Stack

402. Remove K Digits

Greedy; Stack

> Given a non-negative integer num represented as a string, remove k digits from the number so that the new number is the smallest possible.

解法一：每次去除首个比右侧元素大的元素

*时间复杂度：O(k*n)*

参考

这是一个贪心算法，每次寻找整个数列中第一个“峰值”，即比右侧元素大的值，将其去除。因为我们想要得到的数列是尽可能小的，所以开头的元素要尽可能小，去除了最前面的峰值可以最大化地减小这个数，所以每次操作得到的都是可能得到的最小的数。这个算法时间复杂度较高。

解法二：建栈将元素尽可能地升序排列

时间复杂度：O(n)

参考

使用栈来存放需要保留的字符，每次字符尝试进栈，在剩余字符还充足的前提情况下，将所有大于这个数字的字符出栈，再将这个字符进栈，即可得到最佳解。最后需要考虑字符相等的边界条件。这样栈里面的元素始终是升序排列的，除非剩余元素不够我们无法再进行删除。

```
class Solution {
    public String removeKdigits(String num, int k) {
        Stack<Character> nStack = new Stack<>();
        int len = num.length();
        int d = len - k;
        int c = 0;
        for (int i = 0; i < len; ++i)
        {
            //Need enough digits to reach d
               while (!nStack.empty() && c + len - i > d &&
nStack.peek() > num.charAt(i))
```

```
                {
                    nStack.pop();
                    c--;
                }
                nStack.push(num.charAt(i));
                c++;
            }
            //Deal with edge cases of same numbers
            while (c > d)
            {
                nStack.pop();
                c--;
            }
            StringBuilder sb = new StringBuilder();
            while (!nStack.empty())
                sb.append(nStack.pop());
            sb.reverse();
            while(sb.length() > 1 && sb.charAt(0)=='0')
                sb.deleteCharAt(0);
            if (sb.length() == 0)
                return "0";
            return sb.toString();
        }
    }
```

---

901. Online Stock Span

Stack

> Write a class StockSpanner which collects daily price quotes for some stock, and returns the span of that stock's price for the current day.
>
> The span of the stock's price today is defined as the maximum number of consecutive days (starting from today and going backwards) for which the price of the stock was less than or equal to today's price.
>
> For example, if the price of a stock over the next 7 days were [100, 80, 60, 70, 60, 75, 85], then the stock spans would be [1, 1, 1, 2, 1, 4, 6].

解法一：Stack，合并结果，降序排列

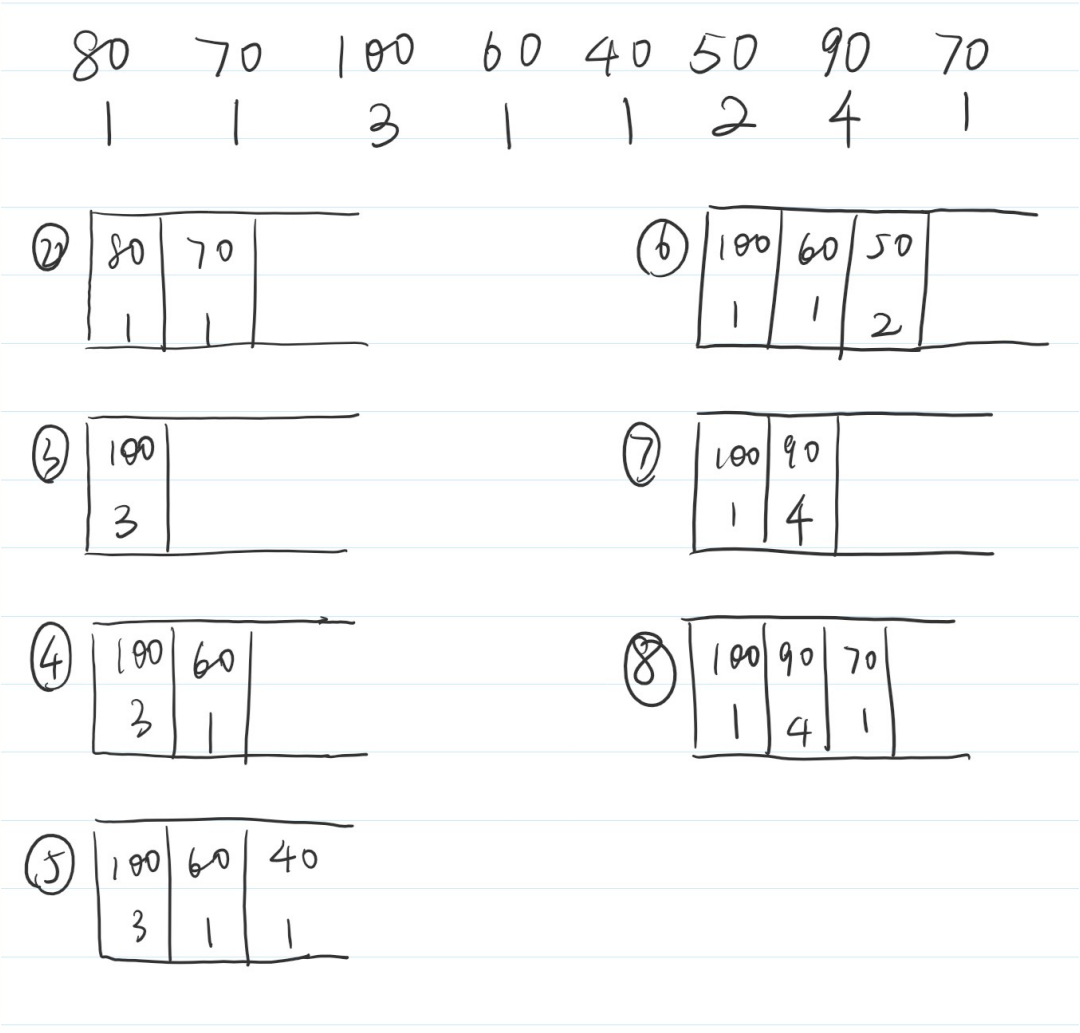
参考

这道题应该使用栈结构，因为每个读入的元素都从今日开始向后考虑小于今日价格的最长连续天数。题目表述的不是很清楚，但是从样例结果来看，这个连续天数一定需要包含今日的数量。

例如，读入85，需要向后考虑发现75小于85，其对应权重是4，紧接着的60，70和60也小于85，但是**他们的权重已经包含在75的4里面了**；相反之后的80也小于85但是比75大，因此80的权重也需要加上。所以得到1+4+1=6.

为了避免重复的线性查找，每次读入之后我们只需要保存一些特定的数值和权重加和即可。如果这个数字的权重可以包含在别的数字的权重里，我们没有必要将其保留在数组中。

引入栈的结构之后，每次入栈前先将栈内所有价值比当前元素价值的元素pop出，并将权重加和，形成一个新的<价值，权重>组合元素，放入栈内。如果没有则直接放入栈内，权重为1。这时的权重即为题目所需答案。栈内所有元素都是**降序排列**的，自然保证了没有重复的查找。




```
class StockSpanner {
```

```
    public StockSpanner() {
        stk = new Stack<Integer> ();
        weights = new Stack<Integer> ();
    }
    Stack<Integer> stk;
    Stack<Integer> weights;

    public int next(int price) {
        int w = 1;
        while (!stk.isEmpty() && stk.peek() <= price)
        {
            w += weights.pop();
            stk.pop();
        }
        stk.push(price);
        weights.push(w);
        return w;
    }
}
```
