Revision

- Target Sum.
- Crossword.

Target Sum.

tar = 8

eg. 5 1 3 2 4 2 7 8 6 6

5 3
2 6
1 7
?

tar = 8

A (1) 2 2 3 4 5 6 6 7 (8) B

i ... j

i < j

1 7
2 6
3 5

A[i] + A[j] == tar

4 + 4

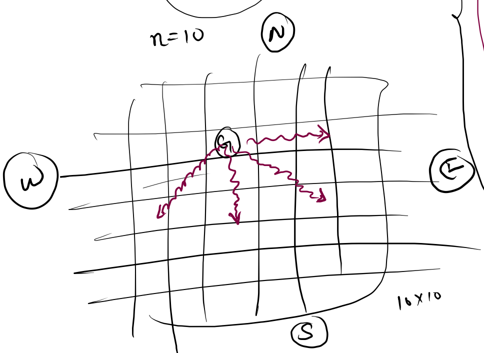
9 > tar

A + B > tar.

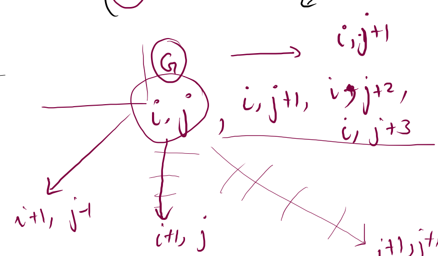
2. Crossword.

GREEN

n = 10



1. → (L → R)
2. ↓ (Top → Bottom)
3. X
   (a) NW → SE ↘
   (b) NE - SW ↙



Count Substring of 0 and 1

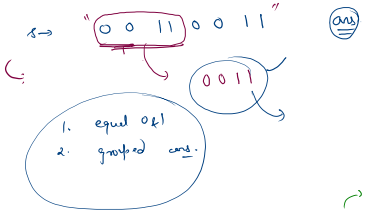
Problem | Submissions | Leaderboard | Discussions

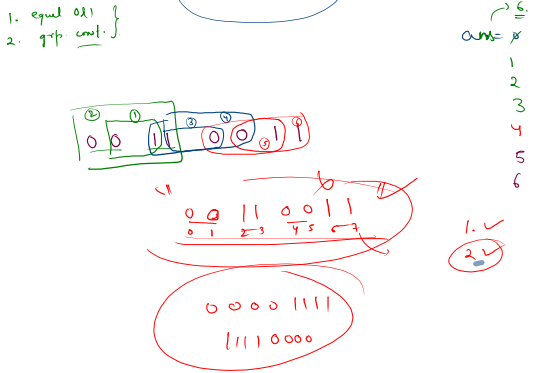
Given a binary string s, return the number of non-empty substrings that have the same number of 0's and 1's, and all the 0's and all the 1's in these substrings are grouped consecutively. Substrings that occur multiple times are counted the number of times they occur.

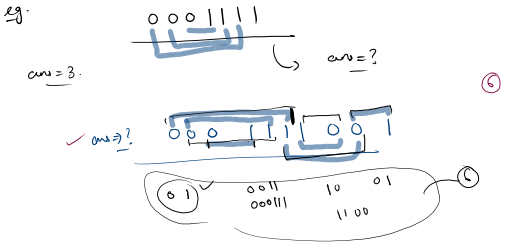
```
00110011
```

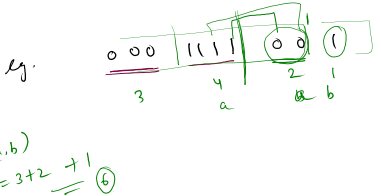
Sample Output 0

```
6
```



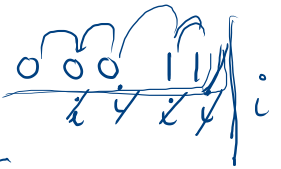






ans = 0

ans = 2



```java
int prev = 0;       // how many elements are there in previous grp (a)
int curr = 1;       // how many elements are there in current grp (b)
int ans = 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;
        curr = 1;

    }

}
ans += Math.min(curr, prev);
System.out.println(ans);
```

ans = 0

p = 3

c = 2

ans = 2

```java
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

public class Solution {

    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{
                if(curr > max){
                    max = curr;
                }
                curr = 1;
            }
        }
        if(curr > max){
            max = curr;
        }



        System.out.println(max);

    }
}
```

Subsequence.

L → R

str → ~~a~~ b c ~~d~~ e = "bce"

→ "bcde"

"acd"

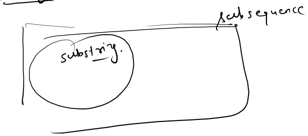
abc

"abcde"

abe

ace

all the substring is subsequence



"abc" → 8

""	bc	c
"a"	b	
"ab"		
"abc"		
"ac"		

abc ⇒

a	b	c	
0	0	0	→ ""
0	0	1	→ "c"
0	1	0	→ "b"
0	1	1	→ "bc"
1	0	0	→ "a"
1	0	1	→ "ac"
1	1	0	→ "ab"
1	1	1	→ "abc"

3

_ _ _

ab. (2)

0	0	→ ""
0	1	→ b
1	0	→ a
1	1	→ ab

$4 \Rightarrow 2^2$

Total str.

n char → 2^n

abc → $2^3 = 8$

abcde → 2^5

Check Subsequence

Problem | Submissions | Leaderboard | Discussions

Take two Strings **str** and **target** as input. Print **"True"** if **str** is a **subsequence** of **target** else print **"False"**.

Note: A **subsequence** of a string is a sequence that can be derived from the given string by deleting zero or more elements *without changing the order* of the remaining elements. (i.e., "ace" is a subsequence of "abcde" while "aec" is not).

Sample Input 0

```
abc
afbghidc
```

Sample Output 0

```
True
```

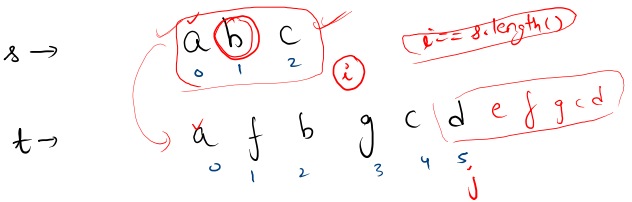
eg. s → abc

t → afbgcd

if (s is a subsequence of t)

↳ True

else ↳ false



if (s[i] == t[j])

i++

j++

else s[i] ≠ t[j]

j++