1.15 FIND LARGE GROUPS IN A STRING

Question:

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.

AIM:

To identify large groups (3 or more consecutive identical characters) in a string s and return their intervals [start, end].

ALGORITHM:

1.Initialize result = [].

2. Use two pointers:
   - i = start index of a group.
   - Traverse the string with j.
3. While traversing:
   - If s[j] != s[i], it means the group ended at j-1.
   - Check if group length (j-1 - i + 1) >= 3.
   - If yes → append [i, j-1] to result.
   - Update i = j (start new group).
4. After loop ends, check the last group as well.
5. Return result.

PROGRAM:

```
def large_group_positions(s):
    result = []
    start = 0
    for i in range(1, len(s)):
        if s[i] != s[i-1]:
            if i - start >= 3:
                result.append([start, i-1])
            start = i
    if len(s) - start >= 3:
        result.append([start, len(s)-1])
    return result

def run_large_groups():
    s = input("Enter string: ")
    print("Large group intervals:", large_group_positions(s))

run_large_groups()
```

Input:

s = "abbxxxxzzy"

Output:

```
Enter string: abbxxxxzzy
Large group intervals: [[3, 6]]
>>>
```

RESULT:

Thus the program is successfully executed, and the output is verified.

PERFORMANCE ANALYSIS:

- Time Complexity: O(n) (one pass through string).
- Space Complexity: O(1) (excluding result storage).