Strings

Content

- Basics.
- Switch Case.
- Substring.
- Check Palindrome.
- Palindromic Substring.
- Immutability.

Avg.

PSP — Wednesday

69.5 $\longrightarrow$ 69.1 $\longrightarrow$ 75%

Personal Goal $\longrightarrow$ as close to 100%

Basics

String $\longrightarrow$ array of characters
~~group of characters~~
~~collection of characters~~

b
ac
bac

a
bc
abc

order is important in a string.

sequence of characters.

"eat" $\neq$ "ate"

"Hello" $\downarrow$ double quotes for string

'a' 'b' $\longrightarrow$ single quote for character

To denote characters we use equivalent int values.
ASCII values.

character	'A'	to	'Z'	'a'	to	'z'	'0'	'9'
	$\downarrow$		$\downarrow$	$\downarrow$		$\downarrow$	$\downarrow$	$\downarrow$
ASCII	65		90	97		122	48	57

```
char ch = (char) 65
     print(ch)   //  A
```

```
char ch = (char) ('a' + 1)
     print(ch)   //  b
```

('a' + 1) $\longrightarrow$ implicit typecast from char to int

(char) $\longrightarrow$ explicit typecast

Note $\longrightarrow$ Smaller data type to larger data type conversion is implicit.

```
int x = 'a'
    print(x)   // 97
```

ASCII

right functions to convert to

HW ⟶ Do the above in your language and find the

Switch Case

Given a string S. Convert lowercase $\rightleftharpoons$ uppercase
Input only contains alphabets.

Eg   Input = "Hello"
      Output = "hELLO"

      Input = " aDgbHJe "
      Output = "AdGBhjE"

      Input = " aBcD "
      Ouput = " AbCd "

How to figure out if a character is lowercase ?

```
given char ch
 // condition of lowercase
if(ch >= 97 && ch <= 122)
↓
if(ch >= 'a' && ch <= 'z')
```

Pseudocode

Given string s

```
for i ⟶ 0 to N-1 {   // N = length of s
    ch = s[i]
    // condition of lowercase
    if (ch >= 'a' && ch <= 'z') {
        s[i] = (char) (ch - 32)
    }
    else {
        s[i] = (char) (ch + 32)
    }
}
```

'A' → 65   'a' → 97

TC : O(N)

SC: O(1)

If you forgot 32

```
int diff = abs ('a' - 'A')
```

abs → absolute value.

Substring

$\rightarrow$ similar to subarray.

$\rightarrow$ a part of a string.

$\longrightarrow$ entire string can be a substring

$\longrightarrow$ a single character is also a substring

"" empty string is not a substring.

s = "abc" $\rightarrow$

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

} 6 substring

s = "bxcd" $\rightarrow$

{ b, bx, bxc, bxcd } 4

{ x, xc, xcd } 3

{ c, cd } 2

{ d } 1

No. of substrings = no. of subarrays = $\frac{n*(n+1)}{2}$

Check Palindrome

aa , noon

Check if the given string s is a palindrome or not.

palindrome → Reads same from left to right & right to left

Eg :

mom	$\longrightarrow$	T
dad	$\longrightarrow$	T
madam	$\longrightarrow$	T
eye	$\longrightarrow$	T
subham	$\longrightarrow$	F

reverse of s == s

Bruteforce create a new string which is reverse of s and now s == reverse of s

TC : O(N)

SC : O(N) $\longrightarrow$ creation of reverse of s

a b b a   a b b a

r l $\longrightarrow$ stop when l >= r

Pseudocode

```
l = 0        r = N-1
while ( l < r ) {
        if ( s[l] != s[r] ) return false
        l += 1      // l++
        r -= 1      // r--
}
return true
```

a b b a

l   r

TC : O(N)

SC : O(1)

How to find if a substring is a palindrome ?

$$\begin{tikzcd} & \text{substring} \arrow[dl] \arrow[dr] & \\ \text{start} & & \text{end} \end{tikzcd}$$

Eg

$$\begin{tikzcd} & \text{aamadamcc} \arrow[dl] \arrow[dr] & \\ \text{start} & & \text{end} \end{tikzcd}$$

```
boolean checkPalindrome ( s, start, end )
        l = start   r = end
        while ( l < r ) {
                if ( s[l] != s[r] ) return false
                l += 1        // l++
                r -= 1        // r--
        }
        return true
}
```

$$\begin{array}{cccccccc} a & a & a & b & b & a & b & b \\ & & l & & & r & & \end{array}$$

TC : O(N)

SC : O(1)

TC : O(N)

SC : O(1)

Given a string s. Find the length of the longest **** odd length palindromic substring.

Eg   a bcbcb x b   Output 5

a na madam   Output 5

fe acabaca bgf   Output 7

ada ebcdfdcbe tggte   output - 9

Bruteforce —— check all odd length substring for palindrome and return max length.

Pseudocode

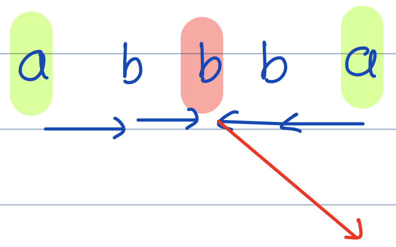
```
  ans = 0
for i ⟶ 0 to N-1 {
    for j ⟶ i to N-1 {
        len = j-i+1

        if ( len % 2 == 1 && checkpalindrome(s,i,j))
            ans = max(ans, len)
        }
    }
}
```

TC : $O(N^3)$

SC : $O(1)$

Break 10:37

Observation



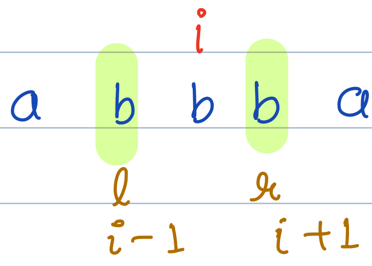
for odd length we will always meet at a single char

anamadam ans = ~~1~~ ~~3~~ 5

For every index consider it as the centre of the palindrome and expand around it.

Pseudocode

```
ans = 0

for i ⟶ 0 to N-1 {
        len = 1
        l = i-1
        r = i+1
        while ( l>=0 && r<N ) {
                if ( s[l] != s[r] ) break
                l -= 1
                r += 1
                len += 2
        }

        ans = max (ans , len)
}
print (ans)
```



TC : $O(N^2)$

SC : $O(1)$

Follow up find the even length palindrome

HW $\longrightarrow$ similar to above

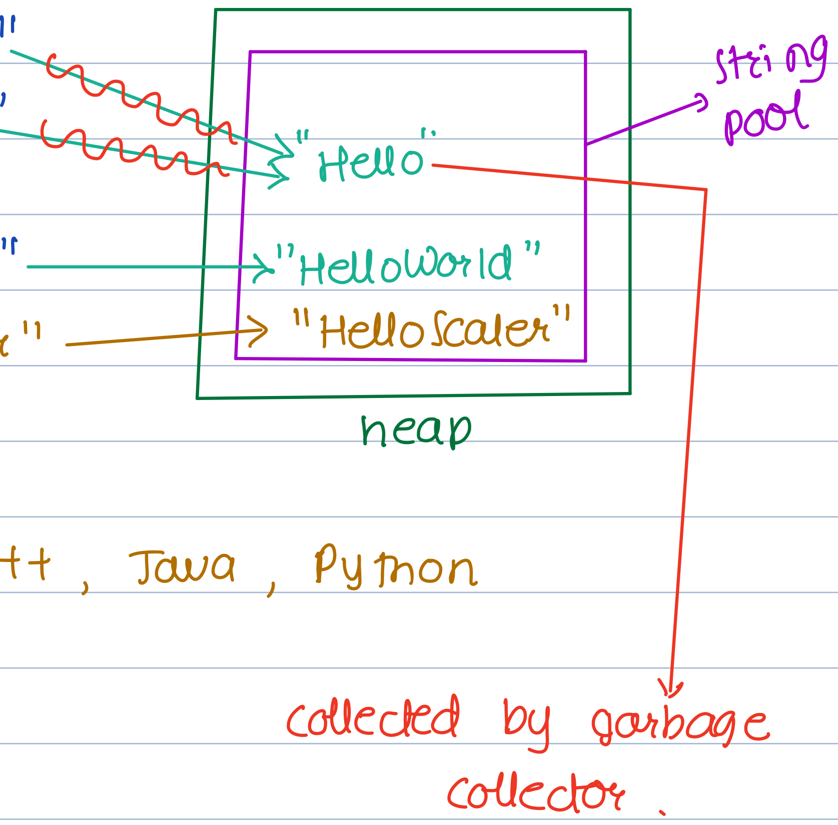
$l = i \quad r = i + 1$

Immutability { In general we have finite no. of meaningful words }

Once created it cannot be changed

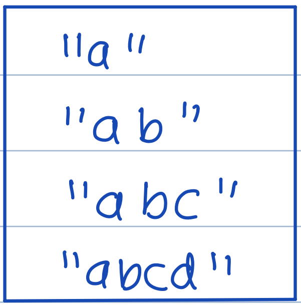
String s1 = "Hello"
String s2 = "Hello"

s1 = s1 + "World"
s2 = s2 + "Scaler"



String is immutable C++, Java, Python

String s = "a"
s = s + "b"
s = s + "c"
s = s + "d"



Given a list of words. Form a sentence adding all the words. word length is at max 10

Input = ["a", "b", "c", "d", "efg"]   A[]
Output = "abcdefg"

```
string s = ""
for i → 0 to N-1 {
    string word = A[i]
    s = s + word
}
print(s)
```

TC : O(N²)

~~TC : O(N)~~

O(N) operation
O(length of s)

String $\longrightarrow$ Immutable
String Builder $\longrightarrow$ Mutable
HW $\nearrow$

Next class Interview problems.

```
s = "abc"                          abc
s = s + "def"                      abcdef
```

```
A =    1  2   3   4   5
```

```
for  i ⟶ 0 to N-1 {                          for (int val : A) {
|       val = A[i]         ⟶                 |      print(val)
|       print(val)                            }
}
```