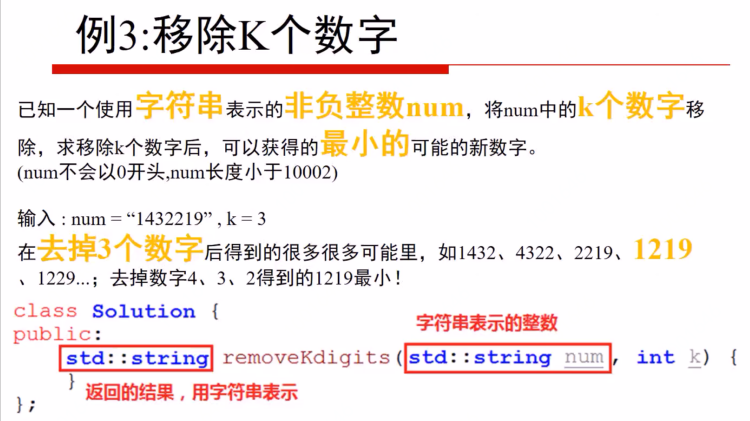
Leetcode\_402\_RemoveKDigits\_移除K个数字

# Leetcode\_402\_RemoveKDigits\_移除K个数字

## 题目介绍

\* 难度：Medium

\* <https://leetcode.com/problems/remove-k-digits/description/>



\* 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 10002 and will be ≥ 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 which is 0.

## 思路分析

\* 思路分析：利用的容器Deque双端队列，首先模拟栈的使用，

\* 由于需要从尾端顺序获取剩下的值，利用的Deque的双端特性；

\* 算法思想：遍历每一个元素，当该元素比栈顶元素小，就删除栈顶元素，

\* 并while循环判断栈顶，直到栈为空或栈顶元素比该值小；

\* while循环之后，若该元素不为0或者栈不为空，则将该元素入栈；

\* 因为栈空时，0不能入栈；

\* 这时候，如栈空且该元素为0，则不入栈，k值并没有减1，这就是神奇之处,

\* 如10069，k-3；:1入栈，1出栈，k-1=2；然后0不入栈，栈空，第二个0来了，啥也不做，

\* 6入栈，然后9来了，9入栈，由于k=2，需要删除9和6，最后栈为空，故返回0；

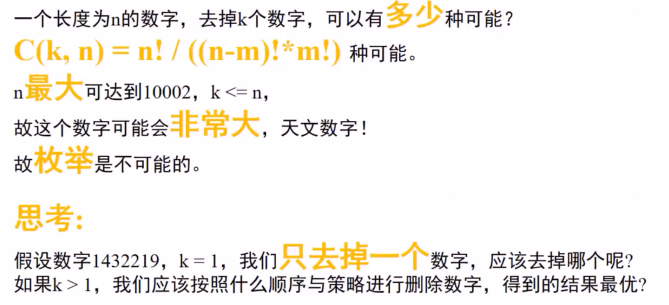
\* 10200，k=1；1入栈，然后0来了，1出栈，k-1=0；0不入栈，k=0了，直接把2/0/0入栈即可；

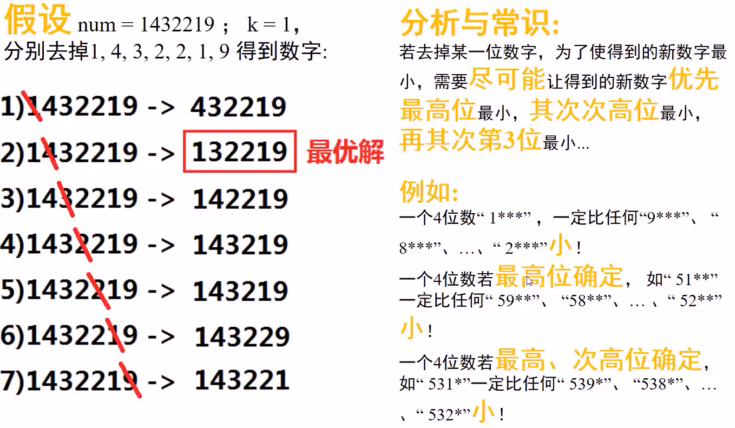
\* 最后返回200。

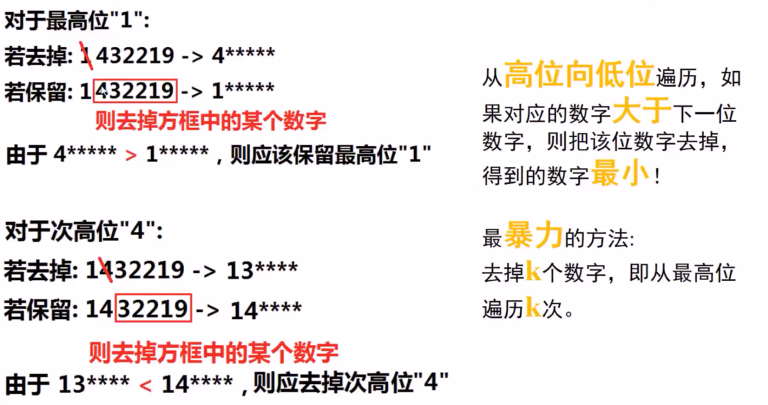
\* 一开始，就是纠结在0，这里设计很巧妙，

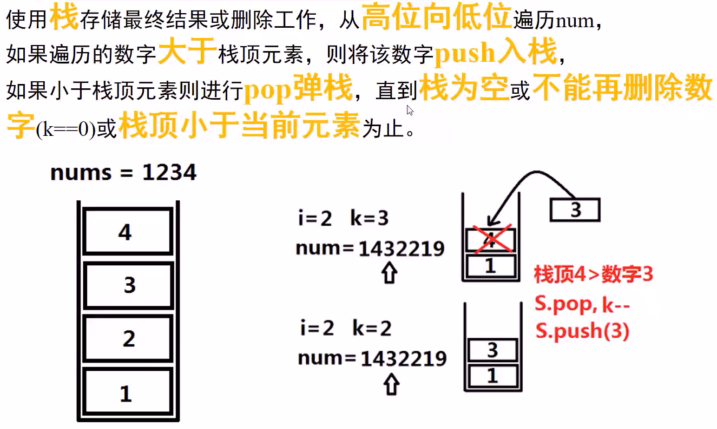
\* 注意理解困难之处：删除k个数字，最后剩下的数字位数，不一定必须是（总位数-k）；出现开头为0情况，

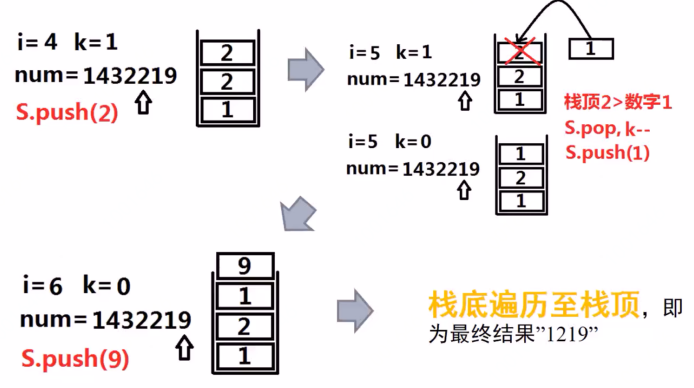
\* 最后结果位数可以小于（总位数-k）。



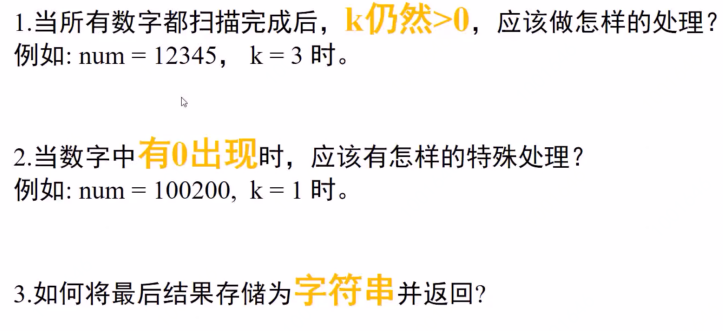








**最关键的三个问题：**



问题1：当扫描完毕，k仍然大于0，此时需要再重新扫描一遍；

利用双端队列实现。

问题2：0不能作为开头。

问题3：利用双端队列的特点从另一端拼接输出即可。

## Java代码

public String **removeKdigits**(String nums, int k) {

if(nums == null || nums.length() == 0) return "0";

if(k == 0) return nums;

char[] numChar = nums.toCharArray();

ArrayDeque<Character> deque = new ArrayDeque<Character>();

for(int i = 0;i < numChar.length;i++){

//注意这里一定是while循环，不是if，保证当前值比前面的数都小

while(!deque.isEmpty() && k > 0 && numChar[i] < deque.peekFirst()) {

deque.pop();//删除栈顶元素

k--;

}

if(!deque.isEmpty()||numChar[i] != '0') deque.push(numChar[i]);//将新数字入栈

}

//至此，栈中已经是排序的情况；若k还大于0，直接删除后面大值即可

while(!deque.isEmpty()&& k>0 ){

deque.pop();

k--;

}

StringBuilder sb = new StringBuilder();

//从队列尾端取出队列中的值

while(!deque.isEmpty()) sb.append(deque.removeLast());

String result = sb.toString();

return "".equals(result)?"0":result;//关键当队列为空时，返回"0"

}