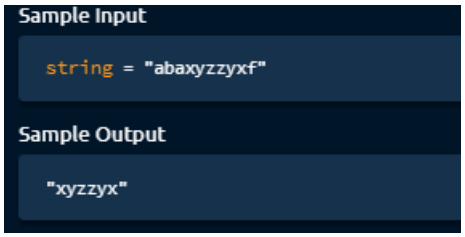
Topic - Strings

Difficulty Medium: Longest Palindromic Substring

Write a function that, given a string, returns its longest palindromic substring. A palindrome is defined as a string that's written the same forward and backward. Note that single-character strings are palindromes. You can assume that there will only be one longest palindromic substring.



Hint:

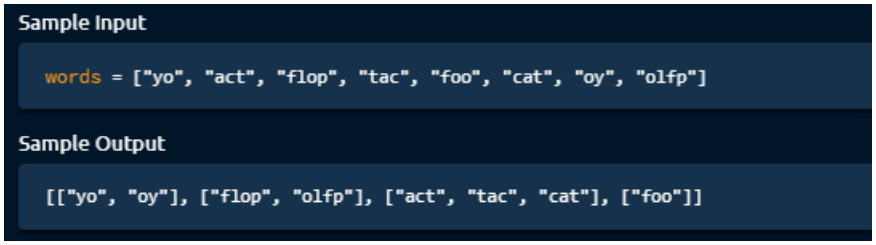
Try generating all possible substrings of the input string and checking for their palindromicity. What is the runtime of the isPalindrome check? What is the total runtime of this approach? Recognize that a palindrome is a string that is symmetrical with respect to its center, which can either be a character (in the case of odd-length palindromes) or an empty string (in the case of even-length palindromes). Thus, you can check the palindromicity of a string by simply expanding from its center and making sure that characters on both sides are indeed mirrored. Traverse the input string, and at each index, apply the logic mentioned in Hint #2. What does this accomplish? Is the runtime of this approach better?

Space-Time Complexity:

O(n^2) time | O(n) space - where n is the length of the input string

Difficulty Medium: Group Anagrams

Write a function that takes in an array of strings and groups anagrams together. Anagrams are strings made up of exactly the same letters, where order doesn't matter. For example, "cinema" and "iceman" are anagrams; similarly, "foo" and "ofo" are anagrams. Your function should return a list of anagram groups in no particular order.



Hint:

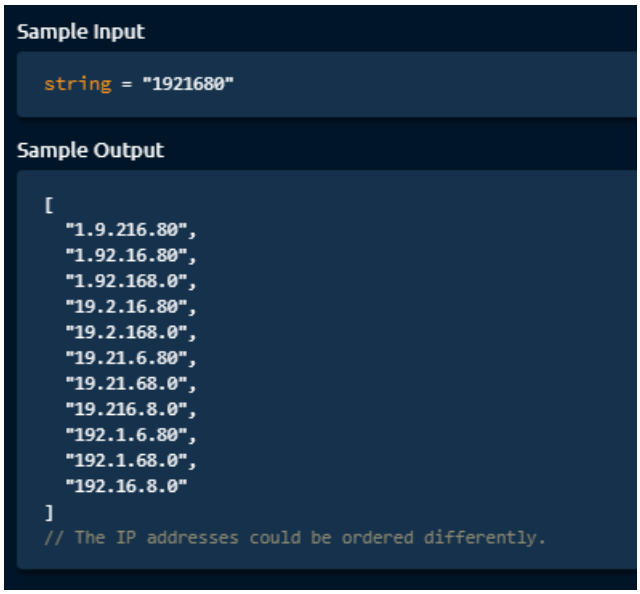
Try rearranging every input string such that each string's letters are ordered in alphabetical order. What can you do with the resulting strings? For any two of the resulting strings mentioned in Hint #1 that are equal to each other, their original strings (with their letters in normal order) must be anagrams. Realizing this, you could bucket all of these resulting strings together, all the while keeping track of their original strings, to find the groups of anagrams. Can you simply store the resulting strings mentioned in Hint above in a hash table and find the groups of anagrams using this hash table?

Space-Time Complexity:

O(w * n * log(n)) time | O(wn) space - where w is the number of words and n is the length of the longest word

Difficulty Medium: Valid IP Addresses

You're given a string of length 12 or smaller, containing only digits. Write a function that returns all the possible IP addresses that can be created by inserting three .s in the string. An IP address is a sequence of four positive integers that are separated by .s, where each individual integer is within the range 0 - 255, inclusive. An IP address isn't valid if any of the individual integers contains leading 0s. For example, "192.168.0.1" is a valid IP address, but "192.168.00.1" and "192.168.0.01" aren't, because they contain "00" and 01, respectively. Another example of a valid IP address is "99.1.1.10"; conversely, "991.1.1.0" isn't valid, because "991" is greater than 255. Your function should return the IP addresses in string format and in no particular order. If no valid IP addresses can be created from the string, your function should return an empty list. Note: check out our Systems Design Fundamentals on SystemsExpert to learn more about IP addresses!



Hint:

How can you split this problem into subproblems to make it easier? Each IP address is comprised of four parts; try finding one valid part at a time and then combining sets of four valid parts to create one valid IP address. Go through all possible combinations of valid IP-address parts. You'll do this by generating a valid first part, then generating all valid second parts given the first part, then finally all valid third and fourth parts given first and second parts. If you find a set of four valid parts, then simply combine them together and add that IP address to some final array. You can start by creating all the possible first parts of an IP address; these will be substrings of the main

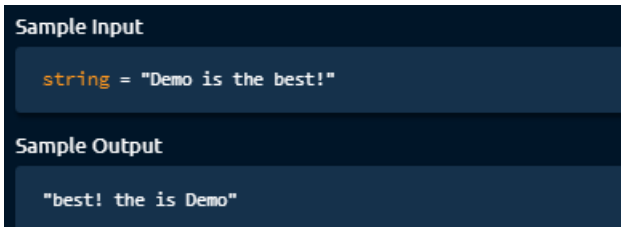
string that start at the first character and that have lengths 1, 2 and 3. Then you can repeat this process for the second part, where the substrings in this part will start where the first part ended. The same thing applies for the third and fourth parts. After going through all possible parts and storing valid IP addresses, you'll have found all of the IP addresses that can be formed from the input string.

Space-Time Complexity:

O(1) time | O(1) space

Difficulty Medium: Reverse Words In String

Write a function that takes in a string of words separated by one or more whitespaces and returns a string that has these words in reverse order. For example, given the string "tim is great", your function should return "great is tim". For this problem, a word can contain special characters, punctuation, and numbers. The words in the string will be separated by one or more whitespaces, and the reversed string must contain the same whitespaces as the original string. For example, given the string "whitespaces   4" you would be expected to return "4   whitespaces". Note that you're not allowed to to use any built-in split or reverse methods/functions. However, you are allowed to use a built-in join method/function. Also note that the input string isn't guaranteed to always contain words.



Hint:

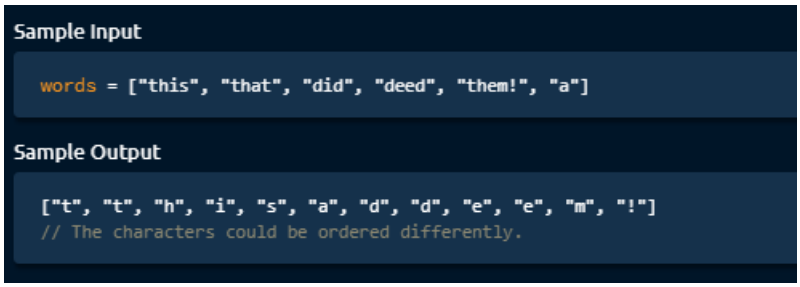
There are at least two ways to solve this problem, and both require locating the words in the string. How can you find all of the words in the string? If you're able to locate all of the words in the string, the next step is to figure out how many spaces are between them. If you can create a list that contains all of the words in the string and all of the spaces between them, then all you need to do is reverse the list and recreate the string using the reversed list. A potentially easier approach to this problem is to start by reversing the entire string. Once the entire string has been reversed, the words will be in the correct order, but each word will also be reversed. From here, all you have to do is reverse all of the individual words in this new string. By doing this, you'll restore each reversed word back to its original order, and you'll have the desired output.

Space-Time Complexity:

O(n) time | O(n) space - where n is the length of the string

Difficulty Medium: Minimum Characters For Words

Write a function that takes in an array of words and returns the smallest array of characters needed to form all of the words. The characters don't need to be in any particular order. For example, the characters ["y", "r", "o", "u"] are needed to form the words ["your", "you", "or", "yo"]. Note: the input words won't contain any spaces; however, they might contain punctuation and/or special characters.



Hint:

There are a few different ways to solve this problem, but all of them use the same general approach. You'll need to determine not only all of the unique characters required to form the input words, but also their required frequencies. What determines the required frequencies of characters to form multiple words? The word that contains the highest frequency of any character dictates how many of those characters are required. For example, given words = ["A", "AAAA"] you need 4 As, because the word that contains the most of amount of As has 4. Use a hash table to keep track of the maximum frequencies of all unique characters that occur across all words. Count the frequency of each character in each word, and use those per-word frequencies to update your maximum-character-frequency hash table. Once you've determined the maximum frequency of each character across all words, you can use the built-up hash table to generate your output array.

Space-Time Complexity:

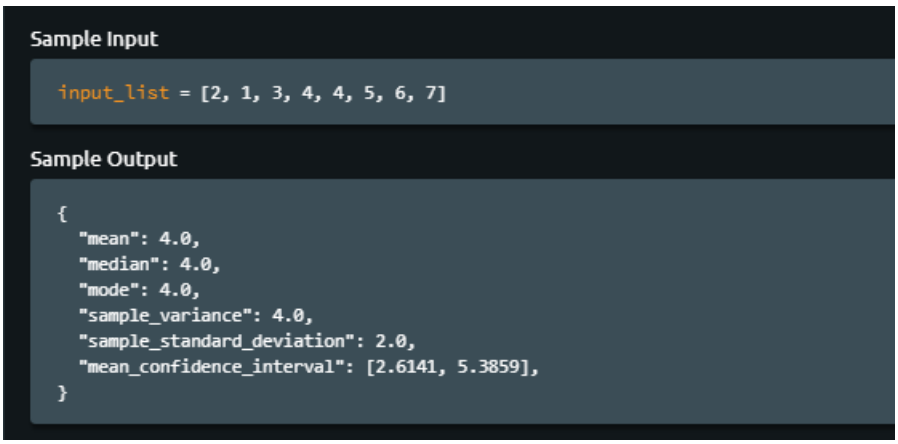
O(n * l) time | O(c) space - where n is the number of words, l is the length of the longest word, and c is the number of unique characters across all words See notes under video explanation for details about the space complexity.

Difficulty Medium: Get Statistics

Write a function that takes in a list of numbers and returns a dictionary containing the following statistics about the numbers: the mean, median, mode, sample variance, sample standard deviation, and 95% confidence interval for the mean.

Note that:

a. **You can assume that the given list contains a large-enough number of samples from a population to use a z-score of 1.96.**
b. **If there's more than one mode, your function can return any of them.**
c. **You shouldn't use any libraries.**
d. **Your output values will automatically be rounded to the fourth decimal.**



Hint:

The mean is the sum of the values divided by the number of values. The median is the middle value of the sorted values. If the input list has an even number of elements, the median is the average of the two middle values. Naturally, you'll first have to sort the input list, and you'll then have to find the middle index (or the two middle indices if the list has an even number of elements).The mode is the most frequent element in the input list. You'll have to count how many times each value appears in the list. What's the difference between a sample variance and a population variance? The variance of a sample is the sum of the squared differences between each element and the mean of the elements. Each term in the sum is divided by the number of input elements minus 1. The sample standard deviation is the square root of the sample variance. To find the 95% confidence interval of

the mean, you first need to find the z-score of 95%, which is 1.96 assuming a two-tailed distribution. This 1.96 is then multiplied by the standard error, which is the standard deviation divided by the square root of the number of input elements. Finally, the lower end of the confidence interval is obtained by subtracting 1.96 * standard_error from the mean, and the upper end is obtained by adding 1.96 * standard_error to the mean.

Space-Time Complexity: