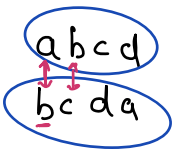
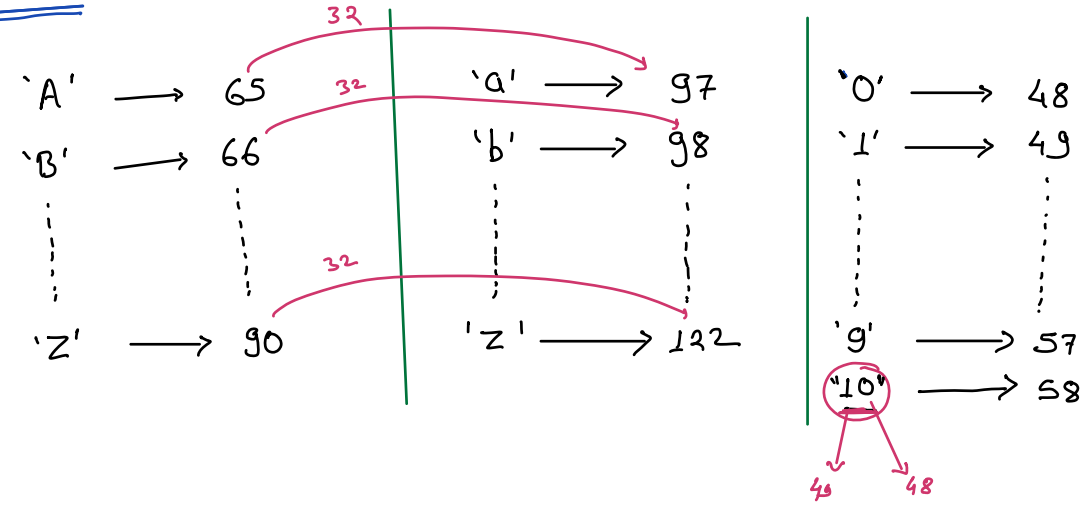
Strings : ~~Group of char~~

Sequence of char ✓

Array / List of char ✓



ASCII

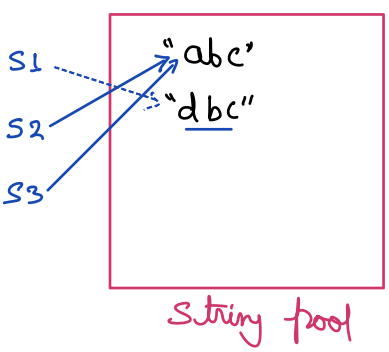


Immutable Strings

(Java / Python & other languages)

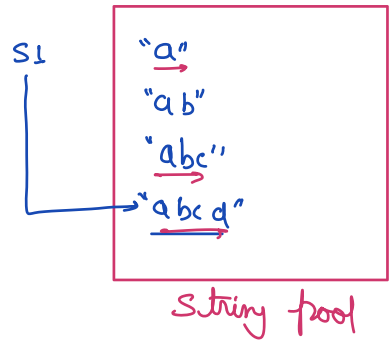
```
String s1 = "abc";
// s1[0] = 'd'
// s1 → "dbc"

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



Append Char to immutable string

```
String s1 = "a";
s1 = s1 + "b";   → O(1) O(N)
s1 = s1 + "c";   → O(1) O(N)
s1 = s1 + "d";   → O(1) O(N)
```



If we append N times

TC : $O(N^2)$
SC : $O(N^2)$

Garbage Collector cleans out useless strings from memory.

StringBuilder

Java & C#

```
StringBuilder sb = new StringBuilder();
sb.Add('c');
sb.Append("abc");
sb.ToString()  ⟶ Java String.
```

Q Given a string s. Toggle the case of every characters.

Upper Case ⟶ Lower case

Lower Case ⟶ Upper case

s : a Bc AE d ⟶ AbCaeD

[Without using any inbuilt method]

```
toggle (s) {

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

s[i] = s[i] ^ diff (32)

abs ('a' - 'A');

'a' → 97

'b' → 98

⋮ ⋮

'z' → 122

7	6	5	4	3	2	1	0
_	1	1	_	_	_	_	_
	↓ 64	↓ 32					

64 + 32 → 96

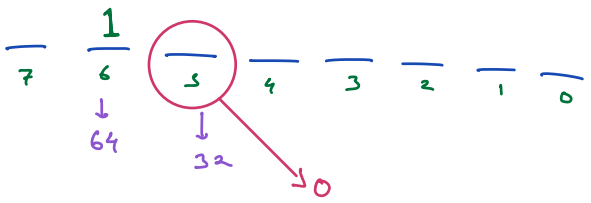
5th bit must be set for [a, z]

'A' ⟶ 65

'B' ⟶ 66

⋮ ⋮

'Z' ⟶ 90



64 + 32 ⟶ 96

5th bit must be unset for [A, Z]

diff ('a' 'A') ⇒ 32

'z' → 122

'Z' → 90

	7	6	5	4	3	2	1	0
'z'	0	1	1	1	1	0	1	0
'Z'	0	1	0	1	1	0	1	0

'z' - 'Z' ⇒ 32

In every pair of small & capital char only 5th bit will be diff.

```
S[i] = S[i] ^ 32;
```

S[i] ^ (1 << 5)

2^5

Q Given a string of lower case char. Sort it in dict order.

s : d a b a e d b

a a b b d d e

Use library sorting method → O(N log N)

lower case alphabets → [a - z] ⇒ 26 unique char.

d a b a e d b

- d : 2
- a : 2
- b : 2
- e : 1

↑ HashMap / Dict

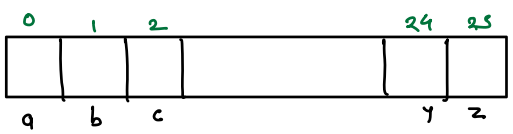
for a ⟶ z

a a b b d d e

int count [26] = {0}

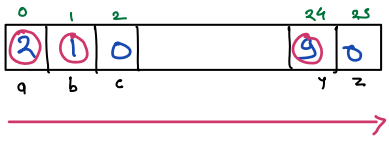
Char		index
a	97 (-97)	0
b	98 (-97)	1
c	99 (-97)	2
⋮		⋮
z	122 (-97)	25

S[i] - 'a'



// Count freq.

- int count[26] = {0};
  for (i=0; i<N; i++){
    // index for S[i] → S[i] - 'a'
    count[S[i] - 'a']++;
  }



// Build the string as per count, in dict order.

```
K = 0;
for (i=0; i<26; i++){
    // count[i] stores freq of i + 'a'
    for (j=0; j< count[i] ; j++){
        S[K] = char(i + 'a');
        K++;
    }
}
```

i	j	# iterations
0	[0, |a|]	|a|
1	[0, |b|]	+ |b|
2	[0, |c|]	+ |c|
⋮	⋮	+ ⋮
25	[0, |z|]	+ |z|
		N

TC : O(N) + O(N) ⇒ O(N)

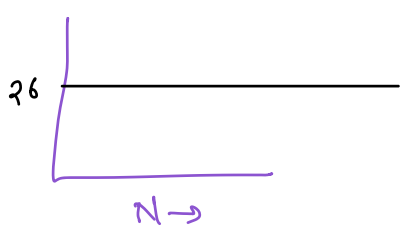
count freq. Build the string

SC (Extra) : O(1) → O(N) (StringBuilder)
(if string is immutable)

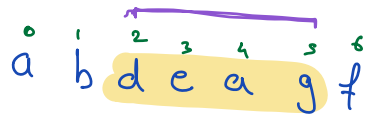
LC chr	26
(LC+UC)	52
All chars	256

if String is mutable

Counting Sort

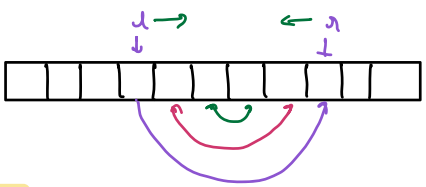


Q Given a string s & two indices l & r.
Reverse the sub-string from l to r



a b g a e d f

l : 2 , r : 5




```
reverse (s, l, r) {
    while ( l < r ) {
        swap( s[l] , s[r] );
        l++;
        r--;
    }
```

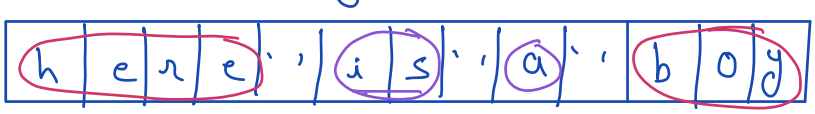
TC : O(N)
SC : O(1) → Mutable Str
O(N) → Immutable Str.

Amazon

Q Given a Character array storing a sentence.
Reverse it word-by-word

* No extra space is allowed
* Every word is separated by a single white space (" ")
* No inbuilt method (reverse(), Split())

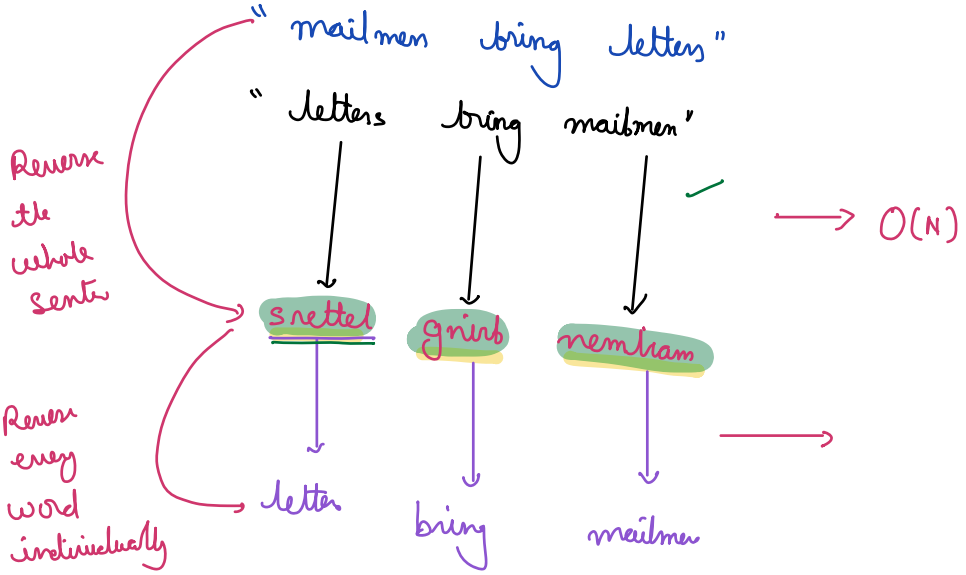
x here - is - a - boy x



boy a is here

"Are you as clever as I am"

"am I as clever as you Are"



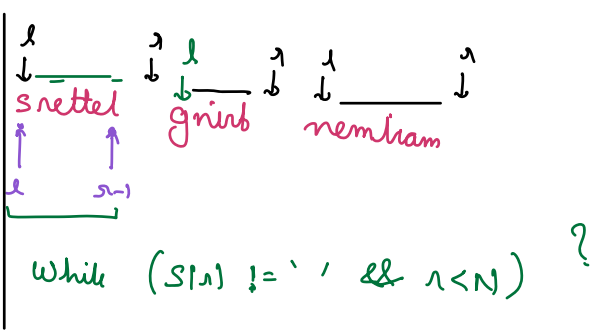
TC : O(N) + O(N)

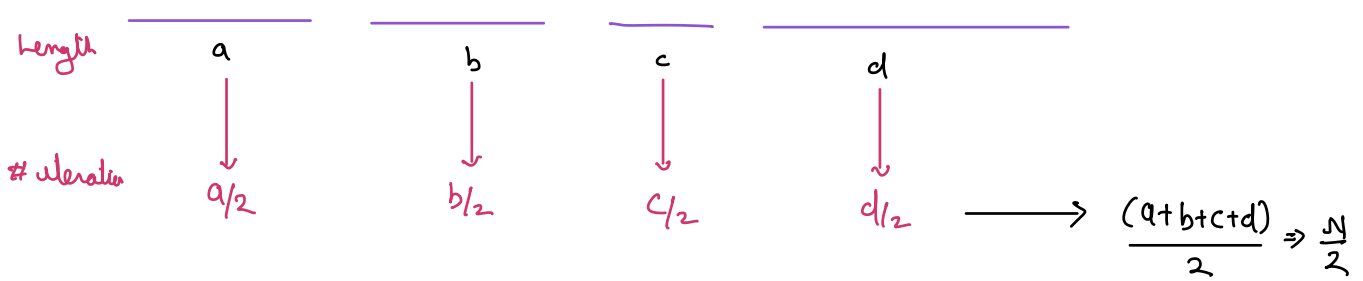
Reverse whole sentence

Reverse all words individually

SC : O(1)

Tricky





Amazon Direct-i Qtn

Q Given a string of size N. (lowercase alphabets)
Return the length of longest palindromic substring.

Palindrome

→ MOM ←
→ MADAM ←
→ MALAYALAM ←

s: a b a c a b

3

⇒ 5

aba → 3
acc → 3
bacab → 5

s: a b c a l m

⇒ 1

Approach 1

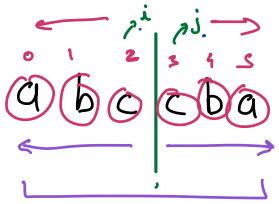
- Iterate over all substring : $O(N^2)$.
- Check if substring is a palindrome : $O(N)$
- Maintain max length.

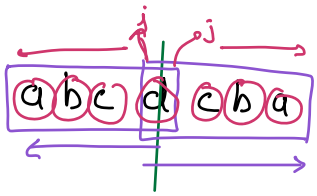
TC: $O(N^2 \times N) \rightarrow O(N^3)$

No of substrings
$N + (N-1) + (N-2) + (N-3) + \cdots + 1$
$= \frac{N \times (N+1)}{2}$

Approach II

⇒ 2^n





//Return length of longest palindrome centered b/w i & j

int getPalLength (s, i, j) {

}