



Gabriel Wu (@lucifer1004)

January 01, 2026

Contents

0001. Two Sum	1
0002. Add Two Numbers	3
0003. Longest Substring Without Repeating Characters	5
0004. Median of Two Sorted Arrays	6
0005. Longest Palindromic Substring	7
0006. Zigzag Conversion	9
0007. Reverse Integer	10
0008. String to Integer (atoi)	12
0009. Palindrome Number	14
0010. Regular Expression Matching	15
0011. Container With Most Water	17
0012. Integer to Roman	19
0013. Roman to Integer	21
0014. Longest Common Prefix	23
0015. 3Sum	24
0016. 3Sum Closest	26
0017. Letter Combinations of a Phone Number	28
0018. 4Sum	29
0019. Remove Nth Node From End of List	30
0020. Valid Parentheses	31
0021. Merge Two Sorted Lists	32
0022. Generate Parentheses	33
0023. Merge k Sorted Lists	34
0024. Swap Nodes in Pairs	35
0025. Reverse Nodes in k-Group	36
0026. Remove Duplicates from Sorted Array (Adapted)	37
0033. Search in Rotated Sorted Array	38
0035. Search Insert Position	40
0039. Combination Sum	41
0042. Trapping Rain Water	43
0046. Permutations	45
0048. Rotate Image	46
0049. Group Anagrams	47

0050. Pow(x, n)	48
0051. N-Queens	50
0053. Maximum Subarray	51
0054. Spiral Matrix	53
0055. Jump Game	54
0056. Merge Intervals	55
0062. Unique Paths	56
0069. Sqrt(x)	57
0070. Climbing Stairs	58
0072. Edit Distance	60
0076. Minimum Window Substring	62
0078. Subsets	64
0094. Binary Tree Inorder Traversal	65
0101. Symmetric Tree	66
0104. Maximum Depth of Binary Tree	68
0110. Balanced Binary Tree	69
0112. Path Sum	70
0113. Path Sum II	72
0116. Populating Next Right Pointers in Each Node	73
0121. Best Time to Buy and Sell Stock	75
0144. Binary Tree Preorder Traversal	76
0145. Binary Tree Postorder Traversal	77
0155. Min Stack	78
0200. Number of Islands	79
0206. Reverse Linked List	81
0207. Course Schedule	82
0218. The Skyline Problem	84
0209. Minimum Size Subarray Sum	88
0210. Course Schedule II	89
0289. Game of Life	91
0347. Top K Frequent Elements	92
0407. Trapping Rain Water II	93
0547. Number of Provinces	96
0785. Is Graph Bipartite?	98

0814. Binary Tree Pruning	100
0997. Find the Town Judge	102

0001. Two Sum

EASYARRAYHASH-TABLE

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 (0 / 5)

Case 1

nums: [2, 7, 11, 15]

target: 9

Because `nums[0] + nums[1] == 9`, we return `[0, 1]`.

Expected
[0, 1]

Your Output
Got none (check your code)

Case 2

nums: [3, 2, 4]

target: 6

Expected
[1, 2]

Your Output
Got none (check your code)

Case 3

nums: [3, 3]

target: 6

Expected
[0, 1]

Your Output
Got none (check your code)

Case 4

nums: [0, 0]

target: 1

Expected
[-1, -1]

Your Output
Got none (check your code)

Case 5

nums: [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]

target: 191

Expected	Your Output
[31, 32]	<i>Got none (check your code)</i>

0002. Add Two Numbers

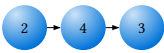
MEDIUMLINKED-LISTMATHRECURSION

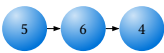
You are given two **non-empty** linked lists representing two non-negative integers. The digits are stored in **reverse order**, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

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

Test Results (0 / 4)

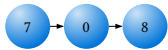
Case 1

l1: 

l2: 

$342 + 465 = 807$.

Expected



Your Output

Got none (check your code)

Case 2

l1: 

l2: 

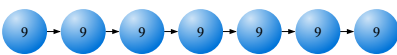
Expected

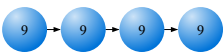


Your Output

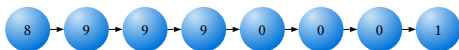
Got none (check your code)

Case 3

l1: 

l2: 

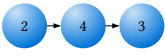
Expected

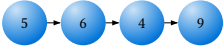


Your Output

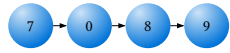
Got none (check your code)

Case 4

l1: 

l2: 

Expected

Your Output
<i>Got none (check your code)</i>

0003. Longest Substring Without Repeating Characters

MEDIUM

STRING

HASH-TABLE

SLIDING-WINDOW

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

Test Results (0 / 3)

Case 1

s: "abcabcbb"

The answer is "abc", with the length of 3. Note that "bca" and "cab" are also correct answers.

Expected	Your Output
3	Got none (check your code)

Case 2

s: "bbbbbb"

The answer is "b", with the length of 1.

Expected	Your Output
1	Got none (check your code)

Case 3

s: "pwwkew"

The answer is "wke", with the length of 3. Notice that the answer must be a substring, "pwke" is a subsequence and not a substring.

Expected	Your Output
3	Got none (check your code)

0004. Median of Two Sorted Arrays

HARDARRAYBINARY-SEARCHDIVIDE-AND-CONQUER

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 (0 / 3)

Case 1

nums1: [1, 3]

nums2: [2]

Merged array = [1, 2, 3] and median is 2.

Expected
2

Your Output

Got none (check your code)

Case 2

nums1: [1, 2]

nums2: [3, 4]

Merged array = [1, 2, 3, 4] and median is $\frac{2+3}{2} = 2.5$.

Expected
2.5

Your Output

Got none (check your code)

Case 3

nums1: [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]

nums2: [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]

Expected
66.0

Your Output

Got none (check your code)

0005. Longest Palindromic Substring

MEDIUM

STRING

DYNAMIC-PROGRAMMING

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

Test Results (0 / 7)

Case 1

s: "abad"

"aba" is also a valid answer.

Expected	Your Output
"bab"	<i>Got none (check your code)</i>

Case 2

s: "cbdd"

Expected	Your Output
"bb"	<i>Got none (check your code)</i>

Case 3

s: "abcdefg fedcbb"

Expected	Your Output
"bcdefg fedcb"	<i>Got none (check your code)</i>

Case 4

s: "accc"

Expected	Your Output
"ccc"	<i>Got none (check your code)</i>

Case 5

s: "a"

Expected	Your Output
"a"	<i>Got none (check your code)</i>

Case 6

s: "aa"

Expected
"aa"

Your Output
<i>Got none (check your code)</i>

Case 7

s: "asasfsafdaasfsaasa"

Expected
"aasfsaa"

Your Output
<i>Got none (check your code)</i>

0006. Zigzag Conversion

MEDIUMSTRING

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 (0 / 3)

Case 1

s: "PAYPALISHIRING"

numRows: 3

Expected	Your Output
"PAHNAPLSIIGYIR"	<i>Got none (check your code)</i>

Case 2

s: "PAYPALISHIRING"

numRows: 4

<pre>P I N A L S I G Y A H R P I</pre>	
Expected	Your Output
"PINALSIGYAHRPI"	<i>Got none (check your code)</i>

Case 3

s: "A"

numRows: 1

Expected	Your Output
"A"	<i>Got none (check your code)</i>

0007. Reverse Integer

MEDIUMMATH

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 (0 / 9)

Case 1

x: 123

Expected
321

Your Output
<i>Got none (check your code)</i>

Case 2

x: -123

Expected
-321

Your Output
<i>Got none (check your code)</i>

Case 3

x: 120

Expected
21

Your Output
<i>Got none (check your code)</i>

Case 4

x: 0

Expected
0

Your Output
<i>Got none (check your code)</i>

Case 5

x: 23498423

Expected
32489432

Your Output
<i>Got none (check your code)</i>

Case 6

x: -213898800

Expected
-8898312

Your Output
<i>Got none (check your code)</i>

Case 7

x: 1534236469

Expected
0

Your Output
<i>Got none (check your code)</i>

Case 8

x: 2147483647

Expected
0

Your Output
<i>Got none (check your code)</i>

Case 9

x: -2147483648

Expected
0

Your Output
<i>Got none (check your code)</i>

0008. String to Integer (atoi)

MEDIUM

STRING

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 (0 / 5)

Case 1

s: "42"

Expected	Your Output
42	Got none (check your code)

Case 2

s: " -042"

Leading whitespace is read and ignored. '-' is read, so the result should be negative. "042" is read in, leading zeros ignored in the result.

Expected	Your Output
-42	Got none (check your code)

Case 3

s: "1337c0d3"

"1337" is read in; reading stops because the next character is a non-digit.

Expected	Your Output
1337	<i>Got none (check your code)</i>

Case 4

s: "0-1"

"0" is read in; reading stops because the next character is a non-digit.

Expected	Your Output
0	<i>Got none (check your code)</i>

Case 5

s: "words and 987"

Reading stops at the first non-digit character 'w'.

Expected	Your Output
0	<i>Got none (check your code)</i>

0009. Palindrome Number

EASYMATH

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

Test Results (0 / 3)

Case 1

x: 121

121 reads as 121 from left to right and from right to left.

Expected	Your Output
true	<i>Got none (check your code)</i>

Case 2

x: -121

From left to right, it reads -121. From right to left, it becomes 121-. Therefore it is not a palindrome.

Expected	Your Output
false	<i>Got none (check your code)</i>

Case 3

x: 10

Reads 01 from right to left. Therefore it is not a palindrome.

Expected	Your Output
false	<i>Got none (check your code)</i>

0010. Regular Expression Matching

HARD

STRING

DYNAMIC-PROGRAMMING

RECURSION

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 (0 / 8)

Case 1

s: "aa"

p: "a"

"a" does not match the entire string "aa".

Expected
false

Your Output

Got none (check your code)

Case 2

s: "aa"

p: "a*"

'*' means zero or more of the preceding element, 'a'. Therefore, by repeating 'a' once, it becomes "aa".

Expected
true

Your Output

Got none (check your code)

Case 3

s: "ab"

p: ".*"

".*" means "zero or more (*) of any character (.)".

Expected
true

Your Output

Got none (check your code)

Case 4

s: "aab"

p: "c*a*b"

Expected
true

Your Output
<i>Got none (check your code)</i>

Case 5

s: "mississippi"

p: "mis*is*p*"

Expected
false

Your Output
<i>Got none (check your code)</i>

Case 6

s: "ab"

p: ".*c"

Expected
false

Your Output
<i>Got none (check your code)</i>

Case 7

s: "ab"

p: ".*c*"

Expected
true

Your Output
<i>Got none (check your code)</i>

Case 8

s: "xxxxx xxx"

p: "xx.*xx"

Expected
true

Your Output
<i>Got none (check your code)</i>

0011. Container With Most Water

MEDIUM

ARRAY

TWO-POINTERS

GREEDY

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 $(i, \text{height}[i])$.

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

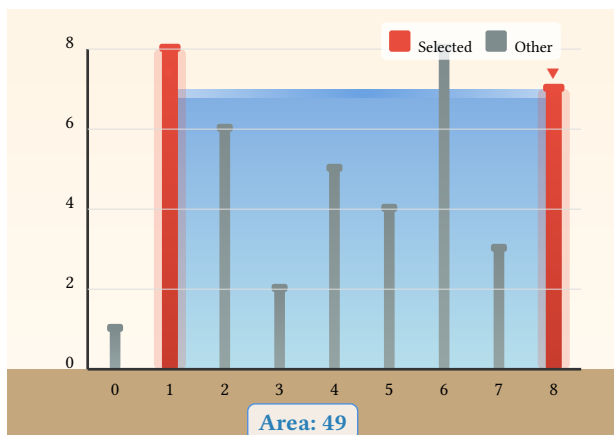
Return the maximum amount of water a container can store.

Notice that you may not slant the container.

Test Results (0 / 3)

Case 1

height: [1, 8, 6, 2, 5, 4, 8, 3, 7]



The above vertical lines are represented by array [1, 8, 6, 2, 5, 4, 8, 3, 7]. In this case, the max area of water (blue section) the container can contain is 49.

Expected

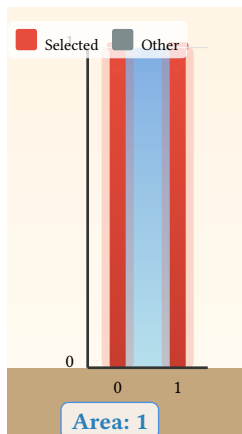
49

Your Output

Got none (check your code)

Case 2

height: [1, 1]



Expected

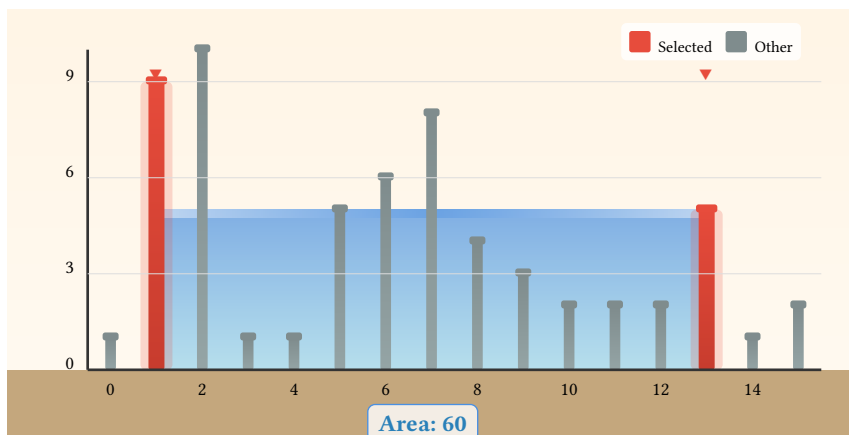
1

Your Output

Got none (check your code)

Case 3

height: [1, 9, 10, 1, 1, 5, 6, 8, 4, 3, 2, 2, 2, 5, 1, 2]



Expected

60

Your Output

Got none (check your code)

0012. Integer to Roman

MEDIUM

HASH-TABLE

MATH

STRING

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 (0 / 3)

Case 1

num: 3749

3000 = MMM as 1000 (M) + 1000 (M) + 1000 (M)
700 = DCC as 500 (D) + 100 (C) + 100 (C)
40 = XL as 10 (X) less of 50 (L)
9 = IX as 1 (I) less of 10 (X)

Expected

"MMMDCCLXIX"

Your Output

Got none (check your code)

Case 2

num: 58

50 = L, 8 = VIII

Expected

"LVIII"

Your Output

Got none (check your code)

Case 3

num: 1994

1000 = M, 900 = CM, 90 = XC, 4 = IV

Expected	Your Output
"MCMXCIV"	<i>Got none (check your code)</i>

0013. Roman to Integer

EASYHASH-TABLEMATHSTRING

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 (0 / 3)

Case 1

s: "III"

III = 3.

Expected
3

Your Output
Got none (check your code)

Case 2

s: "LVIII"

L = 50, V = 5, III = 3.

Expected
58

Your Output
Got none (check your code)

Case 3

s: "MCMXCIV"

M = 1000, CM = 900, XC = 90 and IV = 4.

Expected	Your Output
1994	<i>Got none (check your code)</i>

0014. Longest Common Prefix

EASYSTRINGTRIE

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 (0 / 2)

Case 1

strs: ["flower", "flow", "flight"]

Expected	Your Output
"fl"	<i>Got none (check your code)</i>

Case 2

strs: ["dog", "racecar", "car"]

There is no common prefix among the input strings.

Expected	Your Output
""	<i>Got none (check your code)</i>

0015. 3Sum

MEDIUMARRAYTWO-POINTERSSORTING

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

Notice that the solution set must not contain duplicate triplets.

Test Results (0 / 5)

Case 1

nums: [-1, 0, 1, 2, -1, -4]

```
nums[0] + nums[1] + nums[2] = (-1) + 0 + 1 = 0.  
nums[1] + nums[2] + nums[4] = 0 + 1 + (-1) = 0.  
nums[0] + nums[3] + nums[4] = (-1) + 2 + (-1) = 0.  
The distinct triplets are [-1, 0, 1] and [-1, -1, 2].
```

Expected
[[-1, -1, 2], [-1, 0, 1]]

Your Output
Got none (check your code)

Case 2

nums: [0, 1, 1]

The only possible triplet does not sum up to 0.

Expected
[]

Your Output
Got none (check your code)

Case 3

nums: [0, 0, 0]

The only possible triplet sums up to 0.

Expected
[[0, 0, 0]]

Your Output
Got none (check your code)

Case 4

nums: [-10, -7, -4, -1, 2, 5, 8, 11, 14, 17]

Expected
[[-10, -7, 17], [-10, -4, 14], [-10, -1, 11], [-10, 2, 8], [-7, -4, 11], [-7, -1, 8], [-7, 2, 5], [-4, -1, 5]]

Your Output
Got none (check your code)

Case 5

nums: [-10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

Expected	Your Output
[[[-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]]]	<i>Got none (check your code)</i>

0016. 3Sum Closest

MEDIUMARRAYTWO-POINTERSSORTING

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 (0 / 5)

Case 1

nums: [-1, 2, 1, -4]

target: 1

The sum that is closest to the target is 2. $(-1) + 2 + 1 = 2$.

Expected
2

Your Output
<i>Got none (check your code)</i>

Case 2

nums: [0, 0, 0]

target: 1

The sum that is closest to the target is 0. $0 + 0 + 0 = 0$.

Expected
0

Your Output
<i>Got none (check your code)</i>

Case 3

nums: [0, 1, 1]

target: 2

Expected
2

Your Output
<i>Got none (check your code)</i>

Case 4

nums: [-10, -7, -4, -1, 2, 5, 8, 11, 14, 17]

target: 20

Expected
21

Your Output
<i>Got none (check your code)</i>

Case 5

nums: [-10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

target: 30

Expected	Your Output
24	<i>Got none (check your code)</i>

0017. Letter Combinations of a Phone Number

MEDIUM

HASH-TABLE

STRING

BACKTRACKING

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 (0 / 3)

Case 1

digits: "23"

Expected
["ad", "bd", "cd", "ae", "be", "ce", "af", "bf", "cf"]

Your Output

Got none (check your code)

Case 2

digits: "2"

Expected
["a", "b", "c"]

Your Output

Got none (check your code)

Case 3

digits: ""

Expected
[]

Your Output

Got none (check your code)

0018. 4Sum

MEDIUMARRAYTWO-POINTERSSORTING

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:

- $0 \leq a, b, c, d < n$
- `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 (0 / 3)

Case 1

nums: [1, 0, -1, 0, -2, 2]

target: 0

Expected	Your Output
[[-2, -1, 1, 2], [-2, 0, 0, 2], [-1, 0, 0, 1]]	<i>Got none (check your code)</i>

Case 2

nums: [2, 2, 2, 2, 2]

target: 8

Expected	Your Output
[[2, 2, 2, 2]]	<i>Got none (check your code)</i>

Case 3

nums: [-5, -4, -3, -2, -1, 0, 1, 2, 3, 4]

target: 3

Expected	Your Output
[[-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]]	<i>Got none (check your code)</i>

0019. Remove Nth Node From End of List

MEDIUM

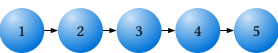
LINKED-LIST

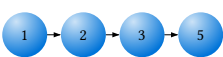
TWO-POINTERS

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

Test Results (0 / 3)


Case 1


head: 
n: 2

Expected


Your Output
<i>Got none (check your code)</i>

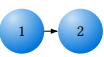
Case 2


head: 
n: 1

Expected


Your Output
<i>Got none (check your code)</i>

Case 3

head: 
n: 1

Expected


Your Output
<i>Got none (check your code)</i>

0020. Valid Parentheses

EASYSTRINGSTACK

Given a string `s` containing just the characters `'('`, `')'`, `'{'`, `'}'`, `'['` and `']'`, determine if the input string is valid.

An input string is valid if:

1. Open brackets must be closed by the same type of brackets.
 2. Open brackets must be closed in the correct order.
 3. Every close bracket has a corresponding open bracket of the same type.
- `s` consists of parentheses only `'()[]{}'`.

Test Results (0 / 5)

Case 1

`s: "()"`

Expected
true

Your Output
Got none (check your code)

Case 2

`s: "()[]{}"`

Expected
true

Your Output
Got none (check your code)

Case 3

`s: "[]"`

Expected
false

Your Output
Got none (check your code)

Case 4

`s: "([])"`

Expected
true

Your Output
Got none (check your code)

Case 5

`s: "([)]"`

Expected
false

Your Output
Got none (check your code)

0021. Merge Two Sorted Lists

EASY

LINKED-LIST

RECURSION

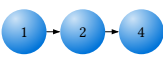
You are given the heads of two sorted linked lists `list1` and `list2`.

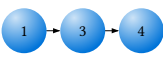
Merge the two lists into one sorted list. The list should be made by splicing together the nodes of the first two lists.

Return the head of the merged linked list.

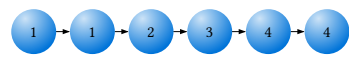
Test Results (0 / 3)

Case 1

list1: 

list2: 

Expected



Your Output

Got none (check your code)

Case 2

list1: 

list2: 

Expected



Your Output

Got none (check your code)

Case 3

list1: 

list2: 

Expected



Your Output

Got none (check your code)

0022. Generate Parentheses

MEDIUM

STRING

DYNAMIC-PROGRAMMING

BACKTRACKING

Given n pairs of parentheses, write a function to generate all combinations of well-formed parentheses.

Test Results (0 / 2)

Case 1

n: 3

Expected	Your Output
<code>["((()))", "(())()", "(())()", "()(())", "()()()"]</code>	<i>Got none (check your code)</i>

Case 2

n: 1

Expected	Your Output
<code>["()"]</code>	<i>Got none (check your code)</i>

0023. Merge k Sorted Lists

HARD

LINKED-LIST

DIVIDE-AND-CONQUER

HEAP

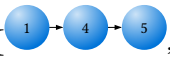
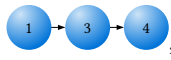
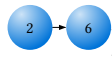
MERGE-SORT

You are given an array of k linked-lists `lists`, each linked-list is sorted in ascending order.

Merge all the linked-lists into one sorted linked-list and return it.

Test Results (0 / 3)

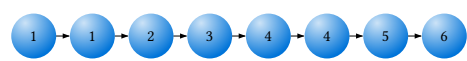
Case 1

lists: [ ,  , ]

The linked-lists are: 1->4->5, 1->3->4, 2->6.

Merging them into one sorted linked list: 1->1->2->3->4->4->5->6.

Expected



Your Output

Got none (check your code)

Case 2

lists: []

Expected



Your Output

Got none (check your code)

Case 3

lists: []

Expected



Your Output

Got none (check your code)

0024. Swap Nodes in Pairs

MEDIUM

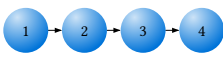
LINKED-LIST

RECURSION

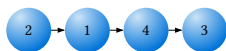
Given a linked list, swap every two adjacent nodes and return its head.

Test Results (0 / 4)

Case 1

head: 

Expected



Your Output

Got none (check your code)

Case 2

head: 

Expected



Your Output

Got none (check your code)

Case 3

head: 

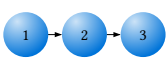
Expected



Your Output

Got none (check your code)

Case 4

head: 

Expected



Your Output

Got none (check your code)

0025. Reverse Nodes in k-Group

HARD

LINKED-LIST

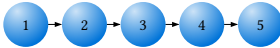
RECURSION

Given the head of a linked list, reverse the nodes of the list k at a time, and return the modified list.

k is a positive integer and is less than or equal to the length of the linked list. If the number of nodes is not a multiple of k then left-out nodes, in the end, should remain as it is.

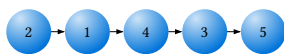
Test Results (0 / 2)

Case 1

head: 

k: 2

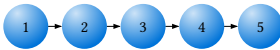
Expected



Your Output

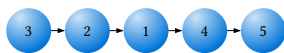
Got none (check your code)

Case 2

head: 

k: 3

Expected



Your Output

Got none (check your code)

0026. Remove Duplicates from Sorted Array (Adapted)

EASYARRAYTWO-POINTERS

- This problem is different from the original version since functions in Typst cannot modify the input array in place.

Given an integer array `nums` sorted in non-decreasing order, remove the duplicates in-place such that each unique element appears only once. The relative order of the elements should be kept the same.

After removing duplicates, return the unique elements in ascending order.

Test Results (0 / 2)

Case 1

nums: [1, 1, 2]

Your function should return $k = 2$, with the first two elements of `nums` being 1 and 2 respectively.

Expected	Your Output
[1, 2]	Got none (check your code)

Case 2

nums: [0, 0, 1, 1, 1, 2, 2, 3, 3, 4]

Your function should return $k = 5$, with the first five elements of `nums` being 0, 1, 2, 3, and 4 respectively.

Expected	Your Output
[0, 1, 2, 3, 4]	Got none (check your code)

0033. Search in Rotated Sorted Array

MEDIUMARRAY

BINARY-SEARCH

There is an integer array `nums` sorted in ascending order (with **distinct** values).

Prior to being passed to your function, `nums` is **possibly left rotated** at an unknown index k ($1 \leq k < n$) such that the resulting array is `[nums[k], nums[k+1], ..., nums[n-1], nums[0], nums[1], ..., nums[k-1]]` (**0-indexed**). For example, `[0,1,2,4,5,6,7]` might be left rotated by 3 indices and become `[4,5,6,7,0,1,2]`.

Given the array `nums` **after** the possible rotation and an integer `target`, return the index of `target` if it is in `nums`, or `-1` if it is not in `nums`.

You must write an algorithm with $\mathcal{O}(\log n)$ runtime complexity.

Test Results (0 / 8)

Case 1

nums: `[4, 5, 6, 7, 0, 1, 2]`

target: `0`

Expected
4

Your Output
<i>Got none (check your code)</i>

Case 2

nums: `[4, 5, 6, 7, 0, 1, 2]`

target: `3`

Expected
-1

Your Output
<i>Got none (check your code)</i>

Case 3

nums: `[1]`

target: `0`

Expected
-1

Your Output
<i>Got none (check your code)</i>

Case 4

nums: `[1]`

target: `1`

Expected
0

Your Output
<i>Got none (check your code)</i>

Case 5

nums: [3, 1]

target: 1

Expected
1

Your Output
<i>Got none (check your code)</i>

Case 6

nums: [5, 1, 3]

target: 5

Expected
0

Your Output
<i>Got none (check your code)</i>

Case 7

nums: [4, 5, 6, 7, 0, 1, 2]

target: 4

Expected
0

Your Output
<i>Got none (check your code)</i>

Case 8

nums: [4, 5, 6, 7, 0, 1, 2]

target: 2

Expected
6

Your Output
<i>Got none (check your code)</i>

0035. Search Insert Position

EASYARRAYBINARY-SEARCH

Given a sorted array of distinct integers and a target value, return the index if the target is found. If not, return the index where it would be if it were inserted in order.

You **must** write an algorithm with $\mathcal{O}(\log n)$ runtime complexity.

Test Results (0 / 5)

Case 1

nums: [1, 3, 5, 6]

target: 5

Expected
2

Your Output
<i>Got none (check your code)</i>

Case 2

nums: [1, 3, 5, 6]

target: 2

Expected
1

Your Output
<i>Got none (check your code)</i>

Case 3

nums: [1, 3, 5, 6]

target: 7

Expected
4

Your Output
<i>Got none (check your code)</i>

Case 4

nums: [1, 3, 5, 6]

target: 0

Expected
0

Your Output
<i>Got none (check your code)</i>

Case 5

nums: [1]

target: 0

Expected
0

Your Output
<i>Got none (check your code)</i>

0039. Combination Sum

MEDIUMARRAYBACKTRACKING

Given an array of **distinct** integers `candidates` and a target integer `target`, return a list of all **unique combinations** of `candidates` where the chosen numbers sum to `target`. You may return the combinations in **any order**.

The **same** number may be chosen from `candidates` an **unlimited number of times**. Two combinations are unique if the frequency of at least one of the chosen numbers is different.

The test cases are generated such that the number of unique combinations that sum up to `target` is less than 150 combinations for the given input.

Test Results (0 / 6)

Case 1

candidates: [2, 3, 6, 7]

target: 7

2 and 3 are candidates, and $2 + 2 + 3 = 7$. 7 is a candidate, and $7 = 7$.

Expected	Your Output
[[2, 2, 3], [7]]	Got none (check your code)

Case 2

candidates: [2, 3, 5]

target: 8

Expected	Your Output
[[2, 2, 2, 2], [2, 3, 3], [3, 5]]	Got none (check your code)

Case 3

candidates: [2]

target: 1

Expected	Your Output
[]	Got none (check your code)

Case 4

candidates: [1]

target: 1

Expected	Your Output
[[1]]	Got none (check your code)

Case 5

candidates: [1]

target: 2

Expected	Your Output
[[1, 1]]	<i>Got none (check your code)</i>

Case 6

candidates: [1, 2]

target: 4

Expected	Your Output
[[1, 1, 1, 1], [1, 1, 2], [2, 2]]	<i>Got none (check your code)</i>

0042. Trapping Rain Water

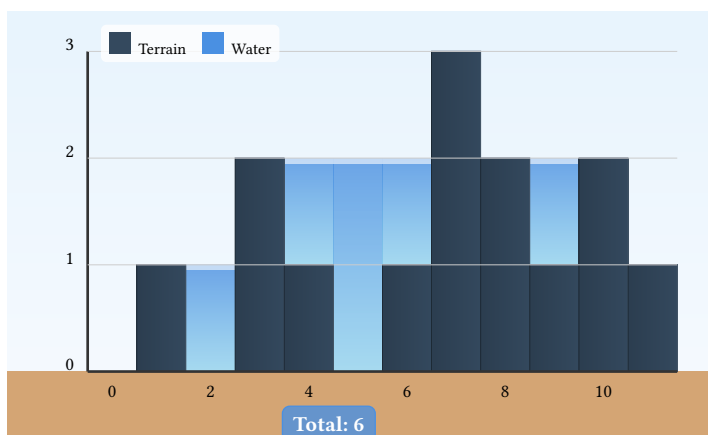
HARDARRAYTWO-POINTERSDYNAMIC-PROGRAMMINGSTACKMONOTONIC-STACK

Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it can trap after raining.

Test Results (0 / 3)

Case 1

height: [0, 1, 0, 2, 1, 0, 1, 3, 2, 1, 2, 1]



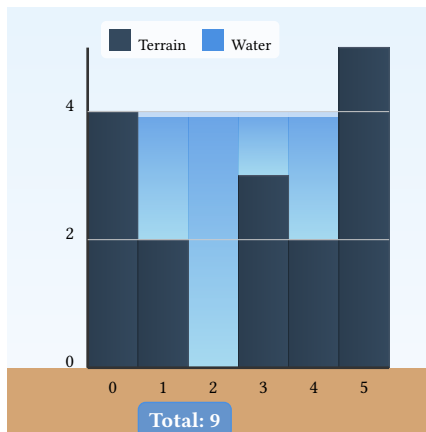
The above elevation map (black section) is represented by array [0, 1, 0, 2, 1, 0, 1, 3, 2, 1, 2, 1]. In this case, 6 units of rain water (blue section) are being trapped.

Expected
6

Your Output
Got none (check your code)

Case 2

height: [4, 2, 0, 3, 2, 5]



Expected

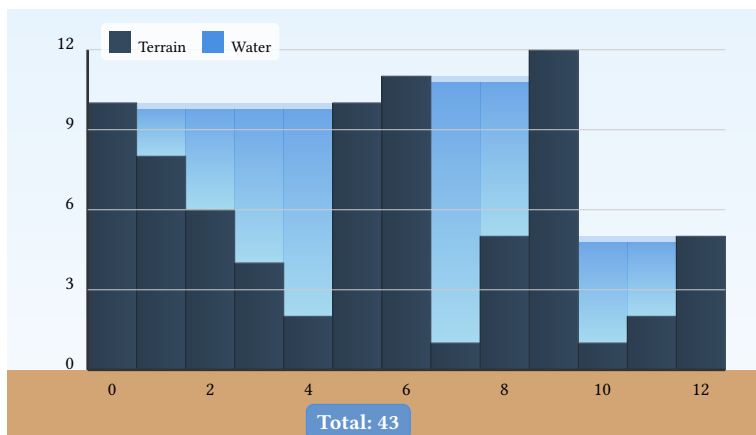
9

Your Output

Got none (check your code)

Case 3

height: [10, 8, 6, 4, 2, 10, 11, 1, 5, 12, 1, 2, 5]



Expected

43

Your Output

Got none (check your code)

0046. Permutations

MEDIUMARRAYBACKTRACKING

Given an array `nums` of distinct integers, return all the possible **permutations**. You can return the answer in **any order**.

Test Results (0 / 3)

Case 1

nums: [1, 2, 3]

Expected	Your Output
[[1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1], [3, 1, 2], [3, 2, 1]]	<i>Got none (check your code)</i>

Case 2

nums: [0, 1]

Expected	Your Output
[[0, 1], [1, 0]]	<i>Got none (check your code)</i>

Case 3

nums: [1]

Expected	Your Output
[[1]]	<i>Got none (check your code)</i>

0048. Rotate Image

MEDIUMARRAYMATRIXMATH

You are given an $n \times n$ 2D matrix representing an image, rotate the image by **90 degrees** (clockwise).

You have to rotate the image **in-place**, which means you have to modify the input 2D matrix directly.

Do not allocate another 2D matrix and do the rotation.

Note: In Typst, we return the rotated matrix instead of modifying in-place.

Test Results (0 / 3)

Case 1

matrix: [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

Expected	Your Output
[[7, 4, 1], [8, 5, 2], [9, 6, 3]]	<i>Got none (check your code)</i>

Case 2

matrix: [[5, 1, 9, 11], [2, 4, 8, 10], [13, 3, 6, 7], [15, 14, 12, 16]]

Expected	Your Output
[[15, 13, 2, 5], [14, 3, 4, 1], [12, 6, 8, 9], [16, 7, 10, 11]]	<i>Got none (check your code)</i>

Case 3

matrix: [[1]]

Expected	Your Output
[[1]]	<i>Got none (check your code)</i>

0049. Group Anagrams

MEDIUMARRAYHASH-TABLESTRINGSORTING

Given an array of strings `strs`, group the **anagrams** together. You can return the answer in **any order**.

An **anagram** is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.

Test Results (0 / 3)

Case 1

`strs: ["eat", "tea", "tan", "ate", "nat", "bat"]`

- There is no string in `strs` that can be rearranged to form "bat".
- The strings "nat" and "tan" are anagrams as they can be rearranged to form each other.
- The strings "ate", "eat", and "tea" are anagrams as they can be rearranged to form each other.

Expected

```
[["eat", "tea", "ate"], ["tan", "nat"], ["bat"]]
```

Your Output

Got none (check your code)

Case 2

`strs: []`

Expected

```
[[""]]
```

Your Output

Got none (check your code)

Case 3

`strs: ["a"]`

Expected

```
[["a"]]
```

Your Output

Got none (check your code)

0050. Pow(x, n)

MEDIUMMATHRECURSION

Implement [pow\(x, n\)](#), which calculates x raised to the power n (i.e., x^n).

- $-100.0 < x < 100.0$
- $-2^{31} \leq n \leq 2^{31} - 1$
- n is an integer.
- $-10^4 \leq x^n \leq 10^4$

Test Results (0 / 5)

Case 1

x: 2

n: 10

Expected
1024

Your Output
<i>Got none (check your code)</i>

Case 2

x: 2.1

n: 3

Expected
9.261000000000001

Your Output
<i>Got none (check your code)</i>

Case 3

x: 2

n: -2

$$2^{-2} = \frac{1}{2^2} = \frac{1}{4} = 0.25$$

Expected
0.25

Your Output
<i>Got none (check your code)</i>

Case 4

x: 0

n: 15

Expected
0

Your Output
<i>Got none (check your code)</i>

Case 5

x: 0.9

n: 199

Expected	Your Output
7.838976787394889e-10	<i>Got none (check your code)</i>

0051. N-Queens

HARDARRAYBACKTRACKING

The **n-queens** puzzle is the problem of placing n queens on an $n \times n$ chessboard such that no two queens attack each other.

Given an integer n , return *all distinct solutions* to the **n-queens puzzle**. You may return the answer in **any order**.

Each solution contains a distinct board configuration of the n -queens' placement, where 'Q' and '.' both indicate a queen and an empty space, respectively.

Test Results (0 / 3)

Case 1

n: 4

There exist two distinct solutions to the 4-queens puzzle as shown above.

Expected	Your Output																																
<table><tr><td>.</td><td>Q</td><td>.</td><td>.</td></tr><tr><td>.</td><td>.</td><td>.</td><td>Q</td></tr><tr><td>Q</td><td>.</td><td>.</td><td>.</td></tr><tr><td>.</td><td>.</td><td>Q</td><td>.</td></tr></table> <hr/> <table><tr><td>.</td><td>.</td><td>Q</td><td>.</td></tr><tr><td>Q</td><td>.</td><td>.</td><td>.</td></tr><tr><td>.</td><td>.</td><td>.</td><td>Q</td></tr><tr><td>.</td><td>Q</td><td>.</td><td>.</td></tr></table>	.	Q	Q	Q	Q	.	.	.	Q	.	Q	Q	.	Q	.	.	<p>Got none (check your code)</p>
.	Q	.	.																														
.	.	.	Q																														
Q	.	.	.																														
.	.	Q	.																														
.	.	Q	.																														
Q	.	.	.																														
.	.	.	Q																														
.	Q	.	.																														

Case 2

n: 1

Expected	Your Output	
<div><table><tr><td>Q</td></tr></table></div>	Q	<p>Got none (check your code)</p>
Q		

Case 3

n: 2

Expected	Your Output	
<div><table><tr><td></td></tr></table></div>		<p>Got none (check your code)</p>

0053. Maximum Subarray

MEDIUM

ARRAY

DIVIDE-AND-CONQUER

DYNAMIC-PROGRAMMING

Given an integer array `nums`, find the contiguous subarray (containing at least one number) which has the largest sum, and return its sum.

Test Results (0 / 7)

Case 1

nums: [-2, 1, -3, 4, -1, 2, 1, -5, 4]

The subarray [4, -1, 2, 1] has the largest sum 6.

Expected
6

Your Output
Got none (check your code)

Case 2

nums: [1]

The subarray [1] has the largest sum 1.

Expected
1

Your Output
Got none (check your code)

Case 3

nums: [5, 4, -1, 7, 8]

The subarray [5, 4, -1, 7, 8] has the largest sum 23.

Expected
23

Your Output
Got none (check your code)

Case 4

nums: [-1]

Expected
-1

Your Output
Got none (check your code)

Case 5

nums: [-2, -1]

Expected
-1

Your Output
<i>Got none (check your code)</i>

Case 6

nums: [1, 2, 3, 4, 5]

Expected
15

Your Output
<i>Got none (check your code)</i>

Case 7

nums: [-1, -2, -3, -4]

Expected
-1

Your Output
<i>Got none (check your code)</i>

0054. Spiral Matrix

MEDIUMARRAYMATRIXSIMULATION

Given an $m \times n$ matrix, return all elements of the matrix in **spiral order**.

Test Results (0 / 4)

Case 1

matrix: `[[1, 2, 3], [4, 5, 6], [7, 8, 9]]`

Expected
<code>[1, 2, 3, 6, 9, 8, 7, 4, 5]</code>

Your Output
<i>Got none (check your code)</i>

Case 2

matrix: `[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]`

Expected
<code>[1, 2, 3, 4, 8, 12, 11, 10, 9, 5, 6, 7]</code>

Your Output
<i>Got none (check your code)</i>

Case 3

matrix: `[[1]]`

Expected
<code>[1]</code>

Your Output
<i>Got none (check your code)</i>

Case 4

matrix: `[[1, 2], [3, 4]]`

Expected
<code>[1, 2, 4, 3]</code>

Your Output
<i>Got none (check your code)</i>

0055. Jump Game

MEDIUMARRAYGREEDYDYNAMIC-PROGRAMMING

You are given an integer array `nums`. You are initially positioned at the array's **first index**, and each element in the array represents your maximum jump length at that position.

Return `true` if you can reach the last index, or `false` otherwise.

Test Results (0 / 4)

Case 1

nums: [2, 3, 1, 1, 4]

Jump 1 step from index 0 to 1, then 3 steps to the last index.

Expected	Your Output
true	<i>Got none (check your code)</i>

Case 2

nums: [3, 2, 1, 0, 4]

You will always arrive at index 3 no matter what. Its maximum jump length is 0, which makes it impossible to reach the last index.

Expected	Your Output
false	<i>Got none (check your code)</i>

Case 3

nums: [0]

Expected	Your Output
true	<i>Got none (check your code)</i>

Case 4

nums: [2, 0, 0]

Expected	Your Output
true	<i>Got none (check your code)</i>

0056. Merge Intervals

MEDIUM

ARRAY

SORTING

Given an array of intervals where $\text{intervals}[i] = [\text{start}, \text{end}]$, merge all overlapping intervals, and return an array of the non-overlapping intervals that cover all the intervals in the input.

Test Results (0 / 4)

Case 1

intervals: `[[1, 3], [2, 6], [8, 10], [15, 18]]`

Since intervals `[1, 3]` and `[2, 6]` overlap, merge them into `[1, 6]`.

Expected	Your Output
<code>[[1, 6], [8, 10], [15, 18]]</code>	<i>Got none (check your code)</i>

Case 2

intervals: `[[1, 4], [4, 5]]`

Intervals `[1, 4]` and `[4, 5]` are considered overlapping.

Expected	Your Output
<code>[[1, 5]]</code>	<i>Got none (check your code)</i>

Case 3

intervals: `[[4, 7], [1, 4]]`

Intervals `[1, 4]` and `[4, 7]` are considered overlapping.

Expected	Your Output
<code>[[1, 7]]</code>	<i>Got none (check your code)</i>

Case 4

intervals: `[[1, 4], [2, 3]]`

Expected	Your Output
<code>[[1, 4]]</code>	<i>Got none (check your code)</i>

0062. Unique Paths

MEDIUMMATHDYNAMIC-PROGRAMMINGCOMBINATORICS

There is a robot on an $m \times n$ grid. The robot is initially located at the **top-left corner** (i.e., `grid[0][0]`). The robot tries to move to the **bottom-right corner** (i.e., `grid[m - 1][n - 1]`). The robot can only move either down or right at any point in time.

Given the two integers m and n , return the number of possible unique paths that the robot can take to reach the bottom-right corner.

Test Results (0 / 4)

Case 1

m: 3

n: 7

Expected	Your Output
28	<i>Got none (check your code)</i>

Case 2

m: 3

n: 2

From the top-left corner, there are a total of 3 ways to reach the bottom-right corner:

1. Right \rightarrow Down \rightarrow Down
2. Down \rightarrow Down \rightarrow Right
3. Down \rightarrow Right \rightarrow Down

Expected	Your Output
3	<i>Got none (check your code)</i>

Case 3

m: 1

n: 1

Expected	Your Output
1	<i>Got none (check your code)</i>

Case 4

m: 7

n: 3

Expected	Your Output
28	<i>Got none (check your code)</i>

0069. Sqrt(x)

EASYMATHBINARY-SEARCH

Given a non-negative integer x , return the square root of x rounded down to the nearest integer. The returned integer should be non-negative as well.

You **must not** use any built-in exponent function or operator.

Test Results (0 / 5)

Case 1

x: 4

The square root of 4 is 2, so we return 2.

Expected
2.0

Your Output

Got none (check your code)

Case 2

x: 8

The square root of 8 is 2.82842..., and since we round it down to the nearest integer, 2 is returned.

Expected
2.75

Your Output

Got none (check your code)

Case 3

x: 0

Expected
0

Your Output

Got none (check your code)

Case 4

x: 1

Expected
1

Your Output

Got none (check your code)

Case 5

x: 2147395599

Expected
46339.09997432213

Your Output

Got none (check your code)

0070. Climbing Stairs

EASYMATHDYNAMIC-PROGRAMMINGMEMOIZATION

You are climbing a staircase. It takes n steps to reach the top.

Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?

Test Results (0 / 8)

Case 1

n: 2

There are two ways to climb to the top:

1. 1 step + 1 step
2. 2 steps

Expected
2

Your Output
<i>Got none (check your code)</i>

Case 2

n: 3

There are three ways to climb to the top:

1. 1 step + 1 step + 1 step
2. 1 step + 2 steps
3. 2 steps + 1 step

Expected
3

Your Output
<i>Got none (check your code)</i>

Case 3

n: 1

Expected
1

Your Output
<i>Got none (check your code)</i>

Case 4

n: 4

Expected
5

Your Output
<i>Got none (check your code)</i>

Case 5

n: 5

Expected
8

Your Output
<i>Got none (check your code)</i>

Case 6

n: 10

Expected
89

Your Output
<i>Got none (check your code)</i>

Case 7

n: 20

Expected
10946

Your Output
<i>Got none (check your code)</i>

Case 8

n: 45

Expected
1836311903

Your Output
<i>Got none (check your code)</i>

0072. Edit Distance

MEDIUM

STRING

DYNAMIC-PROGRAMMING

Given two strings word1 and word2, return the minimum number of operations required to convert word1 to word2.

You have the following three operations permitted on a word:

- Insert a character
- Delete a character
- Replace a character

Test Results (0 / 8)

Case 1

word1: "horse"

word2: "ros"

| horse → rorse (replace 'h' with 'r') → rose (remove 'r') → ros (remove 'e')

Expected	Your Output
3	<i>Got none (check your code)</i>

Case 2

word1: "intention"

word2: "execution"

Expected	Your Output
5	<i>Got none (check your code)</i>

Case 3

word1: ""

word2: ""

Expected	Your Output
0	<i>Got none (check your code)</i>

Case 4

word1: "a"

word2: ""

Expected	Your Output
1	<i>Got none (check your code)</i>

Case 5

word1: ""

word2: "abc"

Expected
3

Your Output
<i>Got none (check your code)</i>

Case 6

word1: "abc"

word2: "abc"

Expected
0

Your Output
<i>Got none (check your code)</i>

Case 7

word1: "abc"

word2: "def"

Expected
3

Your Output
<i>Got none (check your code)</i>

Case 8

word1: "kitten"

word2: "sitting"

Expected
3

Your Output
<i>Got none (check your code)</i>

0076. Minimum Window Substring

HARDHASH-TABLESTRINGSLIDING-WINDOW

Given two strings s and t of lengths m and n respectively, return the **minimum window substring** of s such that every character in t (**including duplicates**) is included in the window. If there is no such substring, return the empty string $""$.

The testcases will be generated such that the answer is **unique**.

Follow up: Could you find an algorithm that runs in $\mathcal{O}(m + n)$ time?

Test Results (0 / 6)

Case 1

s: "ADOBECODEBANC"

t: "ABC"

The minimum window substring "BANC" includes 'A', 'B', and 'C' from string t.

Expected	Your Output
"BANC"	Got none (check your code)

Case 2

s: "a"

t: "a"

Expected	Your Output
"a"	Got none (check your code)

Case 3

s: "a"

t: "aa"

Expected	Your Output
""	Got none (check your code)

Case 4

s: "aa"

t: "aa"

Expected	Your Output
"aa"	Got none (check your code)

Case 5

s: "abc"

t: "b"

Expected	Your Output
"b"	<i>Got none (check your code)</i>

Case 6

s: "cabwefgewcwaefgcf"

t: "cae"

Expected	Your Output
"cwaef"	<i>Got none (check your code)</i>

0078. Subsets

MEDIUMARRAYBACKTRACKINGBIT-MANIPULATION

Given an integer array `nums` of **unique** elements, return all possible **subsets** (the power set).

The solution set **must not** contain duplicate subsets. Return the solution in **any order**.

Test Results (0 / 2)

Case 1

`nums`: [1, 2, 3]

Expected	Your Output
[[], [1], [2], [1, 2], [3], [1, 3], [2, 3], [1, 2, 3]]	<i>Got none (check your code)</i>

Case 2

`nums`: [0]

Expected	Your Output
[[], [0]]	<i>Got none (check your code)</i>

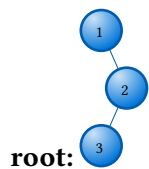
0094. Binary Tree Inorder Traversal

EASYTREESTACKBINARY-TREEDEPTH-FIRST-SEARCH

Given the root of a binary tree, return the inorder traversal of its nodes' values.

Test Results (0 / 4)

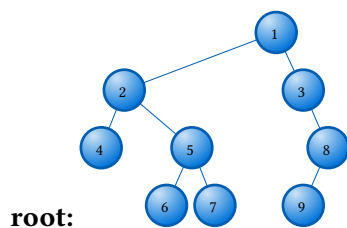
Case 1



Expected
[1, 3, 2]

Your Output
Got none (check your code)

Case 2



Expected
[4, 2, 6, 5, 7, 1, 3, 9, 8]

Your Output
Got none (check your code)

Case 3



Expected
[]

Your Output
Got none (check your code)

Case 4



Expected
[1]

Your Output
Got none (check your code)

0101. Symmetric Tree

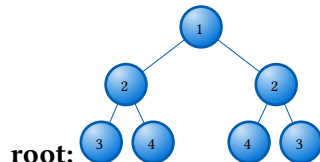
EASYTREEDEPTH-FIRST-SEARCHBREADTH-FIRST-SEARCHBINARY-TREE

Given the root of a binary tree, check whether it is a mirror of itself (i.e., symmetric around its center).

Follow up: Could you solve it both recursively and iteratively?

Test Results (0 / 6)

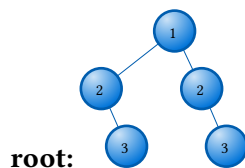
Case 1



Expected
true

Your Output
<i>Got none (check your code)</i>

Case 2



Expected
false

Your Output
<i>Got none (check your code)</i>

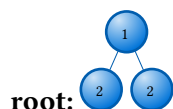
Case 3



Expected
true

Your Output
<i>Got none (check your code)</i>

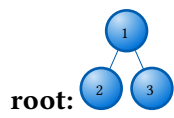
Case 4



Expected
true

Your Output
<i>Got none (check your code)</i>

Case 5



Expected
false

Your Output
<i>Got none (check your code)</i>

Case 6



Expected
true

Your Output
<i>Got none (check your code)</i>

0104. Maximum Depth of Binary Tree

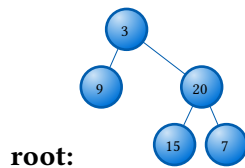
EASYTREEDEPTH-FIRST-SEARCHRECURSION

Given the root of a binary tree, return its **maximum depth**.

A binary tree's **maximum depth** is the number of nodes along the longest path from the root node down to the farthest leaf node.

Test Results (0 / 3)

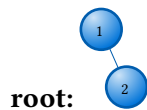
Case 1



Expected
3

Your Output
<i>Got none (check your code)</i>

Case 2



Expected
2

Your Output
<i>Got none (check your code)</i>

Case 3



Expected
1

Your Output
<i>Got none (check your code)</i>

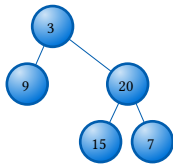
0110. Balanced Binary Tree

EASYTREEBINARY-TREEDEPTH-FIRST-SEARCH

Given a binary tree, determine if it is **height-balanced**¹.

Test Results (0 / 3)

Case 1



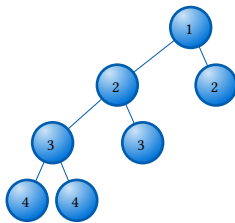
root:

Expected
true

Your Output

Got none (check your code)

Case 2



root:

Expected
false

Your Output

Got none (check your code)

Case 3

root:

Expected
true

Your Output

Got none (check your code)

¹A **height-balanced** binary tree is a binary tree in which the depth of the two subtrees of every node never differs by more than one.

0112. Path Sum

EASYTREEBINARY-TREEDEPTH-FIRST-SEARCHBREADTH-FIRST-SEARCH

Given the root of a binary tree and an integer target-sum, return true if the tree has a root-to-leaf path such that adding up all the values along the path equals target-sum.

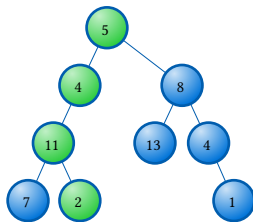
A **leaf** is a node with no children.

Test Results (0 / 3)

Case 1

target-sum: 22

root:



The root-to-leaf path with the target sum is shown.

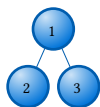
Expected
true

Your Output
Got none (check your code)

Case 2

target-sum: 5

root:



There are two root-to-leaf paths in the tree:

(1 → 2): The sum is 3.

(1 → 3): The sum is 4.

There is no root-to-leaf path with sum = 5.

Expected
false

Your Output
Got none (check your code)

Case 3

target-sum: 0

root:



Since the tree is empty, there are no root-to-leaf paths.

Expected	Your Output
false	<i>Got none (check your code)</i>

0113. Path Sum II

MEDIUM

TREE

BINARY-TREE

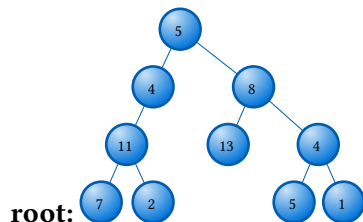
DEPTH-FIRST-SEARCH

BACKTRACKING

Given the root of a binary tree and an integer target-sum, return all root-to-leaf paths where the sum of the node values in the path equals target-sum. Each path should be returned as a list of the node values, not node references.

Test Results (0 / 3)

Case 1



root:

target-sum: 22

There are two paths whose sum equals targetSum:

$$5 + 4 + 11 + 2 = 22$$

$$5 + 8 + 4 + 5 = 22$$

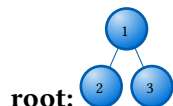
Expected

[[5, 4, 11, 2], [5, 8, 4, 5]]

Your Output

Got none (check your code)

Case 2



root:

target-sum: 5

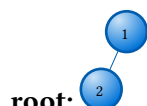
Expected

[]

Your Output

Got none (check your code)

Case 3



root:

target-sum: 0

Expected

[]

Your Output

Got none (check your code)

0116. Populating Next Right Pointers in Each Node

MEDIUM

TREE

DEPTH-FIRST-SEARCH

BREADTH-FIRST-SEARCH

BINARY-TREE

You are given a **perfect** binary tree where all leaves are on the same level, and every parent has two children.

Populate each next pointer to point to its next right node. If there is no next right node, the next pointer should be set to none.

Initially, all next pointers are set to none.

Constraints:

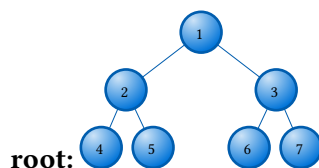
- The number of nodes in the tree is in the range $[0, 2^{12} - 1]$.
- $-1000 \leq \text{val} \leq 1000$

Follow-up:

- You may only use constant extra space.
- The recursive approach is fine. You may assume implicit stack space does not count as extra space for this problem.

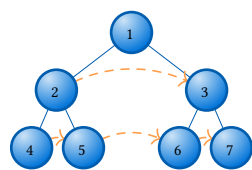
Test Results (0 / 4)

Case 1



Given the above perfect binary tree (Figure A), your function should populate each next pointer to point to its next right node, just like in Figure B. The serialized output is in level order as connected by the next pointers, with '#' signifying the end of each level.

Expected



Your Output

Got none (check your code)

Case 2

Diagram of a perfect binary tree with 1 node. The root is node 1. The root is labeled "root:".

Expected




Your Output

Got none (check your code)

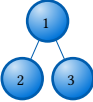
Case 3

root: 

Expected


Your Output
<i>Got none (check your code)</i>

Case 4

root: 

Expected


Your Output
<i>Got none (check your code)</i>

0121. Best Time to Buy and Sell Stock

EASYARRAYDYNAMIC-PROGRAMMING

You are given an array `prices` where `prices[i]` is the price of a given stock on the i th day.

You want to maximize your profit by choosing a single day to buy one stock and choosing a different day in the future to sell that stock.

Return the maximum profit you can achieve from this transaction. If you cannot achieve any profit, return 0.

Test Results (0 / 4)

Case 1

prices: [7, 1, 5, 3, 6, 4]

Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = $6 - 1 = 5$. Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell.

Expected	Your Output
5	Got none (check your code)

Case 2

prices: [7, 6, 4, 3, 1]

In this case, no transactions are done and the max profit = 0.

Expected	Your Output
0	Got none (check your code)

Case 3

prices: [1, 2]

Expected	Your Output
1	Got none (check your code)

Case 4

prices: [2, 1]

Expected	Your Output
0	Got none (check your code)

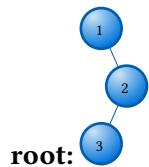
0144. Binary Tree Preorder Traversal

EASYTREESTACKBINARY-TREEDEPTH-FIRST-SEARCH

Given the root of a binary tree, return the preorder traversal of its nodes' values.

Test Results (0 / 4)

Case 1



root:

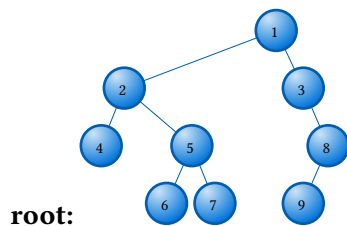
Expected

[1, 2, 3]

Your Output

Got none (check your code)

Case 2



root:

Expected

[1, 2, 4, 5, 6, 7, 3, 8, 9]

Your Output

Got none (check your code)

Case 3



root:

Expected

[]

Your Output

Got none (check your code)

Case 4



root:

Expected

[1]

Your Output

Got none (check your code)

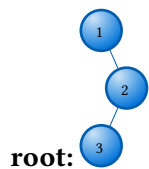
0145. Binary Tree Postorder Traversal

EASYTREESTACKBINARY-TREEDEPTH-FIRST-SEARCH

Given the root of a binary tree, return the postorder traversal of its nodes' values.

Test Results (0 / 4)

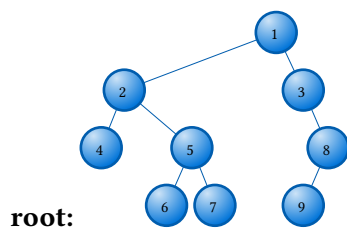
Case 1



Expected
[3, 2, 1]

Your Output
Got none (check your code)

Case 2



Expected
[4, 6, 7, 5, 2, 9, 8, 3, 1]

Your Output
Got none (check your code)

Case 3



Expected
[]

Your Output
Got none (check your code)

Case 4



Expected
[1]

Your Output
Got none (check your code)

0155. Min Stack

MEDIUMSTACKDESIGN

Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.

Implement the MinStack class:

- `MinStack()` initializes the stack object.
- `push(val)` pushes the element `val` onto the stack.
- `pop()` removes the element on the top of the stack.
- `top()` gets the top element of the stack.
- `getMin()` retrieves the minimum element in the stack.

You **must** implement a solution with $\mathcal{O}(1)$ time complexity for each function.

Test Results (0 / 1)

Case 1

operations: ["MinStack", "push", "push", "push", "getMin", "pop", "top", "getMin"]

args: [[], [-2], [0], [-3], [], [], [], []]

```
MinStack minStack = new MinStack();
minStack.push(-2);
minStack.push(0);
minStack.push(-3);
minStack.getMin(); minStack.pop();
minStack.top(); minStack.getMin();
```

Expected
[none, none, none, none, -3, none, 0, -2]

Your Output
Got none (check your code)

0200. Number of Islands

MEDIUMARRAYDEPTH-FIRST-SEARCHBREADTH-FIRST-SEARCHUNION-FINDMATRIX

Given an $m \times n$ 2-D binary grid `grid` which represents a map of 1 (land) and 0 (water), return the number of islands.

An island is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may assume all four edges of the grid are all surrounded by water.

Test Results (0 / 6)

Case 1

grid: `[[1, 1, 1, 1, 0], [1, 1, 0, 1, 0], [1, 1, 0, 0, 0], [0, 0, 0, 0, 0]]`

Expected	Your Output
1	<i>Got none (check your code)</i>

Case 2

grid: `[[1, 1, 0, 0, 0], [1, 1, 0, 0, 0], [0, 0, 1, 0, 0], [0, 0, 0, 1, 1]]`

Expected	Your Output
3	<i>Got none (check your code)</i>

Case 3

grid: `[[0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0]]`

Expected	Your Output
0	<i>Got none (check your code)</i>

Case 4

grid: `[[1, 0, 0, 0, 0], [0, 1, 0, 0, 0], [0, 0, 1, 0, 0], [0, 0, 0, 1, 1]]`

Expected	Your Output
4	<i>Got none (check your code)</i>

Case 5

grid: `[[1, 1, 1, 1, 1], [1, 0, 0, 0, 1], [1, 0, 1, 0, 1], [1, 0, 0, 0, 1], [1, 1, 1, 1, 1]]`

Expected	Your Output
2	<i>Got none (check your code)</i>

Case 6

grid: [[1, 1, 1, 1, 1], [1, 0, 1, 0, 1], [1, 0, 1, 0, 1], [1, 0, 0, 0, 1], [1, 1, 1, 1, 1]]

Expected
1

Your Output
<i>Got none (check your code)</i>

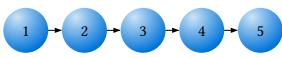
0206. Reverse Linked List

EASYLINKED-LISTRECURSION

Given the head of a singly linked list, reverse the list, and return the reversed list.

Test Results (0 / 3)

Case 1

head: 

Expected



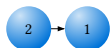
Your Output

Got none (check your code)

Case 2

head: 

Expected



Your Output

Got none (check your code)

Case 3

head: 

Expected



Your Output

Got none (check your code)

0207. Course Schedule

MEDIUM

GRAPH

DEPTH-FIRST-SEARCH

BREADTH-FIRST-SEARCH

TOPOLOGICAL-SORT

There are a total of `numCourses` courses you have to take, labeled from `0` to `numCourses - 1`. You are given an array `prerequisites` where `prerequisites[i] = [a, b]` indicates that you **must** take course `b` first if you want to take course `a`.

For example, the pair `[0, 1]`, indicates that to take course `0` you have to first take course `1`.

Return `true` if you can finish all courses. Otherwise, return `false`.

Test Results (0 / 4)

Case 1

Input: 2 courses, 1 prerequisites



There are a total of 2 courses to take. To take course 1 you should have finished course 0. So it is possible.

Expected

true

Your Output

Got none (check your code)

Case 2

Input: 2 courses, 2 prerequisites



There are a total of 2 courses to take. To take course 1 you should have finished course 0, and to take course 0 you should also have finished course 1. So it is impossible.

Expected

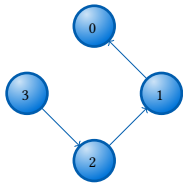
false

Your Output

Got none (check your code)

Case 3

Input: 4 courses, 3 prerequisites



Expected
true

Your Output
<i>Got none (check your code)</i>

Case 4

Input: 1 courses, 0 prerequisites



Expected
true

Your Output
<i>Got none (check your code)</i>

0218. The Skyline Problem

HARDARRAYDIVIDE-AND-CONQUERBINARY-INDEXED-TREESEGMENT-TREELINE-SWEEPSORTINGHEAPORDERED-SET

A city's **skyline** is the outer contour of the silhouette formed by all the buildings in that city when viewed from a distance.

Given the locations and heights of all the buildings, return the **skyline** formed by these buildings collectively.

The geometric information of each building is given in the array `buildings` where `buildings[i] = [left_i, right_i, height_i]`:

- `left_i` is the x coordinate of the left edge of the i^{th} building.
- `right_i` is the x coordinate of the right edge of the i^{th} building.
- `height_i` is the height of the i^{th} building.

You may assume all buildings are perfect rectangles grounded on an absolutely flat surface at height 0.

The **skyline** should be represented as a list of "key points" sorted by their x-coordinate in the form `[[x_1, y_1], [x_2, y_2], ...]`. Each key point is the left endpoint of some horizontal segment in the skyline except the last point, which always has a y-coordinate 0 and is used to mark the skyline's termination.

Note: There must be no consecutive horizontal lines of equal height in the output skyline.

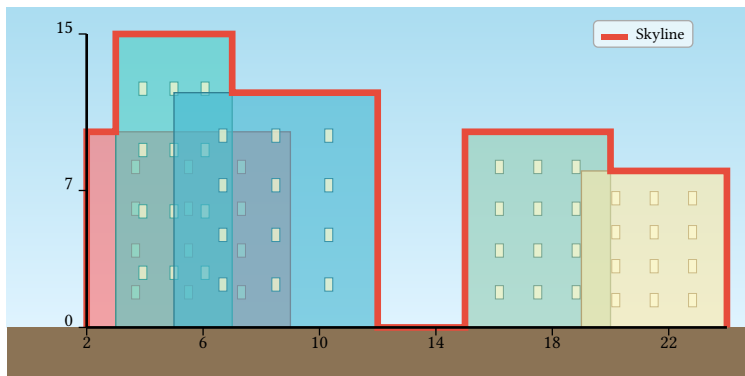
Constraints:

- $1 \leq \text{buildings.length} \leq 10^4$
- $0 \leq \text{left}_i < \text{right}_i \leq 2^{31} - 1$
- $1 \leq \text{height}_i \leq 2^{31} - 1$
- `buildings` is sorted by `left_i` in non-decreasing order.

Test Results (0 / 5)

Case 1

buildings: `[[2, 9, 10], [3, 7, 15], [5, 12, 12], [15, 20, 10], [19, 24, 8]]`



The colored rectangles show the buildings. The red line traces the skyline contour, with key points marking height changes.

Expected

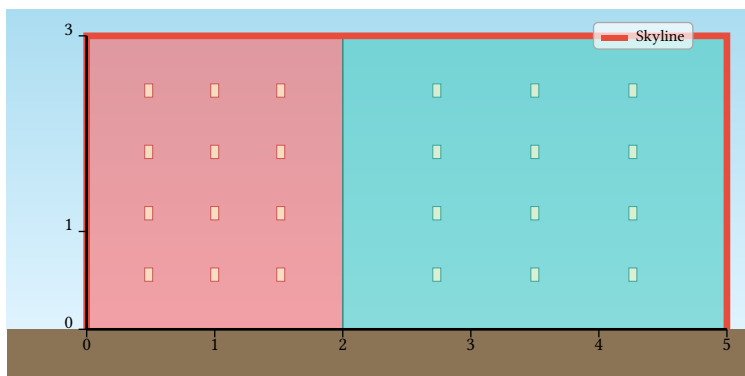
```
[[2, 10], [3, 15], [7, 12], [12, 0], [15, 10], [20, 8],  
[24, 0]]
```

Your Output

Got none (check your code)

Case 2

buildings: `[[0, 2, 3], [2, 5, 3]]`



Expected

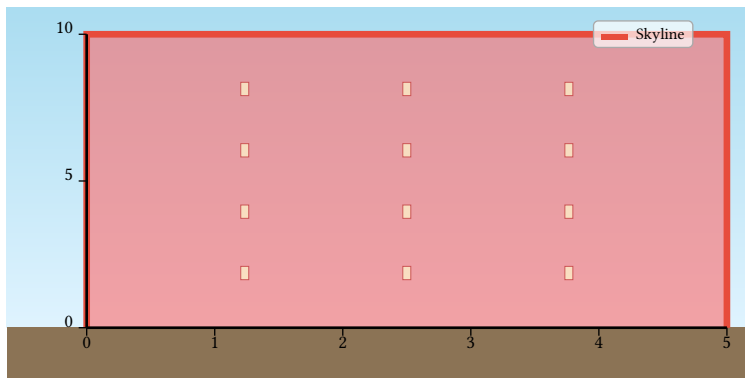
```
[[0, 3], [5, 0]]
```

Your Output

Got none (check your code)

Case 3

buildings: `[[0, 5, 10]]`



Expected

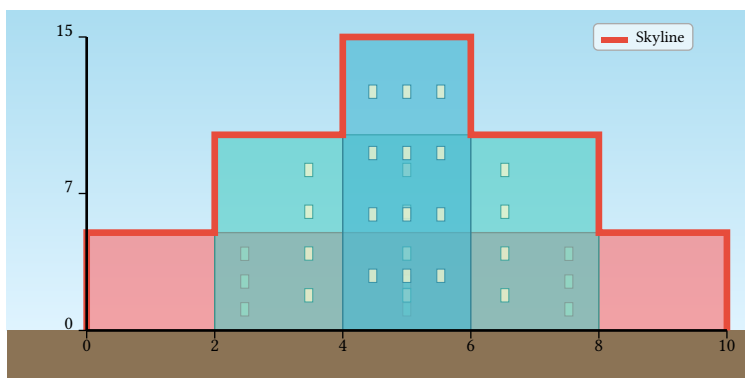
`[[0, 10], [5, 0]]`

Your Output

Got none (check your code)

Case 4

buildings: `[[0, 10, 5], [2, 8, 10], [4, 6, 15]]`



Expected

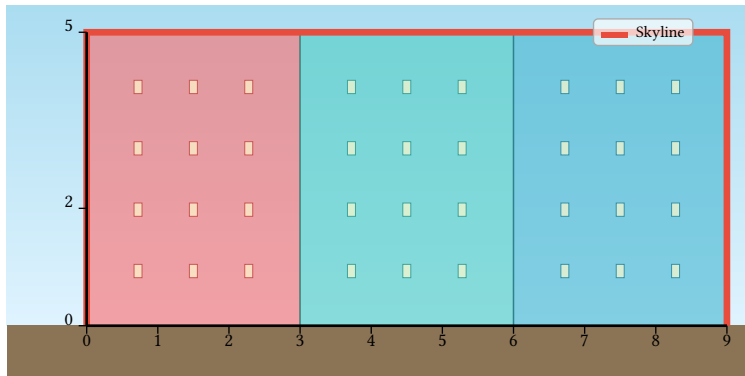
`[[0, 5], [2, 10], [4, 15], [6, 10], [8, 5], [10, 0]]`

Your Output

Got none (check your code)

Case 5

buildings: $[[0, 3, 5], [3, 6, 5], [6, 9, 5]]$



Expected

$[[0, 5], [9, 0]]$

Your Output

Got none (check your code)

0209. Minimum Size Subarray Sum

MEDIUMARRAYSLIDING-WINDOWBINARY-SEARCHPREFIX-SUM

Given an array of positive integers `nums` and a positive integer `target`, return the **minimal length** of a subarray whose sum is greater than or equal to `target`. If there is no such subarray, return 0 instead.

Test Results (0 / 4)

Case 1

target: 7

nums: [2, 3, 1, 2, 4, 3]

The subarray [4, 3] has the minimal length under the problem constraint.

Expected	Your Output
2	<i>Got none (check your code)</i>

Case 2

target: 4

nums: [1, 4, 4]

Expected	Your Output
1	<i>Got none (check your code)</i>

Case 3

target: 11

nums: [1, 1, 1, 1, 1, 1, 1, 1]

Expected	Your Output
0	<i>Got none (check your code)</i>

Case 4

target: 15

nums: [1, 2, 3, 4, 5]

Expected	Your Output
5	<i>Got none (check your code)</i>

0210. Course Schedule II

MEDIUM

GRAPH

TOPOLOGICAL-SORT

DEPTH-FIRST-SEARCH

There are a total of `numCourses` courses you have to take, labeled from 0 to `numCourses - 1`. You are given an array `prerequisites` where `prerequisites[i] = [a, b]` indicates that you **must** take course `b` first if you want to take course `a`.

For example, the pair `[0, 1]`, indicates that to take course 0 you have to first take course 1.

Return the ordering of courses you should take to finish all courses. If there are many valid answers, return **any** of them. If it is impossible to finish all courses, return an empty array.

Constraints:

- $1 \leq \text{numCourses} \leq 2000$
- $0 \leq \text{prerequisites.length} \leq \text{numCourses} \times (\text{numCourses} - 1)$
- All the pairs `[a, b]` are **distinct**.

Test Results (0 / 4)

Case 1

Input: 2 courses, 1 prerequisites



There are a total of 2 courses to take. To take course 1 you should have finished course 0. So the correct course order is `[0, 1]`.

Expected

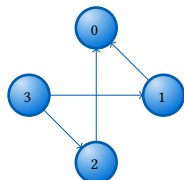
`[0, 1]`

Your Output

Got none (check your code)

Case 2

Input: 4 courses, 4 prerequisites



There are a total of 4 courses to take. To take course 3 you should have finished both courses 1 and 2. Both courses 1 and 2 should be taken after you finished course 0. So one correct course order is `[0, 1, 2, 3]`. Another correct ordering is `[0, 2, 1, 3]`.

Expected

`[0, 1, 2, 3]`

Your Output

Got none (check your code)

Case 3

Input: 1 courses, 0 prerequisites



Expected
[0]

Your Output
<i>Got none (check your code)</i>

Case 4

Input: 2 courses, 2 prerequisites



Expected
[]

Your Output
<i>Got none (check your code)</i>

0289. Game of Life

MEDIUMARRAYMATRIXSIMULATION

According to [Wikipedia](#): "The Game of Life, also known simply as Life, is a cellular automaton devised by the British mathematician John Horton Conway in 1970."

The board is made up of an $m \times n$ grid of cells, where each cell has an initial state: live (represented by a 1) or dead (represented by a 0). Each cell interacts with its eight neighbors (horizontal, vertical, diagonal) using the following four rules (taken from the above Wikipedia article):

1. Any live cell with fewer than two live neighbors dies as if caused by under-population.
2. Any live cell with two or three live neighbors lives on to the next generation.
3. Any live cell with more than three live neighbors dies, as if by over-population.
4. Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.

The next state of the board is determined by applying the above rules simultaneously to every cell in the current state of the $m \times n$ grid board. In this process, births and deaths occur simultaneously.

Given the current state of the board, return the board after applying the above rules.

Test Results (0 / 2)

Case 1

Board: 4 x 3

0	1	0
0	0	1
1	1	1
0	0	0

Expected

0	0	0
1	0	1
0	1	1
0	1	0

Your Output

Got none (check your code)

Case 2

Board: 2 x 2

1	1
1	0

Expected

1	1
1	1

Your Output

Got none (check your code)

0347. Top K Frequent Elements

MEDIUMARRAYHASH-TABLESORTINGHEAP

Given an integer array `nums` and an integer `k`, return the k most frequent elements. You may return the answer in **any order**.

Test Results (0 / 3)

Case 1

nums: [1, 1, 1, 2, 2, 3]

k: 2

Expected	Your Output
[1, 2]	<i>Got none (check your code)</i>

Case 2

nums: [1]

k: 1

Expected	Your Output
[1]	<i>Got none (check your code)</i>

Case 3

nums: [1, 2, 1, 2, 1, 2, 3, 1, 3, 2]

k: 2

Expected	Your Output
[1, 2]	<i>Got none (check your code)</i>

0407. Trapping Rain Water II

HARDARRAYBFSHEAPMATRIX

Given an $m \times n$ integer matrix `heightMap` representing the height of each unit cell in a 2D elevation map, return the volume of water it can trap after raining.

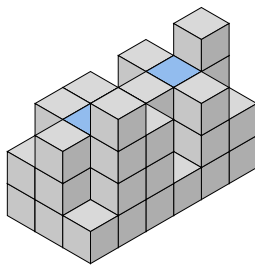
Constraints:

- $m == \text{heightMap.length}$
- $n == \text{heightMap}[i].\text{length}$
- $1 \leq m, n \leq 200$
- $0 \leq \text{heightMap}[i][j] \leq 2 \times 10^4$

Test Results (0 / 5)

Case 1

heightMap: `[[1, 4, 3, 1, 3, 2], [3, 2, 1, 3, 2, 4], [2, 3, 3, 2, 3, 1]]`



After the rain, water is trapped between the blocks. We have two small ponds 1 and 3 units trapped. The total volume of water trapped is 4.

Expected

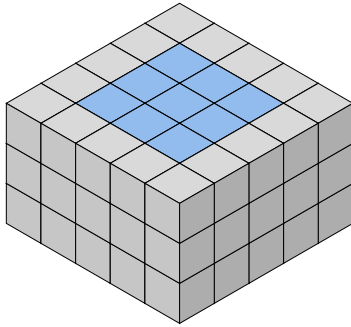
4

Your Output

Got none (check your code)

Case 2

heightMap: `[[3, 3, 3, 3, 3], [3, 2, 2, 2, 3], [3, 2, 1, 2, 3], [3, 2, 2, 2, 3], [3, 3, 3, 3, 3]]`

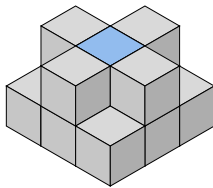


Expected
10

Your Output
<i>Got none (check your code)</i>

Case 3

heightMap: `[[1, 2, 1], [2, 0, 2], [1, 2, 1]]`

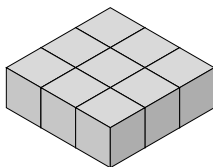


Expected
2

Your Output
<i>Got none (check your code)</i>

Case 4

heightMap: `[[1, 1, 1], [1, 1, 1], [1, 1, 1]]`

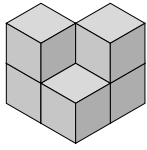


Expected
0

Your Output
<i>Got none (check your code)</i>

Case 5

heightMap: $[[1, 2], [2, 1]]$



Expected	Your Output
0	<i>Got none (check your code)</i>

0547. Number of Provinces

MEDIUM

DEPTH-FIRST-SEARCH

BREADTH-FIRST-SEARCH

UNION-FIND

GRAPH

There are n cities. Some of them are connected, while some are not. If city a is connected directly with city b , and city b is connected directly with city c , then city a is connected indirectly with city c .

A **province** is a group of directly or indirectly connected cities and no other cities outside of the group.

You are given an $n \times n$ matrix `isConnected` where `isConnected[i][j] = 1` if the i -th city and the j -th city are directly connected, and `isConnected[i][j] = 0` otherwise.

Return the total number of **provinces**.

Test Results (0 / 5)

Case 1

Adjacency Matrix: 3×3

1	1	0
1	1	0
0	0	1

Expected

2

Your Output

Got none (check your code)

Case 2

Adjacency Matrix: 3×3

1	0	0
0	1	0
0	0	1

Expected

3

Your Output

Got none (check your code)

Case 3

Adjacency Matrix: 1×1

1

Expected

1

Your Output

Got none (check your code)

Case 4

Adjacency Matrix: 2×2

1	1
1	1

Expected
1

Your Output
<i>Got none (check your code)</i>

Case 5

Adjacency Matrix: 4×4

1	0	0	1
0	1	1	0
0	1	1	1
1	0	1	1

Expected
1

Your Output
<i>Got none (check your code)</i>

0785. Is Graph Bipartite?

MEDIUM

GRAPH

DEPTH-FIRST-SEARCH

BREADTH-FIRST-SEARCH

There is an **undirected** graph with n nodes, where each node is numbered between 0 and $n - 1$. You are given a 2D array `graph`, where `graph[u]` is an array of nodes that node `u` is adjacent to. More formally, for each `v` in `graph[u]`, there is an undirected edge between node `u` and node `v`.

The graph has the following properties:

- There are no self-edges (`graph[u]` does not contain `u`).
- There are no parallel edges (`graph[u]` does not contain duplicate values).
- If `v` is in `graph[u]`, then `u` is in `graph[v]` (the graph is undirected).
- The graph may not be connected, meaning there may be two nodes `u` and `v` such that there is no path between them.

A graph is **bipartite** if the nodes can be partitioned into two independent sets `A` and `B` such that every edge in the graph connects a node in set `A` and a node in set `B`.

Return `true` if and only if it is bipartite.

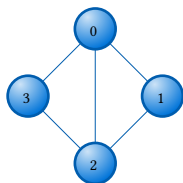
Constraints:

- $1 \leq n \leq 100$
- $0 \leq \text{graph}[u].\text{length} < n$

Test Results (0 / 4)

Case 1

Input: 4 nodes (adjacency list)



There is no way to partition the nodes into two independent sets such that every edge connects a node in one and a node in the other.

Expected

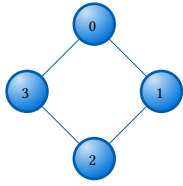
false

Your Output

Got none (check your code)

Case 2

Input: 4 nodes (adjacency list)



We can partition the nodes into two sets: {0, 2} and {1, 3}.

Expected
true

Your Output
<i>Got none (check your code)</i>

Case 3

Input: 2 nodes (adjacency list)



Expected
true

Your Output
<i>Got none (check your code)</i>

Case 4

Input: 2 nodes (adjacency list)



Expected
true

Your Output
<i>Got none (check your code)</i>

0814. Binary Tree Pruning

MEDIUM

TREE

BINARY-TREE

DEPTH-FIRST-SEARCH

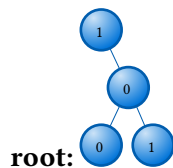
Given the root of a binary tree, return the same tree where every subtree (of the given tree) not containing a 1 has been removed.

A subtree of a node `node` is `node` plus every node that is a descendant of `node`.

- `node.val` is either 0 or 1.

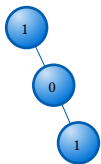
Test Results (0 / 3)

Case 1



Only the red nodes satisfy the property "every subtree not containing a 1". The diagram on the right represents the answer.

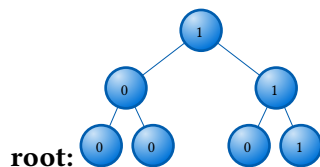
Expected



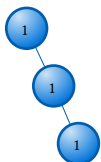
Your Output

Got none (check your code)

Case 2



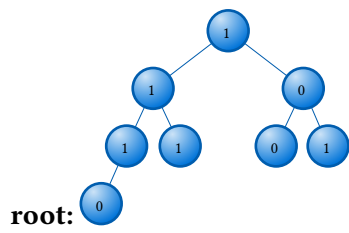
Expected



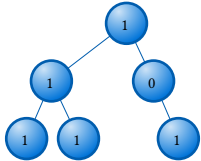
Your Output

Got none (check your code)

Case 3



Expected



Your Output

Got none (check your code)

0997. Find the Town Judge

EASYGRAPHHASH-TABLE

In a town, there are n people labeled from 1 to n . There is a rumor that one of these people is secretly the town judge.

If the town judge exists, then:

1. The town judge trusts nobody.
2. Everybody (except for the town judge) trusts the town judge.
3. There is exactly one person that satisfies properties 1 and 2.

You are given an array `trust` where `trust[i] = [a, b]` representing that the person labeled `a` trusts the person labeled `b`. If a trust relationship does not exist in `trust` array, then such a trust relationship does not exist.

Return the label of the town judge if the town judge exists and can be identified, or return `-1` otherwise.

Constraints:

- $1 \leq n \leq 1000$
- $0 \leq \text{trust.length} \leq 10^4$
- All the pairs of `trust` are **unique**.
- `trust[i]` are all different.
- `trust[i][0] != trust[i][1]`

Test Results (0 / 5)

Case 1

Input: 2 people, 1 trust relationships

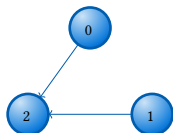


Expected
2

Your Output
Got none (check your code)

Case 2

Input: 3 people, 2 trust relationships

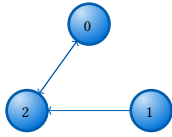


Expected
3

Your Output
Got none (check your code)

Case 3

Input: 3 people, 3 trust relationships



Expected
-1

Your Output
<i>Got none (check your code)</i>

Case 4

Input: 1 people, 0 trust relationships

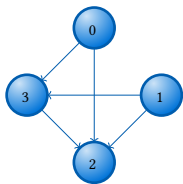


Expected
1

Your Output
<i>Got none (check your code)</i>

Case 5

Input: 4 people, 5 trust relationships



Expected
3

Your Output
<i>Got none (check your code)</i>