
                jinwei = false;
                stack.push(d);
            }
        }
        if(jinwei) {
            stack.push(1);
        }
        int[] result = new int[stack.size()];
        for(int i = 0; i < result.length; i++) {
            result[i] = stack.pop();
        }
        return result;
    }
}
```