Today's content → String.

- → Introduction
- → Flip
- → Sort a given string
- → Reverse string
- → Longest palindromic substring.

String:
- Sequence of chars
- Collection of chars
- Array of chars

abc → The strings are not same
acb → order matters

Characters: → 256 characters.

'a', '$'.

ASCII values → [0 1 2 ... 65 ... 255].

65 → 01000001

'A' → 65
'B' → 66
'C' → 67
...
'Z' → 90.

-32 +32 / +32 -32 / +32 -32

'a' → 97
'b' → 98
'c' → 99
...
'z' → 122

+32 / -32

'0' → 48
'1' → 49
'2' → 50
...
'9' → 57.

'10' → Not a single character.

'9' → ASCII value 57.

7 6 5 4 3 2 1 0
0 0 1 1 1 0 0 1

name = 'Keerthi'.

print('9' + 8).

57 + 8 = 65 → 'A'.

Python → Error.

ord() → ASCII value of character.
chr() → character from ascii value., chr(65) → A.

ord() and chr() method for python.

Should you use a library of string → ? → NO.

IQ: Given a string, Toggle every character.

(alphabets)

Note: Input contains only small & capital letters.

Ex: S = 'aNAcONdA'.

= 'AnaConDa'

Flip i^{th} bit →

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

Code:

```
def toggleword(s, n)
{
    for i in range(0, len(s))
    {
        if( ord(s[i]) >=65 && ord(s[i]) <=90)
        {
            // s[i] is capital.
            s[i] = chr[ord(s[i])+32]
        }
        else
        {
            // s[i] is small.
            s[i] = chr[ord(s[i])-32]
        }
    }
    return s
}
```

Without using 'if' & 'else'.

A → 65.

7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	1

B → 66

7	6	5	4	3	2	1	0
0	1	0	0	0	0	1	0

C → 67

7	6	5	4	3	2	1	0
0	1	0	0	0	0	1	1

Z → 90

7	6	5	4	3	2	1	0
0	1	0	1	1	0	1	0

a → 97

7	6	5	4	3	2	1	0
0	1	1	0	0	0	0	1

b → 98

7	6	5	4	3	2	1	0
0	1	1	0	0	0	1	0

2Q: Given a string, contains only lower case alphabets. Sort given string, in alphabetical order.

Ex: S = d a b a c d b.
= a a b b c d d.

Constraints:

$1 \le N \le 10^5$

$'a' \le s[i] \le 'z'$.

Brute force:

(i) using bubble sort. TC: $O(N^2)$
SC: O(1).

(ii) Inbuilt sort, TC: O(NlogN).
SC: O(1).

idea:

S = d a b a c d b.

S = a a b b c d d

'a' → 2
'b' → 2
'c' → 1
'd' → 2
'y' →
'z' →

256 characters.

[0 0 0 0 2 2 1 2 0 0 0 0 0 0 0 0 255]
0 1 2 a-b c d - - 97

Code:

```
def sortAlphabetically(S, n)
{
    C = [0] * 256 // filled with 0's.

    for i in range(0, len(S))
    |   index = ord(S[i])
    |   C[index] += 1

    k = 0

    for i in range(0, 256)
    {
    |   if(C[i] != 0)
    |   |   ch = chr(i) // 'a', 'b' etc.
    |   |   for j in range(0, C[i]+1)
    |   |   |   S[k] = ch
    |   |   |   k = k+1;
    }

    return S;
}
```

Counts of each character.

[0 0 0 0 3 4 5 0 0 0]
a b c
97 98 99

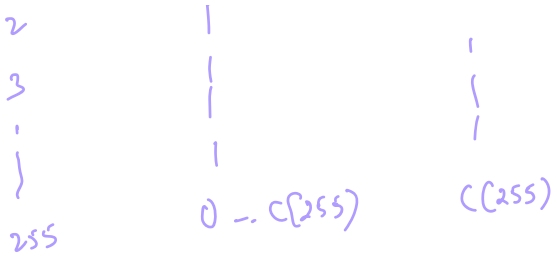
S → 'a a a' b b b b

SC: O(1).

TC: = O(N)

Total iterations
= count of every character.
= no. of characters in string.

i	j	Total iterations.
0	(0 -- C[0])	C(0)
1	0 -- C(1)	C(1)
		C(2)



Substring: is same as subarrays.

- continuous part of string.
- Full string is also a substring.
- A single character is also a substring.

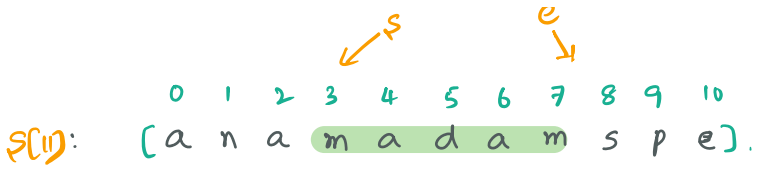
Break: 10:20:00
10:25:00

Q: Check if given substring is palindrome or not?

Ex) madam, nayan, level

mom, civic

dad, radar




```
def isPalindrome( word, s, e, n)
{
        while (s <= e)
        {
                if (word[s] != word[e])
                        return false

                s = s+1
                e = e-1

        }
        return true

}
```

start index

end index

TC: O(n).

SC: O(1).

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

Ex:

0	1	2	3	4	5
a	b	a	c	a	b

ans = 5.

Ex:

0	1	2	3	4
a	b	c	d	e

ans = 1 / single character is also a palindrome.

Brute force:

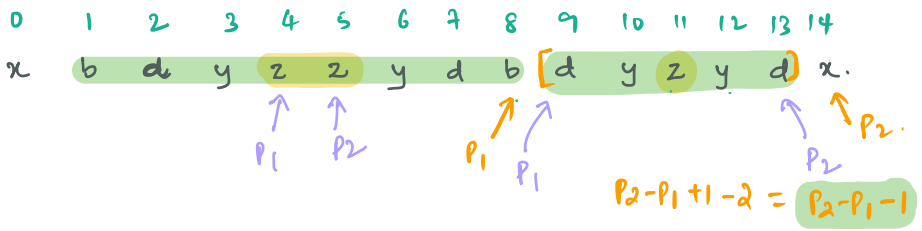
* Generate all substrings ($O(N^2)$) & check if its palindrome ($O(N)$) or not. (keep track of max palindrome length).

TC: $O(N^3)$.

```
def longestPandromicString(s, n)
{
    ans = 0,
    for i in range(0, n)  <-- start index of substring.
        for j in range(i, n)  <-- end index of substring.
        {
            // substring (i, j)
            if( isPalindrome(s, i, j)) // -> length = j-i+1
            {
                ans = max(ans, j-i+1)
            }
        }
    return ans
}
```

TC: $O(N^3)$.
SC: $O(1)$.

Ex:



Hint: If middle element of the palindrome is known, can you find the length of the palindrome.

idea:

(i) Take every element as middle of palindrome and expand as much as possible.] odd length palindromes.

(ii) Take every adjacent characters as middle of palindrome and try to expand.] even length palindromes.

```
def expand(S, p1, p2)
{
    while (p1>=0 && p2<N && S[p1]==S[p2])
    {
        p1 = p1-1;
        p2 = p2+1
    }
    return p2-p1-1;
}
```

```
def lengthOfLongestPalindrome(s, n)
{
    ans = 0
    // odd length palindrome.
    for i in range(0, N)
        ans = max(ans, expand(s, i, i))

    // even length palindrome.
    for i in range(0, N).
        ans = max(ans, expand(s, i, i+1))

    return ans
}
```

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