✅HW 01: Valid Anagram (Leetcode-245)

Example 1:	Example 2:
Input: s = "anagram", t = "nagaram"	Input: s = "rat", t = "car"
Output: true	Output: false

METHOD:01

SORTING

STEP 01 SORT S and T

Exampl:1

S = AAAGMNR

T = AAAGMNR

STEP 02 Compare both S == T

return true

Else

return false

T.C. = O(NlogN)

S.C. = O(1)

METHOD :2

Counting

Number of respective characters in 's' string should be same to that of 't' string

s = "ANAGRAM" t = "NAGARAM"

Step 01: *create frequency Table of 's'*

ASCII

A = 65

a = 97

A = 3

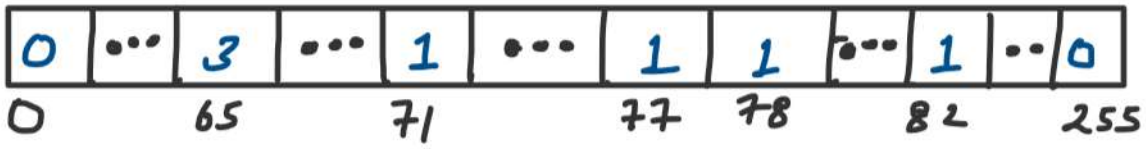
N = 1

G = 1

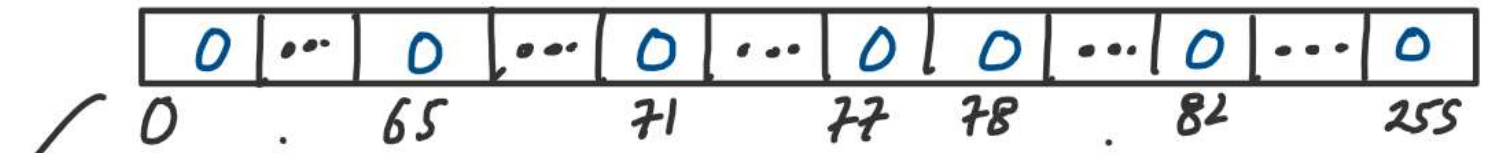
R = 1

M = 1

freqTable



Step 02: *update frequency Table of 's' according to 't' string*



AGAR frequency table ka har ek element equal to zero ho jata hai to --> return true
Else --> return false

```cpp
// ✅HW 01: Valid Anagram (Leetcode-245)
class Solution {
public:
    bool isAnagram(string s, string t) {

        int freqTable[256] = {0};

        // Updating freqTable by 's' string
        for(int i=0; i<s.size(); i++){
            freqTable[s[i]]++;
        }

        // Again, Updating freqTable by 't' string
        for(int i=0; i<t.size(); i++){
            freqTable[t[i]]--;
        }

        // Comparing each elements of freqTable are zero or not
        for(int i=0; i<256; i++){
            if(freqTable[i] != 0){
                // Agar ek abhi element not equal to zero hai to invalid anagram hai
                return false;
            }
        }
        // Agar all elements equal to zero hai to valid anagram hai
        return true;
    }
};
```


$$T.C. = O(N) + O(M) + O(1)$$

$$= O(N+M)$$

$$S.C. = O(1)$$

✅HW 02: Reverse Only Letters (Leetcode-917)

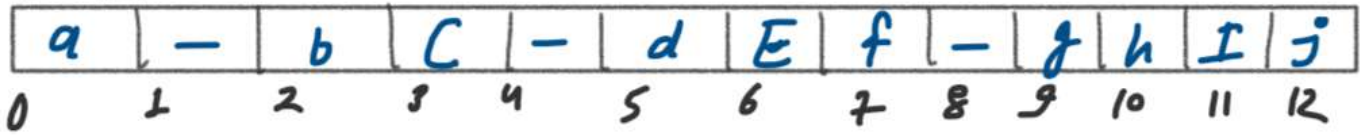
Example 1: *Input: s = "ab-cd"* *Output: "dc-ba"*	***Example 2:*** *Input: s = "a-bC-dEf-ghIj"* *Output: "j-Ih-gfE-dCba"*	***Example 3:*** *Input: s = "Test1ng-Leet=code-Q!"* *Output: "Qedo1ct-eeLg=ntse-T!"*

METHOD

Two pointers Approach

EX:2

Item:01



start = 0

End = 12

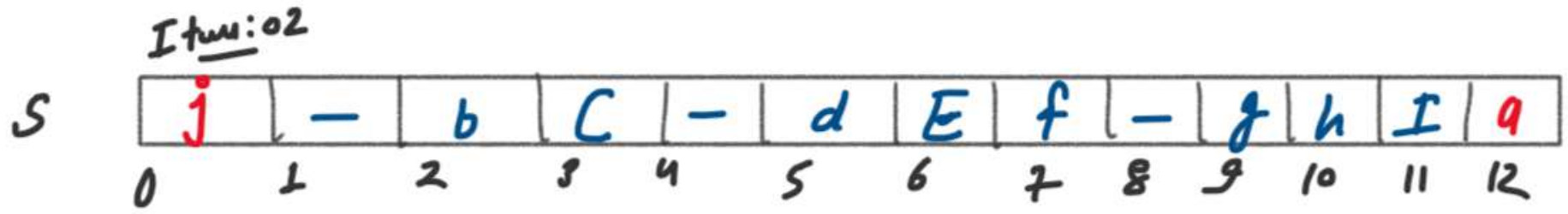
```
if ( isalpha( s[start]) && isalpha (s [End])
       TRUE                  TRUE
```

swap (s[start] , e[End])

a j

start ++

End – –



start = 1

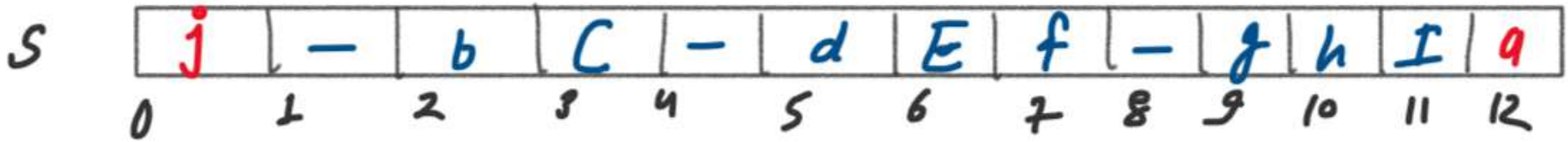
End = 11

if (! isalpha (s [start])

TRUE

start ++

Item: a3



start = 2

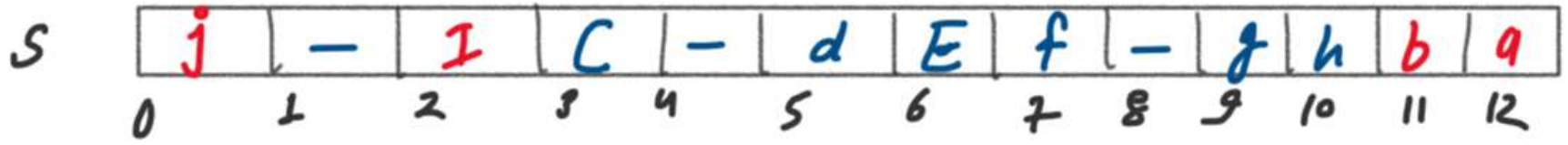
End = 11

```
if ( isalpha( s[start]) && isalpha (s [end])
     TRUE                  TRUE
```

```
swap ( s[start] , e[End] )
       b          I
```

```
start ++
End - -
```

Item:04



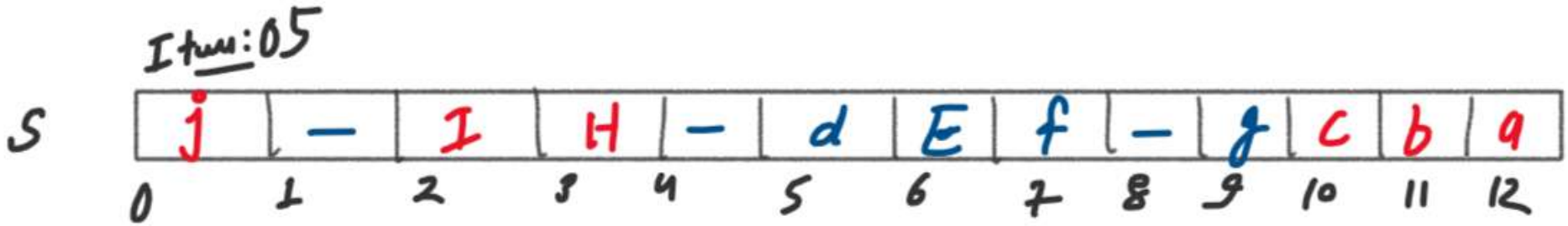
start = 3

End = 10

```
if ( isalpha( s[start]) && isalpha (s [End])
        TRUE                  TRUE
```

```
swap ( s[start] , e[End] )
        C           H
```

```
start ++

End - -
```



start = 4

End = 9

if (! isalpha (s [start])

TRUE

start ++

Itm:06

S	j	-	I	H	-	d	E	f	-	g	c	b	a
	0	1	2	3	4	5	6	7	8	9	10	11	12

start = 5
End = 9

if (isalpha(s[start]) && isalpha (s [End])
TRUE TRUE

swap (s[start] , e[End])
d g

start ++

End - -

Itm:07

start = 6

End = 8

```
if ( ! isalpha ( s [ End ] )
```

TRUE

End--

Item:08

start = 6

End = 7

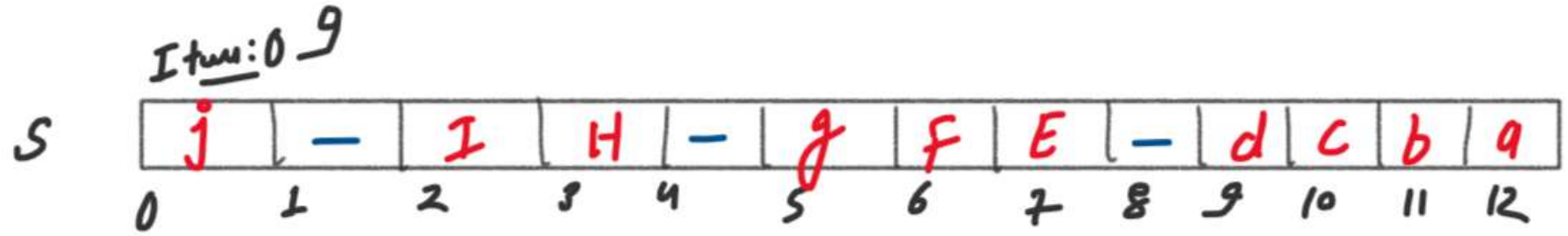
```
if ( isalpha( s[start]) && isalpha (s[End])
```

TRUE TRUE

```
swap ( s[start] , e[End] )
```

E f

```
start ++
End - -
```



start = 7
End = 6

STOP

start > End

Final output

```cpp
// ✅HW 02: Reverse Only Letters (Leetcode-917)
class Solution {
public:
    string reverseOnlyLetters(string s) {
        int start = 0, end = s.size()-1;

        while(start <= end){
            if(isalpha(s[start]) && isalpha(s[end])){
                swap(s[start], s[end]);
                start++, end--;
            }
            else if(!isalpha(s[start])){
                start++;
            }
            else if(!isalpha(s[end])){
                end--;
            }
        }
        return s;
    }
};
```

T.C. ⇒ $O(\frac{n}{2})$

⇒ $O(N)$

S.C. ⇒ $O(1)$

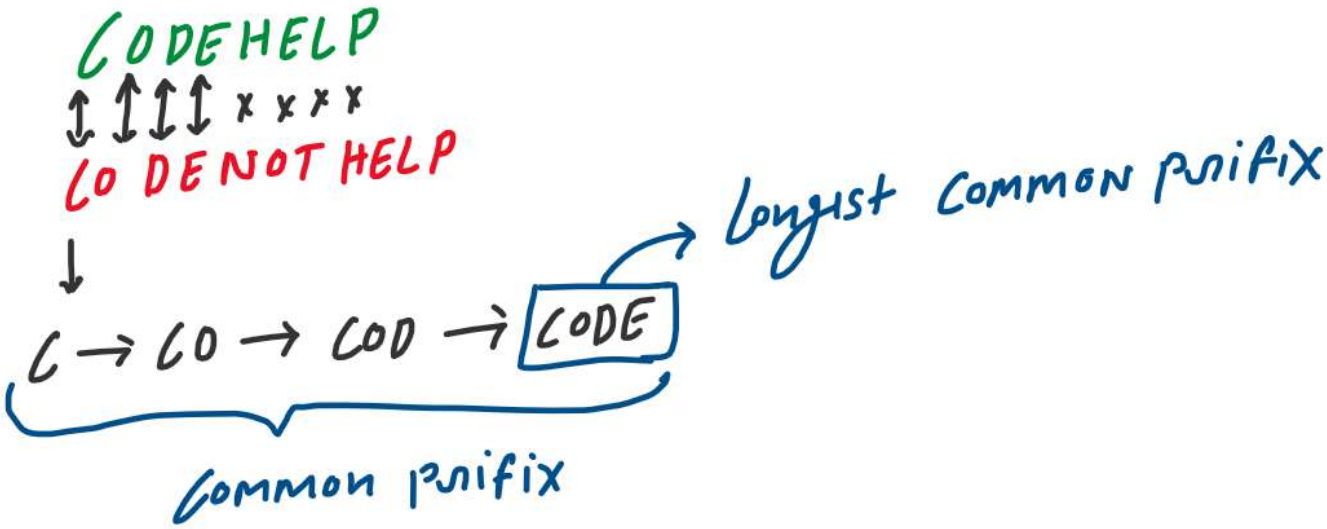
Alternate of isalpha() method

↳ 'a' 'z' 'A' 'Z'

```
if((ch >= 97 && ch <= 122) ||
   (ch >= 65 && ch <= 90))
{
      // Alphabet
}
```

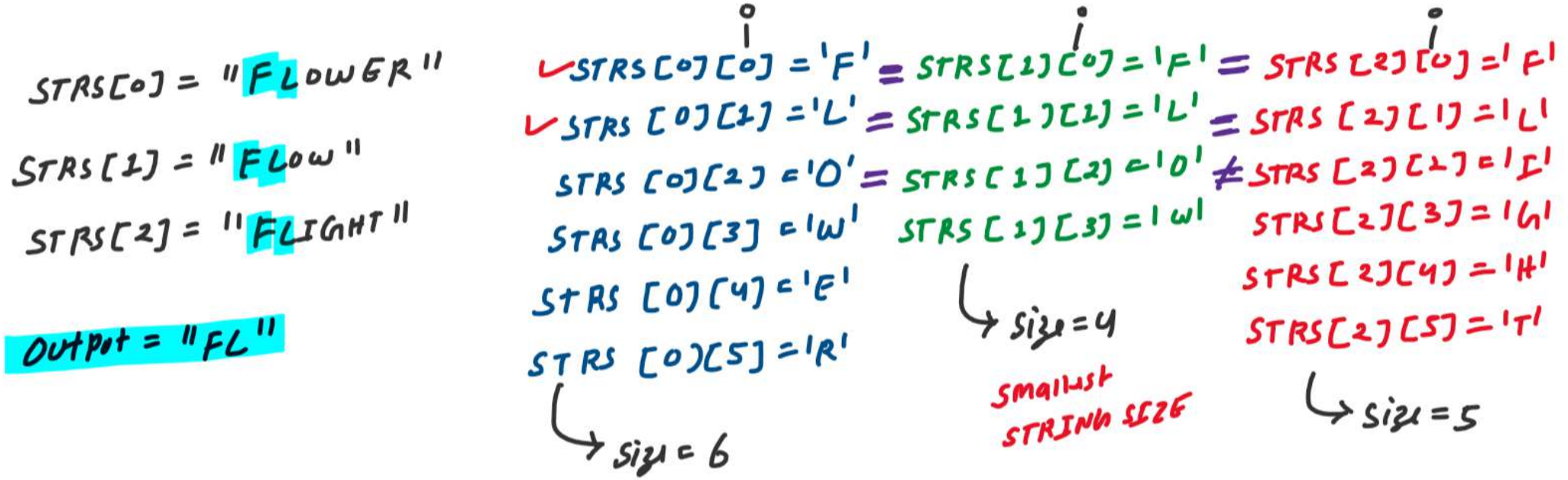
✅HW 03: Longest Common Prefix (Leetcode-14)

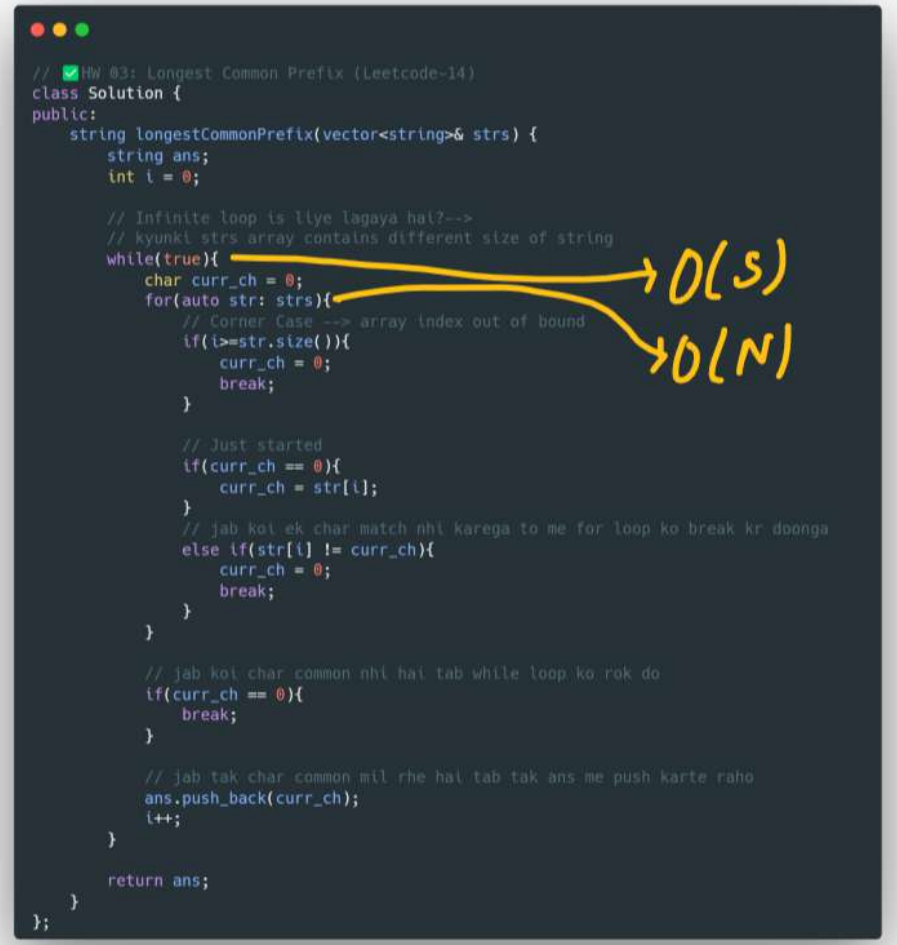
What is Common Prefix?



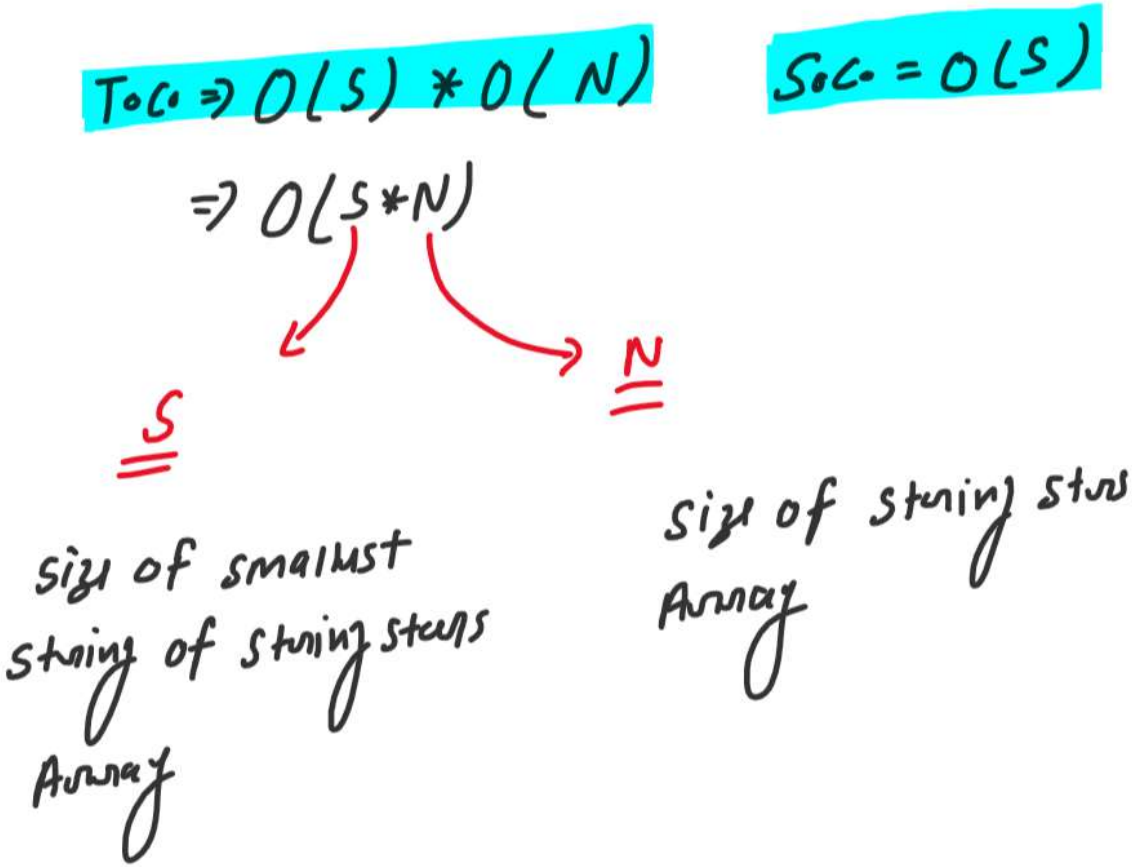
Example 1:
Input: strs = ["flower","flow","flight"]
Output: "fl"

Example 2:
Input: strs = ["dog","racecar","car"]
Output: ""
Explanation: There is no common prefix among the input strings.





```cpp
// ✅HW 03: Longest Common Prefix (Leetcode-14)
class Solution {
public:
    string longestCommonPrefix(vector<string>& strs) {
        string ans;
        int i = 0;

        // Infinite loop is liye lagaya hai?-->
        // kyunki strs array contains different size of string
        while(true){
            char curr_ch = 0;
            for(auto str: strs){
                // Corner Case --> array index out of bound
                if(i>=str.size()){
                    curr_ch = 0;
                    break;
                }

                // Just started
                if(curr_ch == 0){
                    curr_ch = str[i];
                }
                // jab koi ek char match nhi karega to me for loop ko break kr doonga
                else if(str[i] != curr_ch){
                    curr_ch = 0;
                    break;
                }
            }

            // jab koi char common nhi hai tab while loop ko rok do
            if(curr_ch == 0){
                break;
            }

            // jab tak char common mil rhe hai tab tak ans me push karte raho
            ans.push_back(curr_ch);
            i++;
        }

        return ans;
    }
};
```

✅HW 04: Reverse Vowels of a String (Leetcode-345)

Example 1:
Input: s = "hello"
Output: "holle"

Example 2:
Input: s = "Leetcode"
Output: "Leotcede"

SIMILAR TO HW:02

Two pointer Approach

Ex1 S = hello

h o l l e

Ex2 S = leetcode

leotcede

vowels

a
e
i
o
u

method → isVowels(char ch){
ch = tolower(ch);
return ch=='a' ||
ch=='e' ||
ch=='i' ||
ch=='o' ||
ch=='u';
}

```cpp
// ✅HW 04: Reverse Vowels of a String (Leetcode-345)
class Solution {
public:
    bool isVowel(char ch){
        ch = tolower(ch);
        return ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u';
    }
    string reverseVowels(string s) {
        int start = 0;
        int end = s.size()-1;

        while(start<=end){
            if(isVowel(s[start]) && isVowel(s[end])){
                swap(s[start], s[end]);
                start++;
                end--;
            }
            else if(!isVowel(s[start])){
                start++;
            }
            else if(!isVowel(s[end])){
                end--;
            }
        }
        return s;
    }
};
```

O(N/2)

=> O(N)

T.C. = O(N)

S.C. = O(1)