* almost all sorting algorithms are comparison based sorting algo.

3	1	9	6	7	5	10

for asc ⇒ $A[i] \leq A[i+1]$

* we only need to compare two data pts. at a time for any sorting algorithm

$\underbrace{\text{for any sorting algorithm}}_{\text{comparison based}}$

Q1. Sort a given array in Asc order by the no. of factors of each element, if the no. of factor is equal sort by value

arr ⇒	10	4	6	9	3	101
	↓	↓	↓	↓	↓	↓
# factor	4	3	4	3	2	2

sort ⇒ 3 101 4 9 6 10

* compare func:
  i) two arguments [two i/ps that need to be compared]
  ii) based on some logic it will tell us, which is smaller

larger or equal

iii) return 1 | -1 | 0 → equal

$\underset{\text{1st arg smaller}}{\downarrow}$ $\underset{\text{2nd arg smaller}}{\downarrow}$

⇒ C++ ⇒ sort(a.begin(), a.end, comp)

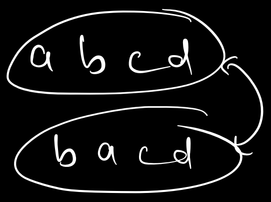
⇒ Java ⇒ Arrays.sort(arr comparator obj)

```
int compare( int a, int b) {
    int f1 = countFactor (a)
    int f2 = countFactor (b)

    if( f1 < f2) {
        return 1
    else if( f1 == f2) {
            if( a < b)
                return 1
            else if( a == b)
                return 0
            else
                return -1
    }
    else {
        return -1
}
```

```
sort ( v.begin() , v.end(), comp )

Java => Arrays.sort( arr, new Comparator <Integer> {
            public int compare ( a, b ) {
                ____
                ____
            }
        })
```

Intermediate :- Strings

Strings => ~~group of chars~~ ←
sequence of chars ← ✓
array / list of chars ← ✓



: Computer only understand binary → number

'A' → 65	'a' → 97	'0' → 48
'B' → 66	'b' → 98	'1' → 49
⋮	⋮	⋮
'Z' → 90	'z' → 122	'9' → 57

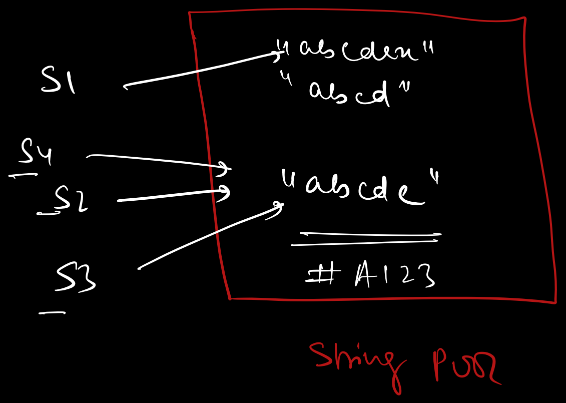
ASCII => American Standard Code for information interchange

In some languages like Java / Python :-

Strings are immutable ⇒ can't be modified

String S1 ⇒ " a b c d"

S1 ⇒ S1 + "e"



S2 = "abcde"

S3 = "abcde"

S4 = "abcde"

* Garbage collector

S1 = S1 + "x"

abcdex

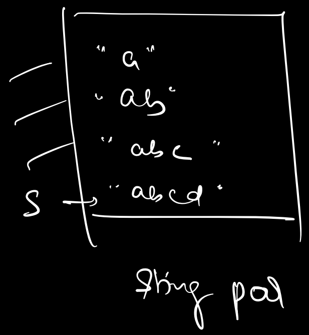
⇒ a, b, c, d

S = "a" → O(N)

S = S + "b" → O(N)

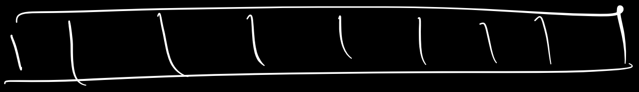
S = S + "c" → O(N)

S = S + "d" → O(N)



TC ⇒ $O(N^2)$

SC ⇒ $O(N^2)$

arr → 

```
for ( i = 0; i < N; i++) {
        s = s + arr[i]
}
            ↳ O(N) | O(N)
```

O(N²)

StringBuilder → mutable

StringBuilder sb = new StringBuilder(" ");

sb.append(arr[i]) → O(1).

Q1. Given a string S, toggle the case of every character.

upper → lower

lower → upper

* no inbuilt func are allowed

S: aBcAEd → AbCaeD.

'A' → 65 $\xleftrightarrow{32}$ 'a' → 97

'B' → 66 $\xleftrightarrow{32}$ 'b' → 98

⋮

'Z' → 90 $\xleftrightarrow{32}$ 'z' → 122

S[i] → charac

```
toggle(s){
    for(i=0; i< s.size(); i++){
        if( s[i] >= 'a' && s[i] <= 'z'){
                // lower case character
                s[i] = s[i] - ('a' - 'A')
        }
        else if( s[i] >= 'A' && s[i] <= 'Z'){
                // upper case character
                S[i] = S[i] + ('a' - 'A')
    }
```

TC ⇒ O(N) → mutable string

SC ⇒ O(1)

** solve with using (+) (-)

'a' → 97
'b' → 98
.
.
'z' → 122

$\underline{\underline{(97-122)}}$

0	1	$\boxed{1}$	_	_	_	_	_
7	6	5	4	3	2	1	0

97 ⇒ 64

97 - 64 ⇒ 33

A → 65

B → 66

Z → 90

65 - 90

$$\begin{array}{cccccccc} & & 32 & & & & & \\ & & \uparrow & & & & & \\ 0 & 1 & \boxed{0} & \alpha \xrightarrow{\quad} & \_ & \_ & \_ & \_ \\ 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \end{array}$$

$65 - 64 = \textcircled{1}$

$$\text{'z'} \Rightarrow 122 \Rightarrow \begin{array}{cccccccc} 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \end{array}$$

$$\text{'Z'} \leftarrow 90 \leftarrow \begin{array}{cccccccc} 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \end{array}$$

```
for( i=0; i<N; i++) {
    if( S[i] => 97 to 122 || S[i] => 65 to 90 ){
        S[i] = S[i] ^ (1<<5)
}
```

Q2. Given a string of lower case characters, sort it in dictionary order.

S = d a b a e d b

O/p = a a b b d d e

Library func ⇒ O(N log N)

d a b a e d b

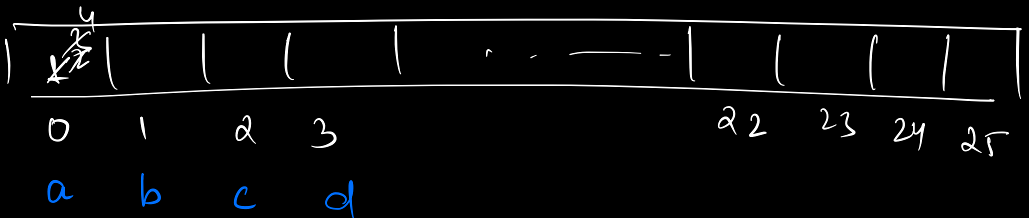
a →	2
b →	2
c →	0
d →	2
e ⇒	1

array

a a b b d d e

String → N

lowercase ⇒ 26.



charac	ascii		index
a	97	−97 ('a')	0
b	98	−97 ('a')	1
c	99	−97	2
d	100	−97	3

+97

S ⇒ b c d a f a

Pseudo

```
int count[26] = {0}
for(i=0; i< s.size(); i++) {
    count[ s[i] - 'a' ]++;          ] → O(N)
}

⇒ k=0
for(i=0; i<26; i++) {
        // count[i] stores the freq of char ⇒ i+'a'
        for(j=0; j< count[i]; j++) {
            s[k] = (char)(i+'a')   → ①
            k++
        }
}
```

$TC \Rightarrow \underbrace{O(N)}_{\text{count array}} + \underbrace{O(N)}_{\text{create string}} \Rightarrow O(N)$

$SC \Rightarrow O(1)$