738-单调递增的数字

题述

738. 单调递增的数字

难度 中等 241

当且仅当每个相邻位数上的数字 `x` 和 `y` 满足 `x <= y` 时，我们称这个整数是**单调递增**的。

给定一个整数 `n` ，返回 *小于或等于* `n` *的最大数字，且数字呈* ***单调递增*** 。

示例 1:

```
输入: n = 10
输出: 9
```

示例 2:

```
输入: n = 1234
输出: 1234
```

示例 3:

```
输入: n = 332
输出: 299
```

提示:

- $0 <= n <= 10^9$

思路

暴力

贪心

题目要求小于等于N的最大单调递增的整数，那么拿一个两位的数字来举例。

例如：98，一旦出现strNum[i - 1] > strNum[i]的情况（非单调递增），首先想让strNum[i - 1]--，然后strNum[i]给为9，这样这个整数就是89，即小于98的最大的单调递增整数。

这一点如果想清楚了，这道题就好办了。

** 局部最优：遇到strNum[i - 1] > strNum[i]的情况，让strNum[i - 1]--，然后strNum[i]给为9，可以保证这两位变成最大单调递增整数**。

全局最优：得到小于等于N的最大单调递增的整数。

** 但这里局部最优推出全局最优，还需要其他条件，即遍历顺序，和标记从哪一位开始统一改成9**。

此时是从前向后遍历还是从后向前遍历呢？

从前向后遍历的话，遇到strNum[i - 1] > strNum[i]的情况，让strNum[i - 1]减一，但此时如果strNum[i - 1]减一了，可能又小于strNum[i - 2]。

这么说有点抽象，举个例子，数字：332，从前向后遍历的话，那么就把变成了329，此时2又小于了第一位的3了，真正的结果应该是299。

** 所以从前后向遍历会改变已经遍历过的结果！**

那么从后向前遍历，就可以重复利用上次比较得出的结果了，从后向前遍历332的数值变化为：332 -> 329 -> 299

确定了遍历顺序之后，那么此时局部最优就可以推出全局，找不出反例，试试贪心。

题解

暴力

```
class Solution {
public:
    // 暴力
    bool checkNum(int num)
    {
        int max = 10;
        while(num)
        {
            int t = num % 10;
            if(max >= t) max = t;
            else return false;
            num = num / 10;
        }
        return true;
    }
    int monotoneIncreasingDigits(int n)
    {
        for(int i = n; i > 0; i--)
        {
            if(checkNum(i)) return i;
        }
        return 0;
    }
};
```

超时！

贪心

```
class Solution {
public:
    int monotoneIncreasingDigits(int n)
    {
        //贪心
        //局部最优 遇到strNum[i - 1] > strNum[i]的情况，让strNum[i - 1]--，然后
strNum[i]给为9，可以保证这两位变成最大单调递增整数。
        //从后向前遍历
        string strNum = to_string(n);
        //flag 用来标记赋值9从哪里开始
        //设置为默认值，为了防止第二个for在flag没有被赋值的情况下执行
        int flag = strNum.size();
        for(int i = strNum.size()-1;i>0;i--)
        {
            if(strNum[i-1] > strNum[i])
            {
                flag = i;
                strNum[i-1]--;
            }
        }
        for(int i = flag;i<strNum.size();i++)
        {
            strNum[i]='9';
        }
        return stoi(strNum);
    }
};
```

Python

```python
class Solution:
    def monotoneIncreasingDigits(self, n: int) -> int:
        a = list(str(n))
        for i in range(len(a)-1,0,-1):
            if int(a[i]) < int(a[i-1]):
                a[i-1] = str(int(a[i-1]) - 1)
                a[i:] = '9' * (len(a) - i)  #python不需要设置flag值，直接按长度给9即可
        return int("".join(a))
```

提交结果	执行用时	内存消耗	语言	提交时间
通过	40 ms	14.8 MB	Python3	2022/03/27 19:32
解答错误	N/A	N/A	Python3	2022/03/27 19:32
通过	4 ms	5.8 MB	C++	2022/03/27 19:29
超出时间限制	N/A	N/A	C++	2022/03/27 19:24

思考

想清楚个例，例如98，一旦出现strNum[i - 1] > strNum[i]的情况（非单调递增），首先想让strNum[i - 1]减一，strNum[i]赋值9，这样这个整数就是89。

考虑遍历顺序，只有从后向前遍历才能重复利用上次比较的结果。