**1** #18

### Problem: 4 Sum

**Description:**

Given an array *S* of *n* integers, are there elements *a*, *b*, *c*, and *d* in *S* such that *a* + *b* + *c* + *d* = target? Find all unique quadruplets in the array which gives the sum of target.

**Note:**

* Elements in a quadruplet (*a*,*b*,*c*,*d*) must be in non-descending order. (ie, *a* ≤ *b* ≤ *c* ≤ *d*)
* The solution set must not contain duplicate quadruplets.

For example, given array S = {1 0 -1 0 -2 2}, and target = 0.

A solution set is:

(-1, 0, 0, 1)

(-2, -1, 1, 2)

(-2, 0, 0, 2)

**2** #3

### Problem: Longest Substring Without Repeating Characters

**Description:**

Given a string, find the length of the longest substring without repeating characters. For example, the longest substring without repeating letters for "abcabcbb" is "abc", which the length is 3. For "bbbbb" the longest substring is "b", with the length of 1.

**3** #36

### Problem: Valid Sudoku (Sudoku定义leetcode上有链接)

### Description:

Determine if a Sudoku is valid, according to: [Sudoku Puzzles - The Rules](http://sudoku.com.au/TheRules.aspx).

The Sudoku board could be partially filled, where empty cells are filled with the character '.'.



A partially filled sudoku which is valid.

**Note:**  
A valid Sudoku board (partially filled) is not necessarily solvable. Only the filled cells need to be validated.

**4** #49

### Problem: Group Anagrams

**Description:**

Given an array of strings, group anagrams together.

For example, given: ["eat", "tea", "tan", "ate", "nat", "bat"],   
Return:

[

["ate", "eat","tea"],

["nat","tan"],

["bat"]

]

**Note:**

1. For the return value, each *inner* list's elements must follow the lexicographic order.
2. All inputs will be in lower-case.

**5** #136

**Problem:** Single Number

**Description:**

Given an array of integers, every element appears *twice* except for one. Find that single one.

**Note:**  
Your algorithm should have a linear runtime complexity. Could you implement it without using extra memory?

**6** #202

### Problem: Happy Number

**Description:**

Write an algorithm to determine if a number is "happy".

A happy number is a number defined by the following process: Starting with any positive integer, replace the number by the sum of the squares of its digits, and repeat the process until the number equals 1 (where it will stay), or it loops endlessly in a cycle which does not include 1. Those numbers for which this process ends in 1 are happy numbers.

**Example:**19 is a happy number

* 12 + 92 = 82
* 82 + 22 = 68
* 62 + 82 = 100
* 12 + 02 + 02 = 1

**7** #204

### Problem: Count Primes

**Description:**

**Description:**

Count the number of prime numbers less than a non-negative number, ***n***.

### 8 #205

### Problem: Isomorphic Strings

**Description:**

Given two strings ***s*** and ***t***, determine if they are isomorphic.

Two strings are isomorphic if the characters in ***s*** can be replaced to get ***t***.

All occurrences of a character must be replaced with another character while preserving the order of characters. No two characters may map to the same character but a character may map to itself.

For example,  
Given "egg", "add", return true.

Given "foo", "bar", return false.

Given "paper", "title", return true.

**Note:**  
You may assume both ***s*** and ***t*** have the same length.

**9 #**187

### Problem: Repeated DNA Sequences

### Description:

All DNA is composed of a series of nucleotides abbreviated as A, C, G, and T, for example: "ACGAATTCCG". When studying DNA, it is sometimes useful to identify repeated sequences within the DNA.

Write a function to find all the 10-letter-long sequences (substrings) that occur more than once in a DNA molecule.

For example,

Given s = "AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT",

Return:

["AAAAACCCCC", "CCCCCAAAAA"].

**10** #166

### Problem: Fraction to Recurring Decimal

### 

**Description:**

Given two integers representing the numerator and denominator of a fraction, return the fraction in string format.

If the fractional part is repeating, enclose the repeating part in parentheses.

For example,

* Given numerator = 1, denominator = 2, return "0.5".
* Given numerator = 2, denominator = 1, return "2".
* Given numerator = 2, denominator = 3, return "0.(6)".