【华为OD机考 统一考试机试C卷】找最小数（C++ Java JavaScript Python C语言）

华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷

目前在考C卷，经过两个月的收集整理，C卷真题已基本整理完毕
抽到原题的概率为2/3到3/3，也就是最少抽到两道原题。请注意：大家刷完C卷真题，最好要把B卷的真题刷一下，因为C卷的部分真题来自B卷。
另外订阅专栏还可以联系笔者开通在线 OJ 进行刷题，提高刷题效率。
真题目录：华为OD机考机试 真题目录（C卷 + D卷 + B卷 + A卷） + 考点说明
专栏：2023华为OD机试(B卷+C卷+D卷)（C++JavaJSPy）
华为OD面试真题精选：华为OD面试真题精选
在线OJ：点击立即刷题，模拟真实机考环境

题目描述

给一个正整数NUM1，计算出新正整数NUM2，NUM2为NUM1中移除N位数字后的结果，需要使得NUM2的值最小。

输入描述

1.输入的第一行为一个字符串，字符串由0-9字符组成，记录正整数NUM1，NUM1长度小于32。
2.输入的第二行为需要移除的数字的个数，小于NUM1长度。

输出描述

输出一个数字字符串，记录最小值NUM2。

用例

输入

```
2615371
4
```

输出

```
131
```

解题思路

原题：https://leetcode.cn/problems/remove-k-digits/solutions/

维护一个单调递增的栈来实现移除数字

1. 初始化一个空栈 `stack`，用于存储需要保留的数字。
2. 遍历输入的正整数 NUM1 中的每个字符。
3. 对于当前字符，检查栈顶元素是否大于当前字符，如果是，则出栈并减少需要移除的数字个数。这样可以确保移除的数字使得新正整数 NUM2 的值最小。
4. 将当前字符入栈。
5. 遍历完成后，如果仍有需要移除的数字个数，从栈顶开始移除剩余的数字。
6. 将栈中的字符连接成一个字符串，去除前导零，输出结果。如果结果为空，则输出 “0”。

C++

```cpp
#include <iostream>
#include <string>
#include <vector>

int main() {
    // 读取输入的正整数 NUM1 和需要移除的数字个数
    std::string num;
    int k;
    std::cin >> num >> k;

    // 使用一个 vector 作为栈来存储结果
    std::vector<char> stack;

    // 遍历输入的数字字符串
    for (char i : num) {
        // 当栈非空、k 大于 0 且栈顶元素大于当前数字时，弹出栈顶元素并减小 k
        while (!stack.empty() && k > 0 && stack.back() > i) {
            k--;
            stack.pop_back();
        }
```

```
        // 将当前数字压入栈中
        stack.push_back(i);
    }

    // 构建结果字符串，移除多余的 k 个数字
    std::string result(stack.begin(), stack.end() - k);
    // 删除结果字符串中的前导零
    result.erase(0, result.find_first_not_of('0'));
    // 如果结果为空，则输出 "0"
    if (result.empty()) {
        result = "0";
    }

    // 输出结果
    std::cout << result << std::endl;

    return 0;
}
```

java

```
import java.util.Scanner;
import java.util.Stack;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // 读取输入的正整数 NUM1 和需要移除的数字个数
        String num = scanner.next();
        int k = scanner.nextInt();

        Stack<Character> stack = new Stack<>();

        // 遍历 NUM1 中的每个字符
        for (char c : num.toCharArray()) {
            // 当栈非空、需要移除的数字个数大于 0 且栈顶元素大于当前字符时，出栈并减少需要移除的数字个数
            while (!stack.isEmpty() && k > 0 && stack.peek() > c) {
```

```
                stack.pop();
                k--;
            }
            stack.push(c);
        }

        // 移除剩余的数字
        while (k > 0) {
            stack.pop();
            k--;
        }

        // 构建结果字符串
        StringBuilder result = new StringBuilder();
        boolean leadingZero = true;
        for (char c : stack) {
            if (c == '0' && leadingZero) {
                continue;
            }
            leadingZero = false;
            result.append(c);
        }

        // 输出结果字符串，如果为空则输出 "0"
        System.out.println(result.length() == 0 ? "0" : result.toString());
    }
}
```

javaScript

```
const readline = require('readline');

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});
```

```
rl.on('line', (num) => {
  rl.on('line', (k) => {
     const stack = [];

    for (const i of num) {
      while (stack.length > 0 && k > 0 && stack[stack.length - 1] > i) {
        k -= 1;
        stack.pop();
      }
      stack.push(i);
    }

    console.log((stack.slice(0, stack.length - k).join('').replace(/^0+/, '') || '0'));
    rl.close();
  });
});
```

python

```
# 读取输入的正整数 NUM1 和需要移除的数字个数
num = input()
k = int(input())

# 初始化一个栈，用于存储需要保留的数字
stack = []

# 遍历 NUM1 中的每个字符
for i in num:
    # 当栈非空、需要移除的数字个数大于 0 且栈顶元素大于当前字符时
    # 出栈并减少需要移除的数字个数
    while stack and k and stack[-1] > i:
        k -= 1
        stack.pop()
    # 将当前字符入栈
    stack.append(i)
```

```
19
20
21
22
23
```

```python
# 输出结果字符串，移除剩余的数字，并去除前导零，如果为空则输出 "0"
print(''.join(stack[:len(stack) - k]).lstrip('0') or "0")
```

C语言

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main() {
    char num[32]; // 存储输入的正整数NUM1，长度小于32
    int k;        // 需要移除的数字个数
    scanf("%s %d", num, &k); // 读取NUM1和k

    char stack[32]; // 使用一个字符数组作为栈来存储结果
    int top = -1;   // 栈顶指针，初始为-1表示空栈

    // 遍历输入的数字字符串
    for (int i = 0; i < strlen(num); i++) {
        char current = num[i];
        // 当栈非空、k大于0且栈顶元素大于当前数字时，弹出栈顶元素并减小k
        while (top >= 0 && k > 0 && stack[top] > current) {
            top--;
            k--;
        }
        // 将当前数字压入栈中
        stack[++top] = current;
    }

    // 移除多余的k个数字
    top -= k;

    // 构建结果字符串
    char result[32];
    for (int i = 0; i <= top; i++) {
        result[i] = stack[i];
    }
```

```
    result[top + 1] = '\0'; // 添加字符串结束符

    // 删除结果字符串中的前导零
    char *start = result;
    while (*start == '0') {
        start++;
    }
    if (*start == '\0') { // 如果所有数字都被移除，输出"0"
        printf("0\n");
    } else {
        printf("%s\n", start); // 输出结果
    }

    return 0;
}
```

文章目录


机考真题
华为OD
No No No
CSDN @算法大师