Power of a String

Take a **String str** as input and calculate the **Power** of the string.

Power of a string is defined as the **maximum length** of **substring** that contains only one **unique** character.

A **substring** is a continuous sequence of characters within a string.

Note: All characters in the string are in **lowercase**.

Sample Input 0

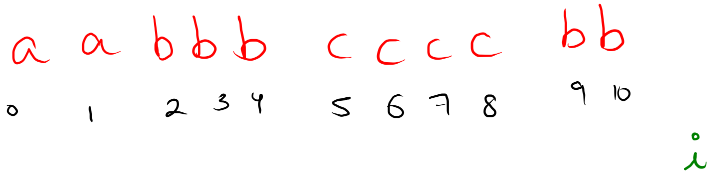
```
abbcccddddeeeeeffgghheecccc
```

Sample Output 0

```
5
```

max = ~~0~~ ~~2~~ ~~3~~ 4

curr = ~~1~~ 2



s[i] == s[i-1]
    curr++

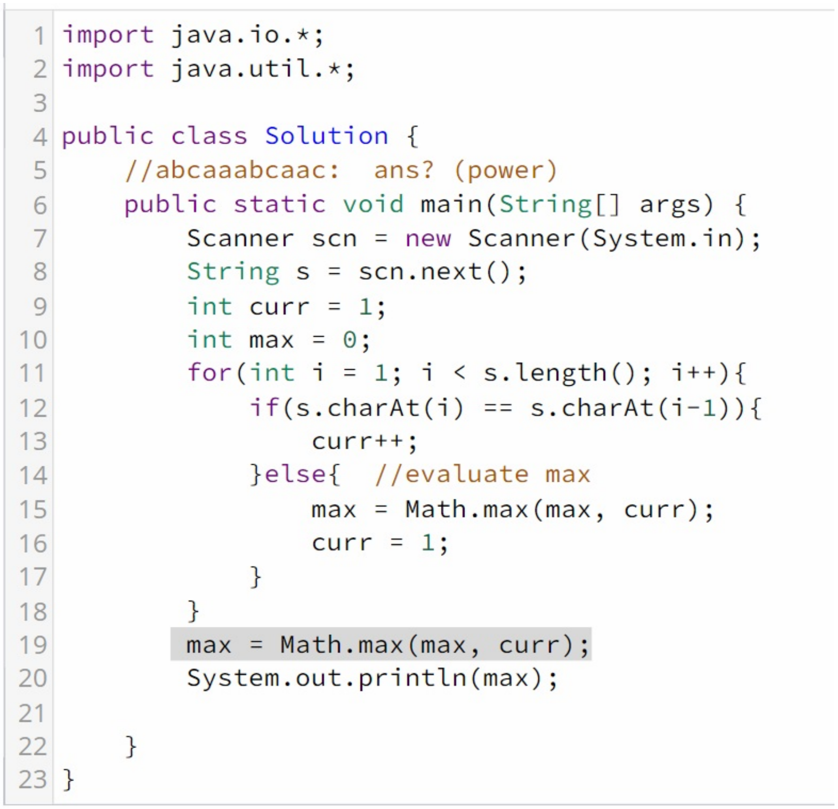
else s[i] ≠ s[i-1]
    ↳ find max
      curr = 1

curr = ~~1~~ ~~2~~ 3

max = ~~0~~ 2

a a b b b

i



```java
import java.io.*;
import java.util.*;

public class Solution {
    //abcaaabcaac:  ans? (power)
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        String s = scn.next();
        int curr = 1;
        int max = 0;
        for(int i = 1; i < s.length(); i++){
            if(s.charAt(i) == s.charAt(i-1)){
                curr++;
            }else{  //evaluate max
                max = Math.max(max, curr);
                curr = 1;
            }
        }
        max = Math.max(max, curr);
        System.out.println(max);

    }
}
```

Count Substring of 0 and 1

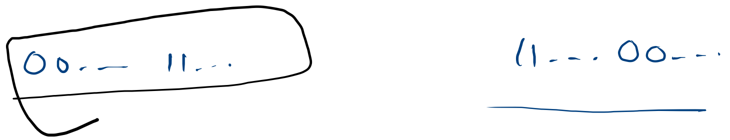
Given a binary string **s**, return the number of **non-empty** substrings that have the same number of **0's** and **1's**, ① and ② all the **0's** and all the **1's** in these substrings are grouped consecutively. Substrings that occur multiple times are counted the number of times they occur.

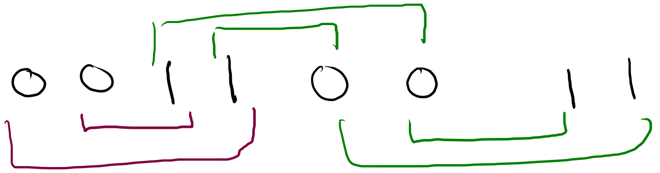
Sample Input 0

```
00110011
```

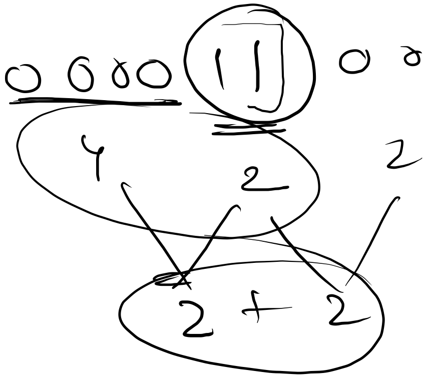
Sample Output 0

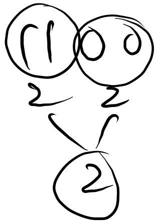
```
6
```

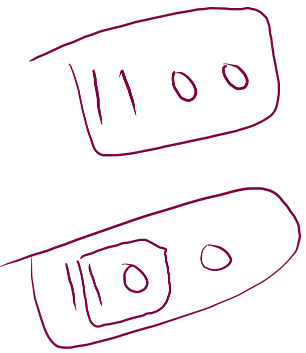


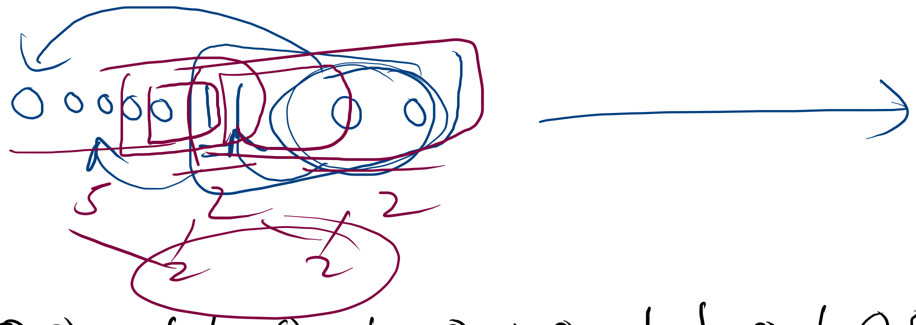


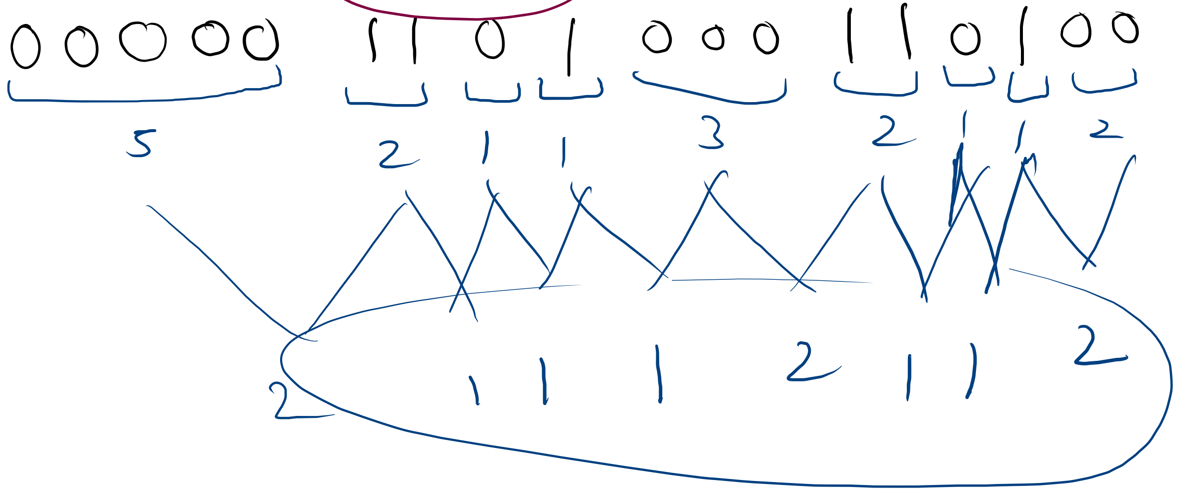
01
0011
10
1100
0011
01





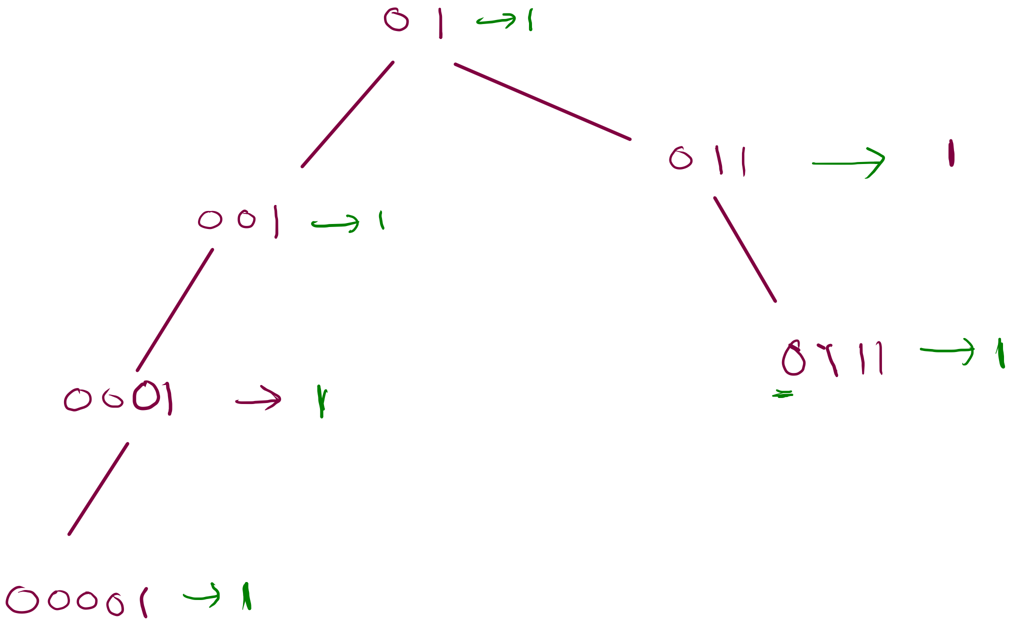






5
2
1
1
3
2
1
1
2
2
1
1
1
2
1
1
2

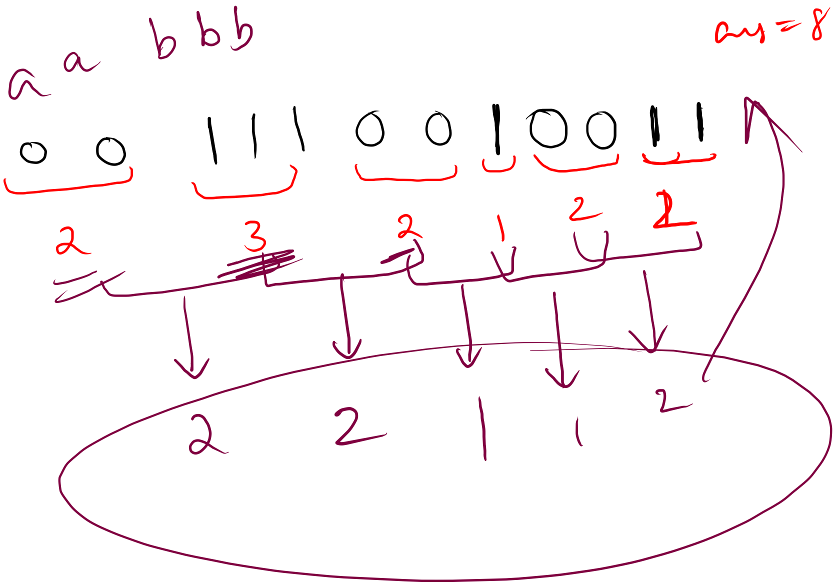
logic

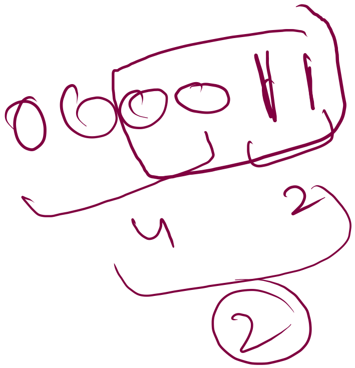


0 0 1 1 1 $\longrightarrow$ 2

1 1 0 0 0 $\longrightarrow$ 2?

0 0 1 1 1 0 0





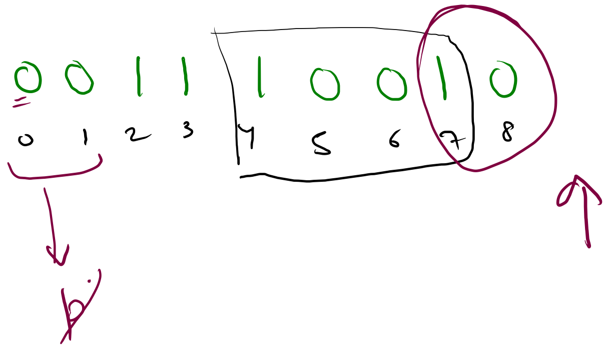
$$8 \rightarrow \underbrace{\quad}_{0} \quad \underbrace{0\ 0}_{2} \quad \underbrace{1\ 1\ 1}_{3} \quad \underbrace{0\ 0}_{2} \quad \underbrace{1}_{1} \quad \underbrace{0\ 0}_{\underset{p}{2}} \quad \underbrace{1\ 1}_{2} \qquad c$$

$\min(p, c) =$

$ans = 0 + 2 + 2 + 1 + 1 + 2$

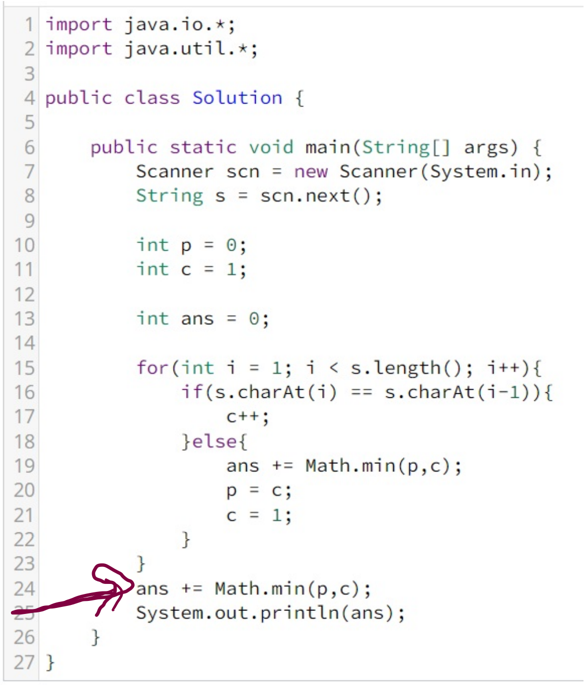
ans = 0 + 2 + 2 + 1 + 1

ans = 6



p = 1

c = 1



```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        String s = scn.next();

        int p = 0;
        int c = 1;

        int ans = 0;

        for(int i = 1; i < s.length(); i++){
            if(s.charAt(i) == s.charAt(i-1)){
                c++;
            }else{
                ans += Math.min(p,c);
                p = c;
                c = 1;
            }
        }
        ans += Math.min(p,c);
        System.out.println(ans);
    }
}
```

eg. 00111

Merge Strings Alternatively

Take two strings as input.

Merge both the strings **alternatively**.

Note: Length of strings will be same.

Sample Input 0

```
GEEK
STER
```

Sample Output 0

```
GSETEEKR
```

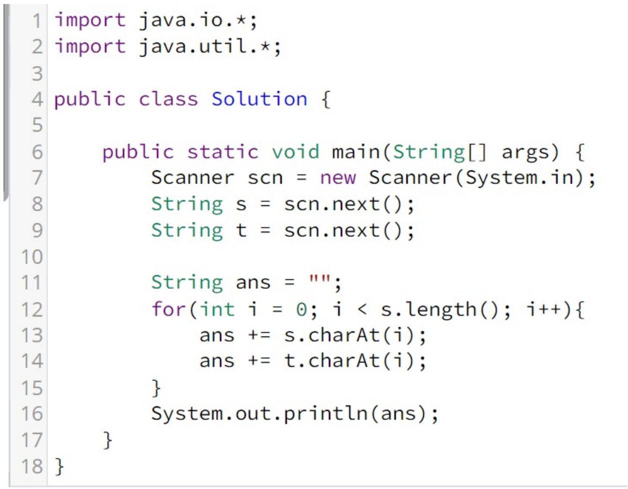
G E E K

0 1 2 3

$\boxed{i}$

S T E R

"GSETEEKR"

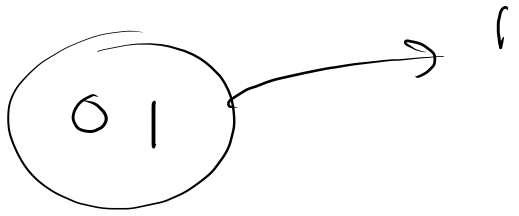


```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        String s = scn.next();
        String t = scn.next();

        String ans = "";
        for(int i = 0; i < s.length(); i++){
            ans += s.charAt(i);
            ans += t.charAt(i);
        }
        System.out.println(ans);
    }
}
```


0 1
(