Leetcode.typ

Contents

0001. Two Sum 1
0002. Add Two Numbers 2
0003. Longest Substring Without Repeating Characters 3
0004. Median of Two Sorted Arrays 4
0005. Longest Palindromic Substring 5
0006. Zigzag Conversion 6
0007. Reverse Integer 7
0008. String to Integer (atoi) 8
0009. Palindrome Number 9
0010. Regular Expression Matching 10
0011. Container With Most Water 11
0012. Integer to Roman 12
0013. Roman to Integer 13
0014. Longest Common Prefix 14
0015. 3Sum 15
0016. 3Sum Closest 16
0017. Letter Combinations of a Phone Number 17
0018. 4Sum 18
0019. Remove Nth Node From End of List 19

0001. Two Sum

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.

Test Results

nums	target	Expected	Your Output	Status
[2, 7, 11, 15]	9	[0, 1]	none	✕ Fail
[3, 2, 4]	6	[1, 2]	none	✕ Fail
[3, 3]	6	[0, 1]	none	✕ Fail
[0, 0]	1	[-1, -1]	none	✕ Fail
[1, 4, 7, 10, 13, 16, 19, 22, 25, 28, 31, 34, 37, 40, 43, 46, 49, 52, 55, 58, 61, 64, 67, 70, 73, 76, 79, 82, 85, 88, 91, 94, 97]	191	[31, 32]	none	✕ Fail

0002. Add Two Numbers

You are given two **non-empty** linked lists representing two non-negative integers. The digits are stored in **reverse order**, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

Test Results

l1	l2	Expected	Your Output	Status
2 -> 4 -> 3	5 -> 6 -> 4	7 -> 0 -> 8	none	✕ Fail
0	0	0	none	✕ Fail
9 -> 9 -> 9 -> 9 -> 9 -> 9 -> 9	9 -> 9 -> 9 -> 9	8 -> 9 -> 9 -> 9 -> 0 -> 0 -> 0 -> 1	none	✕ Fail
2 -> 4 -> 3	5 -> 6 -> 4 -> 9	7 -> 0 -> 8 -> 9	none	✕ Fail

0003. Longest Substring Without Repeating Characters

Given a string `s`, find the length of the **longest substring** without repeating characters.

Test Results

s	Expected	Your Output	Status
"abcabcbb"	3	none	✕ Fail
"bbbbb"	1	none	✕ Fail
"pwwkew"	3	none	✕ Fail

0004. Median of Two Sorted Arrays

Given two sorted arrays `nums1` and `nums2` of size `m` and `n` respectively, return the **median** of the two sorted arrays.

The overall run time complexity should be $\mathcal{O}(\log(m+n))$.

Test Results

nums1	nums2	Expected	Your Output	Status
[1, 3]	[2]	2	none	✕ Fail
[1, 2]	[3, 4]	2.5	none	✕ Fail
[0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 51, 54, 57, 60, 63, 66, 69, 72, 75, 78, 81, 84, 87, 90, 93, 96, 99]	[0, 6, 12, 18, 24, 30, 36, 42, 48, 54, 60, 66, 72, 78, 84, 90, 96, 102, 108, 114, 120, 126, 132, 138, 144, 150, 156, 162, 168, 174, 180, 186, 192, 198]	66.0	none	✕ Fail

0005. Longest Palindromic Substring

Given a string `s`, return the **longest palindromic substring** in `s`.

Test Results

s	Expected	Your Output	Status
"babad"	"bab"	none	✕ Fail
"cbbd"	"bb"	none	✕ Fail
"abcdefgfedcbb"	"bcdefgfedcb"	none	✕ Fail
"accc"	"ccc"	none	✕ Fail
"a"	"a"	none	✕ Fail
"aa"	"aa"	none	✕ Fail
"asasfsafdaasfsaasa"	"aasfsaa"	none	✕ Fail

0006. Zigzag Conversion

The string "PAYPALISHIRING" is written in a zigzag pattern on a given number of rows like this:

```
P   A   H   N
A P L S I I G
Y   I   R
```

And then read line by line: "PAHNAPLSIIGYIR"

Write the code that will take a string and make this conversion given a number of rows.

Test Results

s	numRows	Expected	Your Output	Status
"PAYPALISHIR-ING"	3	"PAHNAPLSI-IGYIR"	none	✕ Fail
"PAYPALISHIR-ING"	4	"PINALSI-GYAHRPI"	none	✕ Fail
"A"	1	"A"	none	✕ Fail

0007. Reverse Integer

Given a signed 32-bit integer x, return x with its digits reversed. If reversing x causes the value to go outside the signed 32-bit integer range $[-2^{31}, 2^{31} - 1]$, then return 0.

Assume the environment does not allow you to store 64-bit integers (signed or unsigned).

Test Results

x	Expected	Your Output	Status
123	321	none	✕ Fail
-123	-321	none	✕ Fail
120	21	none	✕ Fail
0	0	none	✕ Fail
23498423	32489432	none	✕ Fail
-213898800	-8898312	none	✕ Fail
1534236469	0	none	✕ Fail
2147483647	0	none	✕ Fail
-2147483648	0	none	✕ Fail

0008. String to Integer (atoi)

Implement the `myAtoi(string s)` function, which converts a string to a 32-bit signed integer (similar to C/C++'s atoi function).

The algorithm for `myAtoi(string s)` is as follows:

1. Read in and ignore any leading whitespace.
2. Check if the next character (if not already at the end of the string) is '-' or '+'. Read this character in if it is either. This determines if the final result is negative or positive respectively. Assume the result is positive if neither is present.
3. Read in next the characters until the next non-digit character or the end of the input is reached. The rest of the string is ignored.
4. Convert these digits into an integer (i.e. "123" -> 123, "0032" -> 32). If no digits were read, then the integer is 0. Change the sign as necessary (from step 2).
5. If the integer is out of the 32-bit signed integer range $[-2^{31}, 2^{31} - 1]$, then clamp the integer so that it remains in the range. Specifically, integers less than -2^{31} should be clamped to -2^{31}, and integers greater than $2^{31} - 1$ should be clamped to $2^{31} - 1$.
6. Return the integer as the final result.

Note:

- Only the space character ' ' is considered a whitespace character.
- **Do not ignore** any characters other than the leading whitespace or the rest of the string after the digits.

Test Results

s	Expected	Your Output	Status
"42"	42	none	✕ Fail
" -42"	-42	none	✕ Fail
"4193 with words"	4193	none	✕ Fail

0009. Palindrome Number

Given an integer `x`, return `true` if `x` is a **palindrome**, and `false` otherwise.

Test Results

x	Expected	Your Output	Status
121	true	none	✕ Fail
-121	false	none	✕ Fail
10	false	none	✕ Fail

0010. Regular Expression Matching

Given an input string s and a pattern p, implement regular expression matching with support for `'.'` and `'*'` where:

- `'.'` Matches any single character.
- `'*'` Matches zero or more of the preceding element.

The matching should cover the **entire** input string (not partial).

Test Results

s	p	Expected	Your Output	Status
"aa"	"a"	false	none	✕ Fail
"aa"	"a*"	true	none	✕ Fail
"ab"	".*"	true	none	✕ Fail
"aab"	"c*a*b"	true	none	✕ Fail
"mississippi"	"mis*is*p*."	false	none	✕ Fail
"ab"	".*c"	false	none	✕ Fail
"ab"	".*c*"	true	none	✕ Fail
"香蕉 x 牛奶"	"香.*牛."	true	none	✕ Fail

0011. Container With Most Water

You are given an integer array height of length n. There are n vertical lines drawn such that the two endpoints of the i^{th} line are `(i, 0)` and `(i, height[i])`.

Find two lines that together with the x-axis form a container, such that the container contains the most water.

Return the maximum amount of water a container can store.

Notice that you may not slant the container.

Test Results

height	Expected	Your Output	Status
[1, 8, 6, 2, 5, 4, 8, 3, 7]	49	none	✕ Fail
[1, 1]	1	none	✕ Fail

0012. Integer to Roman

Roman numerals are represented by seven different symbols: `I`, `V`, `X`, `L`, `C`, `D` and `M`.

Symbol	Value
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

For example, `2` is written as `II` in Roman numeral, just two one's added together. `12` is written as `XII`, which is simply `X + II`. The number `27` is written as `XXVII`, which is `XX + V + II`.

Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not `IIII`. Instead, the number four is written as `IV`. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as `IX`. There are six instances where subtraction is used:

- `I` can be placed before `V` (5) and `X` (10) to make 4 and 9.
- `X` can be placed before `L` (50) and `C` (100) to make 40 and 90.
- `C` can be placed before `D` (500) and `M` (1000) to make 400 and 900.

Given an integer, convert it to a roman numeral.

Test Results

num	Expected	Your Output	Status
3	"III"	none	✕ Fail
58	"LVIII"	none	✕ Fail
1994	"MCMXCIV"	none	✕ Fail

0013. Roman to Integer

Roman numerals are represented by seven different symbols: `I`, `V`, `X`, `L`, `C`, `D` and `M`.

Symbol	Value
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

For example, `2` is written as `II` in Roman numeral, just two one's added together. `12` is written as `XII`, which is simply `X + II`. The number `27` is written as `XXVII`, which is `XX + V + II`.

Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not `IIII`. Instead, the number four is written as `IV`. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as `IX`. There are six instances where subtraction is used:

- `I` can be placed before `V` (5) and `X` (10) to make 4 and 9.
- `X` can be placed before `L` (50) and `C` (100) to make 40 and 90.
- `C` can be placed before `D` (500) and `M` (1000) to make 400 and 900.

Given a roman numeral, convert it to an integer.

Test Results

s	Expected	Your Output	Status
"III"	3	none	✕ Fail
"LVIII"	58	none	✕ Fail
"MCMXCIV"	1994	none	✕ Fail

0014. Longest Common Prefix

Write a function to find the longest common prefix string amongst an array of strings.

If there is no common prefix, return an empty string `""`.

Test Results

strs	Expected	Your Output	Status
["flower", "flow", "flight"]	"fl"	none	✕ Fail
["dog", "racecar", "car"]	""	none	✕ Fail

0015. 3Sum

Given an integer array nums, return all the triplets `[nums[i], nums[j], nums[k]]` such that `i != j`, `i != k`, and `j != k`, and `nums[i] + nums[j] + nums[k] == 0`.

Notice that the solution set must not contain duplicate triplets.

Test Results

nums	Expected	Your Output	Status
[-1, 0, 1, 2, -1, -4]	[[-1, -1, 2], [-1, 0, 1]]	none	✕ Fail
[0, 1, 1]	[]	none	✕ Fail
[0, 0, 0]	[[0, 0, 0]]	none	✕ Fail
[-10, -7, -4, -1, 2, 5, 8, 11, 14, 17]	[[-10, -7, 17], [-10, -4, 14], [-10, -1, 11], [-10, 2, 8], [-7, -4, 11], [-7, -1, 8], [-7, 2, 5], [-4, -1, 5]]	none	✕ Fail
[-10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]	[[-10, 1, 9], [-10, 2, 8], [-10, 3, 7], [-10, 4, 6], [-9, 0, 9], [-9, 1, 8], [-9, 2, 7], [-9, 3, 6], [-9, 4, 5], [-8, -1, 9], [-8, 0, 8], [-8, 1, 7], [-8, 2, 6], [-8, 3, 5], [-7, -2, 9], [-7, -1, 8], [-7, 0, 7], [-7, 1, 6], [-7, 2, 5], [-7, 3, 4], [-6, -3, 9], [-6, -2, 8], [-6, -1, 7], [-6, 0, 6], [-6, 1, 5], [-6, 2, 4], [-5, -4, 9], [-5, -3, 8], [-5, -2, 7], [-5, -1, 6], [-5, 0, 5], [-5, 1, 4], [-5, 2, 3], [-4, -3, 7], [-4, -2, 6], [-4, -1, 5], [-4, 0, 4], [-4, 1, 3], [-3, -2, 5], [-3, -1, 4], [-3, 0, 3], [-3, 1, 2], [-2, -1, 3], [-2, 0, 2], [-1, 0, 1]]	none	✕ Fail

0016. 3Sum Closest

Given an integer array `nums` of length `n` and an integer `target`, find three integers in `nums` such that the sum is closest to `target`.

Return the sum of the three integers.

You may assume that each input would have exactly one solution.

Test Results

nums	target	Expected	Your Output	Status
[-1, 2, 1, -4]	1	2	none	✕ Fail
[0, 0, 0]	1	0	none	✕ Fail
[0, 1, 1]	2	2	none	✕ Fail
[-10, -7, -4, -1, 2, 5, 8, 11, 14, 17]	20	21	none	✕ Fail
[-10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]	30	24	none	✕ Fail

0017. Letter Combinations of a Phone Number

Given a string containing digits from `2-9` inclusive, return all possible letter combinations that the number could represent. Return the answer in **any order**.

A mapping of digits to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.

Test Results

digits	Expected	Your Output	Status
"23"	["ad", "bd", "cd", "ae", "be", "ce", "af", "bf", "cf"]	none	✕ Fail
""	[]	none	✕ Fail
"2"	["a", "b", "c"]	none	✕ Fail

0018. 4Sum

Given an array `nums` of `n` integers, return an array of all the unique quadruplets `[nums[a], nums[b], nums[c], nums[d]]` such that:

- `0 <= a, b, c, d < n`
- `a`, `b`, `c`, and `d` are **distinct**.
- `nums[a] + nums[b] + nums[c] + nums[d] == target`

You may return the answer in **any order**.

Test Results

nums	target	Expected	Your Output	Status
[1, 0, -1, 0, -2, 2]	0	[[-2, -1, 1, 2], [-2, 0, 0, 2], [-1, 0, 0, 1]]	none	✕ Fail
[2, 2, 2, 2]	8	[[2, 2, 2, 2]]	none	✕ Fail
[-5, -4, -3, -2, -1, 0, 1, 2, 3, 4]	3	[[-5, 1, 3, 4], [-4, 0, 3, 4], [-4, 1, 2, 4], [-3, -1, 3, 4], [-3, 0, 2, 4], [-3, 1, 2, 3], [-2, -1, 2, 4], [-2, 0, 1, 4], [-2, 0, 2, 3], [-1, 0, 1, 3]]	none	✕ Fail

0019. Remove Nth Node From End of List

Given the head of a linked list, remove the nth node from the end of the list and return its head.

Test Results

head	n	Expected	Your Output	Status
1 -> 2 -> 3 -> 4 -> 5	2	1 -> 2 -> 3 -> 5	none	✕ Fail
1	1	∅	none	✕ Fail
1 -> 2	1	1	none	✕ Fail