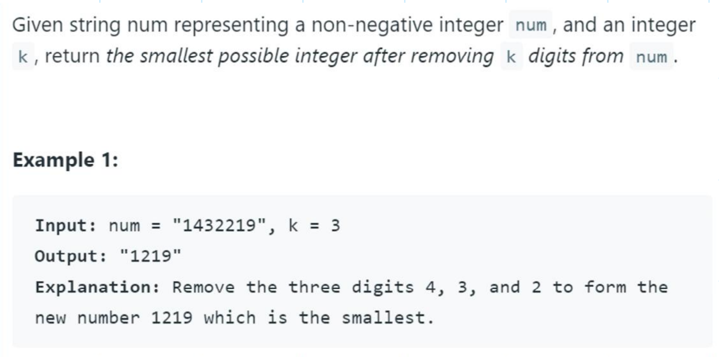
402. Remove K Digits

18 February 2022 09:51 AM



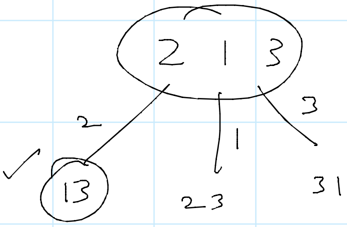
Given string num representing a non-negative integer `num`, and an integer `k`, return *the smallest possible integer after removing* `k` *digits from* `num`.

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.
```

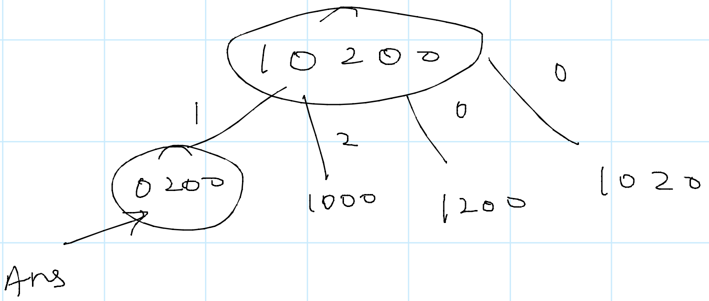
nums = 213

k = 2



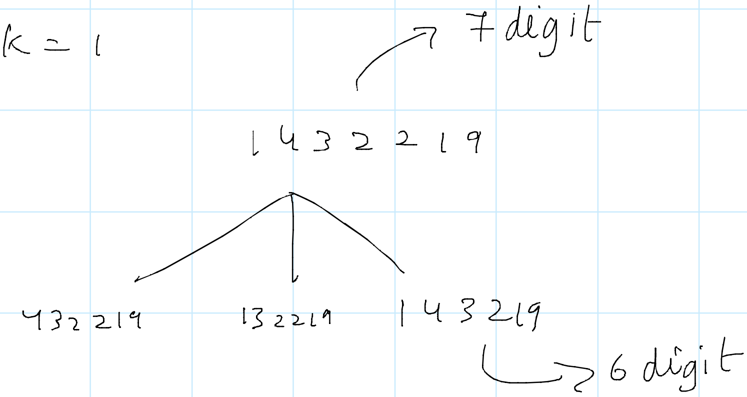
nums = 10200

k = 1



nums = 1432219

k = 1



Observation ⇒ if you remove 1 then 4 is bigger which make it largest element.

Now if you remove 4 then now 3 is less larger than 4 but still larger than 1.

Keep observing the most significant digit

$1\ \cancel{4}\ 3\ 2\ 2\ 2\ 3\ 1 \qquad k=1$

↓

$1\ \cancel{3}\ 2\ 2\ 2\ 3\ 1 \qquad k=2$

↓

$1\ 2\ 2\ 2\ \cancel{3}\ 1 \qquad k=3$

↓

$\boxed{1\ 2\ 2\ 1}$ for $k=3$ Ans

num → $1\ 4\ 3\ 2\ 2\ 2\ 3\ 1$

$K = \cancel{3}\ \cancel{2}\ \cancel{1}\ 0$

$1\ \cancel{4}\ \cancel{3}\ 2\ 2\ 2\ \cancel{3}\ 1$ ds

when 3 comes to the ds the previous element is 4 so you delete the when 3 comes to the ds the previous element is 4 so you delete the

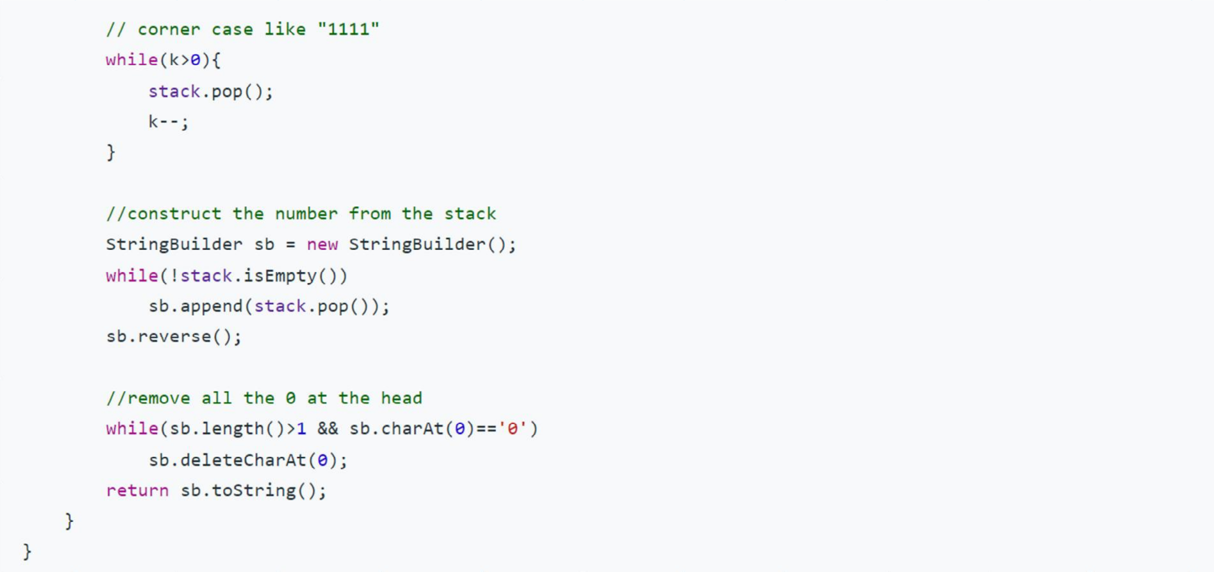
element and then decrease k

Same for 3 and 2

Same for 3 and 1

and K becomes 0.

that the find ans.



```
        // corner case like "1111"
        while(k>0){
            stack.pop();
            k--;
        }

        //construct the number from the stack
        StringBuilder sb = new StringBuilder();
        while(!stack.isEmpty())
            sb.append(stack.pop());
        sb.reverse();

        //remove all the 0 at the head
        while(sb.length()>1 && sb.charAt(0)=='0')
            sb.deleteCharAt(0);
        return sb.toString();
    }
}
```