31、找最小数，考点 or 实现——数据结构/栈

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
说明	无

题目解析

本题和算法 - 移掉 K 位数字_伏城之外的博客-CSDN博客

相同，但是本题题目描述和用例不清晰，建议看上面leetco de算法题后再来写本题

Java算法源码

```java
import java.util.LinkedList;
import java.util.Scanner;

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

    String num1 = sc.nextLine();
    int count = Integer.parseInt(sc.nextLine());

    System.out.println(getResult(num1, count));
  }

  private static String getResult(String num1, int removeCount) {
    if (num1.length() == removeCount) return "0";

    int remainCount = num1.length() - removeCount;

    LinkedList<Character> stack = new LinkedList<>();
    for (int i = 0; i < num1.length(); i++) {
      while (stack.size() > 0 && removeCount > 0 && stack.getLast() > num1.charAt(i)) {
        stack.removeLast();
        removeCount--;
      }
      stack.add(num1.charAt(i));
    }

    while (stack.size() > remainCount) {
      stack.removeLast();
    }

    while (stack.getFirst() == '0' && stack.size() != 1) {
      stack.removeFirst();
    }

    StringBuilder sb = new StringBuilder();
    for (Character c : stack) sb.append(c);
    return sb.toString();
  }
}
```

JS算法源码

```
/* JavaScript Node ACM模式 控制台输入获取 */
const readline = require("readline");

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

const lines = [];
rl.on("line", (line) => {
  lines.push(line);

  if (lines.length === 2) {
    const arr = lines[0].split("").map((ele) => parseInt(ele));
    const removeCount = lines[1];

    console.log(getMinValue(arr, removeCount));
  }
});

function getMinValue(arr, removeCount) {
  if (arr.length === removeCount) return "0";

  let remainCount = arr.length - removeCount;

  let stack = [];
```

```
  for (let i = 0; i < arr.length; i++) {
    while (
      stack.length > 0 &&
      removeCount > 0 &&
      stack[stack.length - 1] > arr[i]
    ) {
      stack.pop();
      removeCount--;
    }
    stack.push(arr[i]);
  }

  while (stack.length > remainCount) {
    stack.pop();
  }

  while (stack[0] === 0 && stack.length !== 1) {
    stack.shift();
  }

  return stack.join("");
}
```

Python算法源码

```python
# 输入获取
num1 = input()
removeCount = int(input())


# 算法入口
def getResult(num1, removeCount):
    if len(num1) == removeCount:
        return "0"

    remainCount = len(num1) - removeCount

    stack = []
    for i in range(len(num1)):
        while len(stack) > 0 and removeCount > 0 and stack[-1] > num1[i]:
            stack.pop()
            removeCount -= 1
        stack.append(num1[i])

    while len(stack) > remainCount:
        stack.pop()

    while stack[0] == '0' and len(stack) != 1:
        stack.pop(0)

    return "".join(stack)

```

```python

# 算法调用
print(getResult(num1, removeCount))
```