


Agenda:

1. String Basics
2. Change case of every character
3. Check if substring [i j] is palindrome
4. Length of longest palindromic substring
5. String Immutability
6. Reverse String Word By Word



**SUCCESS IS NOT FINAL;
FAILURE IS NOT FATAL:
IT IS THE COURAGE TO CONTINUE
THAT COUNTS.**

WINSTON S. CHURCHILL

String → Sequence of characters
Array of characters

"abc" ≠ "bac"

Character → A single symbol representing
letters, digits, special char. etc.

'A' 'a' '_' ' ' '9'

ASCII Values → 'A' - 'Z' → 65 - 90

'a' - 'z' → 97 - 122

'0' - '9' → 48 - 57

```
char ch = (char) 66;
```

```
print(ch); // → 'B'
```

```
int x = 'a';
```

```
print(x); // → 97
```

Q → Given a character array, convert every char. in lower case to its upper case & vice versa.

```
A = ['H', 'e', 'l', 'L', 'o']
```

```
↳ ['h', 'E', 'L', 'l', 'O']
```

```
A = [a D g b H J e]
```

```
↳ [A d G B h j E]
```

		32	
A	65	97	a
B	66	98	b
C	67	99	c
⋮			⋮
Z	90	122	z

```

for i → 0 to (N-1) {
    if ('a' <= A[i] && A[i] <= 'z') // 97 <= A[i] <= 122
        A[i] = (char) (A[i] - 32)
    else
        A[i] = (char) (A[i] + 32)
}

```

TC = O(N) SC = O(1)

Q → Toggle case of every character in String.

solution same as above → will work if
 string is mutable (eg. C++)
 will not work if string is immutable (eg. Java)
 same string is not updated

ans = ""

for i \rightarrow 0 to (N-1) {

if ('a' \leq s[i] && s[i] \leq 'z')

ans += (char) (s[i] - 32) // TC = O(len)

else

ans += (char) (s[i] + 32)

}

return ans

TC = O(N²) SC = O(N²)

a b c ans = "" \rightarrow A

A B

A B C

How to optimise?

- 1) convert string to char array. ✓
- 2) Do operations on char array. ✓
- 3) convert array back to string. ✓

SC = O(N)

(char array)

Substring → continuous part of string.

a b c d e f

$b x c d \rightarrow b \quad x \quad c \quad d$

$b \ x \quad x \ c \quad c \ d$

$$b \times c \quad \times \quad c \times d$$
$$b \times c = d$$

$$\begin{aligned} \# \text{ substrings} &= N + (N-1) + (N-2) \dots + 1 \\ &= \frac{N * (N+1)}{2} \end{aligned}$$

Q → check if the given string is palindrome.

```
str = reverse(str)
```

$s = \text{"racecar"}$

Ans = True

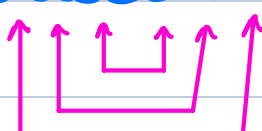
$S = \text{"m a d a m"}$

Ans = True

$s = "abc"$

Ans = false

$s = \text{"racecar"}$



equal



$l = 0$ $r = N - 1$

while ($l < r$) {

 if ($s[l] \neq s[r]$) return false
 else { $l++$ $r--$ }
}

return true

TC = $O(N)$

SC = $O(1)$

Q → Given a string, calculate the length of longest palindromic substring.

$s = \text{"aramadaman"}$ Ans = 5

$s = \text{"xy33y3xz"}$ Ans = 4

$s = "f e \underline{a c a b a c a b g} f"$

Ans = 7

$s = "\underline{a d a} \underline{e b c d f d c b e} \underline{t g g t e}"$

Ans = 9

Brute force $\rightarrow \forall$ substrings check if it is a palindrome.

TC = $O(N^3)$

ans = 0

for $i \rightarrow 0$ to $(N-1)$ {

for $j \rightarrow i$ to $(N-1)$ { // $i - j$

if (isPalindrome(s, i, j)) // TC = $O(N)$

ans = max(ans, $j - i + 1$)

}

} return ans

TC = $O(N^3)$

SC = $O(1)$

Idea $\rightarrow \forall$ char & char pair consider them center & expand to find longest length.

$0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \ 11 \ 12 \ 13 \ 14 \ 15 \ 16$
 $s = "a \ d \ a \ e \ b \ c \ d \ f \ d \ c \ b \ e \ t \underline{g \ g} \ t \ e"$

$ans = 12 \underline{9}$

$l = 2 \quad r = 12$

$(l+1) \text{ --- } (r-1)$

$= (r-1) - (l+1) + 1$

$= \underline{r - l - 1}$

$ans = 0$

for $i \rightarrow 0$ to $(N-1)$ { // odd length

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

while $(l \geq 0 \ \&\& \ r < N \ \&\& \ s[l] == s[r])$ {

| $l-- \quad r++$

}

$ans = \max(ans, r - l - 1)$

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

while $(l \geq 0 \ \&\& \ r < N \ \&\& \ s[l] == s[r])$ {

| $l-- \quad r++$

}

$ans = \max(ans, r - l - 1)$

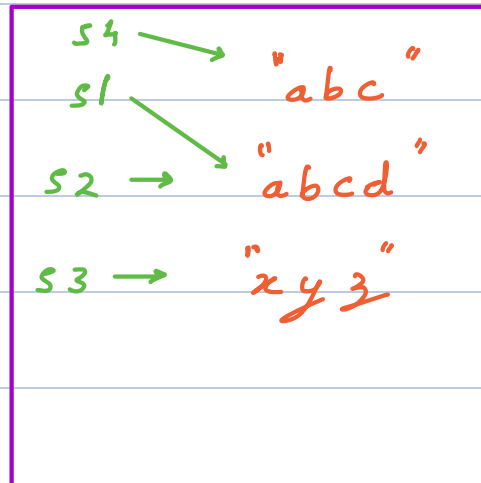
}

return ans

$TC = \underline{O(N^2)} \quad SC = \underline{O(1)}$

Immutability of Strings (Java, Python, C#, ...)

```
String s1 = "abc"  
s2 = "abcd"  
s3 = "xyz"  
s4 = s1 // same  
reference
```

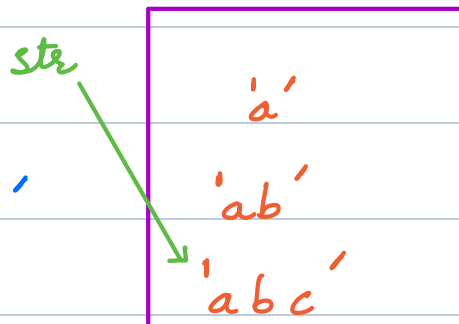


```
s1 = s1 + 'd' //abcd
```

String Pool

Use case → Reusability of same string
in the string pool.

```
String str = 'a'  
str = str + 'b'  
str = str + 'c'
```



Q → Reverse the string word by word.

$s = \text{"He is playing"}$

o/p → "playing is He"

$s = \text{"A car is moving"}$

o/p → "moving is car A"

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

A car is moving

↓ reverse string

gnivom si rac A

↓ ↓ ↓ ↓ reverse every word

moving is car A

TC = $O(N)$

SC = $O(1)$ / $O(N)$
