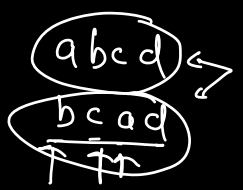
String ⇒ ~~group of char~~

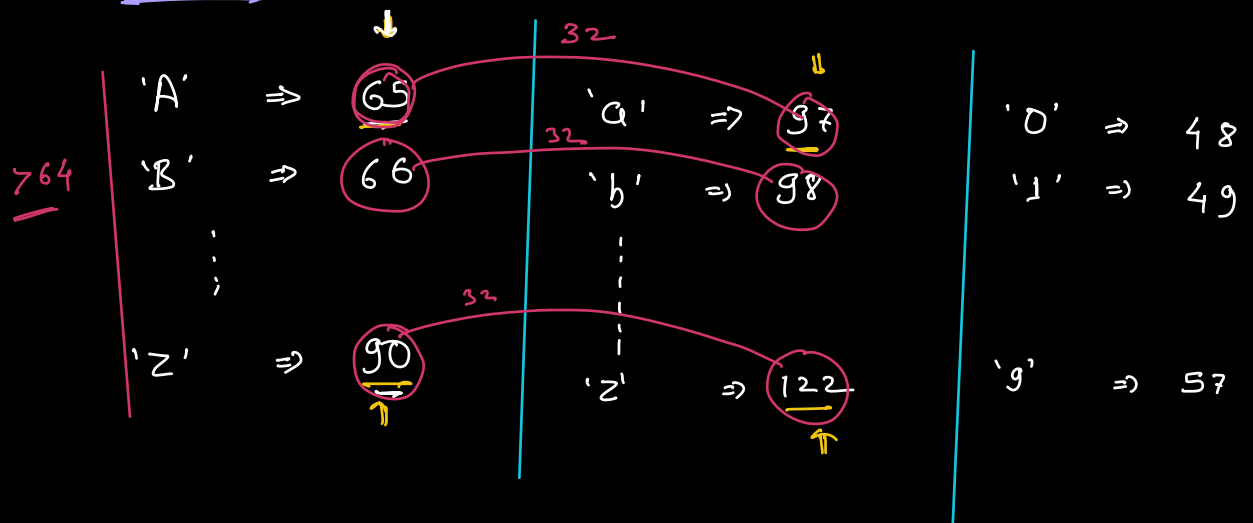
seq of char

Array of char

list of char



ASCII



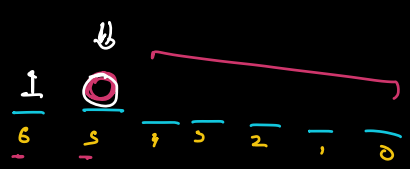
StringBuilder ⇒ Java

String ⇒ Array / List of char

$32 \Rightarrow 2^5 \Rightarrow 100000$

'z' ⇒ 122 ⇒ 1111010

'Z' ⇒ 90 ⇒ 1011010



'Z' ^ 32

⇒ 90 ^ 32 ⇒ 122 ⇒ 'z'

'z' ^ 32 = 122 ^ 32 ⇒ 'Z'

64
32
96

Q Given a string S. Toggle the case of every char.

Upper case → lower

lower case → Upper

S: a B c A E d ⇒ A d C a e D

[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 (S[i] >= 'A' && S[i] <= 'Z') {
            S[i] += 32;
        }
    }
}
```

'a' → 97

'z' → 122

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

'a' - 'A'

TC: O(N)

SC: O(1) (Extra)

Q Given a string. Sort the string in dict order.

↳ lower case alph

s: d a b a e d b

↳ a a b b d d e

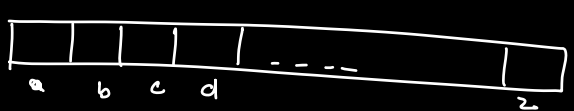
lower case alphabets ⇒ [a-z] ⇒ 26 unique char.

⇓

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

→ a a b b d d e

int count[26] = {0}



a ⇒ 97 $\xrightarrow{-97}$ 0

b ⇒ 98 $\xrightarrow{-97}$ 1

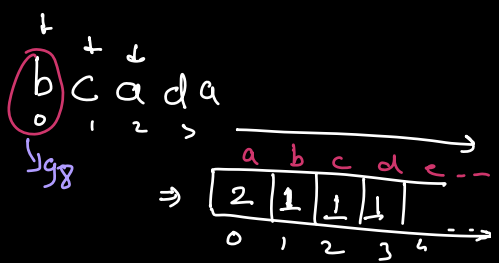
c ⇒ 99 $\xrightarrow{-97}$ 2

d ⇒ 100 $\xrightarrow{-97}$ 3

⋮

z ⇒ 122 $\xrightarrow{-97}$ 25

s[i] - 'a'



98 - 97 ⇒ 1
99 - 97 = 2
97 - 97 = 0
100 - 97 = 3

```
int count[26] = {0}
for(i=0; i<N; i++){
    // index of s[i] in count[]
    int index = s[i] - 'a';
    count[index] = count[index] + 1;
}
```

```
K = 0;
for (i=0; i<26; i++) {
    // C[i] ⇒ freq of (i+97)
    ⇒ for(j=0; j<C[i]; j++) {
        ⇒ [ str[K] = i + 'a';
          K++; ]
    }
}
```

Str

i	j	
0	[0, C[0]]	⇒ |a| ⇒ str
1	[0, C[1]]	⇒ |b|
2	[0, C[2])	⇒ |c|
:	:	:
25	[0, C[25]]	|z|

N

TC : O(N)

SC : O(1) (Extra)

Count Sort

Q Given a string s.

Given a l & r index.

Reverse the string from index l to r.

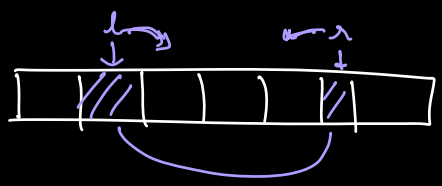
[Without using any inbuilt function]

a b d e a g f

0 1 2 3 4 5 6

[2, 5]

a b g a e d f



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

TC : O(N)

SC : O(1) (Extra)

Amazon, Adobe, MS

Given a character array storing a sentence.
Reverse it word-by-word.

* No extra space before/after the sentence.
* Every word is separated by a single white space.

here_is_a_picture

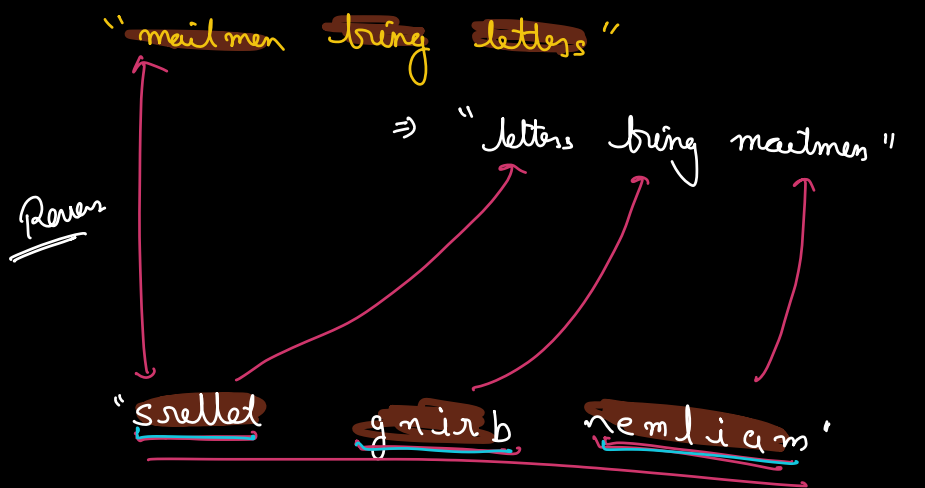
| h | e | r | e | ' ' | i | s | ' ' | a | | | | |

picture_a_is_here

* No inbuilt methods (split(), reverse())
* Using constant extra space.

"are you as clever as I am"

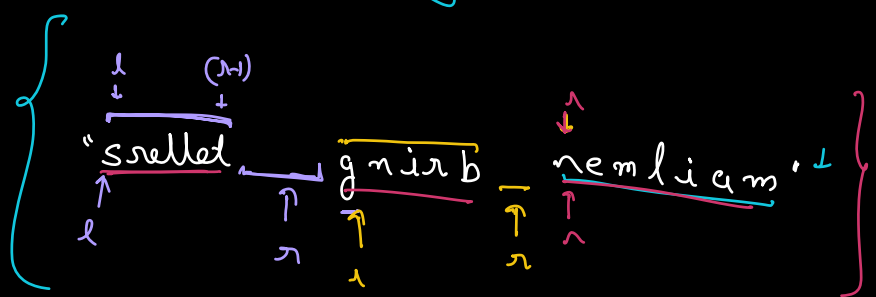
⇒ am I as clever as you are



Step I : Reverse the complete sentence. ⇒ O(N)

Step II : Reverse all words individually. ⇒ O(N)

Tricky



Amazon
Direct-i
Ola

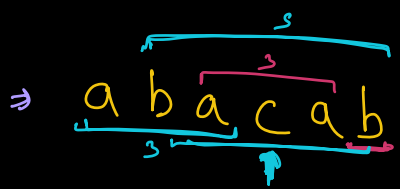
Given a string.

Return the length of longest palindromic sub-string.

→ madam ←

→ aba ←

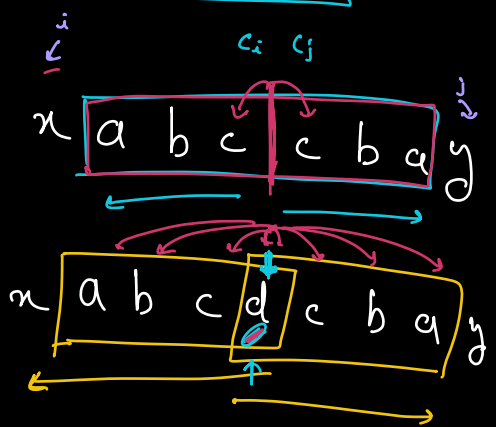
→ NAMAN ←



⇒ 5

a b c d b

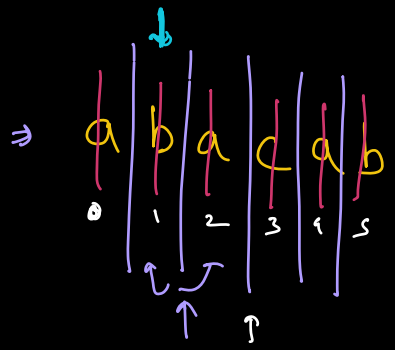
⇒ 1




```
int getPalLength(S, i, j){
    while(i>=0 && j<N && S[i]==S[j]){
        i--;
        j++;
    }
    ret j-i-1;
}
```

(a, b) = j - i - 1

TC: O(N)



getPalLength(S, 0, 0)
getPalLength(S, 0, 1)
getPalLength(S, 1, 1)
getPalLength(S, 1, 2)
getPalLength(S, 2, 2)
getPalLength(S, 2, 3)

```
ans = 1;
for(i=0; i<N; i++){
    // odd length Pal
    ans = max(ans, getPalLength(S, i, i));   // O(N)
    // even length Pal
    ans = max(ans, getPalLength(S, i, i+1));
}
```

TC : $O(N^2)$
SC (Extra) : $O(1)$

DP

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

$O(N) \Rightarrow$ Manacher's Algo