Strings Introduction

- Intro
- Flip
- count occ[]
- Reverse string
- String Palindrome or not

Strings:

- Array of characters
- Sequence of characters
- Bunch of characters

order matters

- {a, d, b}
- {a, b, d}

'A' → 65 → Binary representation

'A' → 65	32 ⟷	'a' = 97	ASCII	
'B' = 66	32 ⟷	'b' = 98	'0' → 48	
'C' = 67		'c' = 99	'1' → 49	
⋮		⋮	⋮	
'Z' = 90	32 ⟷	'z' = 122	'9' = 57	

```
char a = 'A'
print( (int) a ) = 65
```

obs: B/w capital & small char, ASCII value diff is 32

int arr[10]

Strings

- "String" { we cannot change them; Immutable
- char S[]
- String Builder

{ we can change a particular index value.

String S = "Hello"

S[0] = 'A' ✗ in Java

✓ in C, C++

char S[5] = {'H', 'E', 'L', 'L', 'O'}

S[0] = 'A' ✓ Java, C, C++

Q. Given a char[]. Toggle each char.

{ UC → LC
LC → UC }

s[] = AnaConDa

⇓

ans[] = aNAcONdA

only UC & LC chars

```
char[]  Toggle ( char s[])
{
    int n = s.length()
    for( i=0; i<n ; i++)
    {
        // s[i]
        if ( s[i] ≥ 65  && s[i] ≤ 90)
        {
            // s[i] is uppercase
            s[i] = s[i] + 32
        }
        else
        {
            // s[i] is lowercase
            s[i] = s[i] - 32
        }
    }
    return s
}
```

both works

s[i] ≥ 'A' && s[i] ≤ 'z'

TC: O(N)

SC: O(1)

65

'A' = 1 0 0 0 0 0 1

6 5 4 3 2 1 0

↓ +32 ↑ -32

'a' = 1 1 0 0 0 0 1

'97

B - 66 1 0 0 0 0 1 0 'b' 98 1 1 0 0 0 1 0

C - 67 1 0 0 0 0 1 1 'c' 99 1 1 0 0 0 1 1

0 1 0 0 0 0 0

1 1 0 0 0 1 1

0 1 0 0 0 0 0

1 0 0 0 0 1 1

:

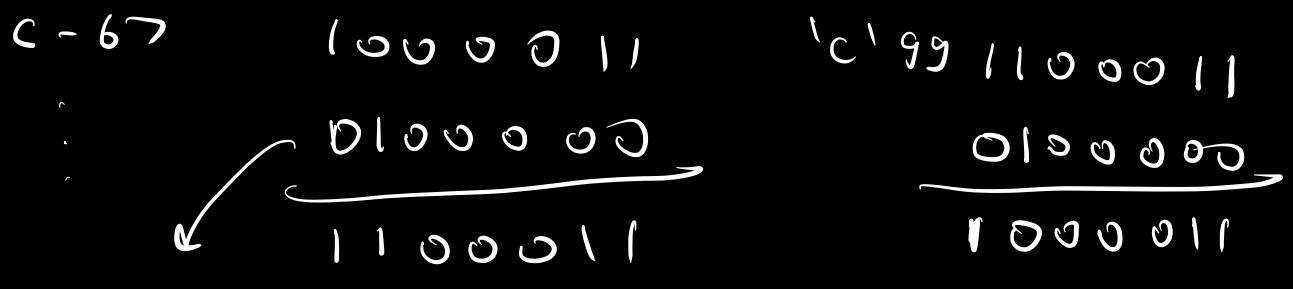
'2' 1 0 1 1 0 1 0 '2' 1 1 1 1 0 1 0

122

0 → 1

1 → 0

S[i] ^ (1<<5)



1<<5

32

Char[] Toggle (Char S[])

int n = S.length()

for(i=0; i<n ; i++)

{ S[i] = S[i] ^ 32

return S

S[i] ^ (1<<5)

TC : O(N) SC: O(1)

hello ^ 32

Q. Given a char ch[], only consists lower case alphabets. ('a' - 'z')
Sort the ch[]

Ans: Ideal → Array sort TC: O(N log N)

Idea 2

d a b a c d b →

'a' → 2
'b' → 2
'c' → 1
'd' → 2

a a b b c d d

int c[26] = {0}

0th → freq of 'a'
1st → freq of 'b'
2nd :
:

a - 'a' → 0
b - 'a' → 1
c - 'a' → 2
⋮
z - 'a' → 25

25th → freq of 'z'

```
int c[26] = {0}

for( i=0; i<n; i++)
{
    // s[i]
    int index = s[i] - 'a'
    c[index]++
}
```

s[i] = 'a' 0
'b' 1
'c' 2

```
int k=0

for( i=0 ; i<26 ; i++)
{
    // char tmp = i+'a'
    // c[i]

    for( j=1 ; j<=c[i] ; j++)
    {
        s[k] = j+'a'
        k++
    }
}
```

aabbccdd
'a' → 2
'b' → 2
'c' → 1
'd' → 2

0 + 'a' = 'a'
1 + 'a' = 'b'
2 + 'a' = 'c'

25 + 'a' = 'z'

No of iterations of inner for loop :

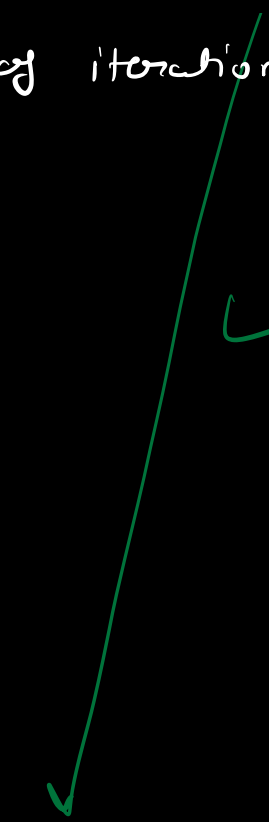
a) 26 N ✗

b) N

c) 26 * 26

d) N^2

e) None of them



c[0] + c[1] + c[2] + ... c[25]

= N

TC: O(N)

SC: O(char size) = O(1) for this question char = 26

d e a b c a d b e f

0 = 2
1 = 2
2 = 1
3 = 2
4 = 2
5 = 1
6 = 0
7 = 0
8 = 0
⋮
5 = 0

a a b b c d d e e f

ans

Q. Given a string / char [], Reverse

a part of string

substring

continous part of string

Entire string is also substring

single char is also substring

Empty substring is not substring.

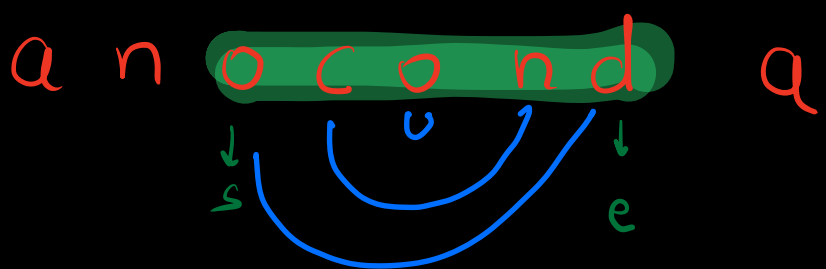
Ex

0	1	2	3	4	5	6	7
a	n	o	c	o	n	d	a

s = 2 e = 6

Both inclusive

ans = a n d n o c o a




```
reverse ( char ss[], int s, int e)
{
    s' = s
    e' = e
    while ( s' < e')
    {
        swap( ss[s'], ss[e'])
        s'++ , e'--
    }
}
```

TC : O(N)
↙
O(size of substring)

reverse (ss, 0, N-1)

Q. Given a ch[], consists of lots of words [each word seperated by '_']

Reverse word by word

[No extra space
No inbuilt functions]

Ex = love_hate_data_structure

ans = structure_data_hate_love

Ex Marry_love_Kill

ans = Kill_love_Marry

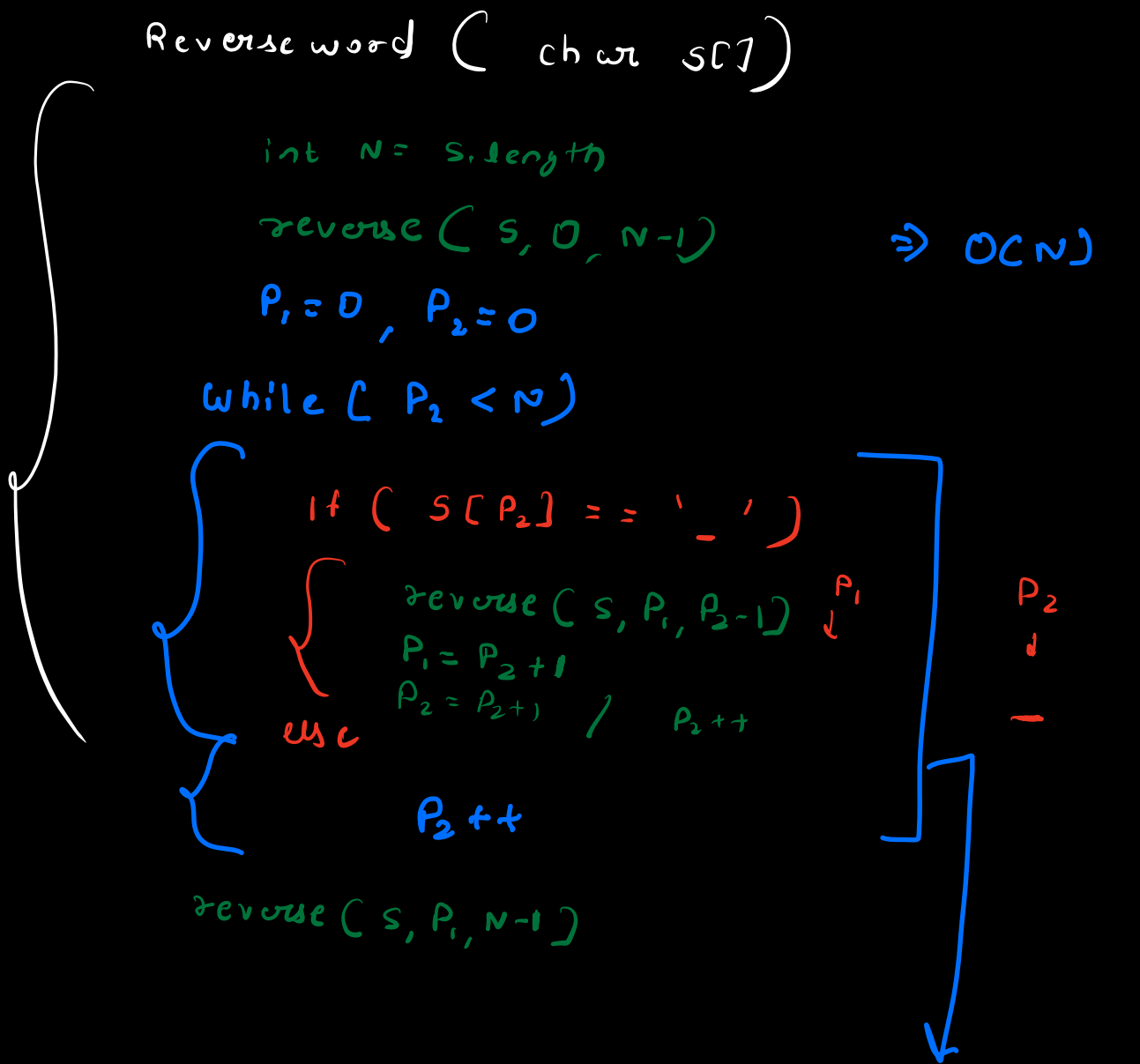
s = love_hate_data_structure

reverse entire string

erutcurts_atad_etah_evol

reverse one by one each word

structure_data_hate_love



$\Rightarrow O(N)$

Tc: $O(N)$

Sc: $O(1)$

$O($ size of word1 $)$

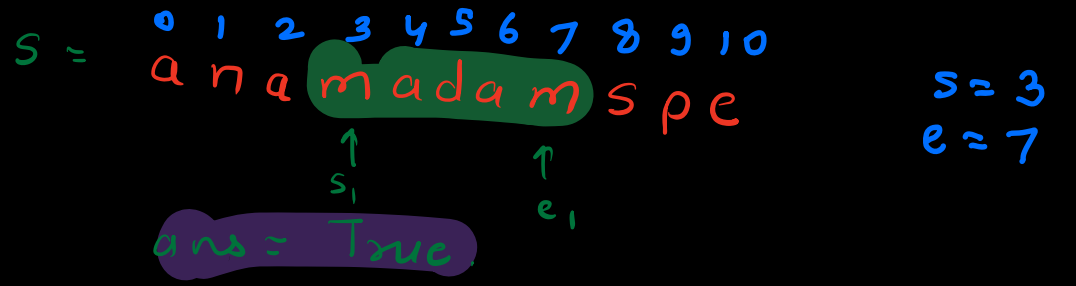
$+ O($ size of word2 $)$

!

!

$= O(N)$

Q. Given a char s[]. check if given substring is Palindrome or not.




```
bool checkPalindrome ( char s[], s, e )
{
    int s' = s
    int e' = e
    while ( s' < e' )
    {
        if ( s[s'] != s'[e'] )  return False
        s'++
        e'--
    }
    return True
}
```

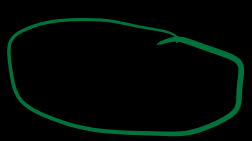
TC : O(N) ~ O(len of substring)

Sc: O(1)

Doubt

LinkedIN Keshav Seksaria Google

4 times



5 times

	Inside
1	5
2	5
3	5
4	5
	20

1 → O(word1)

2 → O(word2)

3 → O(word3)

⋮

$O(word1) + O(word2) + O(word3) + \cdots = O(N)$

0 1 2 3 4

b a b a c

66 − 65

b − a = 1

a − a = 0

c − a = (2)

c[0] = ~~0~~ ~~+~~ 2

c[1] = ~~0~~ + 2

c[2] = ~~0~~ 1

c[3] = 0

4^{th} = 0

5^{th} = 0

S[i] - a

a - a = 0