Strings

Content

- → Intro
- → flip
- → sort chars
- → Reverse string
- → longest palindromic substring

Strings:
- → sequence of chars
- → collection of chars
  - → {a, c, b} : acb
  - → {a, b, c} : abc
  - → not same
  - order is important
- → array of chars

Characters: has ASCII values

'A'	- 65	$\xrightarrow{+32}$ $\xleftarrow{-32}$	'a'	- 97	'0' - 48
'B'	- 66	$\xrightarrow{+32}$ $\xleftarrow{-32}$	'b'	- 98	'1' - 49
'C'	- 67	$\xrightarrow{+32}$ $\xleftarrow{-32}$	'c'	- 99	⋮
⋮			⋮		⋮
					'9' - 57
'Z'	- 90	$\xrightarrow{+32}$ $\xleftarrow{-32}$	'z'	- 122	~~'10'~~ → It is not a single char

char → 1 Byte
8 bits

[0, 255] → unsigned

[-128, 127] → signed

char ch = '9'

1 Byte

ASCII = 57

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

ch = '9'

ch = ch + 8 | 57 + 8 = 65

print(ch)

↳ 'A'

In phyton ← print('9' + '8') | 57 + 56 = 113

print(chr(ord('9') + ord('8')))

↳ 'q'

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 ⇄ lowercase

Note: Input only contains upper/lower case chars.

eg 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] += 32
        else // lower case
            s[i] -= 32
    }
}
```

s[i] ^= 32

OR

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
'b' 98:	0	1	1	0	0	0	1	0
⋮								
'z' 122:	0	1	1	1	1	0	1	0

5th bit is toogled from lower $\rightleftarrows$ upper

Question 2

Given a char array which contains only lowercase letters, sort the array in alphabetical order.

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

sort(s) = a a b b c d d

1. sort s[] with bubble sort

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

2. Use inbuilt function (using comparator)

   TC: $O(N \log N)$ SC: $O(1)$

Idea: there are only 26 lowercase letters

S1 = d a b a c d b

count occurance of each letter

'a' – 2
'b' – 2
'c' – 1
'd' – 2
'e' – 0
⋮
'z' → 0

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

```
int count[26] = {0}
```

(ASCII: 97) 'a' $\xrightarrow{-97,\ -'a'}$ %97 0 index

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

⋮

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

d b c z d

0	1	2	3	...	25
0	1	1	2	...	1

⇓

b c d d z

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) {
        char ch = 'a' + i
        for (j=0; j<count[i]; ++j) {
            s[K] = ch
            K++
        }
    }
}
```

→ TC: O(N)
SC: O(26)
: O(1)

→ TC: O(N)
SC: O(1)

Known as Count Sort

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

Dry Run: s[] = e b c b a c e

count[26] =	1	2	2	0	2	0	0	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	0
	i=0	1	2	3	4	5																			25

s[] =

a	b	b	c	c	e	e	
K=0	1	2	3	4	5	6	7

Total iterations ?

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

Sub-string concept is same as subarray

1. continous part of string
2. full string can be a substring
3. 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, abcba

char ch[] = a n a m a d a m s p c

0 1 2 3 4 5 6 7 8 9 10

s: 3, e: 7

ch[s+1] = ch[e-1]

ch[s] = ch[e]

Code

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

start index

end index

TC : O(N)

SC : O(1)

Question 4

Given a string, calculate length of longest palindromic substring.

eg a b a c a b

ans = 5

a b c d e

ans = 1

Bruteforce

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

TC: ~~O(N²)~~ $O(N^3)$

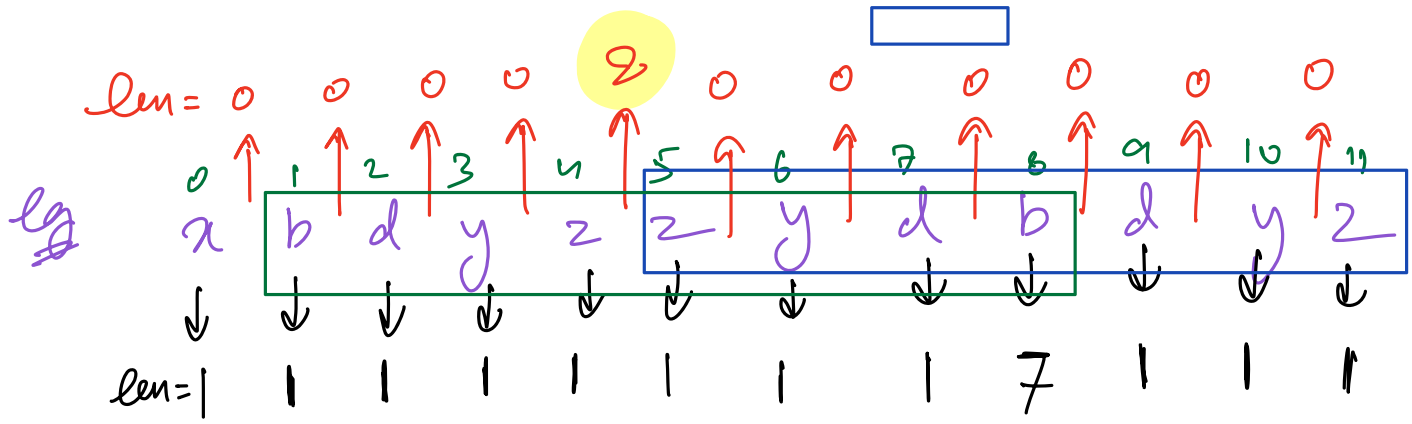
SC: $O(1)$

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

a b b a

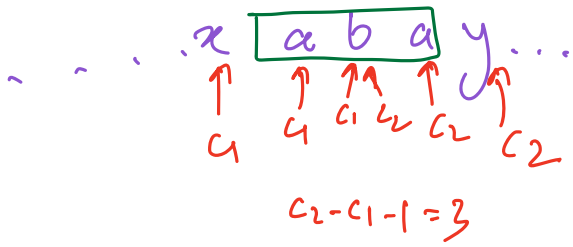
→ Since we don't know the center, consider all possible centers : $O(N)$ centers

looks like we can solve in $O(N^2)$




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

    return c2 - c1 - 1
}
```




```
int longestPalin( char ch[] ) {
    n = a.length
    ans = 1
    for (i=0; i<n; ++i) { // odd length palindrome
        // center : a[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

}
```

TC : $O(N^2)$

SC : $O(1)$

Longest Palindromic Substring

- Brute force — $O(N^3)$
- expand + center — $O(N^2)$
- DP — $O(N^2)$ — advance batch
- Manacher's Algo — $O(N)$ — optional class of advance batch
- binary search + Rabin Karp — $O(N \log N)$ — Not learn at all