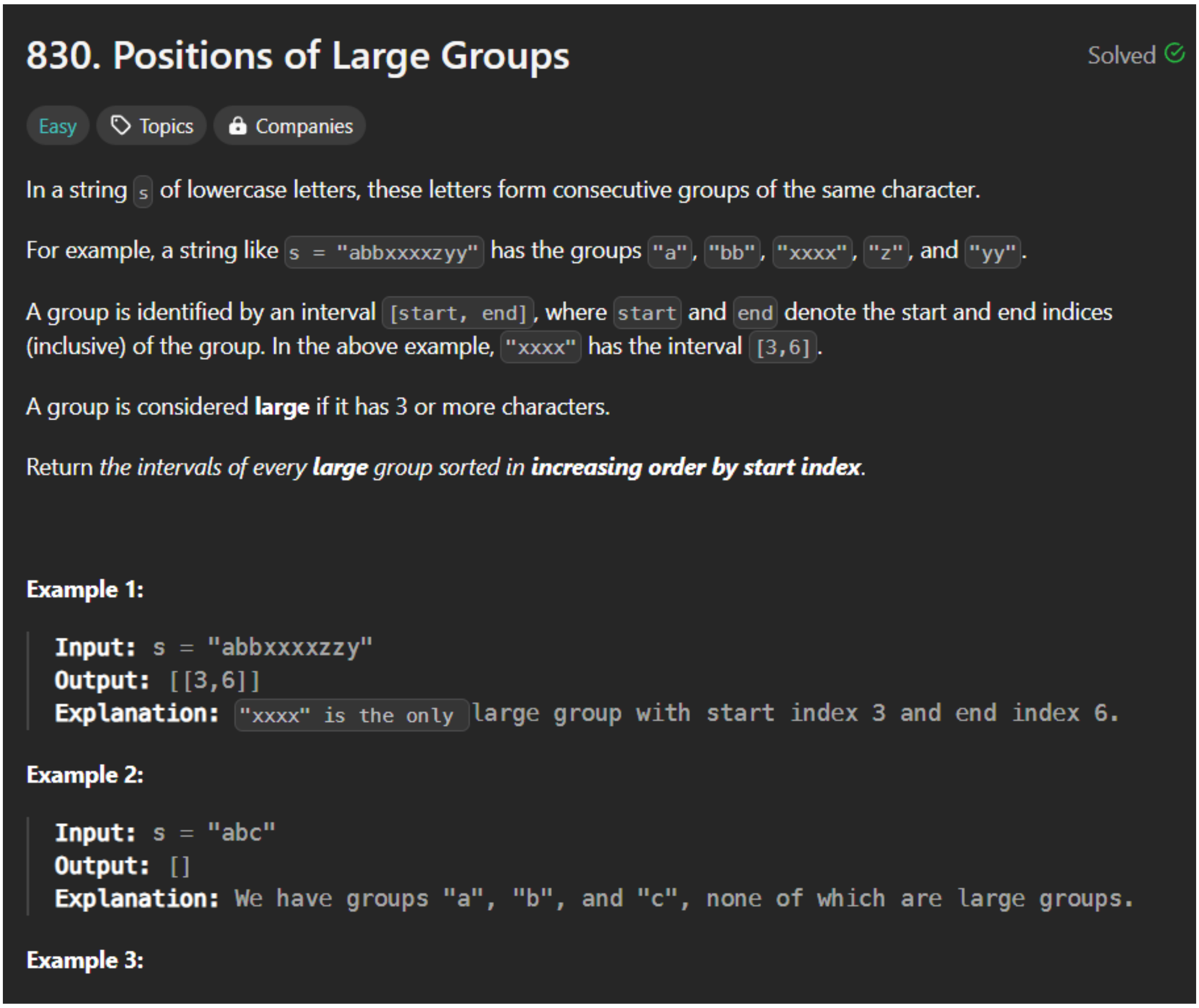


830. Positions of Large Groups

Solved ✓

Easy | Topics | Companies

In a string `s` of lowercase letters, these letters form consecutive groups of the same character.

For example, a string like `s = "abbxxxxzyy"` has the groups `"a"`, `"bb"`, `"xxxx"`, `"z"`, and `"yy"`.

A group is identified by an interval `[start, end]`, where `start` and `end` denote the start and end indices (inclusive) of the group. In the above example, `"xxxx"` has the interval `[3,6]`.

A group is considered **large** if it has 3 or more characters.

Return *the intervals of every **large** group sorted in **increasing order by start index***.

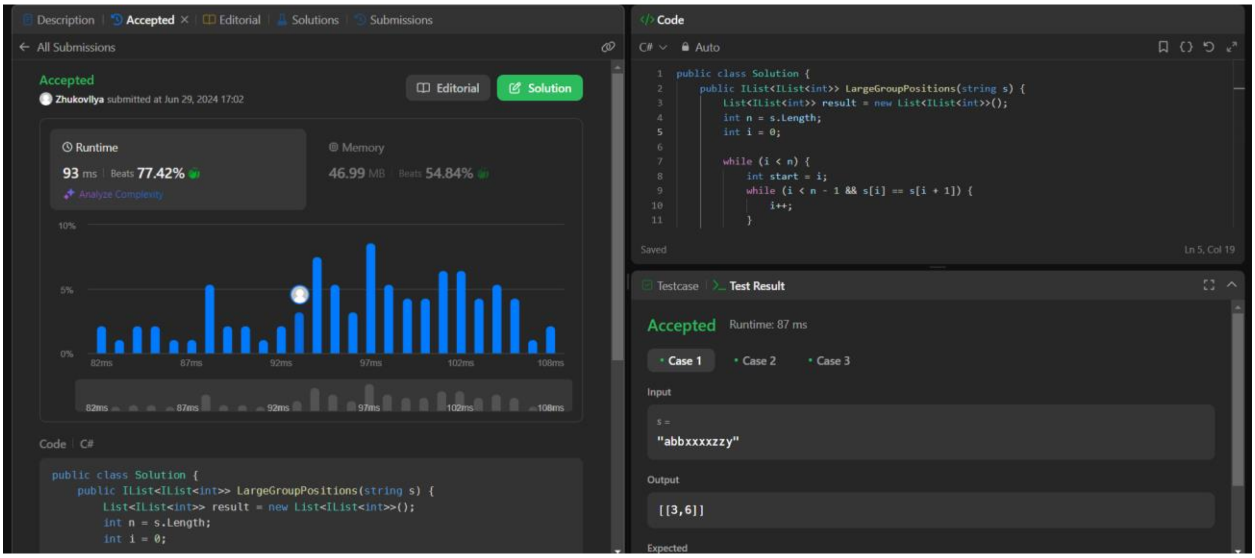
Example 1:

```
Input: s = "abbxxxxzzy"
Output: [[3,6]]
Explanation: "xxxx" is the only large group with start index 3 and end index 6.
```

Example 2:

```
Input: s = "abc"
Output: []
Explanation: We have groups "a", "b", and "c", none of which are large groups.
```

Example 3:



Accepted
Zhukovllya submitted at Jun 29, 2024 17:02

Runtime
93 ms | Beats 77.42%
Analyze Complexity

Memory
46.99 MB | Beats 54.84%

Code | C#

```
public class Solution {
    public IList<IList<int>> LargeGroupPositions(string s) {
        List<IList<int>> result = new List<IList<int>>();
        int n = s.Length;
        int i = 0;
```

```
public class Solution {
    public IList<IList<int>> LargeGroupPositions(string s) {
        List<IList<int>> result = new List<IList<int>>();
        int n = s.Length;
        int i = 0;

        while (i < n) {
            int start = i;
            while (i < n - 1 && s[i] == s[i + 1]) {
                i++;
            }
```

Accepted Runtime: 87 ms

Input

s =
```
"abbxxxxzzy"
```

Output
```
[[3,6]]
```

Код:

```
using System.Collections.Generic;

public class Solution
{
    public IList<IList<int>> LargeGroupPositions(string s)
    {
        List<IList<int>> result = new List<IList<int>>();
        int n = s.Length;
        int i = 0;

        while (i < n)
        {
            int start = i;
            while (i < n - 1 && s[i] == s[i + 1])
            {
                i++;
            }
            int end = i;
            if (end - start + 1 >= 3)
            {
                result.Add(new List<int> { start, end });
            }
            i++;
        }

        return result;
    }
}
```