Q sort an array in asc order by the no. of factors, and if factors are same, sort by value.

array ⇒	✓ 9	3	10	6	4 ✓
	↓	↓	↓	↓	↓
factors ⇒	3	2	4 ✓	4 ✓	3

3 4 9 6 10 ⇒ o/p

Comparators

almost all sorting algo. are comparison based sorting algo.

3	1	9	6	7	5	10

asc order : $arr[i] \leq arr[i+1]$

we only compare two data pts. for any comparison based sorting algo.

$a < b$ ⇒ true
a is smaller

false a is equal or bigger

: compare funcⁿ

- : two arguments (two i/ps that needs to be compared)
- : return result based on some rule
- : return 1 | -1 | 0

1 ↓ 1st arg is smaller

-1 ↓ 2nd arg is smaller

0 ↓ equal

```
int compare (int a, int b) {

    int f1 = countfactors (a)
    int f2 = countfactors (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
    }
```

C++ => sort(a.begin(), a.end(), compare)

Java => Arrays.sort(arr, new Comparator<Integer>(){

```
                public int compare(a,b){

                }
        );
```

Strings

~~group of chars~~

sequence of chars ✓

array | list of chars ✓

a b cd,

b c ad,

∴ Computers can only understand → binary
↓
numbers

Chars are mapped to some numeric value

'A' → 65
'B' → 66
'C' → 67
⋮
'Z' → 90

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

'0' → 48
'1' → 49
'2' → 50
'3' → 51
⋮
'9' → 57
~~'10'~~ → 58
'1' ✓ '0' ✓

ASCII

American
Standard code
for information interchange

In some languages Java/Python

strings are immutable [can't be changed]

String s1 = "abc"

s1 = s1 + "d"

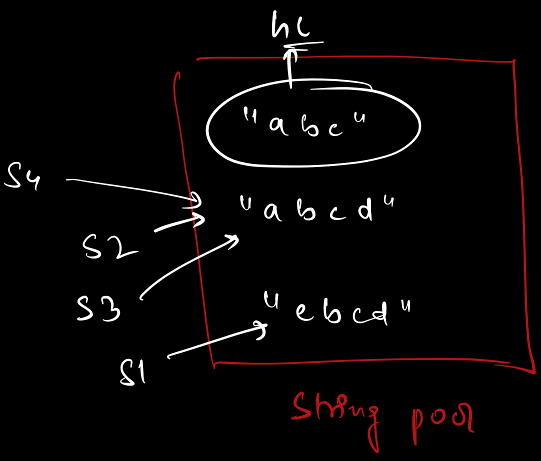
= "abcd"

S2 = "abcd"

S3 = "abcd"

S4 = "abcd"

[Memory Optimisation]



S1[0] = "e"

Garbage Collector

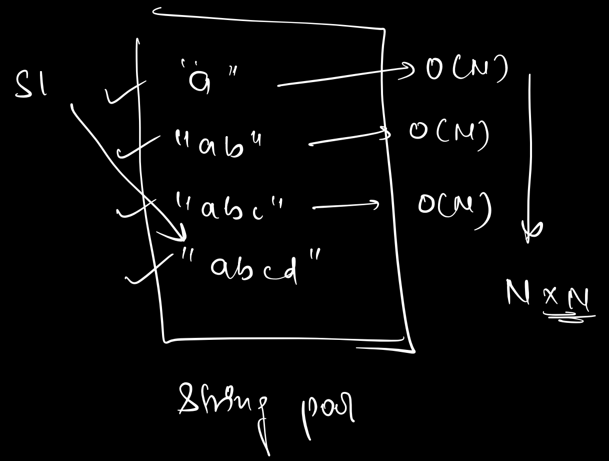
appending multiple chars to a string

String s1 = 'a' → O(N)

S1 = S1 + "b" → O(N)

S1 = S1 + "c" → O(N)

S1 = S1 + "d" → O(N)



TC ⇒ $O(N^2)$

SC ⇒ $O(N^2)$

String abcd → e f gh

abcd → O(N)

abcde

abcdef

abcdefg

abcdefgh

```
S = "abcd"
for ( i=0; i<N; i++ ) {   ← N
        S = S + arr[i]   → O(N)        O(N^2)
}
O(N)
```

⇒ String Builder → mutable

StringBuilder sb = new StringBuilder();

sb.append("c")

sb → "a"

sb → "ab"

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

lowercase → uppercase

upper case → lower case

* no inbuilt method allowed

ex ⇒ abcAed → AbCaED

A → 65 $\xleftrightarrow{32}$ a → 97

B → 66 $\xleftrightarrow{32}$ b → 98

⋮

Z → 90 $\xleftrightarrow{32}$ z → 122

```
toggle(s) {

    for ( i = 0; i < s.size(); i++ ) {

S[i] = S[i] ^ (1<<5)//   //check for lower case
        if( s[i] >= 'a' && s[i] <= 'z' ){
             s[i] = s[i] - ('a' - 'A');

        else if( s[i] >= 'A' && s[i] <= 'z' ){
             s[i] = s[i] + ( 'a' - 'A' );
    }
```

'a' - 'A' = 32

//check for upper case

$$\boxed{\begin{array}{l} TC \Rightarrow O(N) \\ SC \Rightarrow O(1) \end{array}}$$

'a' → 97

'b' → 98

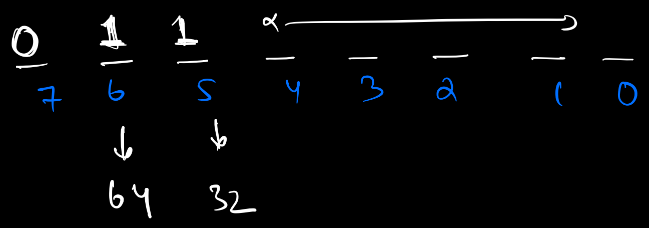
⋮

'z' → 122

97 - 122

97 > 64

lowercase



64 + 32 = 96 < 97

uppercase

'A' → 65

'B' → 66

'Z' → 90

$2^5 = \boxed{32}$

0	1	0	x →				
7	6	5	4	3	2	1	0

64 + 32

= 96 > 90

65 > 64

So, all bits from 0-4 remains same for a pair of small & capital alphabet.

small 'z' ⇒ 122

0	1	1	1	1	0	1	0	⇐ 122
7	6	5	4	3	2	1	0	

'Z' ⇒

0	1	0	1	1	0	1	0	⇐ 90
7	6	5	4	3	2	1	0	

toggle 5th bit → toggle the case

$$S[i] \wedge (1 << 5)$$

Q 2. Given a string of lower case characters, sort it in dictionary order

S ⇒ a, d, b, a, e, b, d

O/p ⇒ a a b b d d e

sorting ⇒ $O(N \log N)$

length ⇒ N

lower case alphabets ⇒ 26 unique characters.

⇒ count of freq of each characters

a → 2

b → 2

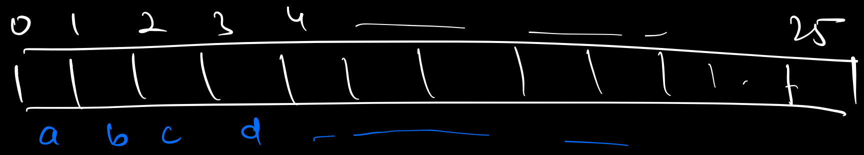
d → 2

e → 1

array

a a b b d d e

int count[26] ⇒



		index	
a → 97	→	0	a - 97
b → 98	→	1	b - 97
c → 99	→	2	c - 97

index ⇒ S[i] ⇒ S[i] - 'a'

```
// count array
  int count[26] = {0}
   for( i=0; i < s.size(); i++){
             count [ s[i] - 'a' ]++
   }
   k=0
   for( i=0; i<26; i++){
            //count[i] stores freq of i+'a' char
              for( j=0; j< count[i]; j++){
                      s[k] = (char) (i+'a')
                      k++
              }
}
```

$$\boxed{\begin{array}{l} TC \Rightarrow \underbrace{O(N)}_{\substack{\text{count arr} \\ \text{or. freq arr}}} + \underbrace{O(N)}_{\text{str}} = O(N) \\ \\ SC \Rightarrow O(1) \end{array}}$$

count sort [1 100]

Q3. Given a string S, and two index l & r, reverse the substring l to r.

[No extra memory
No inbuilt fn]

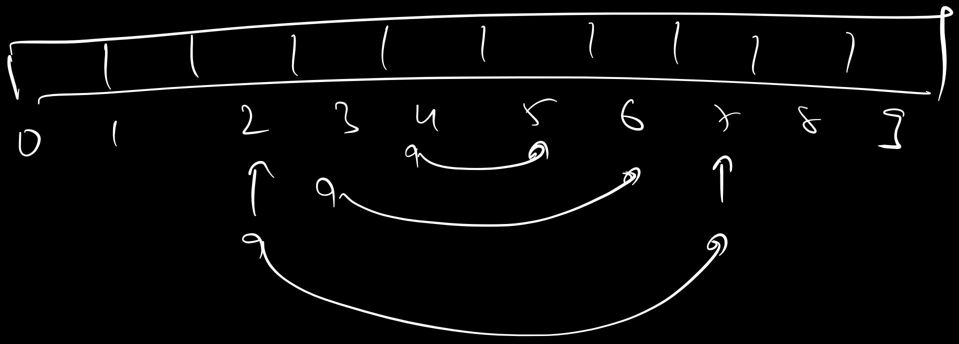
$$\begin{array}{ccccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ S = & a & b & d & e & a & g & h \end{array}$$

$S = a\ b\ \underbrace{d\ e\ a\ g}\ h$

l = 2

r = 5

O/P ⇒ $a\ b\ \underbrace{g\ a\ e\ d}\ h$



pseudo

```
reverse ( S, l, r) {
    while ( l < r){
        swap( S[l], S[r])
        l++
        r--
    }
}
```

Amazon

Q 4. Given a character array, storing a sentence, reverse it word by word.

- no extra space
- every word is separated by "_"
- no inbuilt method
- no space on edges

ex ⇒

h	e	r	e	-	i	s	-	a	-	b	o	y

O/P ⇒ boy - a - is _ here

I/P ⇒ " ARE -YOU _ AS _CLEVER_ AS _ I _ AM"

O/P ⇒ " AM _ I _ AS _ CLEVER_ AS _ YOU_ ARE

I/P ⇒ " mailmen bring letter "

O/P ⇒ " letter bring mailmen ".

I/p ⇒ " mailmen_ bring _ letter "

reverse ⇒ " rettel _ gnirb _ nemliam"

↓ reverse each word

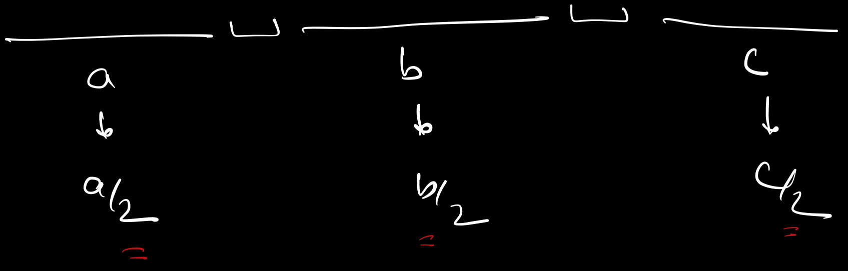
⇒ " letter _ bring _ mailmen "

i) reverse the entire char array ✓

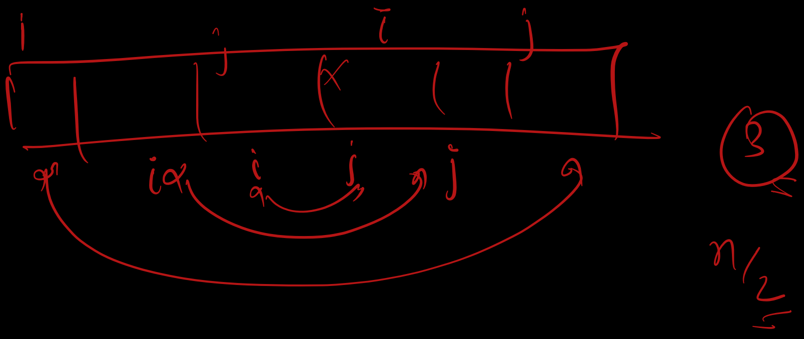
ii) reverse each word

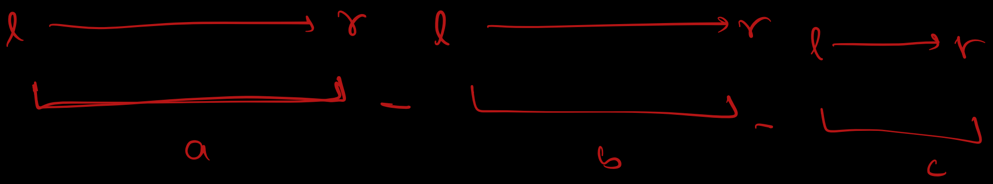
Pseudo

$$TC \Rightarrow \underbrace{O(N)}_{\text{reverse entire array}} + \underbrace{O(N)}_{\text{reverse all the words.}} \Rightarrow O(N)$$



$$\frac{(a+b+c)}{2} \rightarrow \frac{N}{2}$$





$a + a/2$ $b + b/2$ $c + c/2$

$(a+b+c) + \frac{(a+b+c)}{2}$

$N + N/2 = O(N)$

Google, MS, LinkedIn, Amazon

Q Find the longest palindromic substring

$\boxed{TC \Rightarrow O(N^2) \quad SC \Rightarrow O(1)}$

palindrome ⇒ aba → aba ← } same

level → ←   madam → ←

abcba   a

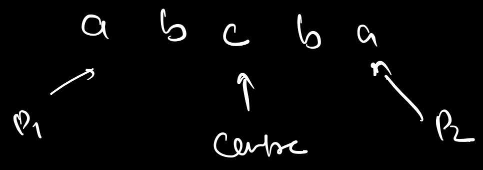
ex ⇒ a b a c a b

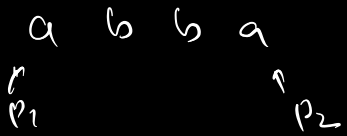
3   5

O/P = 5

a b c d e

O/P = 1.


a b c b a
P1
Centre
P2


a b b a
P1
P2