A string is defined as a Sequence of characters or in other words we can say it is an array of characters.

Ex: $\underbrace{\text{" Welcome to Scaler "}}$

Characters :-

A character is a single symbol that represents a letter, number or any symbol.

ASCII

↳ Standard mapping of character with integer.

A - Z :- 65 - 90

a - z :- 97 - 122

0 - 9 :- 48 - 57

10 = 49 48

```
char ch = (char) 65 ;      //   'A'

char ch = (char) ('a' + 1) ;    //  'b'
                  97 + 1
                 (char)(98)

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

ord ( )

↳ int → char

Q1 Given a string consisting of only alphabets(either lowercase or uppercase). Print all the characters of string in such a way that for all lowercase character, print its uppercase character and for all uppercase character, print its lowercase character.

TestCase

Input

"Hello"

Output

"hELLO"

a D g b H J e → A d G B h j E

Obs: 'a' (97) to 'z' (122) → lower case → -32; else → +32

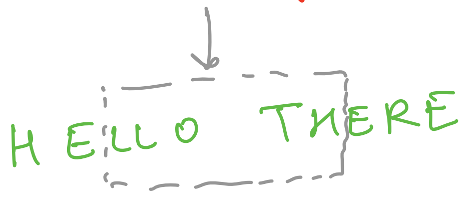
T.C: O(n)

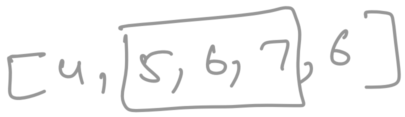
S.C: ~~O(1)~~ O(N)

→ use String builder

```
ans = " "
S = "Hello";
for ( i = 0; i < s.len() ; i++ )
{
    if ( s[i] >= 'a' && s[i] <= 'z')
    {
        ans += s[i] - 32 ;
    }
    else
    {
        ans += s[i] + 32 ;
    }
}
```

Substring





A substring is a continous sequence of character within a string. A substring concept is similar to the subarray concept in array.

A substring can be:

1. Continous part of String
2. Full String can also be a substring.
3. A single character can also be Substring

"abc"

a
b
c
ab
bc
abc

6 substring.

abca

total no. of substring

$$\boxed{\frac{n(n+1)}{2}}$$

Quiz

bxcd.b

b
x
c
d
bx
xc
cd
bxc
xcd
bxcd

10

Q2

Check whether the given substring of string s is palindrome or not.

A palindrome is the sequence of characters that reads the same forward and backward.for example, "nayan", "madam", etc.

== → reference

MALAYALAM.

s.equal(" ") → O(N)

O(N)

O(N) reverse and O(N) compare = O(2N)

O(N)

O(N)

2 pointer and check from both end ⇒ O(N/2)

O(N) N >>> large

O(1)

adda

2 pointer

T.C: O(N/2)

S.C = O(1)

```
s = "DaD";
start = 0
end = s.len() - 1
while (start < end)
{
        if (s[start] != s[end])
                    return false;
        start ++;
        end --;
}
return true;
```

$O(N/2)$

$= O(N)$

Q3 Given a string s, calculate the length of longest palindromic substring in s.

s = " ana madamm " ;

→ longest Palindrome.

ans = 5

Q1 f e a c a b a c a b g f .

ans = 7

Q- a d a e b e d f d c b e t g g t e

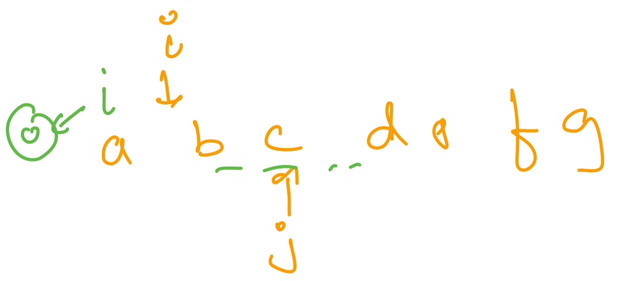
ans = 9

Brute force

(1) → Generate all the substring.

(2) → find palindromes ✓

(3) → find max length.



→ 2 loop to get all i and j

+
palindrom.

```
s= " abcdcbg";
ans= 0
for ( i=0;  i < s.len() ;  i++ ) O(n)
{
    for (j = i ; j < s.len() ; j++) O(n)
    {
        if ( isPalindrom(s, i, j))  → O(n)
        {
            ans = max (ans, j-i+1);
        }
    }
}
return ans;
```

T.C: $O(n^3)$

S.C: $O(1)$

s= " dad "

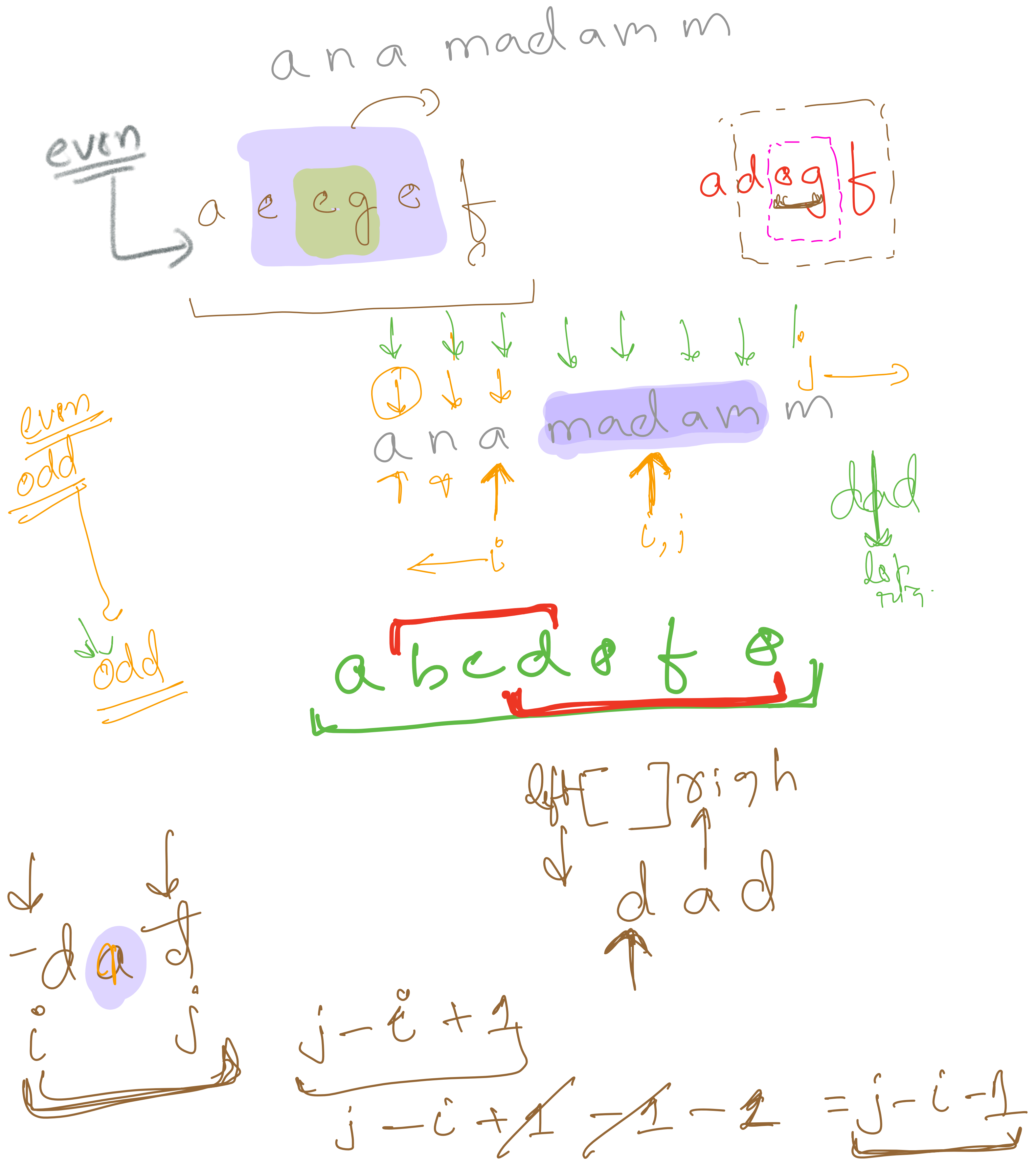
j-i+1
= 0-0+1

i	j	len Palin	ans
0	0	a 1	1
0	1	1	1
0	2	2-0+1 = 3	3
1	1	1-1+1 = 1	3
1	2	.	3
2	2	1	3

→ ans

10:27 pm

optimized approach



maxlength = 0

s = "ana madamm";

```
for ( i = 0;   i < s.len() ; i++)
{
    left = i
    right = i

    while ( left >= 0 and right < N)
    {
        if ( s[left] != s[right])
        {
            break;
        }
        left --;
        right ++;
    }

    maxlength = max(maxlength,
                    right - left - 1)
}
```

eddde

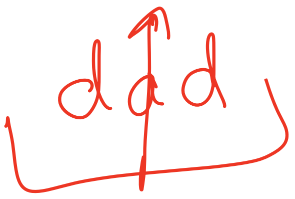
→ 0

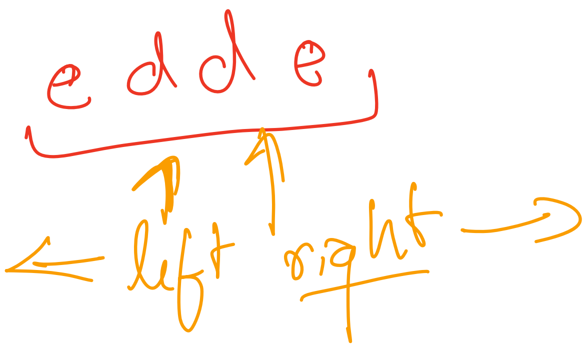
n = 9

i	left	right	maxlength
0	0	0	$1-(-1)-1$
	-1	1	$= 1+\cancel{1}-\cancel{1} = 1$
			$= 1$

1	1	1	$3i - (-1) - 1$
	0	2	$= 3 + 1 - 1$
	-1	3	$= \boxed{3}\ \boxed{3}$

evon palindrom





```
→ for ( i = 0;   i < s.len(); i++)
  {
        left = i
        right= i + 1
        while ( left >= 0 and right<n)
        {
              if ( s[left] != s[righ])
              {    break;
              }
              left --;
              righ ++;
        }
```

max length = max(max length,
right - left - 1).

}

odd and even ⇒ max

max (odd palin , even palindr)

T.C. $O(N^2) + O(N^2)$
even odd

$= O(N^2)$

S.C = $O(1)$

e d e f g h e

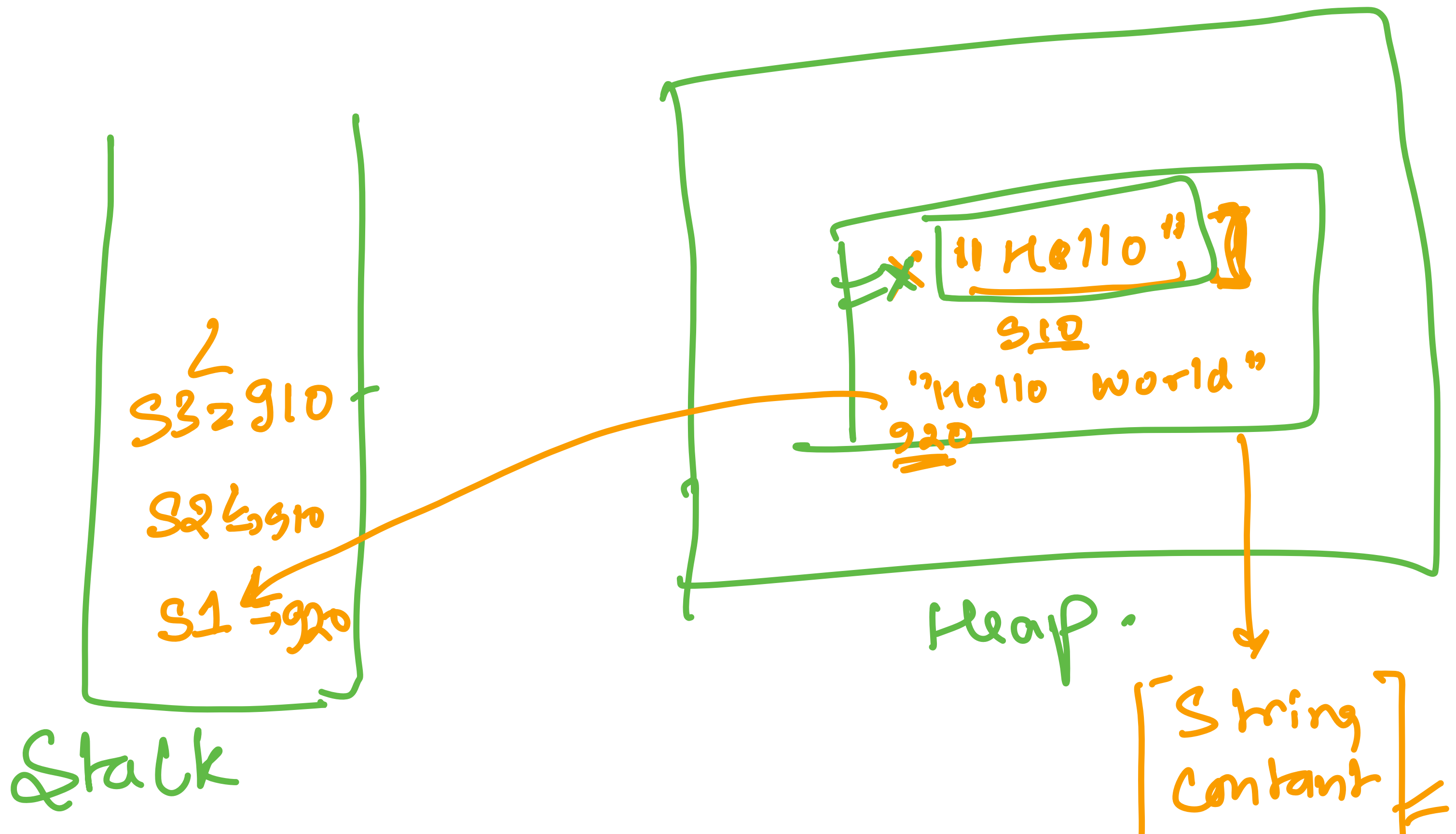
Imutability of String

which once created cannot be change

In python, Java → Strings are immutable

modify a string → Create a new instance

string ↳ class



→ String S1 = "Hello" [Pool]
→ String S2 = "Hello";
→ String S3 = S2;

→ S1.concat("World")
= "Hello World"

```
arr = ["a", "a", "a"]
S = "";
for (i = 0; i < arr.len(); i++) {
    S += (arr[i]);
}
```

O(n) O(n) O(n) O(n)

T.C : $O(n^2)$

Modification of String O(N)

① → copy the old string to the new address

② → add the modificat

Java

① ⇒ $\underline{\text{StringBuilder}}$ → .toString()

② ⇒ s = "abc";

⇒ list

b = ['a', 'b', 'c']

O(n)
+
O(n)
+
O(n)

a = list(s)
= [

a = ''.join(a)

$\underline{\underline{O(n)}}$

List → → ← ←

StringBuilder S = new String()

$\underbrace{\text{s. (} \Leftarrow}$ O(1)

s.append( )

→ StringBuilder S = new String ( )

→ $\underbrace{\text{s. ( )}}$ O(1)

→ $\underbrace{\text{s. toString( )}}$