String

↳ Group of characters. ✗

→ Sequence of characters

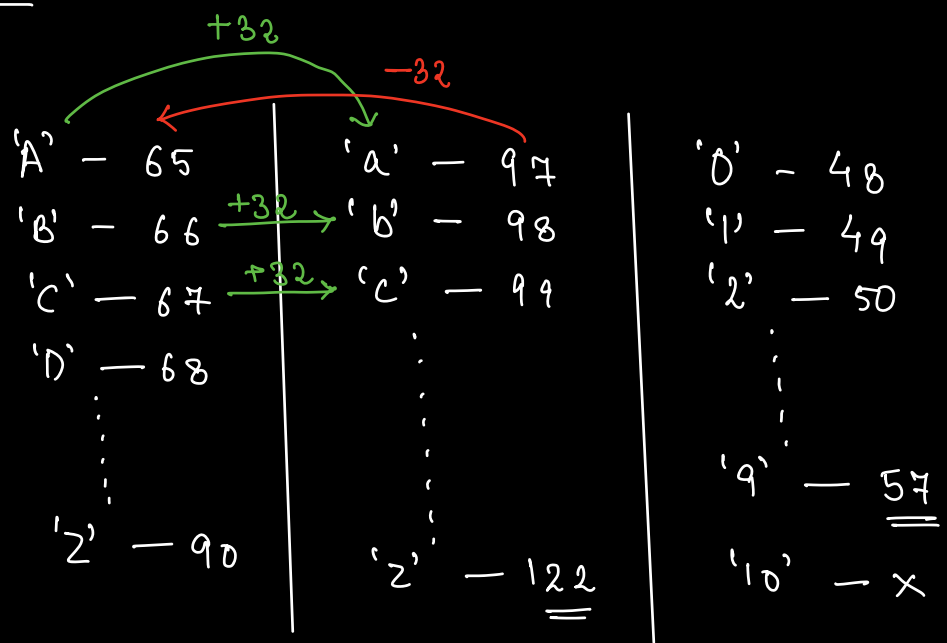
→ Array of characters

→ Set of characters ✗

→ list of characters

⇒ String is an ordered sequence of characters.

⇒ ASCII



⇒ Strings are Immutable (Java / Python / C# ....)

⇒ StringBuilder ⇒ Java

Q. Given a String s, Toggle the case of every character.

Toggle → Small case → Upper case
Upper case → Small case

S = "aAbBcC"
↳ "AaBbCc"

```
String toggle(String s){
    for(i=0; i<n; i++){
        if(s[i]>='A' && s[i]<='Z'){
            // s[i] is an Upper case.
            s[i] += 32;
        }
        else if(s[i]>='a' && s[i]<='z'){
            // s[i] is a lower case
            s[i] -= 32;
        }
    }
    return s;
}
```

s[i] ^= 32 → 1<<5

32 → abs('a' - 'A')

TC: O(N)

$32 = 2^5 = \underset{6}{0}\ \underset{5}{1}\ \underset{4}{0}\ \underset{3}{0}\ \underset{2}{0}\ \underset{1}{0}\ \underset{0}{0}$

'A' - 65
⋮
'Z' - 90

1	0					
6	5	4	3	2	1	0
⇓	⇓					
2^6	32					
64						

'a' - 97
⋮
'Z' - 122
} > 96

1	1					
6	5	4	3	2	1	0
⇓	⇓					
2^6	32					
64						

1) 5th bit will be **unset** in all **upper case** Characters.

2) 5th bit will be **set** in all **lower case** Characters.

⇒ **Toggle** the 5th Bit.
↳ XOR operator

1 ^ 1 = 0
0 ^ 1 = 1

1	1	x	x	x	x	x
0	1	0	0	0	0	0
^						
1	0	x	x	x	x	x

1 ^ 0 = 1
0 ^ 0 = 0

Q. Given a String, String contains only lower case characters. Sort the string.

S: "dabaedb"

s: "a a b b d e"

⇒ sort(S.begin(), S.end())

↳ O(NlogN)

⇒ String only contains lower case alphabets, So String can have 26 distinct characters.

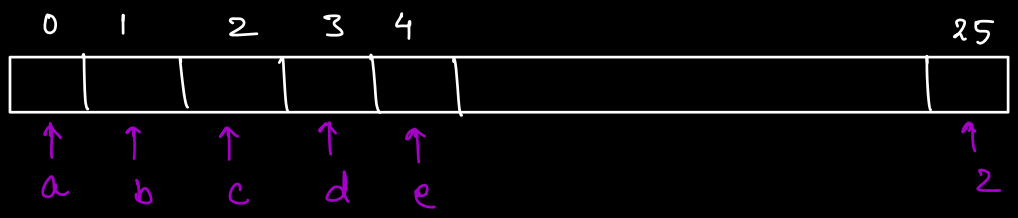
S: "dabaedb"

a → 2
b → 2
c → 0
d → 2
e → 1
f → 0
⋮
z → 0

⇒ a a b b d d e

→ HashMap / Array ...

```
int count[26] = {0};
```



Character → index

a (97) —(-97)→ 0

b —(-97)→ 1

c —(-97)→ 2

⋮

z —(-97)→ 25

s[i] → s[i]-97 (+97 (+'a'))

Count Sort

O(N)

```
for( i = 0; i < n; i++ ){
    // s[i] ⇒ count[s[i]-97]
    count[s[i]-'a']++;
}

for( i = 0; i < 26; i++ ){
    // count[i]
    // char ⇒ i + 'a'
    for(j=0; j<count[i]; j++){
        str.append(i+'a');
    }
}
```

i	j	# iterations
0	[0, c[0]]	c[0] ⇒ |a|
1	[0, c[1]]	c[1] ⇒ |b|
⋮		
25	[0, c[25]]	c[25] ⇒ |z|

$|a| + |b| + |c| + \cdots + |z|$

N

TC : $O(N)$

SC : $O(1)$

↳ Count [26]

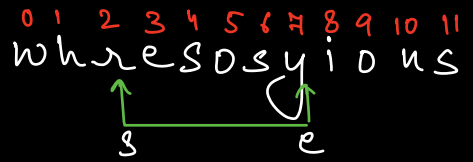
Q: Given a string, start index (s) & end index (e). Reverse the substring from s to e.

Contigous part of the string ≃ Subarrays

S: "a b d e a g f"
0 1 2 3 4 5 6

a b g a e d f

Quiz



w h y s o s e r i o n s .

```
reverse(Str, s, e) {
    while (s < e) {
        swap(Str[s], Str[e])
        s++
        e--
    }
}
```

TC: O(N)

SC: O(1)

Q: Given a String, which stores a sentence.

Amazon
MS
Adobe

→ No extra space before & after
→ Every word of this string is separated by a single space.

→ Reverse the string word by word.

S: "here_is_a_picture"

→ picture a is here

S: "Are you as clever as I am"

→ am I as clever as you Are

S: "Mailmen brings letters"

→ letters brings Mailmen

reverse the string

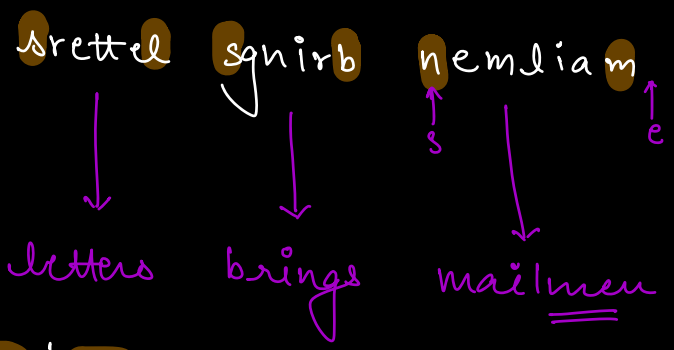
→ srettel sgnirb nemliam

reverse each word individually.

Steps

I) Reverse the complete string.
↳ reverse(str, 0, N-1)

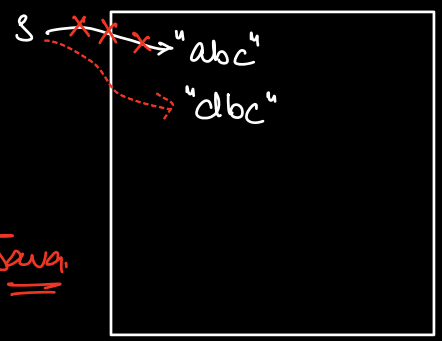
II) Reverse all the words individually.



⇒ Code | TC

#Strings in Java

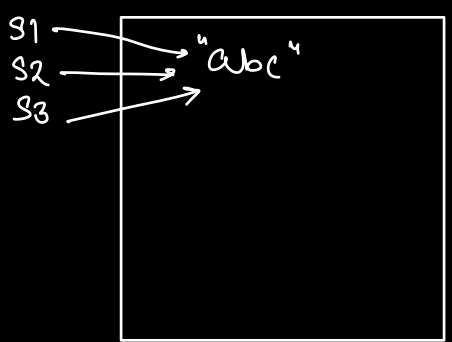
```
String s = "abc"
s[0] = 'd'
```



String Pool

⇒ String is Immutable in Java

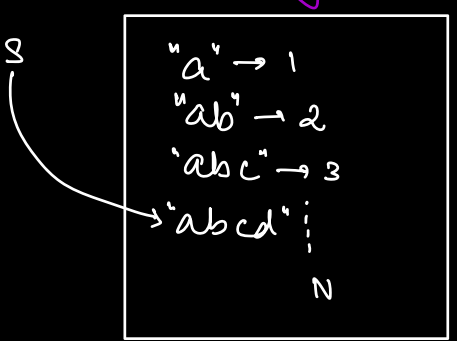
```
String s1 = "abc"
String s2 = "abc"
String s3 = s1;
```



String Pool

```
String s = "a"
s = s + 'b'
s = s + 'c'
s = s + 'd'
.
.
.
N times
```

SC : O(N²)



String Pool

Garbage Collector

⇒ StringBuilder sb = new StringBuilder()