[2870. Minimum Number of Operations to Make Array Empty](https://leetcode.com/problems/minimum-number-of-operations-to-make-array-empty/)

Medium

1.2K

59

Companies

You are given a **0-indexed** array nums consisting of positive integers.

There are two types of operations that you can apply on the array **any** number of times:

* Choose **two** elements with **equal** values and **delete** them from the array.
* Choose **three** elements with **equal** values and **delete** them from the array.

Return *the****minimum****number of operations required to make the array empty, or*-1*if it is not possible*.

**Example 1:**

**Input:** nums = [2,3,3,2,2,4,2,3,4]

**Output:** 4

**Explanation:** We can apply the following operations to make the array empty:

- Apply the first operation on the elements at indices 0 and 3. The resulting array is nums = [3,3,2,4,2,3,4].

- Apply the first operation on the elements at indices 2 and 4. The resulting array is nums = [3,3,4,3,4].

- Apply the second operation on the elements at indices 0, 1, and 3. The resulting array is nums = [4,4].

- Apply the first operation on the elements at indices 0 and 1. The resulting array is nums = [].

It can be shown that we cannot make the array empty in less than 4 operations.

**Example 2:**

**Input:** nums = [2,1,2,2,3,3]

**Output:** -1

**Explanation:** It is impossible to empty the array.

**Constraints:**

* 2 <= nums.length <= 105
* 1 <= nums[i] <= 106

function minOperations($nums) {

        $freq = array\_count\_values($nums);

        $operations = 0;

        foreach ($freq as $count) {

            if($count % 3 == 0){

                $operations += (int) ($count / 3);

            }elseif($count % 2 == 0){

                $operations += (int) ($count / 2);

            }else{

                return -1;

            }

        }

        return $operations;

    }

[300. Longest Increasing Subsequence](https://leetcode.com/problems/longest-increasing-subsequence/)

Given an integer array nums, return *the length of the longest****strictly increasing***

***subsequence***

.

**Example 1:**

**Input:** nums = [10,9,2,5,3,7,101,18]

**Output:** 4

**Explanation:** The longest increasing subsequence is [2,3,7,101], therefore the length is 4.

**Example 2:**

**Input:** nums = [0,1,0,3,2,3]

**Output:** 4

**Example 3:**

**Input:** nums = [7,7,7,7,7,7,7]

**Output:** 1

function lengthOfLIS($nums) {

        if (empty($nums)) {

            return 0;

        }

        $n = count($nums);

        $dp = array\_fill(0, $n, 1);

        for ($i = 1; $i < $n; $i++) {

            for ($j = 0; $j < $i; $j++) {

                if ($nums[$i] > $nums[$j]) {

                    $dp[$i] = max($dp[$i], $dp[$j] + 1);

                }

            }

        }

        return max($dp);

    }

[1. Two Sum](https://leetcode.com/problems/two-sum/)

Easy

54.2K

1.8K

Companies

Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to *target*.

You may assume that each input would have **exactly one solution**, and you may not use the same element twice.

You can return the answer in any order.

**Example 1:**

**Input:** nums = [2,7,11,15], target = 9

**Output:** [0,1]

**Explanation:** Because nums[0] + nums[1] == 9, we return [0, 1].

**Example 2:**

**Input:** nums = [3,2,4], target = 6

**Output:** [1,2]

**Example 3:**

**Input:** nums = [3,3], target = 6

**Output:** [0,1]

**Constraints:**

* 2 <= nums.length <= 104
* -109 <= nums[i] <= 109
* -109 <= target <= 109
* **Only one valid answer exists.**

**Follow-up:**Can you come up with an algorithm that is less than O(n2) time complexity?

**Answar:**

function twoSum($nums, $target) {

        // $size = count($nums);

        // for($i=0; $i<$size; $i++){

        //     for($j=$i+1; $j<$size; $j++){

        //         if($nums[$i] + $nums[$j] == $target){

        //             return array($i,$j);

        //         }

        //     }

        // }

        $sampArr = [];

        foreach($nums as $key => $num){

            $restNum = $target - $num;

            if (array\_key\_exists($restNum, $sampArr)) {

                return [$sampArr[$restNum], $key];

            }

            $sampArr[$num] = $key;

        }

    }

[**27. Remove Element**](https://leetcode.com/problems/remove-element/)

Solved

Easy

Topics

Companies

Hint

Given an integer array nums and an integer val, remove all occurrences of val in nums [**in-place**](https://en.wikipedia.org/wiki/In-place_algorithm). The order of the elements may be changed. Then return the number of elements in nums which are not equal to val.

Consider the number of elements in nums which are not equal to val be k, to get accepted, you need to do the following things:

* Change the array nums such that the first k elements of nums contain the elements which are not equal to val. The remaining elements of nums are not important as well as the size of nums.
* Return k.

**Custom Judge:**

The judge will test your solution with the following code:

int[] nums = [...]; // Input array

int val = ...; // Value to remove

int[] expectedNums = [...]; // The expected answer with correct length.

// It is sorted with no values equaling val.

int k = removeElement(nums, val); // Calls your implementation

assert k == expectedNums.length;

sort(nums, 0, k); // Sort the first k elements of nums

for (int i = 0; i < actualLength; i++) {

assert nums[i] == expectedNums[i];

}

If all assertions pass, then your solution will be **accepted**.

function removeElement(&$nums, $val) {

        $count = 0; // Variable to keep track of the count of elements not equal to val

        foreach ($nums as $num) {

            if ($num != $val) {

                $nums[$count++] = $num; // Update the array with elements not equal to val

            }

        }

        return $count;

    }

You are given a string s of even length. Split this string into two halves of equal lengths, and let a be the first half and b be the second half.

Two strings are **alike** if they have the same number of vowels ('a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U'). Notice that s contains uppercase and lowercase letters.

Return true if a and b are ***alike***. Otherwise, return false.

**Example 1:**

**Input:** s = "book"

**Output:** true

**Explanation:** a = "bo" and b = "ok". a has 1 vowel and b has 1 vowel. Therefore, they are alike.

**Example 2:**

**Input:** s = "textbook"

**Output:** false

**Explanation:** a = "text" and b = "book". a has 1 vowel whereas b has 2. Therefore, they are not alike.

Notice that the vowel o is counted twice.

function halvesAreAlike($s) {

        $total = strlen($s);

        $stringArr = str\_split($s);

        $leftCount = 0;

        $rightCount = 0;

        for($i=0; $i<$total; $i++){

            if(stripos("aeiou", $stringArr[$i]) !== false){

                $i < $total/2 ? $leftCount++ : $rightCount++;

            }

        }

        return $leftCount==$rightCount;

    }

[**1347. Minimum Number of Steps to Make Two Strings Anagram**](https://leetcode.com/problems/minimum-number-of-steps-to-make-two-strings-anagram/)

Solved

Medium

Topics

Companies

Hint

You are given two strings of the same length s and t. In one step you can choose **any character** of t and replace it with **another character**.

Return the minimum number of steps to make t an anagram of s.

An **Anagram** of a string is a string that contains the same characters with a different (or the same) ordering.

**Example 1:**

**Input:** s = "bab", t = "aba"

**Output:** 1

**Explanation:** Replace the first 'a' in t with b, t = "bba" which is anagram of s.

**Example 2:**

**Input:** s = "leetcode", t = "practice"

**Output:** 5

**Explanation:** Replace 'p', 'r', 'a', 'i' and 'c' from t with proper characters to make t anagram of s.

**Example 3:**

**Input:** s = "anagram", t = "mangaar"

**Output:** 0

**Explanation:** "anagram" and "mangaar" are anagrams.

function minSteps($s, $t) {

        $count1 = array\_count\_values(str\_split($s));

        $count2 = array\_count\_values(str\_split($t));

        $steps = 0;

        foreach ($count1 as $char => $occurrences) {

            if (!isset($count2[$char])) {

                $steps += $occurrences;

            } else {

                $steps += max($occurrences - $count2[$char], 0);

            }

        }

        return $steps;

    }

[**1657. Determine if Two Strings Are Close**](https://leetcode.com/problems/determine-if-two-strings-are-close/)

Solved

Medium

Topics

Companies

Hint

Two strings are considered **close** if you can attain one from the other using the following operations:

* Operation 1: Swap any two **existing** characters.
  + For example, abcde -> aecdb
* Operation 2: Transform **every** occurrence of one **existing** character into another **existing** character, and do the same with the other character.
  + For example, aacabb -> bbcbaa (all a's turn into b's, and all b's turn into a's)

You can use the operations on either string as many times as necessary.

Given two strings, word1 and word2, return true if word1 and word2 are ***close***, and false otherwise.

**Example 1:**

**Input:** word1 = "abc", word2 = "bca"

**Output:** true

**Explanation:** You can attain word2 from word1 in 2 operations.

Apply Operation 1: "abc" -> "acb"

Apply Operation 1: "acb" -> "bca"

**Example 2:**

**Input:** word1 = "a", word2 = "aa"

**Output:** false

**Explanation:** It is impossible to attain word2 from word1, or vice versa, in any number of operations.

**Example 3:**

**Input:** word1 = "cabbba", word2 = "abbccc"

**Output:** true

**Explanation:** You can attain word2 from word1 in 3 operations.

Apply Operation 1: "cabbba" -> "caabbb"

Apply Operation 2: "caabbb" -> "baaccc"

Apply Operation 2: "baaccc" -> "abbccc"

**<?php**

function closeStrings($word1, $word2) {

    $len1 = strlen($word1);

    $len2 = strlen($word2);

    if ($len1 !== $len2) {

        return false;

    }

    // Count character frequencies for both words

    $charCount1 = array\_count\_values(str\_split($word1));

    $charCount2 = array\_count\_values(str\_split($word2));

    // Check if character frequencies are the same

    if (count($charCount1) !== count($charCount2)) {

        return false;

    }

    // Check if character patterns are the same

    $pattern1 = array\_keys($charCount1);

    $pattern2 = array\_keys($charCount2);

    sort($pattern1);

    sort($pattern2);

    if ($pattern1 !== $pattern2) {

        return false;

    }

    // Check if character frequencies in both words are the same

    $frequency1 = array\_values($charCount1);

    $frequency2 = array\_values($charCount2);

    sort($frequency1);

    sort($frequency2);

    return $frequency1 === $frequency2;

}

// Example usage:

$word1 = "abc";

$word2 = "bca";

$output = closeStrings($word1, $word2);

echo $output ? "true" : "false"; // Output: true

$word1 = "a";

$word2 = "aa";

$output = closeStrings($word1, $word2);

echo $output ? "true" : "false"; // Output: false

$word1 = "cabbba";

$word2 = "abbccc";

$output = closeStrings($word1, $word2);

echo $output ? "true" : "false"; // Output: true

$word1 = "abbzzca";

$word2 = "babzzcz";

$output = closeStrings($word1, $word2);

echo $output ? "true" : "false"; // Output: false

**?>**