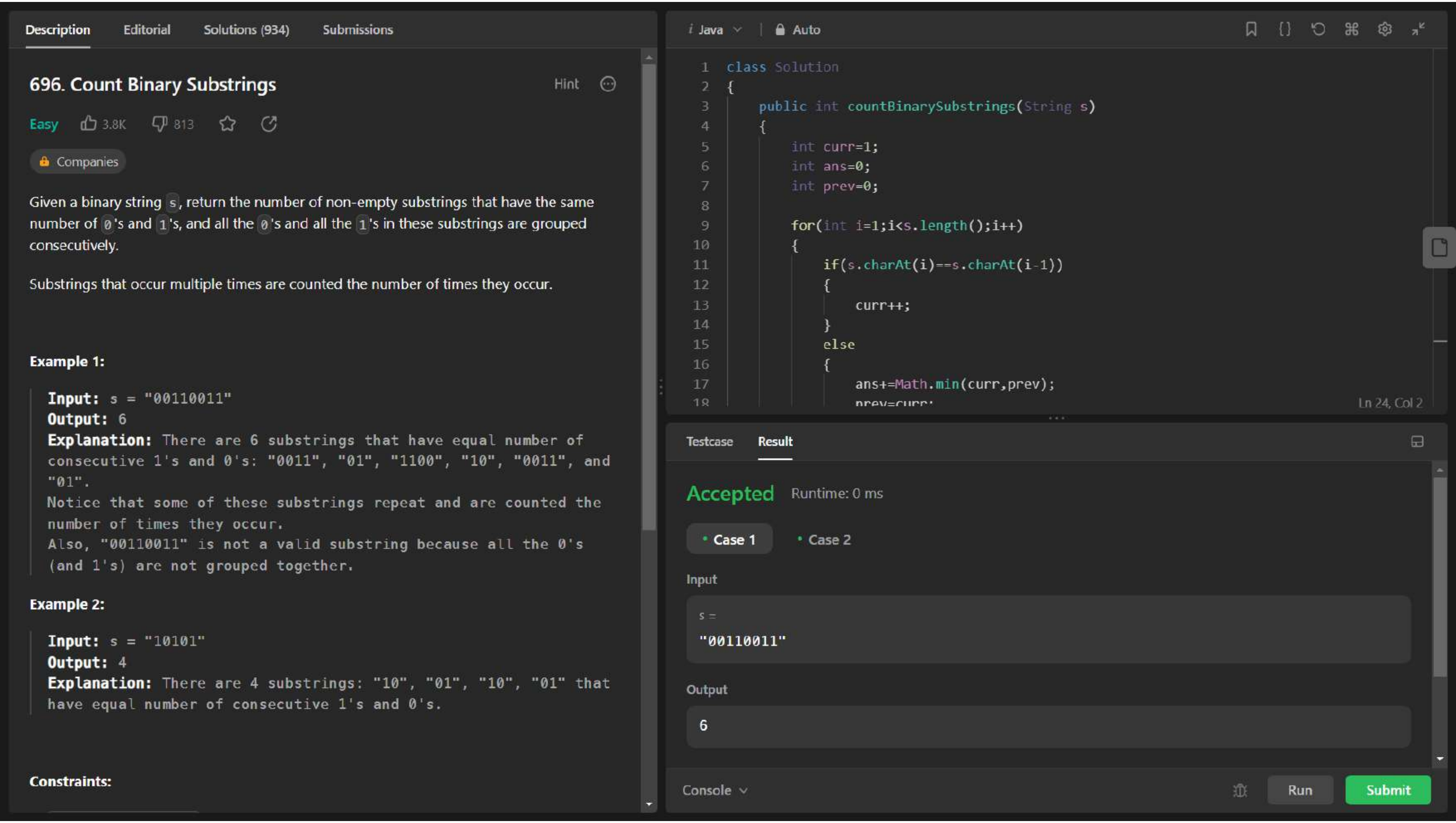


696. Count Binary Substrings

Given a binary string `s`, return the number of non-empty substrings that have the same number of `0`'s and `1`'s, and all the `0`'s and all the `1`'s in these substrings are grouped consecutively.

Substrings that occur multiple times are counted the number of times they occur.

Example 1:

```
Input: s = "00110011"
Output: 6
Explanation: There are 6 substrings that have equal number of
consecutive 1's and 0's: "0011", "01", "1100", "10", "0011", and
"01".
Notice that some of these substrings repeat and are counted the
number of times they occur.
Also, "00110011" is not a valid substring because all the 0's
(and 1's) are not grouped together.
```

Example 2:

```
Input: s = "10101"
Output: 4
Explanation: There are 4 substrings: "10", "01", "10", "01" that
have equal number of consecutive 1's and 0's.
```

Constraints:

```java
class Solution
{
    public int countBinarySubstrings(String s)
    {
        int curr=1;
        int ans=0;
        int prev=0;

        for(int i=1;i<s.length();i++)
        {
            if(s.charAt(i)==s.charAt(i-1))
            {
                curr++;
            }
            else
            {
                ans+=Math.min(curr,prev);
                prev=curr;
```

Accepted Runtime: 0 ms

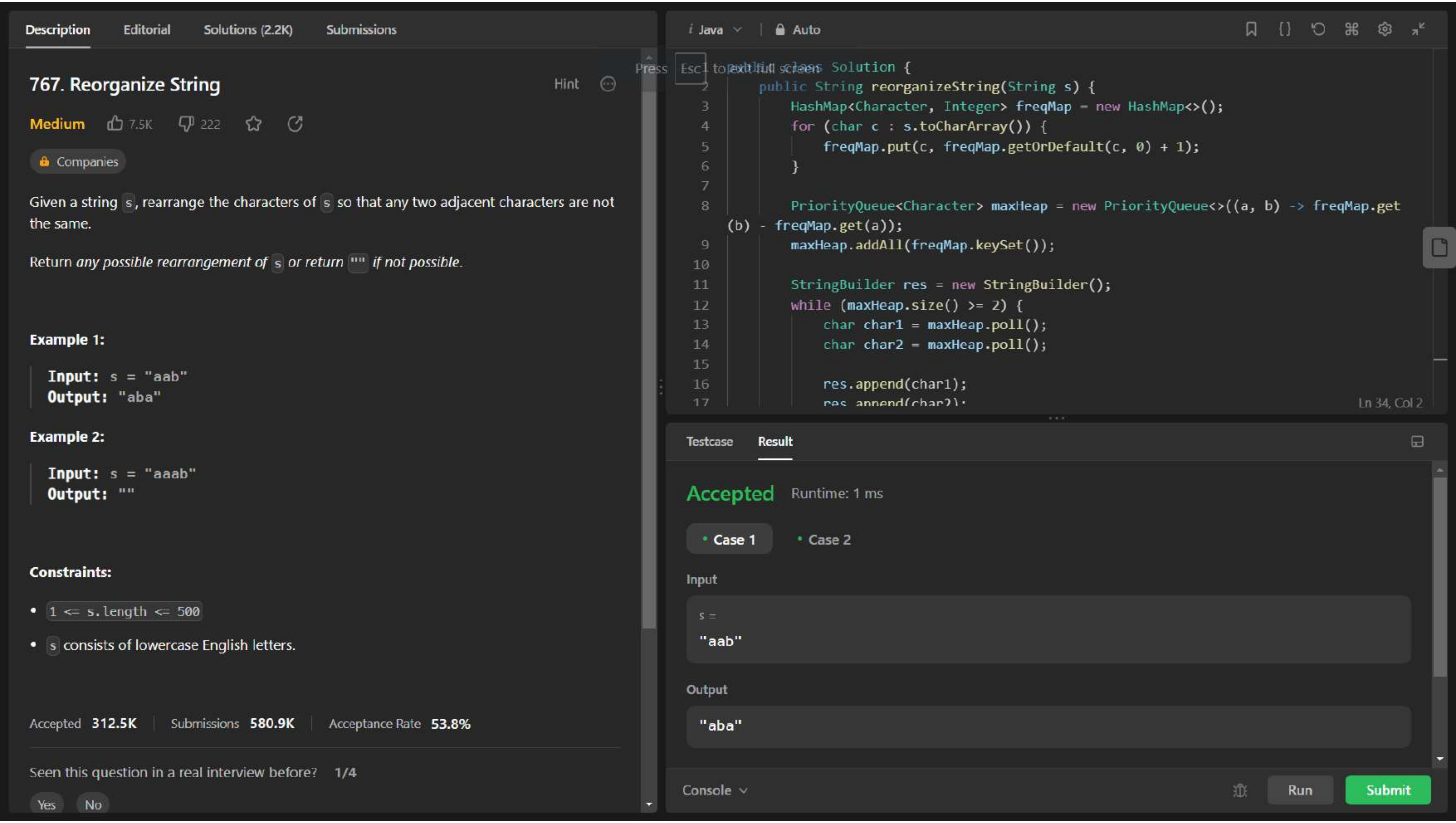
Input

s =

"00110011"

Output

6



767. Reorganize String

Medium

Given a string `s`, rearrange the characters of `s` so that any two adjacent characters are not the same.

Return *any possible rearrangement of* `s` *or return* `""` *if not possible.*

Example 1:

```
Input: s = "aab"
Output: "aba"
```

Example 2:

```
Input: s = "aaab"
Output: ""
```

Constraints:

- `1 <= s.length <= 500`
- `s` consists of lowercase English letters.

Accepted 312.5K | Submissions 580.9K | Acceptance Rate 53.8%

Seen this question in a real interview before? 1/4

```java
public class Solution {
    public String reorganizeString(String s) {
        HashMap<Character, Integer> freqMap = new HashMap<>();
        for (char c : s.toCharArray()) {
            freqMap.put(c, freqMap.getOrDefault(c, 0) + 1);
        }

        PriorityQueue<Character> maxHeap = new PriorityQueue<>((a, b) -> freqMap.get
(b) - freqMap.get(a));
        maxHeap.addAll(freqMap.keySet());

        StringBuilder res = new StringBuilder();
        while (maxHeap.size() >= 2) {
            char char1 = maxHeap.poll();
            char char2 = maxHeap.poll();

            res.append(char1);
            res.append(char2);
```

Accepted Runtime: 1 ms

• Case 1 • Case 2

Input

```
s =
"aab"
```

Output

```
"aba"
```