You are given a lowercase string s.

You must:

1. Find the **highest frequency among vowels** (a, e, i, o, u)
2. Find the **highest frequency among consonants**
3. Return **sum of both frequencies**

If:

- no vowels → vowel frequency = 0
- no consonants → consonant frequency = 0

✅ **Key Idea (Efficient Approach)**

- Use an **array of size 26** → freq[26]
- Traverse the string **once**
- For each character:
    - Increase its frequency
    - Check if it is vowel or consonant
    - Update maxV or maxC immediately

✓ Time Complexity: **O(n)**
✓ Space Complexity: **O(26) ≈ O(1)**

🧪 Dry Run (Example 1)

Input

s = "successes"

Step-by-step

char	freq	vowel/consonant	maxV	maxC
s	1	consonant	0	1
u	1	vowel	1	1
c	1	consonant	1	1
c	2	consonant	1	2
e	1	vowel	1	2
s	2	consonant	1	2

char freq vowel/consonant maxV maxC

s 3 consonant 1 3

e 2 vowel 2 3

s 4 consonant 2 4

Final

maxV = 2

maxC = 4

Answer = 6

✅ **Java Solution (Array + Immediate Update)**

```
class Solution {
    public int maxFreqSum(String s) {
        int[] freq = new int[26];
        int maxV = 0, maxC = 0;

        for (char ch : s.toCharArray()) {
            int idx = ch - 'a';
            freq[idx]++;

            if (isVowel(ch)) {
                maxV = Math.max(maxV, freq[idx]);
            } else {
                maxC = Math.max(maxC, freq[idx]);
            }
        }
        return maxV + maxC;
    }

    private boolean isVowel(char c) {
        return c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u';
```

```
    }
}
```

✅ **C++ Solution (Array + Immediate Update)**

```
class Solution {
public:
    int maxFreqSum(string s) {
        int freq[26] = {0};
        int maxV = 0, maxC = 0;

        for (char ch : s) {
            int idx = ch - 'a';
            freq[idx]++;

            if (isVowel(ch)) {
                maxV = max(maxV, freq[idx]);
            } else {
                maxC = max(maxC, freq[idx]);
            }
        }
        return maxV + maxC;
    }

    bool isVowel(char c) {
        return c=='a'||c=='e'||c=='i'||c=='o'||c=='u';
    }
};
```

✅ **Python Solution (Array + Immediate Update)**

```
class Solution:
    def maxFreqSum(self, s: str) -> int:
```

```
    freq = [0] * 26
    maxV = maxC = 0
    vowels = {'a', 'e', 'i', 'o', 'u'}

    for ch in s:
        idx = ord(ch) - ord('a')
        freq[idx] += 1

        if ch in vowels:
            maxV = max(maxV, freq[idx])
        else:
            maxC = max(maxC, freq[idx])

    return maxV + maxC
```

□ Dry Run (Example 2)

Input

"aeiaeia"

Vowel frequencies:

a → 3

e → 2

i → 2

Consonants:

none → 0

Output

3 + 0 = 3

🎯 Interview Notes (Important)

✓ You **do NOT** need sorting
✓ You **do NOT** need two passes
✓ Updating max during iteration is optimal
✓ Array beats HashMap for fixed alphabets