```
Alg longestBalancedSubstring(S):
    maxLength <- 0
    left <- 0
    char1 <- '\0'
    count1 <- 0
    char2 <- '\0'
    count2 <- 0

    for right <- 0 to s.length():
        c <- S[right]
        if char1 == null or c == char1:
            char1 <- c
            count1 <- count1 + 1
        else if char2 == or c == char2:
            char2 <- c
            count2 <- count2 + 1
        else:
            if count1 < count2:
                currentLength <- count1 * 2
            else:
                currentLength <- count2 * 2

            if currentLength > maxLength:
                maxLength <- currentLength
            left <- right - 1
            char1 <- S[left]
            count1 <- 1
            char2 <- c
            count2 <- 1

        if char1 != null and char2 != null and count1 == count2:
            currentLength <- right - left + 1
            if currentLength is greater than maxLength:
                maxLength <- currentLength

    Return maxLength
```

algorithm analysis:

this algorithm has only one iteration over the string and its complexity is O(n)
and it has some assignment inside and outside the loop also it has some comparason each of complexity of O(1)

so the overall complexity of the algorithm is O(n)