Strings → sequence

group of chars ✗

sequence of chars

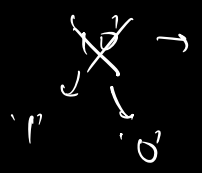
array / list of chars ✓

⇒ 'A' → 65 'a' → 97 '0' → 48

'B' → 66 'b' → 98 '1' → 49

⋮ ⋮ '2' → 50

'Z' → 90 'z' → 122 ⋮



Strings being immutable ←→

N → N

⇒ TC ⇒ O(N2)

⇒ SC ⇒ O(N2)

Immutable String → StringBuilder ✓

Q 1. toggle the case

↑ -32 | +32

↑ toggle the 5th bit

Q 2. given lowercase chars in dict order.

freq array | count array

Q 3. Given a string S and two indices l & r.
reverse the substring from l to r.

0 1 2 3 4 5 6

ex ⇒ S = a b d e a g f

l

r

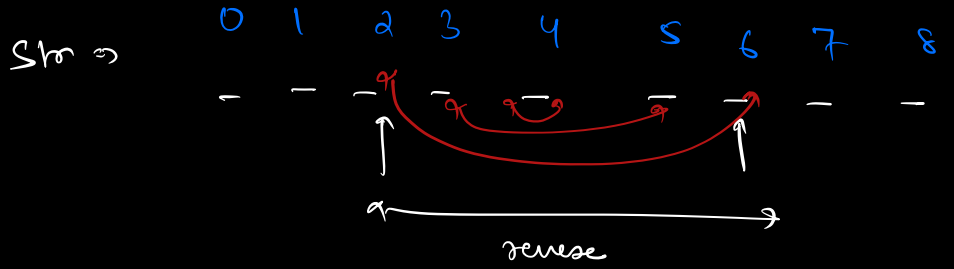
l = 2

r = 5

o/p ⇒ a b g a e d f

l

r



pseudo

```
reverse PartStr ( str, l, r) {

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

TC ⇒ O(N)
SC ⇒ O(1)

Amazon

Q4. Given a character array storing a sentence, reverse it word by word.

* no extra space
* every word is separated by a single space (" ")
* no inbuilt method

eg ⇒

I/P ⇒ ch ⇒ | h | e | r | e | - | i | s | - | a | - | b | o | y |

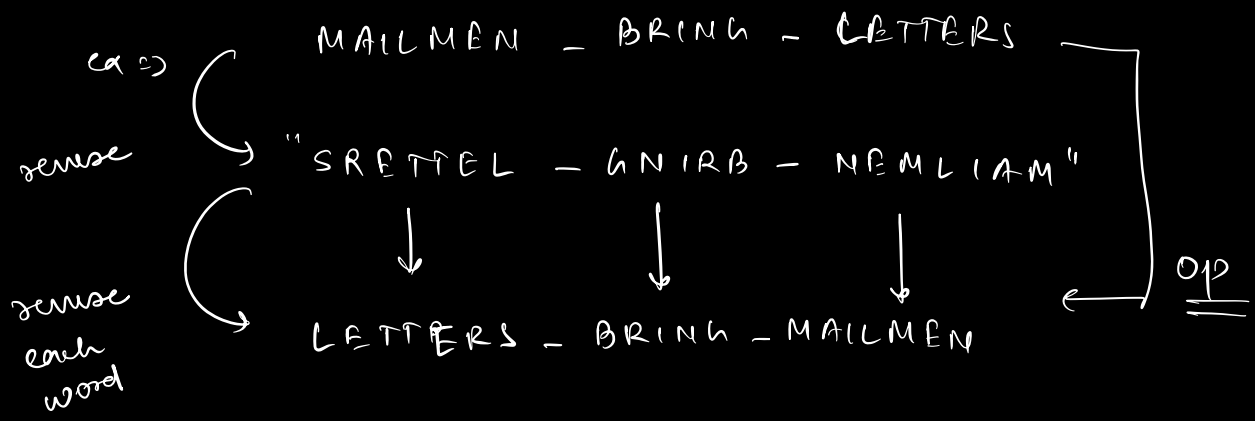
O/P ⇒ | b | o | y | - | a | - | i | s | - | h | e | r | e |

Ip ⇒ ch ⇒ ARE _YOU _AS_CLEVER_AS _I_AM

Op ⇒ "AM_I_AS_CLEVER_AS_YOU_ARE"

Ip ⇒ ch ⇒ "MAILMEN _BRINGS_LETTERS"

Op ⇒ "LETTERS_BRINGS_MAILMEN"



1) reverse the entire string

2) reverse each word

1) "SRETTEL _ SGNIRB _ NEMLIAM"

l r

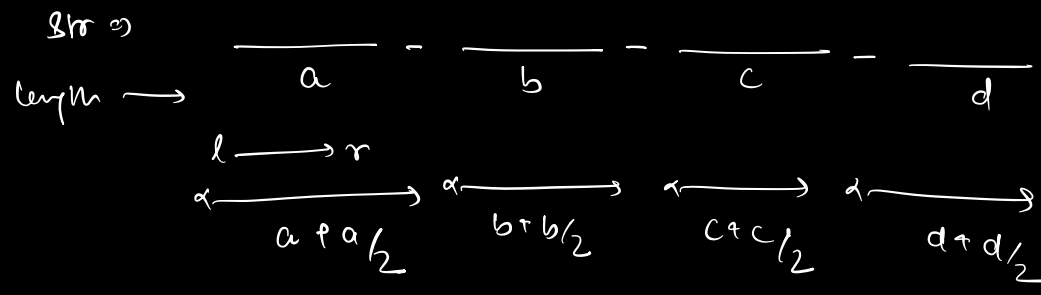
reversePart(str, l, r-1)

(l, r-1)

(l, r-1)

iterations ⇒ $7 + 7/2 + 5 + 5/2 + 3 + 3/2 = 3/2(7+5+3)$

$= 3/2(N)$



iterate ⇒ $a + a/2 + b + b/2 + c + c/2 + d + d/2$

$= \frac{3a}{2} + \frac{3b}{2} + \frac{3c}{2} + \frac{3d}{2}$

$= \frac{3}{2}(a+b+c+d)$

$= \frac{3}{2}(N)$

TC ⇒ $O(N)$

SC ⇒ $O(1)$

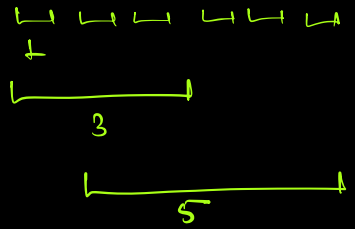
✓

Q. Given a string, calculate length of longest palindromic substring.

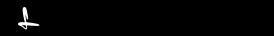
palindrome ⇒ aba → aba ✓

madam → madam

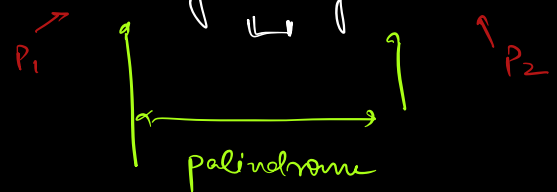
ex ⇒ a b a c a b O/P ⇒ 5



ex ⇒ a b c d e O/P ⇒ 1



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
str ⇒	x	b	d	y	z	z	y	d	b	d	y	z	y	d	x



$l = P_2 - P_1 - 1$

* z is a centre, how?
* only for odd len, palindrome.

```
int expand(str, P1, P2) {
    while(P1 >= 0 && P2 < N &&
          s[P1] == s[P2]) {
        P1--  P2++
    }
    return P2 - P1 - 1;
}
```

⇒ O(N)

a b c b a

P1 P2

1) take every index as a odd len. palindrome centre, and expand on it, finding the len.

return the maxm length

maxm length of odd length palindromic substring

ex: b d y z z y d b

P1 P2

pseudo

```
ans = 1
for( i=0; i<N; i++){   // maxm odd length palindrome
    ans = max(ans, expand(str, i, i))
}

for( i=0; i<N-1; i++){   // maxm even length palindrome
    ans = max(ans, expand(str, i, i+1))
}

return ans
}
```

N (first loop), N (second loop)

TC ⇒ $O(2N^2) \simeq O(N^2)$	
SC ⇒ $O(1)$	

∴ Possible soln

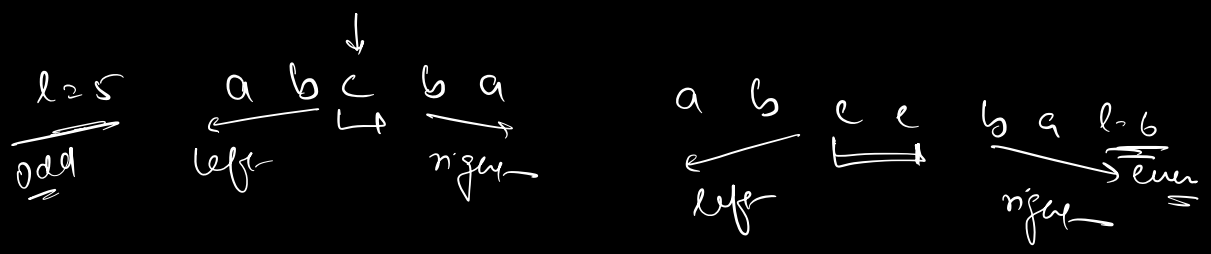
$O(N^3)$ ⇒ Create all possible substrings & check palindrome

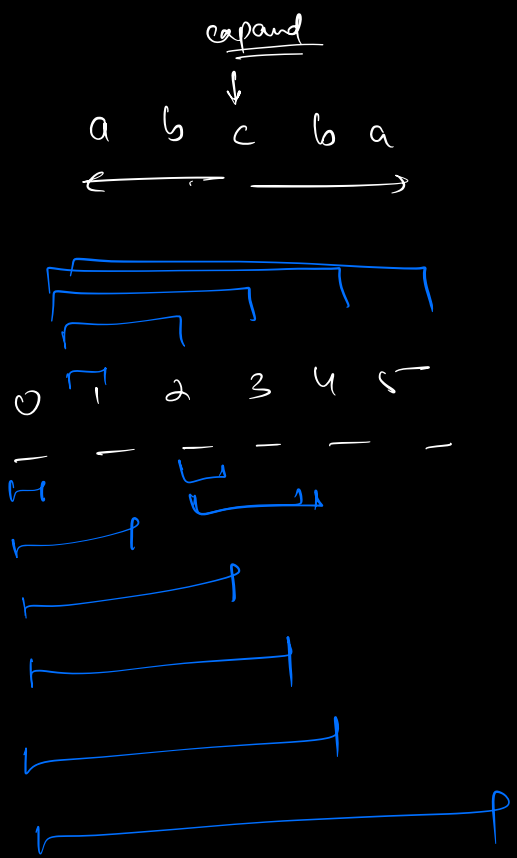
$O(N^2)$ → expand func.

$O(N^2)$ → DP (SC → $O(N^2)$)

Dynamic Programming

TC ⇒ $O(N)$
SC ⇒ $O(N)$] Manacher's Algo.





$\Rightarrow N^2$

$\approx N$

$= O(N^3)$