Java solution using Stack with comments

54

dwaijam
Reputation: 54

```
public String removeDuplicateLetters(String sr) {

    int[] res = new int[26]; //will contain number of occurences of character (i+'a')
    boolean[] visited = new boolean[26]; //will contain if character (i+'a') is present in curren
    char[] ch = sr.toCharArray();
    for(char c: ch){  //count number of occurences of character
        res[c-'a']++;
    }
    Stack<Character> st = new Stack<>(); // answer stack
    int index;
    for(char s:ch){
        index= s-'a';
        res[index]--;   //decrement number of characters remaining in the string to be analysed
        if(visited[index]) //if character is already present in stack, dont bother
            continue;
        //if current character is smaller than last character in stack which occurs later in the
        //it can be removed and  added later e.g stack = bc remaining string abc then a can pop b
```

```
        while(!st.isEmpty() && s<st.peek() && res[st.peek()-'a']!=0){
            visited[st.pop()-'a']=false;
        }
        st.push(s); //add current character and mark it as visited
        visited[index]=true;
    }

    StringBuilder sb = new StringBuilder();
    //pop character from stack and build answer string from back
    while(!st.isEmpty()){
        sb.insert(0,st.pop());
    }
    return sb.toString();
}
```

3

SergeyTachenov
Reputation: ★ 349

Shouldn't have used `Stack` because it's obsolete, `ArrayDeque` is better. But in this case, might as well just use the resulting `StringBuilder` as a stack! No boxing required, and one loop less.

12

A

addy_boy
Reputation: ★ 12

@dwaijam .. Thanks for sharing this solution. I had not been able to wrap my head around the exact requirement of the solution from the examples provided. Using the StringBuilder as a stack, the run-time reduces from 5 ms to 3 ms.

@stachenov .. Thanks for your suggestion of using the StringBuilder as a stack.

```
public class Solution {
    public String removeDuplicateLetters(String s) {

        int[] res = new int[26]; // will contain number of occurences of character (i+'a')
        boolean[] visited = new boolean[26]; // will contain if character ('a' + i) is present in
        char[] ch = s.toCharArray();
        for(char c : ch){  // count number of occurences of character
            res[c-'a']++;
        }
        StringBuilder sb = new StringBuilder();; // answer stack
        int index;
        for(char c : ch){
            index = c - 'a';
            res[index]--;   // decrement number of characters remaining in the string to be analy
            if(visited[index]) // if character is already present in stack, dont bother
                continue;
            // if current character is smaller than last character in stack which occurs later in
            // it can be removed and  added later e.g stack = bc remaining string abc then a can
            while( (sb.length() > 0) && c < sb.charAt(sb.length()-1) && res[sb.charAt(sb.length()
                visited[sb.charAt(sb.length()-1) - 'a'] = false;
                sb.deleteCharAt(sb.length()-1);
            }
            sb.append(c); // add current character and mark it as visited
            visited[index] = true;
        }

        return sb.toString();

    }
}
```

4

jaqenhgar
Reputation: ★ 754

I thought of the same approach, which is applied to another problem.

For an in[] array, and int K, form maximum number, while maintaining order.

Example - `[2,3,9,8,2,6]`, and `K = 3`, the maximum number formed is `[9,8,6]`.

`[2]` - Add, while remaining K = 2, Remaining elements is 5

`[2,3]` - Add 3, while remaining K = 1, Remaining elements is 4

`[9]` - Pop 3 and 2, since 9 is greater than both and remaining K = 2 < elements = 3

`[9,8]` - Add 8, less than 9, and remainging K = 1 < elements = 2

`[9,8,2]` - Add 2, less than 8, and remainging K = 0 < elements = 1

`[9,8,6]` - Pop 2, Add 6, since popping 2 makes K = 1, and element left is 1, which is 6

0

J

junhong202
Reputation: ★ 10

great solution. I like it.

0

R

renhai ↩ @jaqenhgar
Reputation: ★ 22

@jaqenhgar Similar to https://leetcode.com/problems/create-maximum-number/, Thanks

0

Derek_Han
Reputation: ★ 4

The best simple and fast solution. Thanks for sharing!

7 POSTS

4685 VIEWS

Log in to reply