402. Remove K Digits

Medium 4420 187 Add to List Share

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:

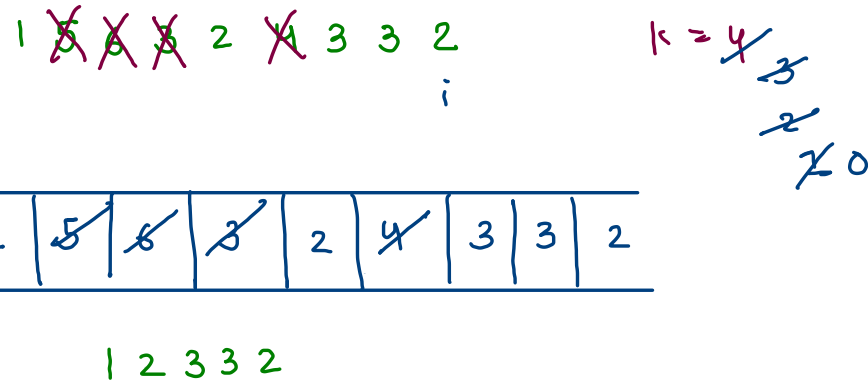
```
Input: num = "1432219", k = 3
Output: "1219"
Explanation: Remove the three digits 4, 3, and 2 to form the
new number 1219 which is the smallest.
```

Example 2:

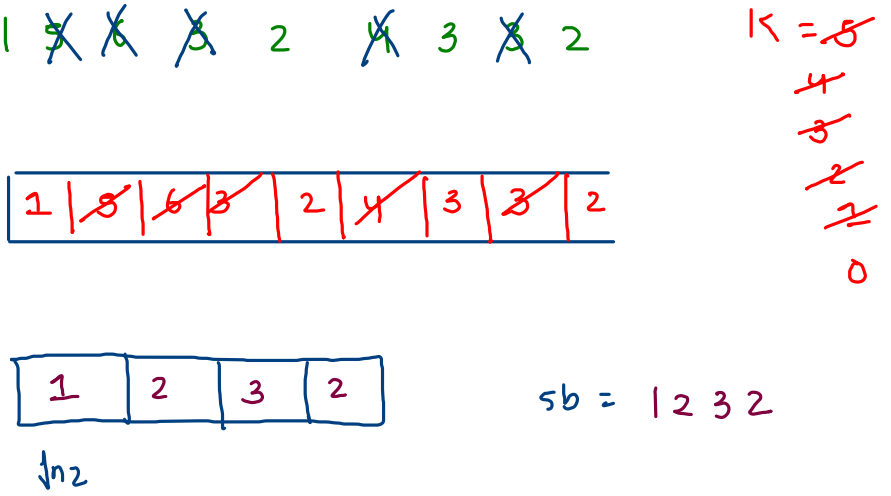
```
Input: num = "10200", k = 1
Output: "200"
Explanation: Remove the leading 1 and the number is 200. Note
that the output must not contain leading zeroes.
```

Example 3:

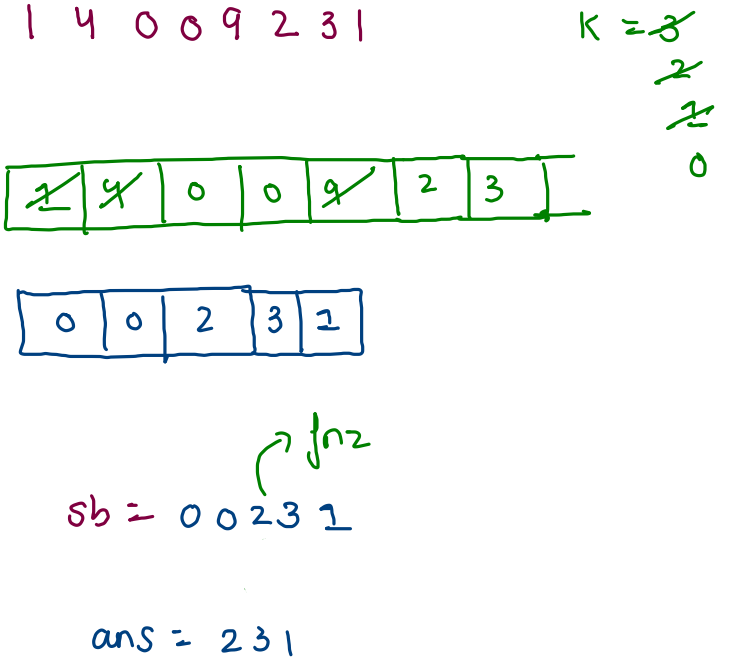
```
Input: num = "10", k = 2
Output: "0"
```



```
for(int i=0; i < num.length();i++) {
    char ch = num.charAt(i);

    while(st.size() > 0 && k > 0 && st.peek() > ch) {
        st.pop();
        k--;
    }

    st.push(ch);
}

while(k > 0) {
    st.pop();
    k--;
}

char[]arr = new char[st.size()];
int idx = arr.length-1;

while(st.size() > 0) {
    arr[idx--] = st.pop();
}


StringBuilder sb = new StringBuilder("");
int fnz = arr.length-1;

for(int i=0; i < arr.length;i++) {
    sb.append(arr[i]);

    if(fnz == arr.length-1 && arr[i] != '0') {
        fnz = i;
    }
}

return sb.toString().substring(fnz);
```



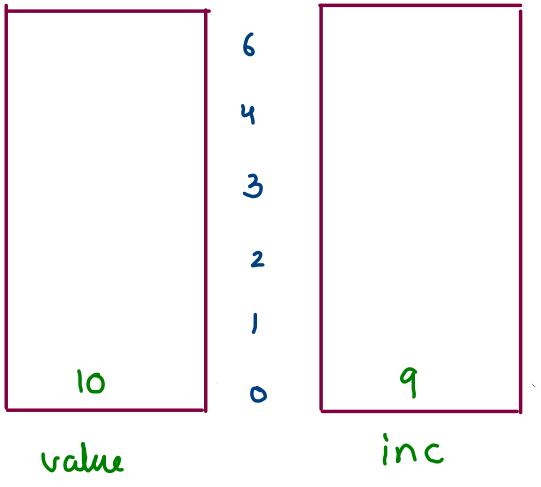
```
for(int i=0; i < num.length();i++) {
    char ch = num.charAt(i);

    while(st.size() > 0 && k > 0 && st.peek() > ch) {
        st.pop();
        k--;
    }

    st.push(ch);
}

while(k > 0) {
    st.pop();
    k--;
}
char[]arr = new char[st.size()];
int idx = arr.length-1;

while(st.size() > 0) {
    arr[idx--] = st.pop();
}


StringBuilder sb = new StringBuilder("");
int fnz = arr.length-1;

for(int i=0; i < arr.length;i++) {
    sb.append(arr[i]);

    if(fnz == arr.length-1 && arr[i] != '0') {
        fnz = i;
    }
}

return sb.toString().substring(fnz);
```



1381. Design a Stack With Increment Operation

maxSize = 6

```
class CustomStack {

    public CustomStack(int maxSize) {

    }

    public void push(int x) {

    }

    public int pop() {

    }

    public void increment(int k, int val) {

    }
}
```



push(10)
push(20)
push(5)
push(12)
push(8)
increment(2, 3)
pop() → 8
increment(4, 6)
pop() → 18
pop() → 11
pop() → 29

inc[k-1] += val

push(10)
push(20)
push(5)
push(12)
push(8)
increment(2, 3)
pop() → 8
increment(4, 6)
pop() → 18
pop() → 11
pop() → 29
pop() → 19

```
public void push(int x) {
    if(tos + 1 == value.length) {
        //overflow
        return;
    }

    tos++;
    value[tos] = x;
}
```

```
public int pop() {
    if(tos == -1) {
        //underflow
        return -1;
    }

    int rv = value[tos] + increment[tos];
    int inc = increment[tos];

    value[tos] = increment[tos] = 0;
    tos--;

    if(tos >= 0)
    increment[tos] += inc;

    return rv;
}
```

```
public void increment(int k, int inc) {
    if(tos == -1) {
        return;
    }

    if(k > tos+1) {
        increment[tos] += inc;
    }
    else {
        increment[k-1] += inc;
    }
}
```