345. Reverse Vowels of a String

<https://leetcode.com/problems/reverse-vowels-of-a-string/>

1. **Listen**

**Problem Statement:**

Given a **string** ***s***, **reverse** only all the **vowels** in the string and **return it**.

The vowels are 'a', 'e', 'i', 'o', and 'u', and they can appear in both cases.

**Input:**

**string** *s* is any word that may or may not contain vowels.

**Goal:**

reverse only **all** the vowels in the string.

**Return:**

return the *string that has it’s vowels reversed.*

1. **Examples**

Example 1:

**Input:** s = "hello"

**Output:** "holle"

**Constraints:**

* 1 <= s.length <= 3 \* 105
* s consist of printable ASCII characters.

**Test Cases:**

* string with only consonants
* string with all vowels
* mix of consonants and vowels
* duplicate adjacent vowels

1. **Brute Force**

Solution 1:

We can use a two-pointer technique to solve this problem.

We have one pointer at the front of the string.

We have another pointer at the end of the string.

We move the front pointer forward as long as it does not see a vowel

We move the end pointer backward as long as it does not see a vowel

After the front and end pointers are both on vowels, swap and iterate.

O(N) time and O(N) space.

1. **Optimize**
2. **Walkthrough**

Initialize a hashset to store all of the vowels. We can check to see if the current index of the string has a character (is a vowel) in the hashset. Takes constant time and space.

In order to manipulate the array, we must convert the string into a character array. One of the drawbacks of java. This will take O(N) space.

We have one pointer at the front of the string.

We have another pointer at the end of the string.

while the front pointer is behind the end pointer

We move the front pointer forward as long as it does not see a vowel

We move the end pointer backward as long as it does not see a vowel

After the front and end pointers are both on vowels, swap and iterate.

1. **Implement**

public String reverseVowels(String s) {

// two-pointer technique: O(N) time and O(N) space

// O(1) time and space

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

// O(N) time and space

char[] sca = s.toCharArray();

// O(N) time

int lo = 0, hi = sca.length - 1;

while(lo < hi)

{

while(lo < hi && !set.contains(sca[lo])) lo++;

while(lo < hi && !set.contains(sca[hi])) hi--;

swap(sca, lo, hi);

lo++;

hi--;

}

return String.valueOf(sca);

}

private void swap(char[] arr, int index1, int index2)

{

char temp = arr[index1];

arr[index1] = arr[index2];

arr[index2] = temp;

}

1. **Test**