Today's Quote

Dream big, stay positive, work hard, and enjoy the journey.

Today's content

→ Intro

→ flip

→ sort ch[ ]

→ Reverse string

→ Longest palindromic substring

String.
- → array of characters
- → sequence of characters
- → bunch of characters → {a b c}
  → {b a c}

* Not same.
order is also important.

characters → ASCII value.

'A' - 65	$\xrightarrow{+32}$ $\xleftarrow[-32]{}$	'a' - 97	'0' - 48		$\boxed{32 \to 2^5}$
'B' - 66	$\xrightarrow{+32}$ $\xleftarrow[-32]{}$	'b' - 98	'1' - 49		
'C' - 67		'c' - 99	'2' - 5		
'D' - 68		'd' - 100			
|		|			
'Z' - 90		'z' - 122	'9' - 57		

'10' → It is not a single character

char ch = '9'
$\underbrace{\text{char ch}}_{2\text{ bytes.}}$
ASCII = 57

ch = '9'
ch = ch+8
print(ch) Ascii → 65
'A'

string. → array of characters.

String s = " a b c d"

s →

a	b	c	d
0	1	2	3

→ print(s[2])
→ c will get printed.

Q) Given a char[ ], toggle every character.
↳ Capital ⇌ Small

Note → Input contains only small & capital characters.

Eg → AnaConDa

o/p → aNAcONdA

```
toggleCharacters( s[ ] , N){
    for( i = 0 ; i < N ; i++) {
        if (s[i] >= 65 && s[i] <= 90){
            // s[i] is capital
            s[i] += 32 ;
        }
        else{ // s[i] is small
            s[i] -= 32 ;
    }   }
}
```

s[i] = s[i] ^ 32

or

s[i] = s[i] ^ (1 << 5)

T.C → O(N)
S.C → O(1)

	2^7	2^6	2^5	2^4	2^2	2^2	2^1	2^0
A: 65	0	1	0	0	0	0	0	1
B: 66	0	1	0	0	0	0	1	0
C: 67	0	1	0	0	0	0	1	1
|								
Z: 90	0	1	0	1	1	0	1	0

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
a: [97]	0	1	1	0	0	0	0	1
b: [98]	0	1	1	0	0	0	1	0
c: [99]	0	1	1	0	0	0	1	1
|								
z: [122]	0	1	1	1	1	0	1	0

Q: Given a char array, which contains only lower-case alphabets. Sort given ch[] in alphabetical order.

Constraints

$1 <= N <= 10^5$

$'a' <= ch[i] <= 'z'$

Ex: s = d a b a c d b

After ⇓ sorting

s = a a b b c d d

ideas

① Sort char[] using bubble sort.

T.C → $O(N^2)$

S.C → $O(1)$

② Use inbuilt sort function + custom comparator. {if needed}

T.C → $O(n\log n)$.

③ s = d a b a c d b

'a' → 2

'b' → 2

'c' → 1

'd' → 2

'a' − 'a' = 0

'b' − 'a' = 1

'c' − 'a' = 2

⋮

'z' − 'a' = 25

S = a a b b c d d

count[26] →

2	2	1	2	0	----	0	0
0	1	2	3	4	----	24	25
'a'	'b'	'c'	'd'	'e'		'y'	'z'

count[0] → frequency of 'a'

count[1] → frequency of 'b'

count[2] → frequency of 'c'

| |

count[25] → frequency of 'z'

```
sortString ( char[] s , N) {
    count [26] ;
    for ( i = 0 ;  i < N ; i++) {              N
        idx = s[i] - 'a'
        count [idx]  += 1 ;
    }
    K = 0
    for( i = 0 ; i <= 25 ; i++){               n
        ch = i + 'a' ;
        for( j = 1 ; j <= count[i]; j++){
            s[K] = ch
            K += 1
        }
    }
}
```

T.C → O(N)

S.C → O(1)

i	j	iteration
0	[1, c[0]]	c[0] +
1	[1, c[1]]	c[1] +
2	[1, c[2]]	c[2] +
⋮		⋮ +
25	[1, c[25]]	c[25]

= $\boxed{n}$.

Eg : s → b b c c d d

0 1 2 3 4 5 6

count →

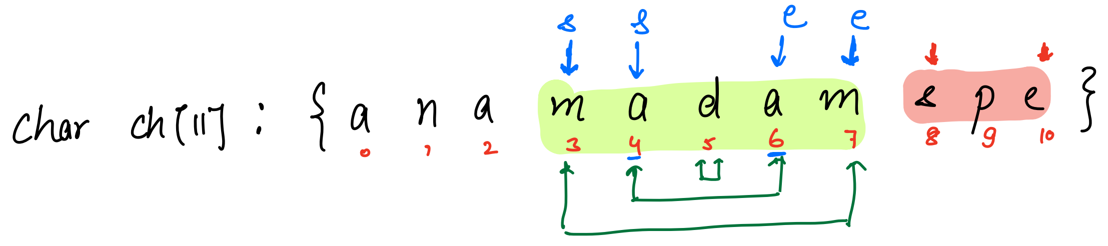
0	2	2	2	1	0	—	—	0
0	1	2	3	4	5	—	—	25

Substring concept is same as sub-array.

1) continuous part of a string
2) full string can be sub-string.
3) A single character can be a substring.

Q: Check if given substring is palindrome or not.

Eg: madam nayan level
mam civic malayalam
dad radar


Char ch[11] : { a n a m a d a m s p e }

start index of the substring

end index of the substring.

```
boolean isPalindrome ( ch[] str, s , e ){
        while ( s < e ){
            if (ch[s] != ch[e]){
                return false;
            }
            s++ , e--
        }
        return true;
}
```

T.C → O(N)
S.C → O(1)

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

Eg: a b a c a b
0 1 2 3 4 5
[ans=5]

Eg: a b c d e
0 1 2 3 4
[ans = 1]

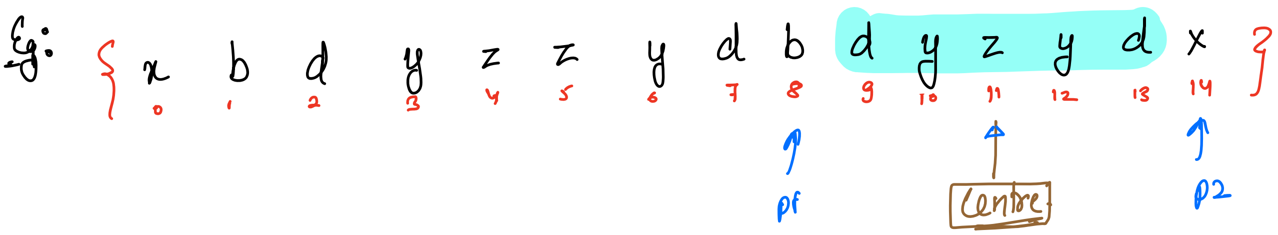
idea-1: For all the substrings, check if it is palindrome or not & get the max-length.

$$\frac{n(n+1)}{2} * n \Rightarrow \boxed{\begin{array}{l} T.C = O(N^3) \\ S.C = O(1) \end{array}}$$

constraints $1 <= N <= 3 \times 10^3$

```
int longestPal ( char[] s , N) {
    ans = 0;
    for( i = 0 ; i < N ; i++) {          // i → start index of the substring
        for( j = i ; j < N ; j++) {      // j → end index of the substring.
            // substring [i, j]
            if( isPalindrome ( s, i , j ) ){
                ans = Max (ans, j - i + 1 );
            }
        }
    }
    return ans
}
```

Eg:



length = (P_1, P_2)

$= P_2 - P_1 - 1$

idea:

Take any idx as the centre and try to expand to find the length of longest palindromic substring. ⇒ max odd length palindrome.

$\boxed{T.C \rightarrow O(N^2)}$

max length palindrome.

Take any two consecutive characters as the centre & try to expand to find the length of longest palindromic substring. ⇒ max even length palindrome.

$\boxed{T.C \rightarrow O(N^2)}$

$\boxed{\begin{array}{l} T.C \rightarrow O(N^2) \\ S.C \rightarrow O(1) \end{array}}$

```
int  expand ( char s[] ,  p1 , p2 ){
    while( p1 ≥ 0 && p2 < N && s[p1] == s[p2] ){
            p1--
            p2++
    }
    return  p2-p1-1 ;
}

int  longestPal  ( char[] s , n ){
    ans = 0/1

    for( i = 0 ; i < n ; i++){    // max odd-length palindrome.
                //centre -> s[i]
          p1 = i  , p2 = i
        ans =  max  (ans  , expand ( s , p1 , p2 ));
    }
                                    i = N-1 , p1 = N-1 , p2 = N,
    for( i = 0 ; i < n-1 ; i++) {
            //centres -> s[i] , s[i+1]
          p1 = i   , p2 = i+1
        ans =  max ( ans , expand(s, p1, p2));
    }

    return ans;
}
```

Doubts ->

Eg:

			P_1 ↓	~~P_1~~ ↓			~~P_2~~ ↓	P_2 ↓								
{	x	b	d	y	z	z	y	d	b	d	y	z	y	d	x	}
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	

i = 4

P_1 = ~~4~~ ~~3~~ 2

P_2 = ~~5~~ ~~6~~ 7

```
for( i=0 ; i < n-1 ; i++) {
    if (a[i] > a[i+1]) {
        swap a[i], a[i+1] ✓
        i = -1
    }
}
```

1+2+3+ — — n

$$\Rightarrow \frac{n(n+1)}{2} = n^2$$

		4			
		~~3~~	2		
	3	~~5~~	~~5~~	5	
	~~4~~				
arr →	~~5~~	~~4~~	~~3~~	~~2~~	1
	0	1	2	3	4

i = ~~0~~ ~~-1~~ ~~0~~ ~~1~~ ~~-1~~ ~~0~~ ~~-1~~ ~~0~~ ~~1~~ ~~2~~ 0

$\boxed{X = A \& B.}$ proof

CustomComparator.

$$X \wedge A \quad + \quad X \wedge B$$

$$(A \& B) \wedge A \quad + \quad (A \& B) \wedge B$$