Update:

" Extra Class " → next Session (Wed)

↑

Doubt Problems / Extra Questions

- Popular Algorithm
  - moove voting Algo
  - DNF Sort
  - Noble Integer
  -
  -

Pattern Matching + Adv Batches (Later)

Strings Introduction (1 session)

↳ sequence of chars

Read only ↳ Java Strings are immutable. (String)

↳ char[] arr, ← mutable it has fixed size

Frequent updates

↳ String Builder ← mutable —— dynamic size + inbuilt method insertion, deletion etc.

✓

American Standard Code for Information Interchange

Storage$ [letter → ASCII value → Binary No]

'A'	65	11011101 101
'B'	66	"
'C'	67	"
'+'	?	⋮
Z	90	
'a'	97	⋮
'b'	98	
'5'	53	
'0'	48	
','	49	
'$'	?	

$$\begin{tikzcd} \text{'A'} \arrow[r, "+32"] & \text{'a'} \\ 65 \; U & 97 \; L \arrow[l, bend left, "-32"] \end{tikzcd}$$

$S[i] = S[i] + 32$ ✓

U → L

$-65 + 97$ ✓

$\Rightarrow S[i] = S[i] - \text{'A'} + \text{'a'}$

$S[i] \Rightarrow \underbrace{\text{'B'} - \text{'A'}} + \text{'a'}$

$= \text{'b'}$

lower case ↔ UpperCase

Ⓠ Given a Char[] , toggle case of ever character in the array

[without using any library function]

S[] =

S	n	A	k	e

↓

s	N	a	K	E

'A' ≤ G ≤ 'z'

↓

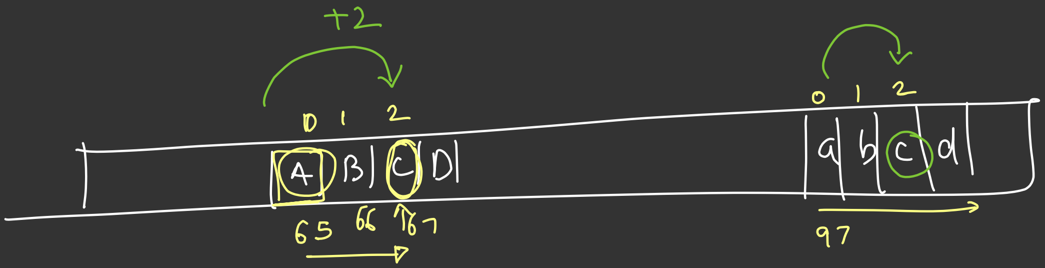
g

① Check → U or L

② Toggle step

```
for(i=0, i< S.length, i++){

    if( s[i] >= 'A' && s[i] ≤ 'z' ){        // upper

        S[i] = (char)(S[i] +32);

    }
    else {   S[i]= (char) (S[i] -32),
                                            }
}
```

clear ☺



$$'C' - 'A'$$

$$= 67 - 65$$

$$= \boxed{2}$$

$$= 'a' + 2$$

$$= 'c'$$

$$\Rightarrow \underbrace{S[i] - 'A'} + 'a'$$

= S[i] in <u>lower case</u>

Q Given a character arr [ ], which contains only lowercase letters

Sort the given character array.

'a' > 't'
↓ ↓
65 > 92 (automatic)

Input

↳ | b a t m a n |

↓

output

| a a b m n t |

↑ ↑

Solution →

~~1~~ Bubble Sort O(N^2) time

Comparison

↳ if ('a' > 't') ✓

↳ false

$a[j] > a[j+1]$

~~2~~ Arrays.Sort(- - -) inbuilt fn

O(N Log N) ← proof : later in adv batch.

int [ ] freq = new int [26];

Java

3 step-1

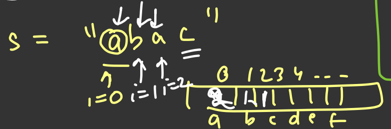
O(N) step

```
freq [26] = 0;
for(i=0, i< s.length(); i++){
    freq [ s[i] - 'a' ] += 1,
```

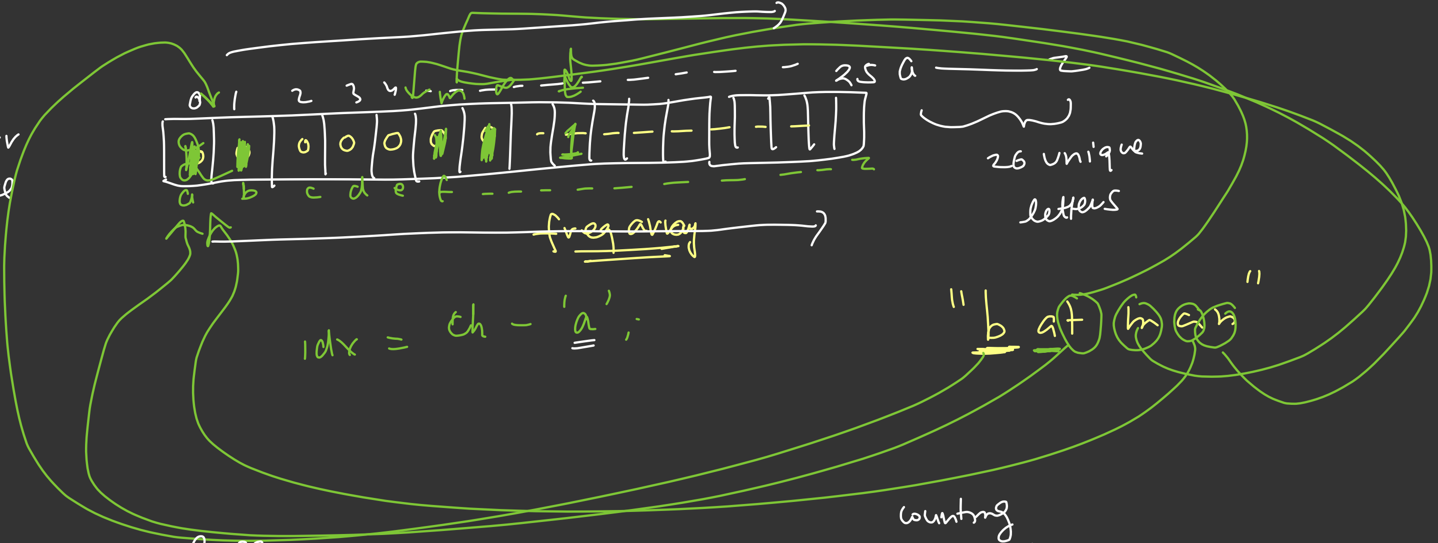
+=

a → 0

c → 2



freq arr of size 26 init with 0


0 1 2 3 4 m - - - - - - - 25 a —— z
0 0 0 0 0 0 1
a b c d e f - - - - - - - - - - z
freq array
26 unique letters
" b a t m a n "

idx = ch - 'a';

Step-2

Range

```
for( i=0 , i<26 , i++){
    while( freq[i]!=0){
        => print( (char)(i+'a'));
        freq[i]--;
    }
}
```

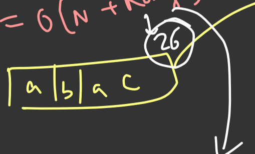
K → or some work

Put in back in orig Array

Total work across is O(N+26) = O(N + Range)

counting

a -	6
b -	0
c -	6
d -	0
e -	0
f -	0



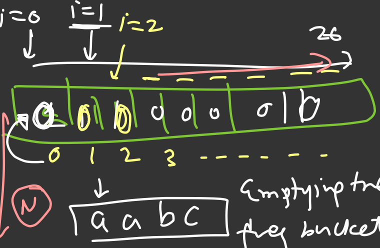
0 => (0+'a') => 'a'

0 => (0+'a') => 'a'

(0+1) => b

(0+2) => c

N



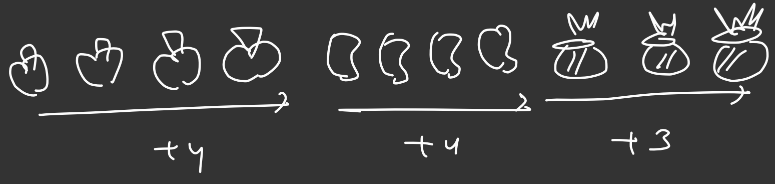
Emptying the freq buckets

= O(N) for this problem

3 Fruits

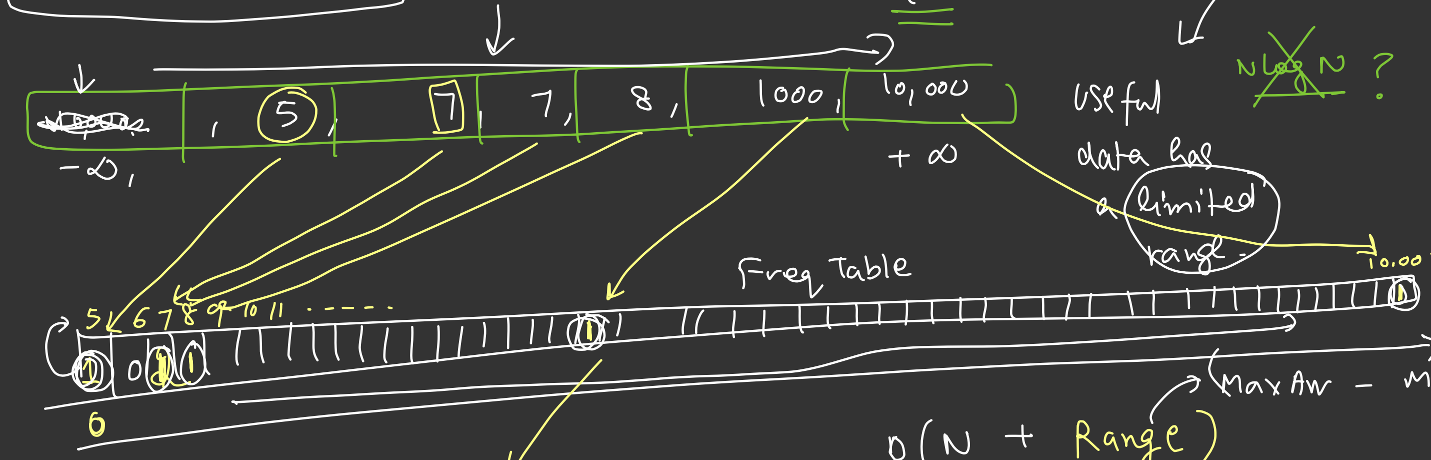
O(N)

Apples -	~~3~~	$\boxed{4}$
mangoes -	~~3~~	$\boxed{4}$
Pineapples -	~~2~~	3



= O(N) time

Counting Sort



Can't work with all kinds of data

5, 7, 7, 8, 1000, 10,000

O(N + Range)

Range → (Max Arr - Min Arr)

↓ Small NO

~~(-∞ to ∞)~~

Example

1 M sat for an exam. 300 marks

↳ 0 — 300 for every student

S1 — 280

S2 — 30

S3 — 160

S4 — 70

S5 — 75

⋮

same marks
↓
same Rank

Marks freq [301]

Arrays.sort (N log N)

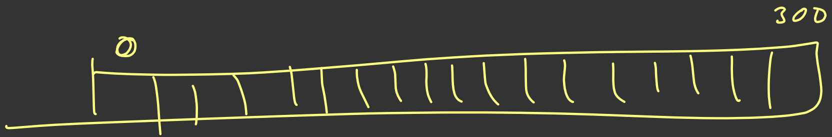
Count Sort (ΦN + Range)

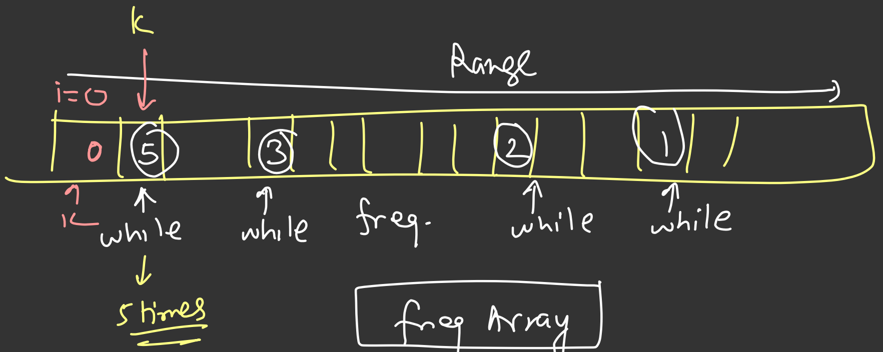
↓ 1M

↓ 300

//

Linear = O(N)





fundamental

freq Array

i=0 → k

k

i=1 → 5k

⋮

k

→ 3k

k

→ 2k

k

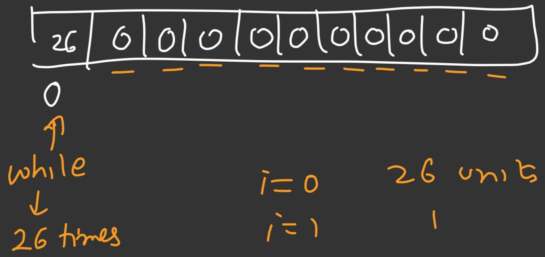
→ 1k

k

k

⋮

⇒ a a a a a a a a a a a a a



i=0 26 units

i=1 1

i=2 1

⋮

$\boxed{k}$ + $(5+3+2+1)$ k

26 times

i = 26 $\frac{1}{52 \text{ unit}}$

= O(N + Range

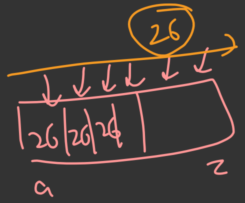
= Range·(K) + N (K)

↓ constant

len = N = 26 x 26 =

O(Range + N)

= ✓



freq

O($\underline{\boxed{N}}$ + Range)

↓

26 x26

Linear in terms of string length.

26 x 26

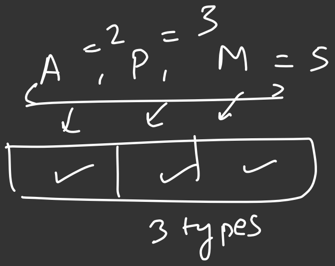
0 —— 10,000

0 —— 300

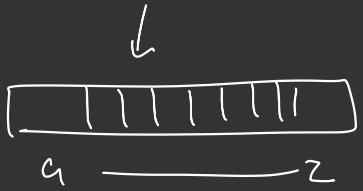
Space

O (Range) in general

O(1) for a-z



3 types

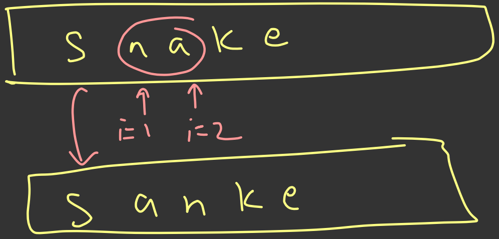


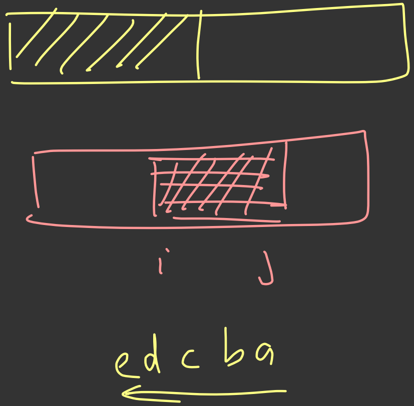
O (Range)

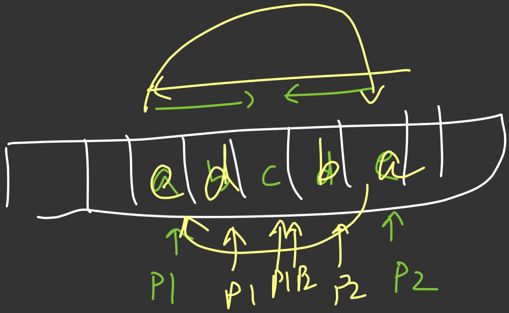
Q Given a String, i want to reverse a part of it.

Char []

Substring







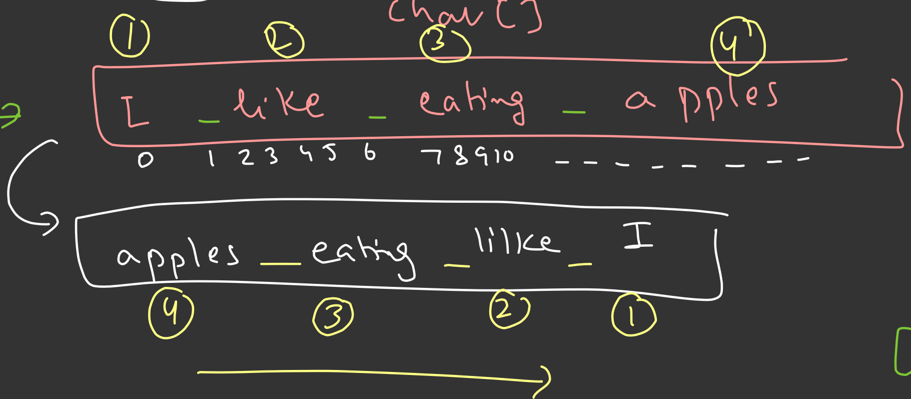
2 Pointer Approach

entire or a part

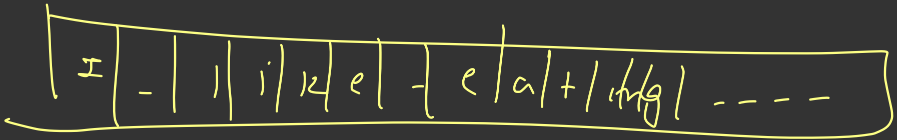
```
while (P1 < P2) {
        swap( arr[P1], arr[P2] )
        P1++,  P2--
}
```

Q Given a char[], Reverse it word by word => without using [any library function]

single char array →

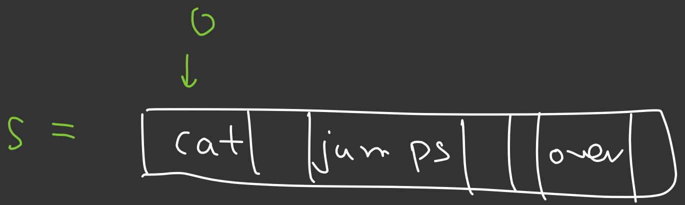


~~["I", "like", "eating"~~

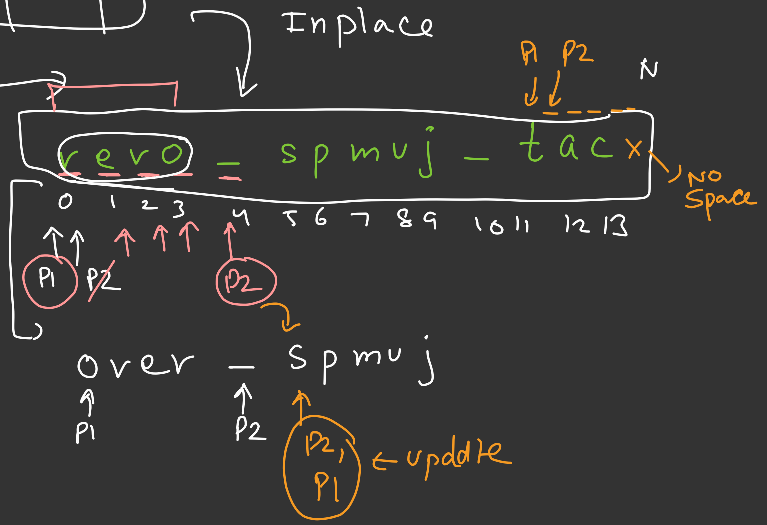


10:35

Break :)



→ Reverse (s, 0, n-1)



Next step → Rev all words sep by space

P1 = 0 , P2 = 0

```
while ( P1 != N) {
    // move P2 until you see a space
    → while (P2<N && s[P2] != '_') { P2 = P2+1 },
    reverse (s, P1, P2-1);
    P2 = P2+1 ,
}
```

Do bounds check first

p2 ==n

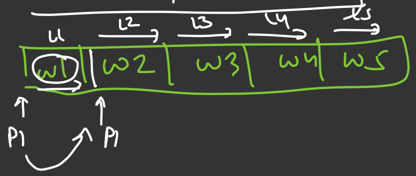
→ ① (s[p2] != '_') && (p2<N) ②

Array out of Bounds exception

```
F
(P2 < N) && s[p2] != 'space'
              ↑
        never evaluated
    ↳ P1 = P2 ;
}
```

Short circuit in AND gate

Step-1 Rev → O(N)



L1
L2
L3
L4
L5

L1+L2+---L+

= O(N)

will be eval

T		T/F		Rest
exp1	&&	exp2	=	T/F
F				
exp1	&&	exp2	=	F

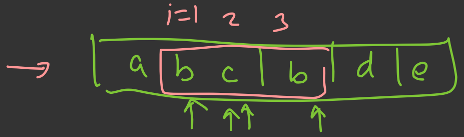
Time → O(N)

Space → O(1)

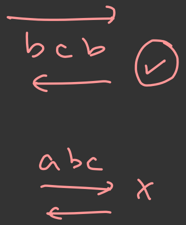
↑

inplace

Q) Given a char[], check if a given substring is palindrome.



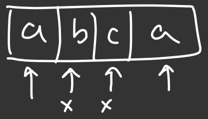
substring(s, 1, 3)



```
bool isPalindrome(char s[], int s, int e) {
        ≡        TODO
3
```

O(N) time

O(1) space



$O(\frac{n}{2}) \rightarrow O(n)$

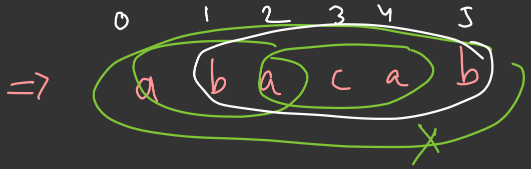
↑

x

Q) Given a string, calc the longest palindromic substring?

→ min L = 1

→ max L = N



"b"

"aba"

"aca"

"bacab" ✓

[1, 5]

5 - 1 + 1

=

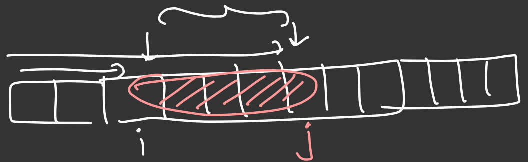
j - (i - 1)

= j - i + 1

$j \geq 1$

✓

Brute force → App-1



L = 0

Find out all subarrays N^2

&

Track the largest N

```
for ( i=0 , i≤N-1 , i++ ){
    for (j=i ; j≤N-1 ; j++) {
        //Subarray → (i, j)
        if ( is Palindrome (s, i, j) ) { L = max (L, j-i+1) }
    }
}
```

O(N)

→ overall $O(N^3)$ time

→ N^2 substrings × N time for palindrome

$\boxed{a\ b\ a\ d}$

i=0 i=1

$L = \cancel{0}\ 3$

j=0 $[0,0]$ = "a" → $0-0+1$ → ①

j=1 ab ×

j=2 aba → $2-0+1$ → ③

j=3 abad → ×

i=1

j=1 b

j=2 ba

j=3 bad

i=2

a

ad

i=3

d

→ Large value of N that can pass 1 s (TLE) for this solution

1~~00~~0

$\boxed{N = 10^3}$ → TLE

$N^3 = 10^9 \gg 10^8$

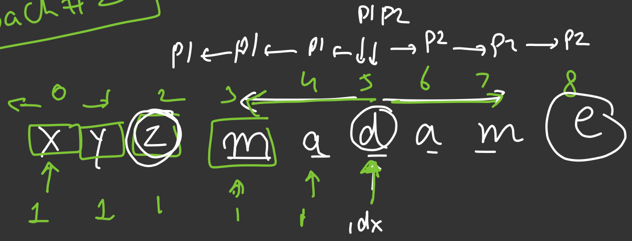
$N = 200 \times 200 \times 200$

$= 8\,00\,00\,00$

↓

$= \boxed{10^7}$

Hint for Approach#2

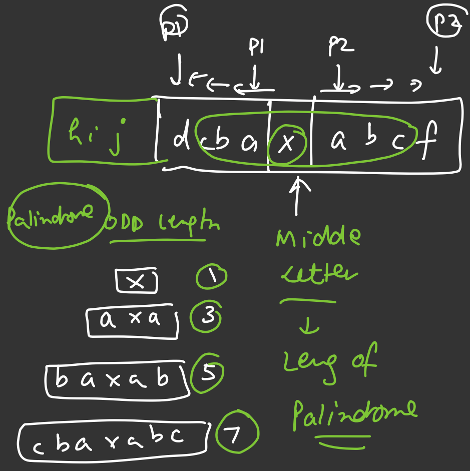


Goal → Find largest palindromic substring

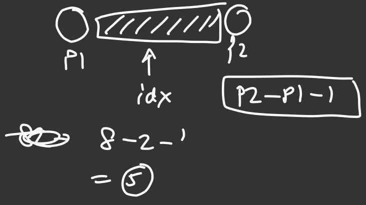
① Go to every idx, treat this as middle idx and find the largest palindrome

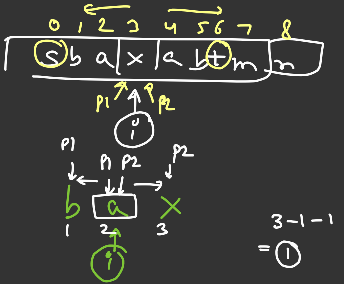
```
int expand ( char s[], int p1, int p2 ) {
        N = s.length;

    while ( p1 >= 0 && p2 < N && s[p1] == s[p2] ) {
            p1-- ,   p2++
    3

    return p2 - p1 - 1;
}
```

O(N)



leng =





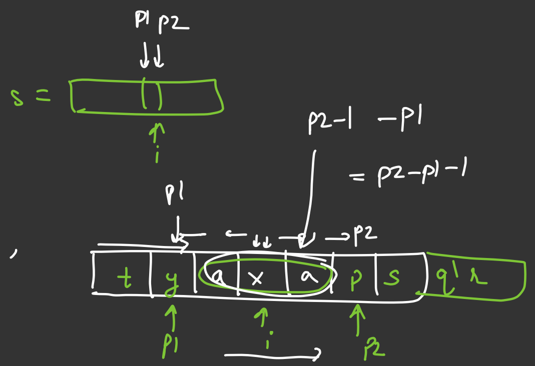
$O(N^2)$

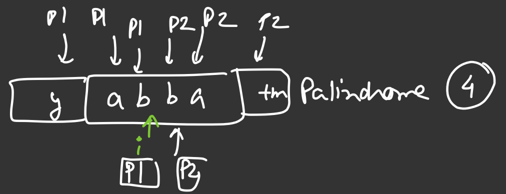
```
ans = 0

for ( i=0 , i< N , i++ ) {
    // s[i] is the center
    p1 = p2 = i

    (N) ans = max (ans, expand(s,p1,p2)) ,
}
```

N^2

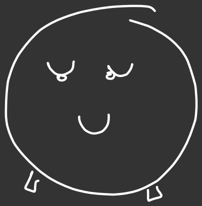
```
for ( i=0 , i< N , i++ ) {
    // s[i] is the center of even length
    p1 = i, p2 = i+1 ,  ← even

    (N) ans = max (ans, expand(s,p1,p2)) ,
}
```

N^2





At every idx,
we can find
a palindrom in
O(N)



Good Night