```
Alg getLongestBalancedSubstring(String S):
    maxLength <- 0
    for I <- 0 to S.length()
        count1 <- 0
        count2 <- 0
        a <- S[I]
        b <- ""
        for j <- I to S.length():
            c <- S[j]
            if c == a
                count1 = count1 + 1
                if count1 == count2 and maxLength < count1 * 2:
                    maxLength = count1 * 2
            else:
                if b == ' '
                    b = c
                    count2++
                    if count1 == count2 and maxLength < count1 * 2:
                        maxLength = count1 * 2
                else if c == b
                    count2++
                    if count1 == count2 and maxLength < count1 * 2:
                        maxLength = count1 * 2
                else
                    Break
    Return maxLength
```

algorithm analysis:
The outer loop loops over each character in the string s so its time complexity is O(n)

The inner loop loops over characters of the string from the current position
of the outer loop
so in its worst case loops over the whole string so its time complexity is O(n)

each assignment or comparason is O(1)

since they are nested loops, the time complexity of the algorithm is O(n^2)