


Ques Given a string. Find the minimum and maximum distance b/w two vowels.

Input string \Rightarrow a b e c e f g a d e f

min distance = 2 [b/w index (1 and 9) or (2 and 4)]
 max distance = 9 [b/w index 0 and 9]

o 1 2 3 4 5 6 7 8 9 10
 a b e c e f g a d e f
 ch =

```
int n=str.length();
int first_occurrence=-1;
int last_occurrence=-1;

int last=-1;
int min_distance=n+1;

for(int i=0; i<n; i++){
    char ch=str.charAt(i);

    if(isVowel(ch)==true){
        if(first_occurrence== -1){ // i have not yet discovered any vowel
            first_occurrence=i;
        }

        last_occurrence=i;

        // finding min distance
        if(last== -1){
            last=i;
        } else {
            int distance = i - last;
            min_distance=Math.min(min_distance,distance);
            last=i;
        }
    }
}
```

first_occurrence = 0
 last_occurrence = 0 2 4
 i = 9 10 7 9
 last = -1 0 2 4 7 9
 min_distance = 1 2 2 1

Ques Given a string of numeric characters. Convert it to integer data type from String data type.

Input \Rightarrow " 134 "
 Output \Rightarrow 134

function

int val = Integer.parseInt(str);
↓
consisting of numbers

Ex: my str = "345";
int val = 345;

Ques Given a numeric string. Check if it is balanced.

Sum of first half is equal to second half
String str \Rightarrow "2433" $\xrightarrow{\text{true}}$ false
 $\begin{array}{c} 2 \\ \downarrow \\ 4 \\ \downarrow \\ 3 \\ \downarrow \\ 3 \end{array}$

str = "1231"
 $\begin{array}{c} 1 \\ \downarrow \\ 2 \\ \downarrow \\ 3 \\ \downarrow \\ 1 \end{array}$

str \Rightarrow "2457335" $\quad (n=7)$

fh = 0 < 6 ||
sh = 2 > 4

```
public static boolean checkBalanced(String str){  
    int n=str.length();  
    int first_half=0;  
    int second_half=0;  
  
    if(n%2==0){  
        // calculating first half  
        for(int i=0; i<n/2; i++){  
            char ch=str.charAt(i);  
  
            int val=ch-'0';  
            first_half=first_half+val;  
        }  
  
        // calculating second half  
        for(int i=n/2; i<n; i++){  
            char ch=str.charAt(i);  
  
            int val=ch-'0';  
            second_half=second_half+val;  
        }  
    } else {  
    }  
  
    if(first_half==second_half){  
        return true;  
    } else {  
        return false;  
    }  
}
```

$$i = \left(\frac{n}{2} + 1\right)$$

$$\sum_{3+1}^{7} + 1 \neq 4$$

$$'a' = 0$$

$$'a' = 1$$

?

$$\begin{aligned} '0' - '0' &= 0 \\ '1' - '0' &= 1 \end{aligned}$$

$$\begin{array}{rcl} '0' & \rightarrow & 0 \\ '1' & \rightarrow & 1 \end{array}$$

Ques Given two strings. Check if sum of ascii values of both strings are equal or not.

\Rightarrow str1 = "abcef" str2 = "bcdzh"
↓
 $97 + 98 + 99 + 100 + 102$
 $98 + 99 + 100 + 104 + 105$

0 1 2 3 4 5 6 7 8 9 10 11
z x v c b t x y z v y
ch= 'b'

h e e l l
 $\begin{matrix} z & \nearrow & v & \downarrow & c & \nearrow & b & \nearrow & t & \nearrow & y \\ \downarrow & & \downarrow \\ t_1 & t_2 & t_3 & = & t_4 & t_5 & t_6 & t_7 & t_8 & t_9 & t_{10} \end{matrix}$

26 0-2

```
start: char correspondingCharacter(String str){  
    int max=0;  
    int freq=0;  
    // storing frequencies of each character  
    for(int i=0; i<str.length(); i++){  
        char ch=str.charAt(i);  
        int freqe=0;  
        for(int j=i+1; j<str.length(); j++){  
            if(str.charAt(j)==ch){  
                freqe++;  
            }  
        }  
        if(freqe>0){  
            freq=freqe+1;  
        }  
    }  
    return freq;
```

Ques Remove all the duplicate from sorted string.

Input
a a b b c c d e e f f g g h h

Output
a b c d e f g h

```
2 public static String removeDuplicateSorted(String str){  
3     String ans="";  
4  
5     ans=ans+str.charAt(0);  
6  
7     for(int i=1; i<str.length(); i++){  
8         char ch=str.charAt(i);  
9         char ch_prev=str.charAt(i-1);  
10  
11         if(ch!=ch_prev){  
12             ans=ans+ch;  
13         }  
14     }  
15  
16     return ans;  
17 }  
18 }
```

String ans = " "

ans = a b
ch = a b
ch_prev = a

Input \Rightarrow baccabfgdf
 $ch = 'b'$) $\downarrow \downarrow \downarrow$

```

4 public static String removeDuplicateUnsorted(String str){
5     String ans="";
6
7     int[] fre=new int[26];
8
9     for(int i=0; i<str.length(); i++){
10         char ch=str.charAt(i);
11
12         int idx=ch-'a';
13
14         if(fre[idx]==0){
15             ans=ans+ch;
16         }
17
18         fre[idx]++;
19
20     }
21
22     return ans;
23 }
```

$ans = b a c f g d$

$a \rightarrow \cancel{a}^2$

$b \rightarrow \cancel{b}^2$

$c \rightarrow \cancel{c}^2$

$d \rightarrow \cancel{d}^1 \quad ch = 'c'$
 $|idx < 2$

$f \rightarrow \cancel{f}^1$

$g \rightarrow \cancel{g}^1$

String str = a b c d e
 \uparrow

$sp = 0$
 $ep = \cancel{ex}^{23}$

..
abc b c d e
ab c d bc cd de
abc de bcd cde

$ep = sp \ ; ep < n$
 $\therefore sp < n; sp++ \{$

```

15 public static void generateAllSubstrings(String str){
16     int n=str.length();
17
18     for(int sp=0; sp<n; sp++){
19         for(int ep=sp; ep<n; ep++){
20             String sub=makeString(str, sp, ep);
21             System.out.print(sub+" ");
22         }
23     }
24 }
```

a b c d e

$$sp = \alpha + 3$$
$$ep = 2, 3, 4$$
$$Sub = cd$$

```
public static void generateAllSubstrings_better(String str){  
    int n=str.length();  
  
    for(int sp=0; sp<n; sp++){  
        String sub="";  
        for(int ep=sp; ep<n; ep++){  
            sub=sub+str.charAt(ep);  
            System.out.print(sub+" ");  
        }  
    }  
}
```

a b c d e
ab bc cd de
abc bcd cde
abcd bcd e
ab cde

Ques Given a string, find the number of substrings with no vowels. 0, 2345
Count = ~~0 * 2 * 3 * 4 * 5~~

b a f c d e

O(n)

```
public static int countSubstringsWithoutVowel(String str){  
    int n=str.length();  
  
    int count=0;  
  
    for(int sp=0; sp<n; sp++){  
        for(int ep=sp; ep<n; ep++){  
            char ch=str.charAt(ep);  
  
            if(isVowel(ch)){  
                break;  
            }  
            count++;  
        }  
    }  
  
    return count;  
}
```

True
False
y()

Ques Given a number string, find the sum of all the substrings.

⇒

int num = 123

$$\text{num} = \text{num} * 10 + \text{digit}$$

"1234"

```
public static int sumOfSubstrings_better(String str){  
    int n=str.length();  
  
    int ans=0;  
  
    for(int sp=0; sp<n; sp++){  
        int num=0;  
        for(int ep=sp; ep<n; ep++){  
            char ch=str.charAt(ep);  
  
            int digit=ch-'0';  
            num=num*10+digit;  
  
            ans=ans+num;  
        }  
    }  
    return ans;  
}
```

$$\begin{aligned} sp &= 0 \\ ep &= sp+1 \\ num &= 1 \times 123 \\ dn &= 123 \\ digit &= 123 \end{aligned}$$

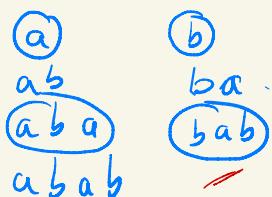
$$\text{ans} = 1 \times 135 + 136 + 1370 + 1372 + 1395$$

$$\text{ans} = 1 \times 135 + 136$$

Ques Given a string, find number of substrings starting and ending with same character.

⇒

a b a b ⇒ 6



① a
② b
③ a
④ b
sp=0 → 'a'
ep=2 → 'a'

$$\begin{aligned} str &= abcd e \\ n &= 5 \\ s &= abcde \quad tar = cdeab \end{aligned}$$

```
public static String getSubString(String s, int idx, int len){  
    String ans="";  
  
    int si=idx;  
    int ei=idx+len-1;  
  
    for(int i=si; i<ei; i++){  
        ans=ans+s.charAt(i);  
    }  
  
    return ans;  
}  
  
public static boolean check(String str, String tar){  
    String s=str+str;  
    int n=str.length();  
  
    for(int i=0; i<n; i++){  
        String sub=getSubString(s,i,n);  
  
        if(sub.equals(tar)){  
            return true;  
        }  
    }  
    return false;  
}
```

String sub = str. substring (si, ei); // won't include ei
 ↳ O(n)

$\text{str} = "abcde"$
 $(0, s)$
 $\text{sub} = \text{str.substring}(2, 4)$
 $\text{sub} = "cd"$
 $\text{sub} = \text{str.substring}(0, 3);$
 $\text{sub} = "abc"$

String sub = str.substring(5); // whole string starting from index 5
eg. sub = str.substring(3);
sub = "de"

. contains
 String s1 = "abcdef"
 String s2 = "cde" // checks if s2 is a substring of s1
 boolean ans = s1.contains(s2);
 ans = true
 s2 = "cdf"
 ans = false

String builder

\Rightarrow Initial

`StringBuilder sb = new StringBuilder();`

String str = "abc"; empty

String sb = new StringBuilder(str);
StringBuilder sb = new StringBuilder(str);

char ch = 'a';
sb.append(ch) → O(1) ⇒ sb = abc

add

get

$$ch = sb \cdot \text{charAt}(2);$$

lengths

$$l_{\text{m}} = \text{s.b. length}()$$

sit down At

sb. setCharAt (index, ch); $\rightarrow O(1)$

$$\Rightarrow O(n)$$

~~#~~ ~~delete~~

to String: → converted to string
String ans =

0 1 2 3 4 5
str = a b a b c
tar = a c d

sub (5, 8)

```
// https://www.hackerrank.com/contests/may-practice-java-dsa/challenge
public static int startingIndex(String str, String tar){
    int n=str.length();
    int tar_len=tar.length();

    for(int i=0; i<n; i++){
        if(str.charAt(i)==tar.charAt(index: 0)){
            String sub=str.substring(i, i+tar_len);

            if(sub.equals(tar)){
                return i;
            }
        }
    }

    return -1;
}
```