1.3 Design an algorithm and write code to remove the duplicate characters in a string without using any additional buffer. NOTE: One or two additional variables are fine. An extra copy of the array is not.

FOLLOW UP

Write the test cases for this method.

pg 48

SOLUTION

First, ask yourself, what does the interviewer mean by an additional buffer? Can we use an additional array of constant size?

Algorithm—No (Large) Additional Memory:

1. For each character, check if it is a duplicate of already found characters.
2. Skip duplicate characters and update the non duplicate characters.

Time complexity is $O(N^2)$.

```
1   public static void removeDuplicates(char[] str) {
2       if (str == null) return;
3       int len = str.length;
4       if (len < 2) return;
5
6       int tail = 1;
7
8       for (int i = 1; i < len; ++i) {
9           int j;
10          for (j = 0; j < tail; ++j) {
11              if (str[i] == str[j]) break;
12          }
13          if (j == tail) {
14              str[tail] = str[i];
15              ++tail;
16          }
17      }
18      str[tail] = 0;
19  }
```

Test Cases:

1. String does not contain any duplicates, e.g.: abcd
2. String contains all duplicates, e.g.: aaaa
3. Null string
4. String with all continuous duplicates, e.g.: aaabbb