Strings

Content

→ Intro

→ flip

→ sort ch[]

→ Reverse string

→ longest Palindromic Substring

$$\begin{tikzcd}
 & \text{sequence of chars} & \\
\text{Strings :} \arrow[ur] \arrow[r] \arrow[dr] & \text{collection of chars} \arrow[r] \arrow[dr] & \{a,c,b\} : acb \\
 & \text{array of chars} & \{a,b,c\} : abc
\end{tikzcd}$$

] → not same

Order is also important

Characters : has ASCII values

'A' - 65 $\underset{-32}{\overset{+32}{\rightleftarrows}}$ 'a' - 97

'B' - 66 $\underset{-32}{\overset{+32}{\rightleftarrows}}$ 'b' - 98

'C' - 67 $\underset{-32}{\overset{+32}{\rightleftarrows}}$ 'c' - 99

.
.
.

'Z' - 90 $\underset{-32}{\overset{+32}{\rightleftarrows}}$ 'z' - 122

'0' - 48

'1' - 49

'2' - 50

.
.
.

'9' - 57

'~~10~~' - It is not a single char

char → 1 Byte

8 bits

[0, 255]

[-128, 127]

char ch = '9'
1Byte ↑ ASCII = 57

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

ch = '9'
ch = ch + 8 | 57 + 8 = 65 → 'A'
print(ch)
↳ 'A'

In python
print(chr(ord('9') + ord('8'))) ⇐ print('9' + '8') | 57 + 56 = 115
↳ Smallcase char ⇒ 's'

String → array of characters

String s = "adba"
print(s[0]) = a

Char s[] = "abda"
print(s[0]) = a

Question 1 :

Given a char array, toggle every char.
↳ uppercase ⇄ lower case

Note: Input only contains small & capital letters.

input → AnaConDa
output → aNAcONdA

Code

```
def toggle ( char s[] ) {
    n = s.length
    for (i=0; i<n; ++i) {
        if ( s[i] >= 65  &&  s[i] <= 90 )   // upper case
            s[i] = s[i] + 32   → TODO [modify in your language if needed]
        else   // lower case
            s[i] = s[i] - 32
    }
}
```

s[i] = s[i]^32

OR

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

TC : O(N)

SC : O(1)

	7	6	5	4	3	2	1	0
'A' 65 :	0	1	0	0	0	0	0	1
(1<<5) ⇒ ^32	0	0	1	0	0	0	0	0
'B' 66 :	0	1	0	0	0	0	1	0
⋮								
'Z' 90	0	1	0	1	1	0	1	0

$\xrightarrow{+2^5}$ $\xleftarrow{-2^5}$

	7	6	5	4	3	2	1	0
'a' 97 :	0	1	1	0	0	0	0	1
^32	0	0	1	0	0	0	0	0
'b' 98 :	0	1	1	0	0	0	1	0
⋮								
'z' 122 :	0	1	1	1	1	0	1	0

Question 2 :

Given a char array, which contains only lower case alphabets, sort the array in alphabetical order.

eg s[] = d a b a c d b

$\hookrightarrow$ sort   a a b b c d d

Brute force

1. Sort s[] with bubble sort
   TC: $O(N^2)$ SC: $O(1)$

2. Using inbuilt sort
   TC: $O(N \log N)$ SC: $O(1)$

Idea: total only 26 lowercase alphabets

S = d a b a c d b

count occurance of each char

'a' — 2
'b' — 2
'c' — 1
'd' — 2

$\xrightarrow{\text{sort}}$ a a b b c d d

S = a b c e b c b a c e

'a'	- 2
'b'	- 3
'c'	- 3
'd'	- 0
'e'	- 2

$\xrightarrow{\text{sort}}$ aa bbb ccc ee

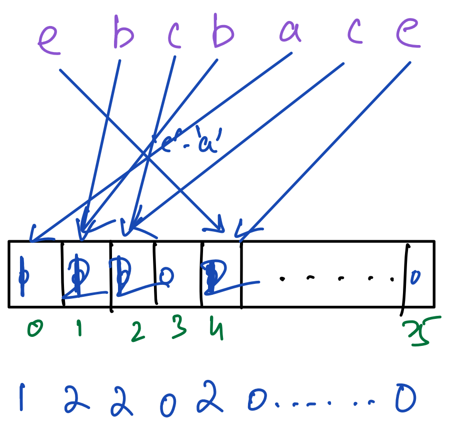
int count[26] = {0}

'a' $\xrightarrow{-97\ ,\ -\text{'a'}}$ 0 index

'b' $\xrightarrow{-97\ ,\ -\text{'a'}}$ 1 index

⋮

'z' $\xrightarrow{-97\ ,\ -\text{'a'}}$ 25 index



Code

```
sortString ( char s[] ) {
    n = s.length
    count[26] = {0}
    for (i=0; i<n; ++i) {
        index = s[i] - 'a'
        count[index]++
    }
    K=0 // index to update s[]
    for ( i=0; i<26; ++i) {
```

→ TC: O(N)   SC: O(26)
: O(1)

```
        // c[i] indicates freq. of ('a'+i)
        char ch = 'a'+i
        for (j=0; j< count[i]; ++j) {
            S[K] = ch
            K++
        }
    }
}
```

→ TC : O(N)
SC : O(1)

total iterations ?

TC: O(N)
SC: O(1)

Also known as Count Sort

i	j : [0, c[i]-1]	total
0	[0, c[0]-1]	c[0]
1	[0, c[1]-1]	+ c[1]
2	.	+ .
.	.	.
.	.	.
25	[0, c[25]-1]	+ c[25]

total iterations = c[0] + c[1] + . . . + c[25]

= total freq of all chars

= N

Dry Run

S[] = e b c b a c e

K=0

C[26] = 1 2 2 0 2 0 0 0 0

i=0 1 2 3 4 5 . . . . .

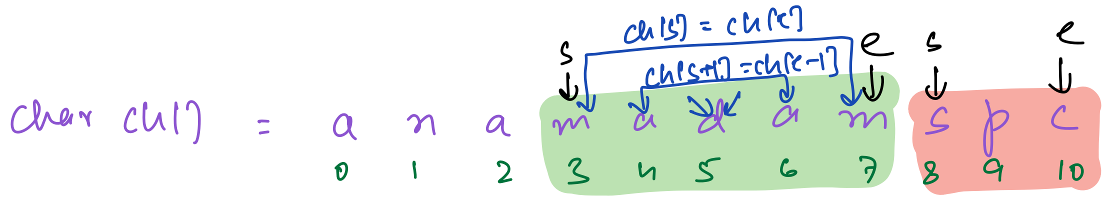
$$S[] = \boxed{\begin{array}{ccccccc} a & b & b & c & c & e & e \\ K=0 & 1 & 2 & 3 & 4 & 5 & 6 \end{array}} 7$$

Sub-string concept is same as subarray
↳ 1. Continous part of string
2. full string can be a substring
3. A single char can also be a substring

Question 3

Check if a given substring is Palindrome or not ?
Left→right ≡ Right→left

eg madam , OPPO , MOM , naman ,
malayalam



for substring we need start & end index

Code
```
bool isPalin ( char ch[], int s, int e ) {
    while (s < e) {
        if (ch[s] != ch[e])
            return false
        s++, e--
```

TC : O(N)
SC : O(1)

```
        }
        return true
    }
```

Question 4

Given a string, calculate length of longest palindromic substring.

eg a $\boxed{b\ a\ c\ a\ b}$ len=5 $\boxed{a}$ b c $\boxed{d}$ e len=1

Bruteforce

```
int longestPalin( char ch[] ){
    n = ch.length
    ans = 1
    for (i=0; i<n; ++i){      // i is start index → N iteration
        for (j=i; j<n; ++j){      // j is end index → N iteration
            // substring ch[i,j]
            if ( isPalin(ch, i, j) ){      // len = j-i+1      → TC: O(N)
                ans = max( ans, j-i+1 )
            }
        }
    }
    return ans
}
```

TC: N × N × N

$= O(N^3)$

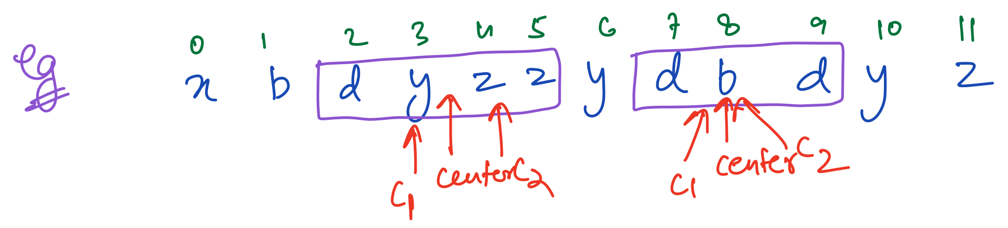
SC: O(1)

→ If the center of a palindromic substring is given, can we find length ? TC : O(N)

→ Consider all possible centers : O(N) centers

So looks like we can solve in O(N²)

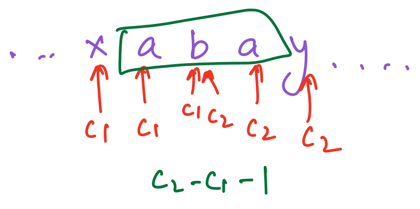
eg



```
int expand (char ch[], c1, c2) {
    while (c1 >= 0 && c2 < n && ch[c1] == ch[c2])
        c1--, c2++

    return c2 - c1 - 1
}
```

TC : O(N)
SC : O(1)



```
int longestPalin (char ch[]) {
    n = ch.length
    ans = 1
    for (i=0; i<n; ++i) {    // odd length palindrome
        // center : ch[i]
        c1 = i , c2 = i
        ans = max(ans, expand(ch, c1, c2))
    }

    for (i=0; i<n-1; ++i) {    // even length palindrome
        //center: ch[i], ch[i+1]
        c1 = i, c2 = i+1
        ans = max(ans, expand(ch, c1, c2))
    }

    return ans
}
```

$$\begin{tikzcd}
 & & \text{longest palindromic substring} \arrow[r] \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] & \text{Mancher's Algo } O(N) \\
\text{bruteforce } O(N^3) & \text{expand } O(N^2) & \text{DP } O(N^2) & \text{binary search + RabinKarp } O(N\log N)
\end{tikzcd}$$

optional class of advance batch

DP: advance batch

binary search + RabinKarp: Not learn at all