Remove K Digits

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.

Note:

- The length of *num* is less than 10^5 and will be $\geq k$.
- The given *num* does not contain any leading zero.

Example 1:

```
Input: num = "1432219", k = 3
Output: "1219"
Explanation: Remove the three digits 4, 3, and 2 to form the new number 1219 which
is the smallest.
```

Example 2:

```
Input: num = "10200", k = 1
Output: "200"
Explanation: Remove the leading 1 and the number is 200. Note that the output must
not contain leading zeroes.
```

Example 3:

```
Input: num = "10", k = 2
Output: "0"
Explanation: Remove all the digits from the number and it is left with nothing whic
h is 0.
```

Solution 1

O(n) Solution

Go through the digits from left to right, remove previous digits if that makes the number smaller (and if we still have to remove digits). Almost the same as the `prep` function from my solution to an earlier problem, which did basically the same except it **maximized** the number.

```
def removeKdigits(self, num, k):
    out = []
    for d in num:
        while k and out and out[-1] > d:
            out.pop()
            k -= 1
        out.append(d)
    return ''.join(out[:-k or None]).lstrip('0') or '0'
```

Regex Solution

k times remove the leftmost digit followed by a smaller digit (or remove the last digit). Didn't think this would be fast enough, but it is :-)

```
def removeKdigits(self, num, k):
    sub = re.compile('1[0]|2[01]|3[0-2]|4[0-3]|5[0-4]|6[0-5]|7[0-6]|8[0-7]|9[0-8]
|.$').sub
    for _ in range(k):
        num = sub(lambda m: m.group()[1:], num, 1)
    return num.lstrip('0') or '0'
```

written by StefanPochmann original link here

Solution 2

```
public class Solution {
    public String removeKdigits(String num, int k) {
        int digits = num.length() - k;
        char[] stk = new char[num.length()];
        int top = 0;
        // k keeps track of how many characters we can remove
        // if the previous character in stk is larger than the current one
        // then removing it will get a smaller number
        // but we can only do so when k is larger than 0
        for (int i = 0; i < num.length(); ++i) {
            char c = num.charAt(i);
            while (top > 0 && stk[top-1] > c && k > 0) {
                top -= 1;
                k -= 1;
            }
            stk[top++] = c;
        }
        // find the index of first non-zero digit
        int idx = 0;
        while (idx < digits && stk[idx] == '0') idx++;
        return idx == digits? "0": new String(stk, idx, digits - idx);
    }
}
```

written by keyulai92gmail.com original link here

Solution 3

k is the number of char needs dropping, 'keep' is the number of char that should keep.
We need to remove the beginning zeroes, and finally check if the result is empty of not.

```
class Solution {
public:
    string removeKdigits(string num, int k) {
        string res = "";
        int n = num.size(), keep = n - k;
        for (char c : num) {
            while (k && res.size() && res.back() > c) {
                res.pop_back();
                --k;
            }
            res.push_back(c);
        }
        res.resize(keep);
        while (!res.empty() && res[0] == '0') res.erase(res.begin());
        return res.empty() ? "0" : res;
    }
};
```

written by grandyang original link here