<https://leetcode.com/problems/sort-vowels-in-a-string/description/>

Given a **0-indexed** string s, **permute** s to get a new string t such that:

All consonants remain in their original places. More formally, if there is an index i with 0 <= i < s.length such that s[i] is a consonant, then t[i] = s[i].

The vowels must be sorted in the **nondecreasing** order of their **ASCII** values. More formally, for pairs of indices i, j with 0 <= i < j < s.length such that s[i] and s[j] are vowels, then t[i] must not have a higher ASCII value than t[j].

Return *the resulting string*.

The vowels are 'a', 'e', 'i', 'o', and 'u', and they can appear in lowercase or uppercase. Consonants comprise all letters that are not vowels.

**Example 1:**

**Input:** s = "lEetcOde"

**Output:** "lEOtcede"

**Explanation:** 'E', 'O', and 'e' are the vowels in s; 'l', 't', 'c', and 'd' are all consonants. The vowels are sorted according to their ASCII values, and the consonants remain in the same places.

**Example 2:**

**Input:** s = "lYmpH"

**Output:** "lYmpH"

**Explanation:** There are no vowels in s (all characters in s are consonants), so we return "lYmpH".

**Constraints:**

1 <= s.length <= 105

s consists only of letters of the English alphabet in **uppercase and lowercase**.

**Attempt 1: 2025-07-13**

**Solution 1: Hash Table + Sorting (10 min)**

class Solution {

    public String sortVowels(String s) {

        Set<Character> vowels = new HashSet<>(Arrays.asList('a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U'));

        List<Character> vowelsList = new ArrayList<>();

        List<Integer> vowelsPos = new ArrayList<>();

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

            if(vowels.contains(s.charAt(i))) {

                vowelsList.add(s.charAt(i));

                vowelsPos.add(i);

            }

        }

        char[] chars = s.toCharArray();

        Collections.sort(vowelsList);

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

            chars[vowelsPos.get(i)] = vowelsList.get(i);

        }

        return new String(chars);

    }

}

Time Complexity: O(n log n) due to sorting (where n is the number of vowels)

Space Complexity: O(n) for storing vowels and their positions

**Refer to Deepseek**

**Problem Statement**

Given a string s, sort only the vowels in the string in non-decreasing order while keeping the consonants in their original positions. The vowels are 'a', 'e', 'i', 'o', 'u' (both lowercase and uppercase).

**Approach**

**Extract Vowels**: Collect all vowels from the string while noting their positions

**Sort Vowels**: Sort the collected vowels in ascending order

**Reconstruct String**: Place the sorted vowels back into their original positions

**Solution Code**

class Solution {

public String sortVowels(String s) {

Set<Character> vowels = new HashSet<>(Arrays.asList('a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U'));

List<Character> vowelList = new ArrayList<>();

List<Integer> vowelPositions = new ArrayList<>();

char[] chars = s.toCharArray();

// Collect vowels and their positions

for (int i = 0; i < chars.length; i++) {

if (vowels.contains(chars[i])) {

vowelList.add(chars[i]);

vowelPositions.add(i);

}

}

// Sort the vowels

Collections.sort(vowelList);

// Place sorted vowels back into their positions

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

chars[vowelPositions.get(i)] = vowelList.get(i);

}

return new String(chars);

}

}

**Explanation**

**Vowel Identification**: We first identify vowels using a HashSet for O(1) lookups

**Collection Phase**: We store both the vowels and their positions in separate lists

**Sorting**: The collected vowels are sorted in ascending order

**Reconstruction**: The sorted vowels are placed back into their original positions in the string

**Optimization**

**Time Complexity**: O(n log n) due to sorting (where n is the number of vowels)

**Space Complexity**: O(n) for storing vowels and their positions

**Example**

Input: "lEetcOde"

Vowels found: ['E', 'e', 'O'] at positions [1, 2, 5]

After sorting: ['O', 'E', 'e']

Reconstructed string: "lOtecEde"

This solution efficiently handles the vowel sorting while maintaining the original consonant positions.

**Refer to**

[L345.Reverse Vowels of a String (Ref.L344,L1119)](note://CA879E2192A94099996642D7560B6FB9)