DATA TYPES:

Write a Python program to find the union of two sets.

Write a Python program to sort a list in descending order.

Write a Python program to remove all occurrences of a specified element from a list.

Write a Python program to find the intersection of two sets.

Write a Python program to find the length of a string.

Write a Python program to concatenate two strings.

24. Write a Python program to convert a tuple to a list.

23. Write a Python program to convert a list to a tuple.

21. Write a Python program to convert a string to an integer.

22. Write a Python program to convert a string to a float.

FUNCTIONS:

20. Write a Python program to print the multiplication table of a given number.

19. Write a Python program to find the factorial of a number using a loop.

17. Write a Python program to find the sum of digits in a number.

18. Write a Python program to check if a number is a palindrome or not.

16. Write a Python program to find the sum of the first N natural numbers.

14. Write a Python program to find the LCM of two numbers.

15. Write a Python program to find the square root of a number using the Newton-Raphson method.

12. Write a Python program to find the second largest element in a list.

13. Write a Python program to find the greatest common divisor (GCD) of two numbers.

10. Write a Python function to remove duplicate elements from a list.

OPERATORS IN PYTHON:

50. Write a Python program to remove duplicate elements from a list.

47. Write a Python program to find the GCD of two numbers.

48. Write a Python program to reverse a string.

49. Write a Python program to sort a list in ascending order.

46. Write a Python program to find the LCM of two numbers.

44. Write a Python program to check if a string is a palindrome or not.

45. Write a Python program to find the factorial of a number.

42. Write a Python program to calculate the area of a triangle.

43. Write a Python program to calculate the circumference of a circle.

40. Write a Python program to find the maximum of two numbers using the ternary operator.

CONDITIONAL STATEMENTS:

70. Write a Python program to check if a given number is a palindrome or not using recursion

68. Write a Python program to check if a given string is a valid password.

Questions
69. Write a Python program to find the area of a circle, given the radius.

Questions
66. Write a Python program to find the roots of a quadratic equation.

67. Write a Python program to find the greatest common divisor (GCD) of three numbers.

Questions
63. Write a Python program to find the average of a list of numbers.

64. Write a Python program to check if a given number is a perfect square or not.

65. Write a Python program to find the largest prime number less than a given number.

Questions
60. Write a Python program to check if a given string is a valid email address.

Questions
61. Write a Python program to find the sum of all even numbers in a list.

Questions
62. Write a Python program to find the sum of all odd numbers in a list.

58. Write a Python program to check if a given number is divisible by 3 or 5.

59. Write a Python program to check if a given character is a vowel or consonant.

55. Write a Python program to check if a given string is a palindrome or not.

56. Write a Python program to find the smallest of three numbers.

57. Write a Python program to check if a given number is prime or not.

52. Write a Python program to check if a number is even or odd.

53. Write a Python program to check if a year is a leap year or not.

Questions
54. Write a Python program to find the largest of three numbers.

51. Write a Python program to check if a number is positive, negative or zero.

LOOPING STATEMENTS:

89. Write a Python program to find the LCM of two numbers using a while loop.

90. Write a Python program to generate a Pascal's triangle using nested loops.

87. Write a Python program to check if a given string is a palindrome or not using a for loop.

88. Write a Python program to find the first n Fibonacci numbers using a for loop.

86. Write a Python program to print the sum of digits of a given number using a while loop.

84. Write a Python program to check if a given number is prime or not using a for loop.

85. Write a Python program to find the sum of all odd numbers between two given numbers using a while loop.

82. Write a Python program to find the GCD of two numbers using a for loop.

83. Write a Python program to generate a multiplication table using nested loops.

81. Write a Python program to find the factorial of a number using a for loop.

79. Write a Python program to find the sum of all even numbers between two given numbers using a while loop.

80. Write a Python program to count the number of vowels in a given string using a for loop.

77. Write a Python program to check if a given number is a palindrome or not using a while loop.

78. Write a Python program to reverse a string using a for loop.

76. Write a Python program to print all the prime numbers within a given range using a for loop.

74. Write a Python program to print the Fibonacci sequence up to a given number using a while loop.

75. Write a Python program to find the factorial of a number using a while loop.

72. Write a Python program to print the multiplication table of a given number using a for loop.

73. Write a Python program to find the sum of all the numbers in a list using a for loop.

71. Write a Python program to print the numbers from 1 to 10 using a for loop.

TYPES OF FUNCTIONS:

100. Write a Python program to concatenate two lists using a generator function.

99. Write a Python program to find the length of the longest word in a given sentence using a reduce function.

Questions
98. Write a Python program to check if a given number is a prime number using a filter function.

Questions
97. Write a Python program to count the number of vowels in a given string using a map function.

Questions
96. Write a Python program to find the LCM of two numbers using a built-in function.

Questions
95. Write a Python program to find the sum of two numbers using a recursive function.

93. Write a Python program to check if a given string is a valid password using a lambda function.

94. Write a Python program to sort a list of numbers in ascending order using a built-in function.

91. Write a Python program to find the area of a circle using a user-defined function.

Questions
92. Write a Python program to check if a given number is a perfect number using a user-defined function.

STRINGS:

119. Write a Python program to find the frequency of all characters in a given string.

120. Write a Python program to find the number of words in a given string.

118. Write a Python program to find the largest palindrome substring in a given string.

117. Write a Python program to check if a given string is a valid email address.

116. Write a Python program to find the index of a given substring in a given string.

115. Write a Python program to remove all whitespace characters from a given string.

114. Write a Python program to find the common characters between two given strings.

113. Write a Python program to find the second most frequent character in a given string.

112. Write a Python program to check if a given string is an anagram of another string.

111. Write a Python program to find the first non-repeating character in a given string.

110. Write a Python program to replace all occurrences of a given word in a string with another word.

109. Write a Python program to find the most frequent word in a given sentence.

108. Write a Python program to check if a given string contains only digits.

107. Write a Python program to remove all the vowels from a given string.

106. Write a Python program to concatenate two strings without using the '+' operator.

105. Write a Python program to find the length of the longest word in a given sentence.

104. Write a Python program to capitalize the first letter of each word in a given string.

103. Write a Python program to check if a given string is a palindrome or not.

102. Write a Python program to reverse a string using slicing.

Questions
101. Write a Python program to count the number of occurrences of a character in a given string.

LISTS:

140. Write a Python program to find the kth smallest element in a given list.

139. Write a Python program to find the difference between two lists.

138. Write a Python program to find the intersection of two lists.

137. Write a Python program to find the union of two lists.

136. Write a Python program to find the common elements between two lists.

135. Write a Python program to find the first n Fibonacci numbers and store them in a list.

134. Write a Python program to find the largest subsequence sum in a given list of integers.

133. Write a Python program to merge two sorted lists into a single sorted list.

132. Write a Python program to remove all even numbers from a given list.

131. Write a Python program to find the maximum and minimum elements in a given list.

129. Write a Python program to shuffle a given list.

130. Write a Python program to concatenate two lists.

Questions
128. Write a Python program to sort a given list in ascending order.

127. Write a Python program to find the frequency of all elements in a given list.

126. Write a Python program to find the index of a given element in a given list.

125. Write a Python program to reverse a given list.

124. Write a Python program to remove duplicates from a given list.

123. Write a Python program to find the smallest number in a given list.

Questions
122. Write a Python program to find the second largest number in a given list.

121. Write a Python program to find the sum of all elements in a given list.

TUPLES AND DICTIONARY:

160. Write a Python program to convert a dictionary to a list of tuples.

159. Write a Python program to sort a given dictionary by its values.

Questions
158. Write a Python program to find the frequency of all elements in a given dictionary.

Questions
157. Write a Python program to find the keys with the maximum and minimum values in a given dictionary.

156. Write a Python program to find the maximum and minimum values in a given dictionary.

155. Write a Python program to remove a given key from a given dictionary.

154. Write a Python program to check if a given value exists in a given dictionary.

152. Write a Python program to concatenate two dictionaries.

153. Write a Python program to check if a given key exists in a given dictionary.

151. Write a Python program to find the length of a given dictionary.

149. Write a Python program to sort a given tuple in ascending order.

150. Write a Python program to find the sum of all elements in a given tuple.

148. Write a Python program to convert a list of tuples to a list of lists.

147. Write a Python program to check if a given element exists in a given tuple.

Questions
145. Write a Python program to convert a tuple to a list.

Questions
146. Write a Python program to find the index of a given element in a given tuple.

144. Write a Python program to concatenate two tuples.

143. Write a Python program to find the maximum and minimum elements in a given tuple.

142. Write a Python program to reverse a given tuple.

141. Write a Python program to find the length of a given tuple.

FILES:

166. Write a Python program to write a list of strings to a file.

165. Write a Python program to copy the contents of one file to another.

Questions
164. Write a Python program to count the number of characters in a file.

162. Write a Python program to count the number of lines in a file.

163. Write a Python program to count the number of words in a file.

161. Write a Python program to read a file and display its contents on the screen.

EXECPTIONAL HANDLING:

179. Write a Python program to raise a custom exception.

178. Write a Python program to handle multiple exceptions using a single except block.

176. Write a Python program to handle a KeyError exception.

Questions
177. Write a Python program to handle a NameError exception.

175. Write a Python program to handle a FileNotFoundError exception.

174. Write a Python program to handle a ValueError exception.

173. Write a Python program to handle a TypeError exception.

172. Write a Python program to handle an IndexError exception.

Questions
171. Write a Python program to handle a ZeroDivisionError exception.

EASY:

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.



Example 1:

Input: nums = [2,7,11,15], target = 9
Output: [0,1]
Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].
Example 2:

Input: nums = [3,2,4], target = 6
Output: [1,2]
Example 3:

Input: nums = [3,3], target = 6
Output: [0,1]
```

Palindrome Number

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



Example 1:

Input: x = 121
Output: true
Explanation: 121 reads as 121 from left to right and from right to left.
Example 2:

Input: x = -121
Output: false
Explanation: From left to right, it reads -121. From right to left, it
becomes 121-. Therefore it is not a palindrome.
```

Example 3:

Input: x = 10
Output: false
Explanation: Reads 01 from right to left. Therefore it is not a palindrome.

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 ones 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.

Example 1:

Input: s = "III"
Output: 3
Explanation: III = 3.
Example 2:

Input: s = "LVIII"
Output: 58
Explanation: L = 50, V= 5, III = 3.
Example 3:

Input: s = "MCMXCIV"
Output: 1994
Explanation: M = 1000, CM = 900, XC = 90 and IV = 4.

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 "".



Example 1:

Input: strs = ["flower","flow","flight"]
Output: "fl"
Example 2:

Input: strs = ["dog","racecar","car"]
Output: ""
Explanation: There is no common prefix among the input strings.
```

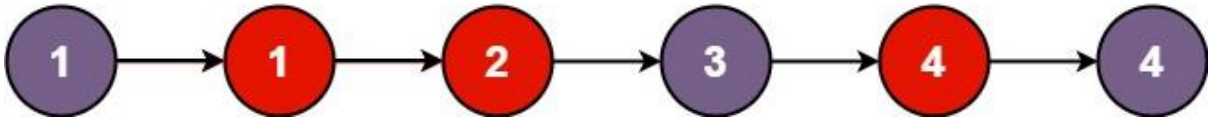
Valid Parentheses

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

An input string is valid if:

Open brackets must be closed by the same type of brackets.
Open brackets must be closed in the correct order.
Every close bracket has a corresponding open bracket of the same type.


Example 1:

Input: s = "()"
Output: true
Example 2:

Input: s = "()[]{}"
Output: true
Example 3:

Input: s = "(]"
Output: false
```

Merge Two Sorted Lists

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

Merge the two lists in a 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.
```

Example 1:




```
Input: list1 = [1,2,4], list2 = [1,3,4]
Output: [1,1,2,3,4,4]
Example 2:

Input: list1 = [], list2 = []
Output: []
Example 3:

Input: list1 = [], list2 = [0]
Output: [0]
```

Remove Duplicates from Sorted Array

```
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. Then return the
number of unique elements in nums.

Consider the number of unique elements of nums to be k, to get accepted,
you need to do the following things:

Change the array nums such that the first k elements of nums contain the
unique elements in the order they were present in nums initially. The
remaining elements of nums are not important as well as the size of nums.
Return k.
Custom Judge:

The judge will test your solution with the following code:

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

int k = removeDuplicates(nums); // Calls your implementation

assert k == expectedNums.length;
for (int i = 0; i < k; i++) {
    assert nums[i] == expectedNums[i];
}
```

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

Example 1:

Input: nums = [1,1,2]
Output: 2, nums = [1,2,_]
Explanation: Your function should return k = 2, with the first two elements of nums being 1 and 2 respectively.
It does not matter what you leave beyond the returned k (hence they are underscores).
Example 2:

Input: nums = [0,0,1,1,1,2,2,3,3,4]
Output: 5, nums = [0,1,2,3,4,_,_,_,_,_]
Explanation: Your function should return k = 5, with the first five elements of nums being 0, 1, 2, 3, and 4 respectively.
It does not matter what you leave beyond the returned k (hence they are underscores).

Remove Element

Given an integer array nums and an integer val, remove all occurrences of val in nums in-place. The order of the elements may be changed. Then return the number of elements in nums which are not equal to val.

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

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

The judge will test your solution with the following code:

```
int[] nums = [...]; // Input array
int val = ...; // Value to remove
int[] expectedNums = [...]; // The expected answer with correct length.
                            // It is sorted with no values equaling val.

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

assert k == expectedNums.length;
sort(nums, 0, k); // Sort the first k elements of nums
for (int i = 0; i < actualLength; i++) {
    assert nums[i] == expectedNums[i];
}
```

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

Example 1:

Input: nums = [3,2,2,3], val = 3
Output: 2, nums = [2,2,_,_]

```
Explanation: Your function should return k = 2, with the first two elements
of nums being 2.
It does not matter what you leave beyond the returned k (hence they are
underscores).
Example 2:

Input: nums = [0,1,2,2,3,0,4,2], val = 2
Output: 5, nums = [0,1,4,0,3,_,_,_]
Explanation: Your function should return k = 5, with the first five
elements of nums containing 0, 0, 1, 3, and 4.
Note that the five elements can be returned in any order.
It does not matter what you leave beyond the returned k (hence they are
underscores).
```

Find the Index of the First Occurrence in a String

```
Given two strings needle and haystack, return the index of the first
occurrence of needle in haystack, or -1 if needle is not part of haystack.



Example 1:

Input: haystack = "sadbutsad", needle = "sad"
Output: 0
Explanation: "sad" occurs at index 0 and 6.
The first occurrence is at index 0, so we return 0.
Example 2:

Input: haystack = "leetcode", needle = "leeto"
Output: -1
Explanation: "leeto" did not occur in "leetcode", so we return -1.
```

Search Insert Position

```
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 O(log n) runtime complexity.



Example 1:

Input: nums = [1,3,5,6], target = 5
Output: 2
Example 2:

Input: nums = [1,3,5,6], target = 2
Output: 1
```

```
Example 3:

Input: nums = [1,3,5,6], target = 7
Output: 4
```

Length of Last Word

```
Given a string s consisting of words and spaces, return the length of the
last word in the string.

A word is a maximal
substring
 consisting of non-space characters only.



Example 1:

Input: s = "Hello World"
Output: 5
Explanation: The last word is "World" with length 5.
Example 2:

Input: s = "   fly me   to   the moon  "
Output: 4
Explanation: The last word is "moon" with length 4.
Example 3:

Input: s = "luffy is still joyboy"
Output: 6
Explanation: The last word is "joyboy" with length 6.
```

Plus One

```
You are given a large integer represented as an integer array digits, where
each digits[i] is the ith digit of the integer. The digits are ordered from
most significant to least significant in left-to-right order. The large
integer does not contain any leading 0's.

Increment the large integer by one and return the resulting array of
digits.



Example 1:

Input: digits = [1,2,3]
Output: [1,2,4]
Explanation: The array represents the integer 123.
Incrementing by one gives 123 + 1 = 124.
Thus, the result should be [1,2,4].
Example 2:

Input: digits = [4,3,2,1]
```

```
Output: [4,3,2,2]
Explanation: The array represents the integer 4321.
Incrementing by one gives 4321 + 1 = 4322.
Thus, the result should be [4,3,2,2].
Example 3:

Input: digits = [9]
Output: [1,0]
Explanation: The array represents the integer 9.
Incrementing by one gives 9 + 1 = 10.
Thus, the result should be [1,0].
```

Add Binary

```
Given two binary strings a and b, return their sum as a binary string.



Example 1:

Input: a = "11", b = "1"
Output: "100"
Example 2:

Input: a = "1010", b = "1011"
Output: "10101"
```

Sqrt(x)

```
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.

For example, do not use pow(x, 0.5) in c++ or x ** 0.5 in python.


Example 1:

Input: x = 4
Output: 2
Explanation: The square root of 4 is 2, so we return 2.
Example 2:

Input: x = 8
Output: 2
Explanation: The square root of 8 is 2.82842..., and since we round it down
to the nearest integer, 2 is returned.
```

Climbing Stairs

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?

Example 1:

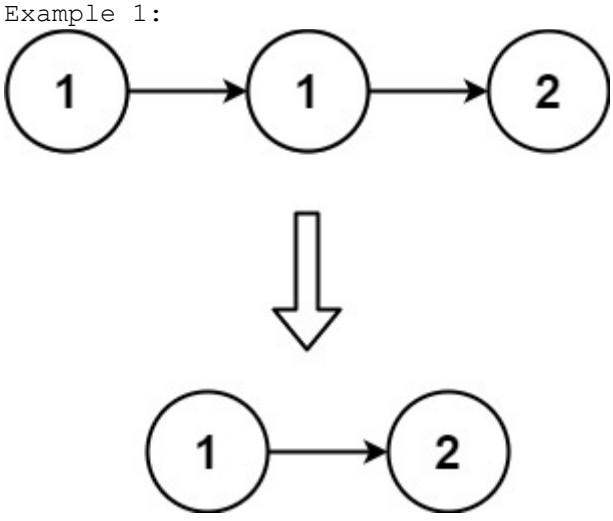
Input: n = 2
Output: 2
Explanation: There are two ways to climb to the top.
1. 1 step + 1 step
2. 2 steps
Example 2:

Input: n = 3
Output: 3
Explanation: 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

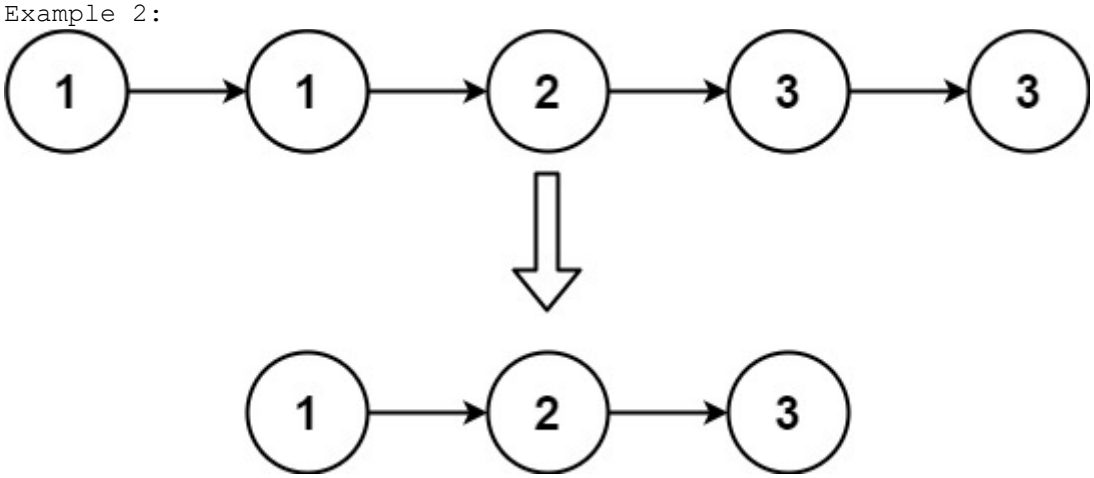
Remove Duplicates from Sorted List

Given the head of a sorted linked list, delete all duplicates such that each element appears only once. Return the linked list sorted as well.

Example 1:



```
Input: head = [1,1,2]
Output: [1,2]
Example 2:
```




```
Input: head = [1,1,2,3,3]
Output: [1,2,3]
```

Merge Sorted Array

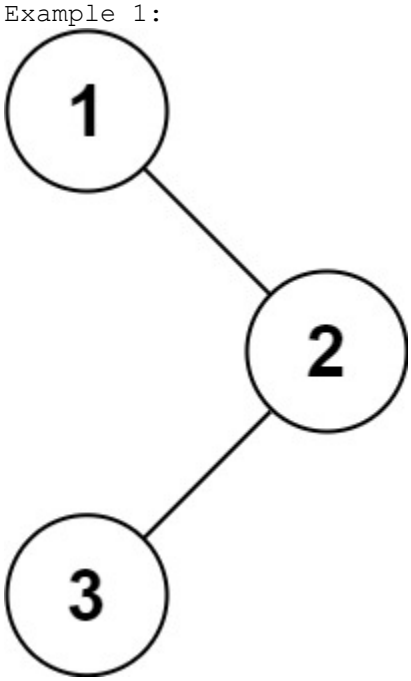
```
You are given two integer arrays nums1 and nums2, sorted in non-decreasing
order, and two integers m and n, representing the number of elements in
nums1 and nums2 respectively.

Merge nums1 and nums2 into a single array sorted in non-decreasing order.

The final sorted array should not be returned by the function, but instead
be stored inside the array nums1. To accommodate this, nums1 has a length
of m + n, where the first m elements denote the elements that should be
merged, and the last n elements are set to 0 and should be ignored. nums2
has a length of n.



Example 1:

Input: nums1 = [1,2,3,0,0,0], m = 3, nums2 = [2,5,6], n = 3
Output: [1,2,2,3,5,6]
Explanation: The arrays we are merging are [1,2,3] and [2,5,6].
The result of the merge is [1,2,2,3,5,6] with the underlined elements
coming from nums1.
Example 2:

Input: nums1 = [1], m = 1, nums2 = [], n = 0
Output: [1]
Explanation: The arrays we are merging are [1] and [].
The result of the merge is [1].
Example 3:

Input: nums1 = [0], m = 0, nums2 = [1], n = 1
Output: [1]
Explanation: The arrays we are merging are [] and [1].
```

The result of the merge is [1].
Note that because m = 0, there are no elements in nums1. The 0 is only there to ensure the merge result can fit in nums1.

Binary Tree Inorder Traversal

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

Example 1:



Input: root = [1,null,2,3]
Output: [1,3,2]
Example 2:

Input: root = []
Output: []
Example 3:

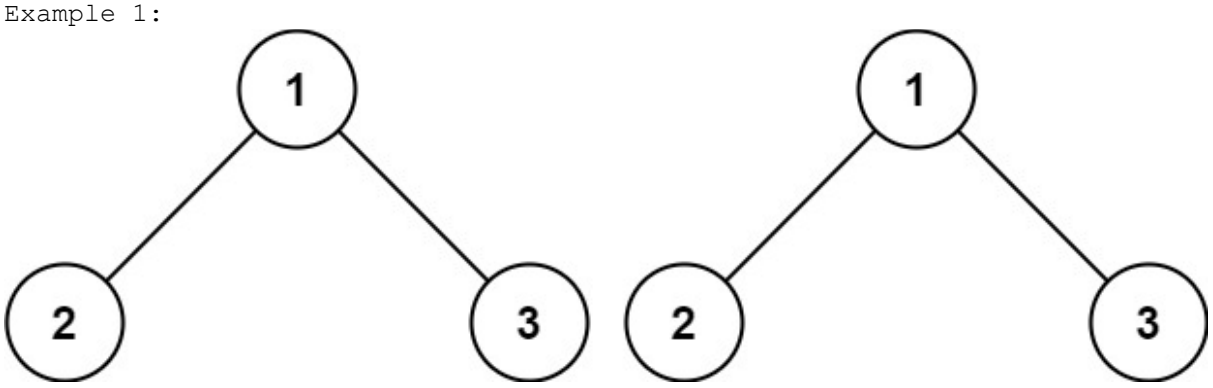
Input: root = [1]
Output: [1]

Same Tree

Given the roots of two binary trees p and q, write a function to check if they are the same or not.

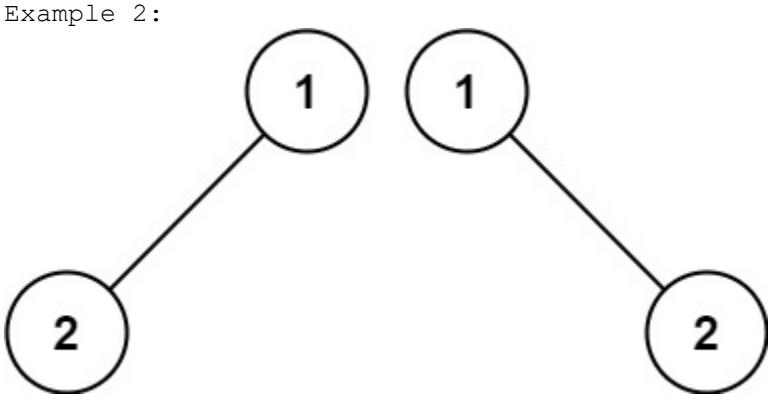
Two binary trees are considered the same if they are structurally identical, and the nodes have the same value.

Example 1:



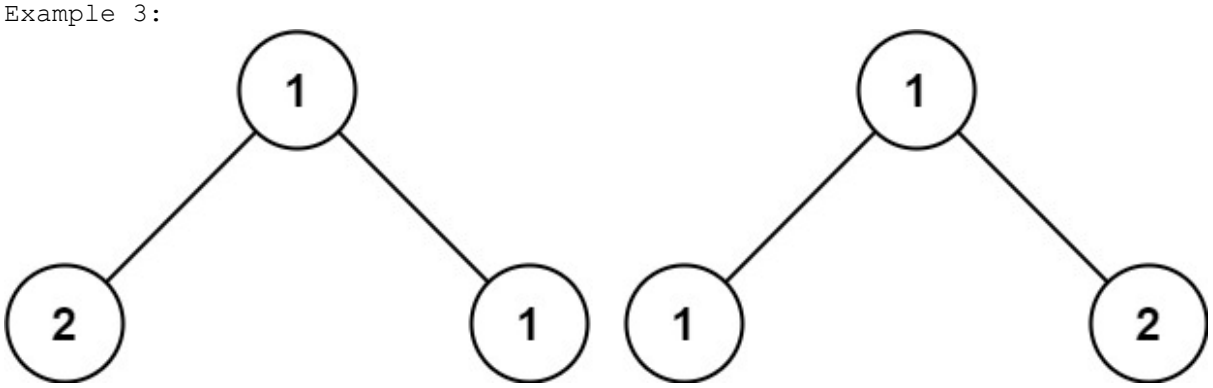

```
Input: p = [1,2,3], q = [1,2,3]
Output: true
```

Example 2:




```
Input: p = [1,2], q = [1,null,2]
Output: false
```

Example 3:

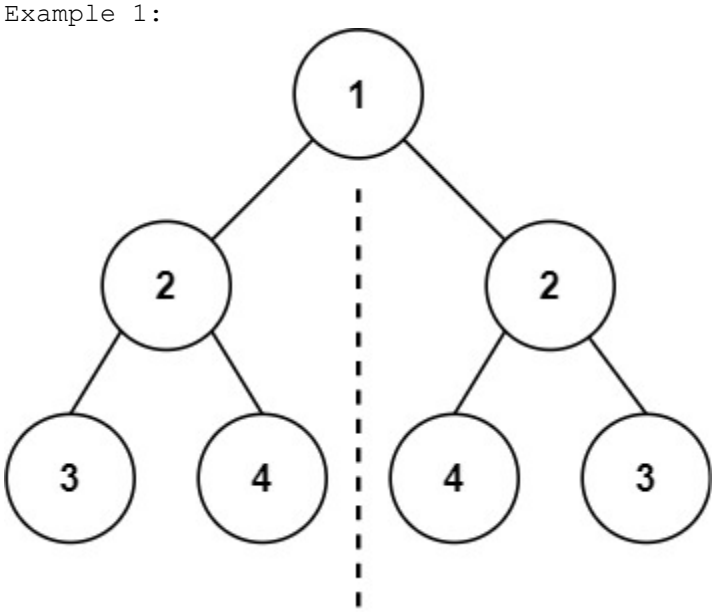



```
Input: p = [1,2,1], q = [1,1,2]
Output: false
```

Symmetric Tree

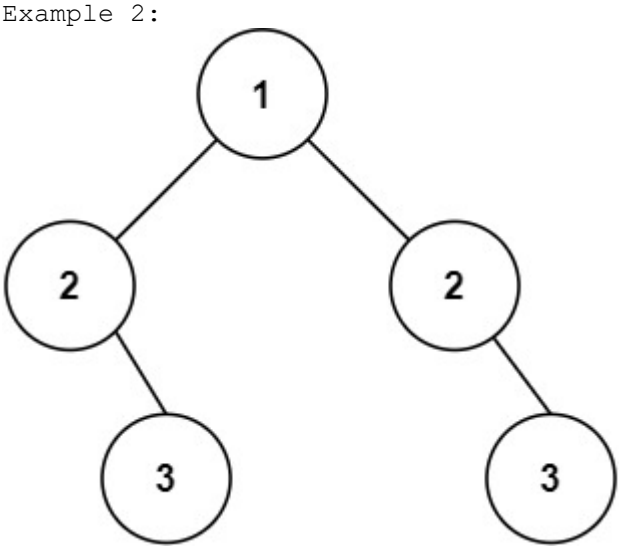
Given the root of a binary tree, check whether it is a mirror of itself (i.e., symmetric around its center).

Example 1:




```
Input: root = [1,2,2,3,4,4,3]
Output: true
```

Example 2:



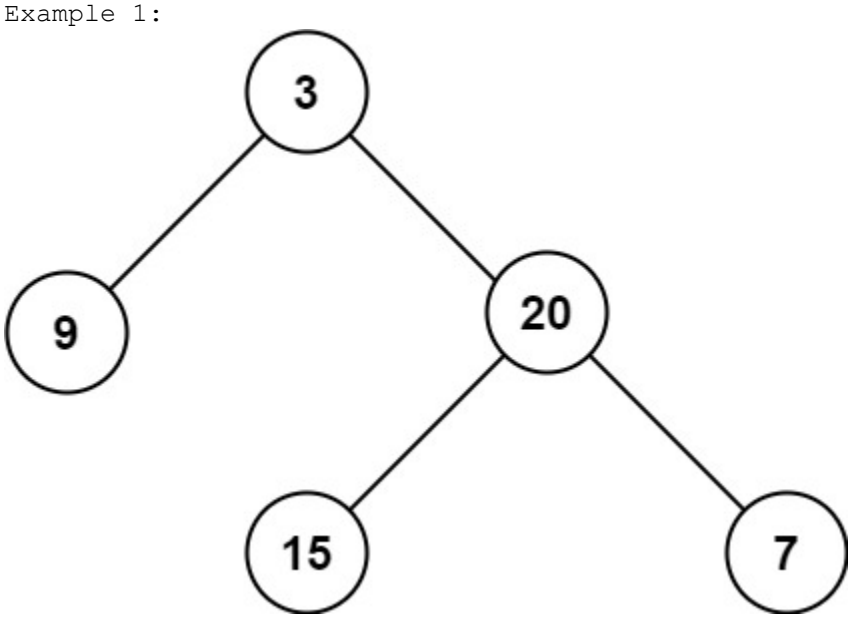

```
Input: root = [1,2,2,null,3,null,3]
Output: false
```

Maximum Depth of Binary Tree

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.

Example 1:

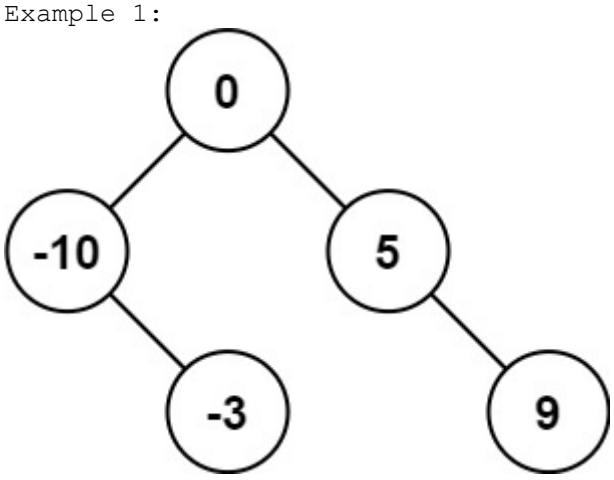



```
Input: root = [3,9,20,null,null,15,7]
Output: 3
Example 2:

Input: root = [1,null,2]
Output: 2
```

Convert Sorted Array to Binary Search Tree

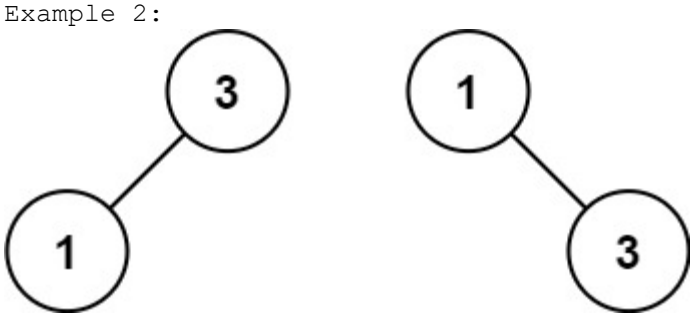
```
Given an integer array nums where the elements are sorted in ascending
order, convert it to a
height-balanced
 binary search tree.
```

Example 1:




```
Input: nums = [-10,-3,0,5,9]
Output: [0,-3,9,-10,null,5]
Explanation: [0,-10,5,null,-3,null,9] is also accepted:
```

Example 2:

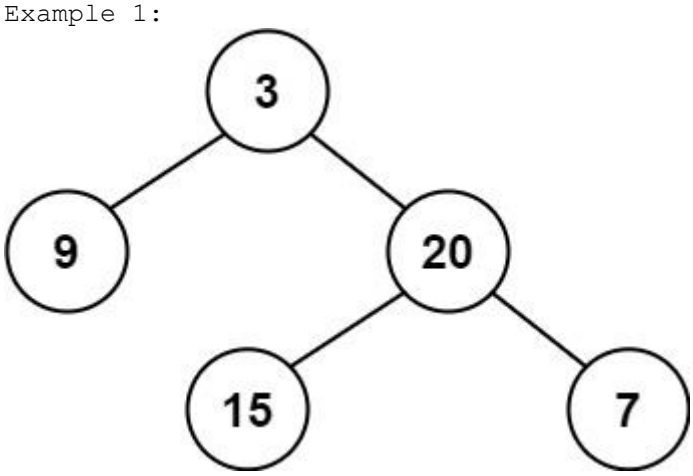



```
Input: nums = [1,3]
Output: [3,1]
Explanation: [1,null,3] and [3,1] are both height-balanced BSTs.
```

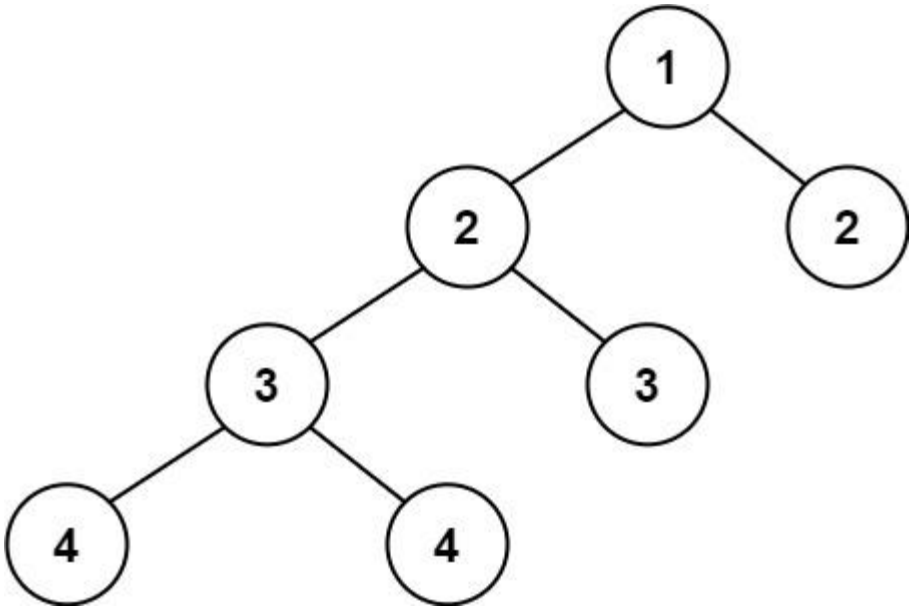
Balanced Binary Tree

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

Example 1:



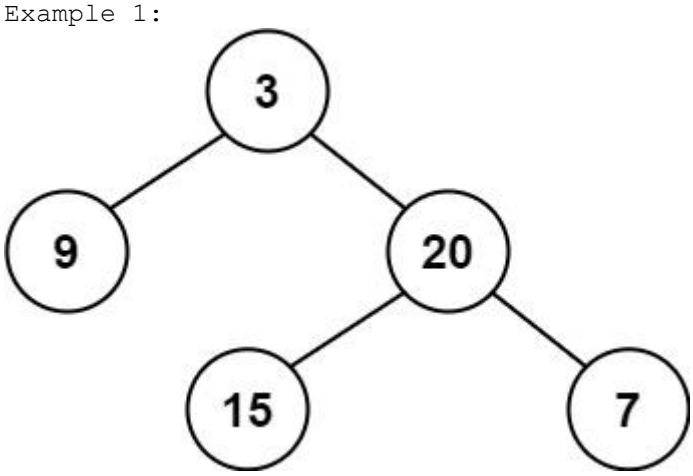

```
Input: root = [3,9,20,null,null,15,7]
Output: true
Example 2:
```



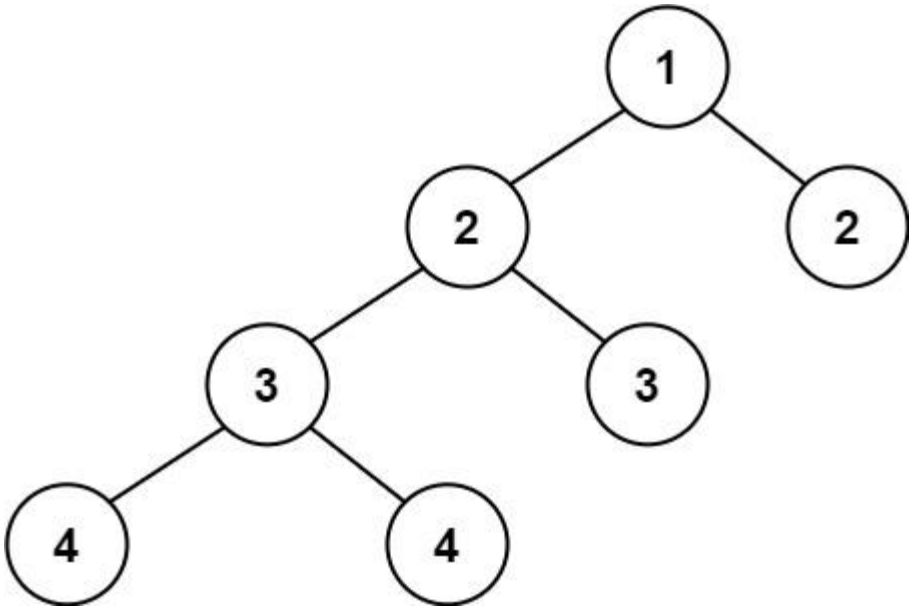

```
Input: root = [1,2,2,3,3,null,null,4,4]
Output: false
Example 3:

Input: root = []
Output: true
```

Balanced Binary Tree

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


Example 1:
```




```
Input: root = [3,9,20,null,null,15,7]
Output: true
Example 2:
```




```
Input: root = [1,2,2,3,3,null,null,4,4]
Output: false
Example 3:

Input: root = []
Output: true
```

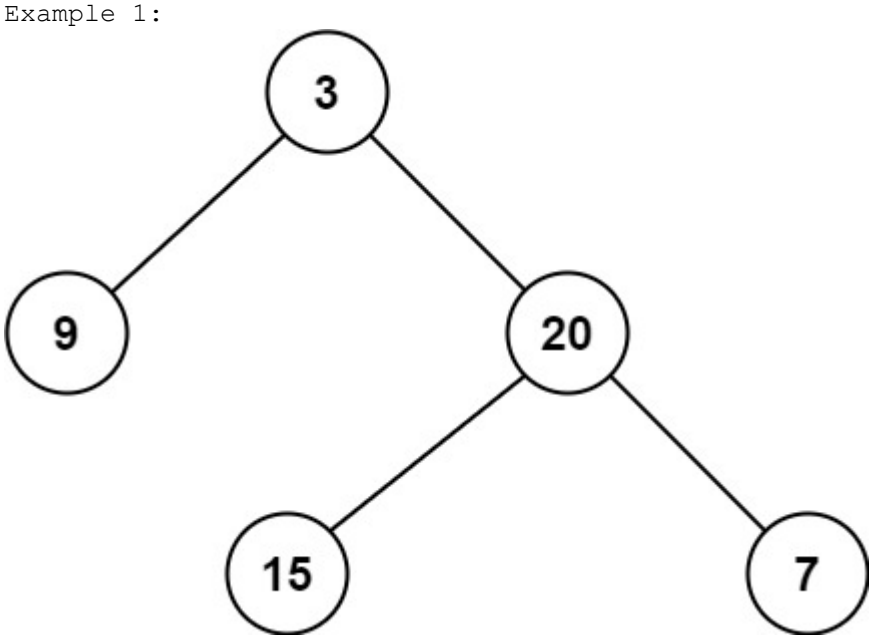
Minimum Depth of Binary Tree

Given a binary tree, find its minimum depth.

The minimum depth is the number of nodes along the shortest path from the root node down to the nearest leaf node.

Note: A leaf is a node with no children.

Example 1:



Input: root = [3,9,20,null,null,15,7]
Output: 2
Example 2:

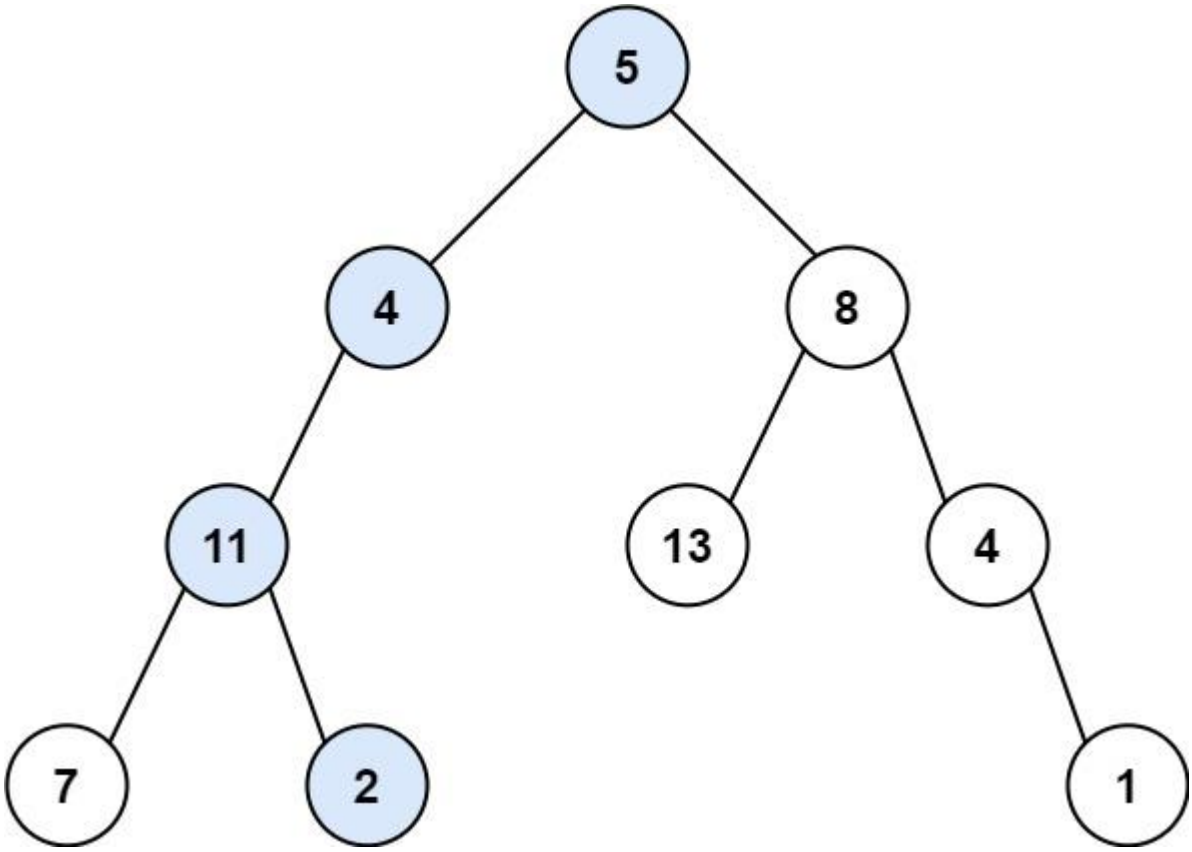
Input: root = [2,null,3,null,4,null,5,null,6]
Output: 5

Path Sum

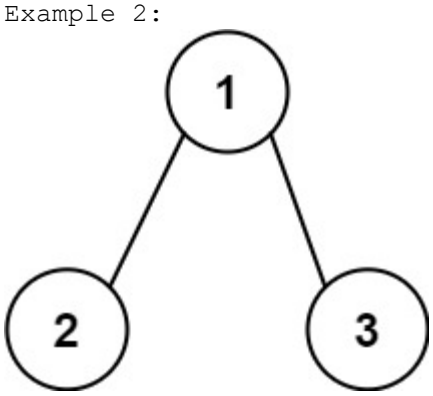
Given the root of a binary tree and an integer targetSum, return true if the tree has a root-to-leaf path such that adding up all the values along the path equals targetSum.

A leaf is a node with no children.

Example 1:




```
Input: root = [5,4,8,11,null,13,4,7,2,null,null,null,1], targetSum = 22
Output: true
Explanation: The root-to-leaf path with the target sum is shown.
Example 2:
```



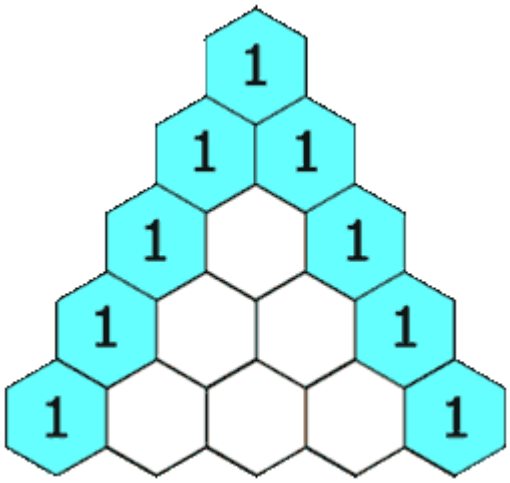

```
Input: root = [1,2,3], targetSum = 5
Output: false
Explanation: There two root-to-leaf paths in the tree:
(1 --> 2): The sum is 3.
(1 --> 3): The sum is 4.
There is no root-to-leaf path with sum = 5.
Example 3:

Input: root = [], targetSum = 0
Output: false
Explanation: Since the tree is empty, there are no root-to-leaf paths.
```

Pascal's Triangle

```
Given an integer numRows, return the first numRows of Pascal's triangle.
```

In Pascal's triangle, each number is the sum of the two numbers directly above it as shown:

Example 1:



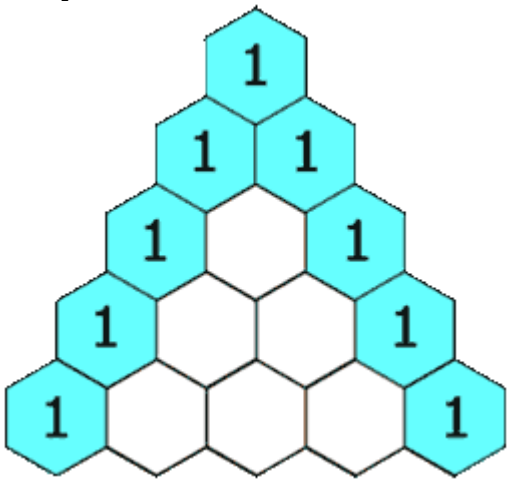

```
Input: numRows = 5
Output: [[1],[1,1],[1,2,1],[1,3,3,1],[1,4,6,4,1]]
Example 2:

Input: numRows = 1
Output: [[1]]
```

Pascal's Triangle II

Given an integer rowIndex, return the rowIndexth (0-indexed) row of the Pascal's triangle.

In Pascal's triangle, each number is the sum of the two numbers directly above it as shown:

Example 1:




```
Input: rowIndex = 3
Output: [1,3,3,1]
Example 2:
```

```
Input: rowIndex = 0
Output: [1]
Example 3:

Input: rowIndex = 1
Output: [1,1]
```

Best Time to Buy and Sell Stock

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



Example 1:

Input: prices = [7,1,5,3,6,4]
Output: 5
Explanation: 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.
Example 2:

Input: prices = [7,6,4,3,1]
Output: 0
Explanation: In this case, no transactions are done and the max profit = 0.
```

Valid Palindrome

```
A phrase is a palindrome if, after converting all uppercase letters into
lowercase letters and removing all non-alphanumeric characters, it reads
the same forward and backward. Alphanumeric characters include letters and
numbers.

Given a string s, return true if it is a palindrome, or false otherwise.



Example 1:

Input: s = "A man, a plan, a canal: Panama"
Output: true
Explanation: "amanaplanacanalpanama" is a palindrome.
Example 2:

Input: s = "race a car"
Output: false
Explanation: "raceacar" is not a palindrome.
Example 3:

Input: s = " "
Output: true
Explanation: s is an empty string "" after removing non-alphanumeric
characters.
```

Since an empty string reads the same forward and backward, it is a palindrome.

Single Number

Given a non-empty array of integers nums, every element appears twice except for one. Find that single one.

You must implement a solution with a linear runtime complexity and use only constant extra space.

Example 1:

Input: nums = [2,2,1]
Output: 1
Example 2:

Input: nums = [4,1,2,1,2]
Output: 4
Example 3:

Input: nums = [1]
Output: 1

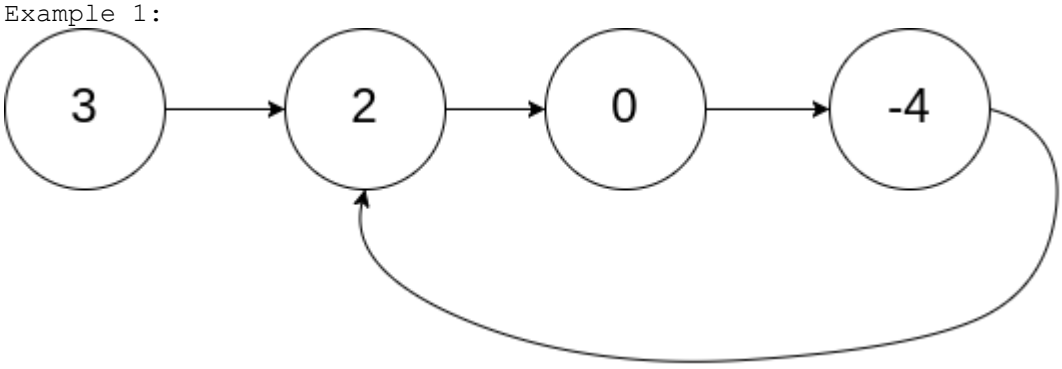
Linked List Cycle

Given head, the head of a linked list, determine if the linked list has a cycle in it.

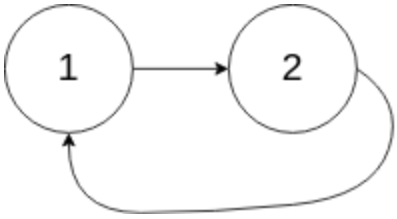
There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the next pointer. Internally, pos is used to denote the index of the node that tail's next pointer is connected to. Note that pos is not passed as a parameter.

Return true if there is a cycle in the linked list. Otherwise, return false.

Example 1:



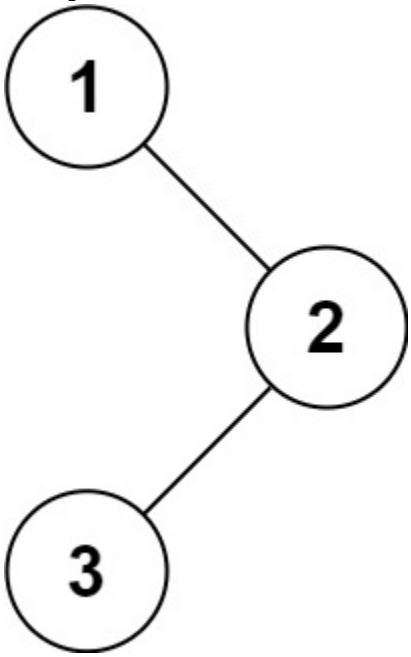
Input: head = [3,2,0,-4], pos = 1
Output: true
Explanation: There is a cycle in the linked list, where the tail connects to the 1st node (0-indexed).
Example 2:



Binary Tree Preorder Traversal

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

Example 1:



Input: root = [1,null,2,3]
Output: [1,2,3]
Example 2:

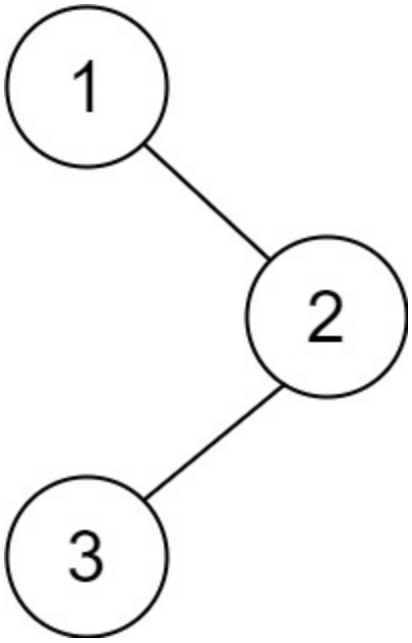
Input: root = []
Output: []
Example 3:

Input: root = [1]
Output: [1]

Binary Tree Postorder Traversal

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

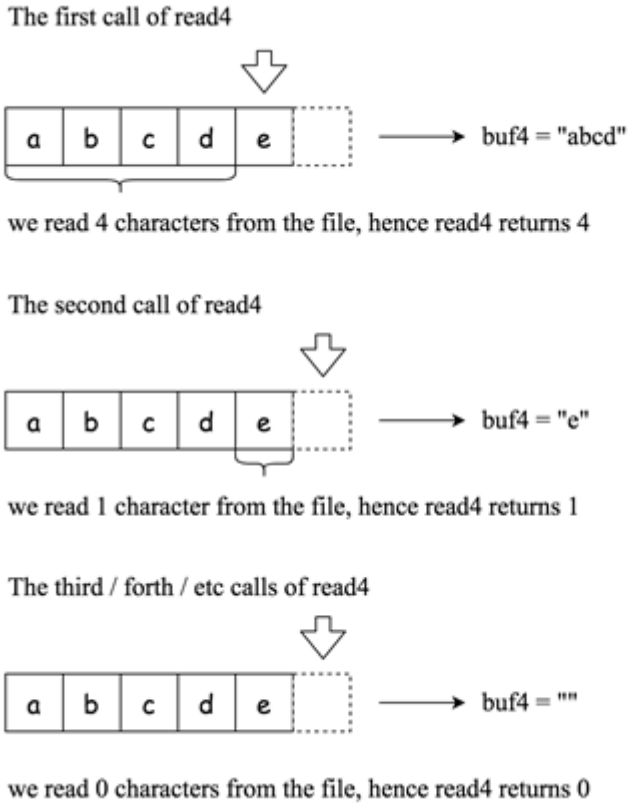
Example 1:




```
Input: root = [1,null,2,3]
Output: [3,2,1]
Example 2:

Input: root = []
Output: []
Example 3:

Input: root = [1]
Output: [1]
```

Read N Characters Given Read4

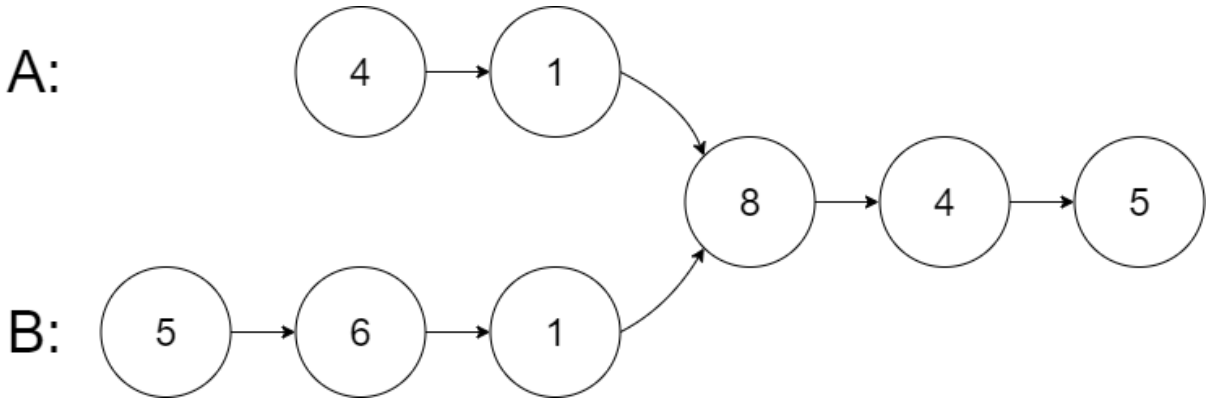
```
Given a file and assume that you can only read the file using a given
method read4, implement a method to read n characters.
Method read4:
The API read4 reads four consecutive characters from file, then writes
those characters into the buffer array buf4.
The return value is the number of actual characters read.
Note that read4() has its own file pointer, much like FILE *fp in C.
Definition of read4:
   Parameter:  char[] buf4
    Returns:    int

buf4[] is a destination, not a source. The results from read4 will be
copied to buf4[].

Below is a high-level example of how read4 works:
```



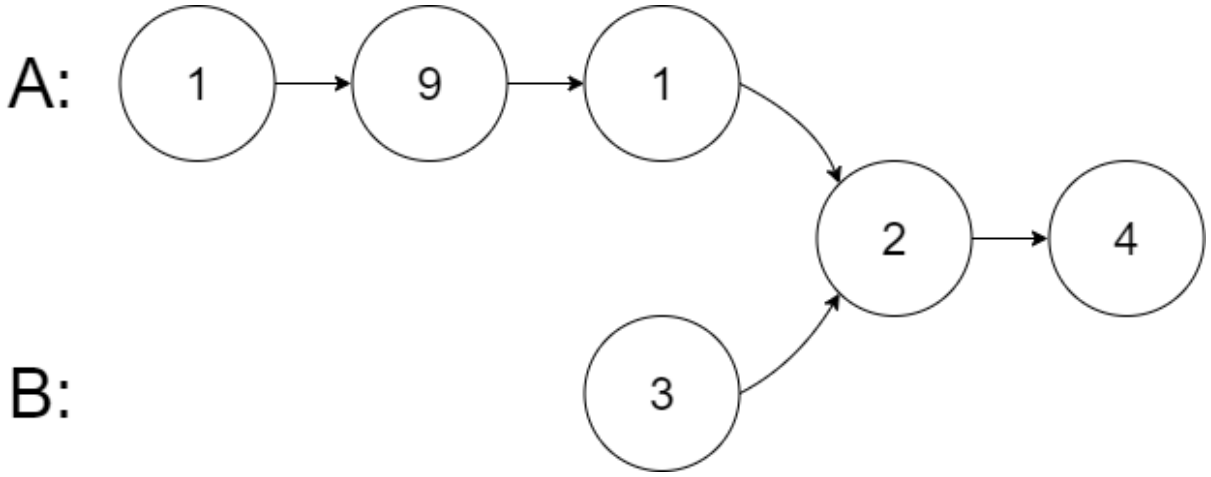

```
File file("abcde"); // File is "abcde", initially file pointer (fp) points
to 'a'
char[] buf4 = new char[4]; // Create buffer with enough space to store
characters
read4(buf4); // read4 returns 4. Now buf4 = "abcd", fp points to 'e'
read4(buf4); // read4 returns 1. Now buf4 = "e", fp points to end of file
read4(buf4); // read4 returns 0. Now buf4 = "", fp points to end of file
```

Intersection of Two Linked Lists

Given the heads of two singly linked-lists headA and headB, return the node at which the two lists intersect. If the two linked lists have no intersection at all, return null. For example, the following two linked lists begin to intersect at node c1: The test cases are generated such that there are no cycles anywhere in the entire linked structure. Note that the linked lists must retain their original structure after the function returns. Custom Judge: The inputs to the judge are given as follows (your program is not given these inputs): intersectVal - The value of the node where the intersection occurs. This is 0 if there is no intersected node. listA - The first linked list. listB - The second linked list. skipA - The number of nodes to skip ahead in listA (starting from the head) to get to the intersected node. skipB - The number of nodes to skip ahead in listB (starting from the head) to get to the intersected node. The judge will then create the linked structure based on these inputs and pass the two heads, headA and headB to your program. If you correctly return the intersected node, then your solution will be accepted. Example 1:



Input: intersectVal = 8, listA = [4,1,8,4,5], listB = [5,6,1,8,4,5], skipA = 2, skipB = 3 Output: Intersected at '8' Explanation: The intersected node's value is 8 (note that this must not be 0 if the two lists intersect). From the head of A, it reads as [4,1,8,4,5]. From the head of B, it reads as [5,6,1,8,4,5]. There are 2 nodes before the intersected node in A; There are 3 nodes before the intersected node in B. - Note that the intersected node's value is not 1 because the nodes with value 1 in A and B (2nd node in A and 3rd node in B) are different node references. In other words, they point to two different locations in memory, while the nodes with value 8 in A and B (3rd node in A and 4th node in B) point to the same location in memory. Example 2:



Input: intersectVal = 2, listA **Excel Sheet Column Title**

```
Given an integer columnNumber, return its corresponding column title as it
appears in an Excel sheet.

For example:

A -> 1
B -> 2
C -> 3
...
Z -> 26
AA -> 27
AB -> 28
...


Example 1:

Input: columnNumber = 1
```

```
Output: "A"
Example 2:

Input: columnNumber = 28
Output: "AB"
Example 3:

Input: columnNumber = 701
Output: "ZY"
```

Majority Element

```
Given an array nums of size n, return the majority element.

The majority element is the element that appears more than ⌊n / 2⌋ times.
You may assume that the majority element always exists in the array.



Example 1:

Input: nums = [3,2,3]
Output: 3
Example 2:

Input: nums = [2,2,1,1,1,2,2]
Output: 2
```

Excel Sheet Column Number

```
Given a string columnTitle that represents the column title as appears in
an Excel sheet, return its corresponding column number.

For example:

A -> 1
B -> 2
C -> 3
...
Z -> 26
AA -> 27
AB -> 28
...


Example 1:

Input: columnTitle = "A"
Output: 1
Example 2:

Input: columnTitle = "AB"
Output: 28
Example 3:

Input: columnTitle = "ZY"
Output: 701
```

Combine Two Tables

```
Table: Person
```

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| personId    | int     |
| lastName    | varchar |
| firstName   | varchar |
+-------------+---------+
```
personId is the primary key column for this table.
This table contains information about the ID of some persons and their first and last names.

Table: Address

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| addressId   | int     |
| personId    | int     |
| city        | varchar |
| state       | varchar |
+-------------+---------+
```
addressId is the primary key column for this table.
Each row of this table contains information about the city and state of one person with ID = PersonId.

Write an SQL query to report the first name, last name, city, and state of each person in the Person table. If the address of a personId is not present in the Address table, report null instead.

Return the result table in any order.

The query result format is in the following example.

Example 1:

Input:
Person table:
```
+----------+----------+-----------+
| personId | lastName | firstName |
+----------+----------+-----------+
| 1        | Wang     | Allen     |
| 2        | Alice    | Bob       |
```

Employees Earning More Than Their Managers

Table: Employee

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| id          | int     |
| name        | varchar |
| salary      | int     |
| managerId   | int     |
+-------------+---------+
```
id is the primary key column for this table.
Each row of this table indicates the ID of an employee, their name, salary, and the ID of their manager.

Write an SQL query to find the employees who earn more than their managers.

Return the result table in any order.

The query result format is in the following example.

Example 1:

Input:
Employee table:

id	name	salary	managerId
1	Joe	70000	3
2	Henry	80000	4
3	Sam	60000	Null
4	Max	90000	Null

Output:

Employee
Joe

Explanation: Joe is the only employee who earns more than his manager.

Duplicate Emails

Table: Person

Column Name	Type
id	int
email	varchar

id is the primary key column for this table.
Each row of this table contains an email. The emails will not contain uppercase letters.

Write an SQL query to report all the duplicate emails. Note that it's guaranteed that the email field is not NULL.

Return the result table in any order.

The query result format is in the following example.

Example 1:

Input:
Person table:

id	email
1	a@b.com

```
| 2  | c@d.com |
| 3  | a@b.com |
+----+---------+
Output:
+---------+
| Email   |
+---------+
| a@b.com |
+---------+
Explanation: a@b.com is repeated two times.
```

Customers Who Never Order

```
Table: Customers

+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| id          | int     |
| name        | varchar |
+-------------+---------+
id is the primary key column for this table.
Each row of this table indicates the ID and name of a customer.


Table: Orders

+-------------+------+
| Column Name | Type |
+-------------+------+
| id          | int  |
| customerId  | int  |
+-------------+------+
id is the primary key column for this table.
customerId is a foreign key of the ID from the Customers table.
Each row of this table indicates the ID of an order and the ID of the
customer who ordered it.


Write an SQL query to report all customers who never order anything.

Return the result table in any order.

The query result format is in the following example.



Example 1:

Input:
Customers table:
+----+-------+
| id | name  |
+----+-------+
| 1  | Joe   |
| 2  | Henry |
| 3  | Sam   |
| 4  | Max   |
```

Reverse Bits

```
Reverse bits of a given 32 bits unsigned integer.
```

Note:

Note that in some languages, such as Java, there is no unsigned integer type. In this case, both input and output will be given as a signed integer type. They should not affect your implementation, as the integer's internal binary representation is the same, whether it is signed or unsigned.
In Java, the compiler represents the signed integers using 2's complement notation. Therefore, in Example 2 above, the input represents the signed integer -3 and the output represents the signed integer -1073741825.

Example 1:

Input: n = 00000010100101000001111010011100
Output: 964176192 (00111001011110000010100101000000)
Explanation: The input binary string 00000010100101000001111010011100 represents the unsigned integer 43261596, so return 964176192 which its binary representation is 00111001011110000010100101000000.
Example 2:

Input: n = 11111111111111111111111111111101
Output: 3221225471 (10111111111111111111111111111111)
Explanation: The input binary string 11111111111111111111111111111101 represents the unsigned integer 4294967293, so return 3221225471 which its binary representation i

Number of 1 Bits

Write a function that takes the binary representation of an unsigned integer and returns the number of '1' bits it has (also known as the Hamming weight).

Note:

Note that in some languages, such as Java, there is no unsigned integer type. In this case, the input will be given as a signed integer type. It should not affect your implementation, as the integer's internal binary representation is the same, whether it is signed or unsigned.
In Java, the compiler represents the signed integers using 2's complement notation. Therefore, in Example 3, the input represents the signed integer. -3.

Example 1:

Input: n = 00000000000000000000000000001011
Output: 3
Explanation: The input binary string 00000000000000000000000000001011 has a total of three '1' bits.
Example 2:

Input: n = 00000000000000000000000010000000
Output: 1
Explanation: The input binary string 00000000000000000000000010000000 has a total of one '1' bit.
Example 3:

Input: n = 11111111111111111111111111111101
Output: 31
Explanation: The input binary string 11111111111111111111111111111101 has a total of thirty one '1' bits.

Valid Phone Numbers

Given a text file file.txt that contains a list of phone numbers (one per line), write a one-liner bash script to print all valid phone numbers.

You may assume that a valid phone number must appear in one of the following two formats: (xxx) xxx-xxxx or xxx-xxx-xxxx. (x means a digit)

You may also assume each line in the text file must not contain leading or trailing white spaces.

Example:

Assume that file.txt has the following content:

```
987-123-4567
123 456 7890
(123) 456-7890
```

Your script should output the following valid phone numbers:

```
987-123-4567
(123) 456-7890
```

Tenth Line

Given a text file file.txt, print just the 10th line of the file.

Example:

Assume that file.txt has the following content:

```
Line 1
Line 2
Line 3
Line 4
Line 5
Line 6
Line 7
Line 8
Line 9
Line 10
```

Your script should output the tenth line, which is:

```
Line 10
```

Delete Duplicate Emails

Table: Person

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| id          | int     |
| email       | varchar |
+-------------+---------+
```

id is the primary key column for this table.
Each row of this table contains an email. The emails will not contain uppercase letters.

Write an SQL query to delete all the duplicate emails, keeping only one unique email with the smallest id. Note that you are supposed to write a DELETE statement and not a SELECT one.

After running your script, the answer shown is the Person table. The driver will first compile and run your piece of code and then show the Person table. The final order of the Person table does not matter.

The query result format is in the following example.

Example 1:

Input:
Person table:

id	email
1	john@example.com
2	bob@example.com
3	john@example.com

Output:

id	email
1	john@example.com
2	bob@example.com

Explanation: john@example.com is repeated two times. We keep the row with the smallest Id = 1.

Rising Temperature

Table: Weather

Column Name	Type
id	int
recordDate	date
temperature	int

id is the primary key for this table.
This table contains information about the temperature on a certain day.

Write an SQL query to find all dates' Id with higher temperatures compared to its previous dates (yesterday).

Return the result table in any order.

The query result format is in the following example.

Example 1:

Input:
Weather table:

```
| id | recordDate | temperature |
+----+------------+-------------+
| 1  | 2015-01-01 | 10          |
| 2  | 2015-01-02 | 25          |
| 3  | 2015-01-03 | 20          |
| 4  | 2015-01-04 | 30          |
+----+------------+-------------+
Output:
+----+
| id |
+----+
| 2  |
| 4  |
+----+
Explanation:
In 2015-01-02, the temperature was higher than the previous day (10 -> 25).
In 2015-01-04, the temperature was higher than the previous day (20 -> 30).
```

Hard Questions

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 O(log (m+n)).



Example 1:

Input: nums1 = [1,3], nums2 = [2]
Output: 2.00000
Explanation: merged array = [1,2,3] and median is 2.
Example 2:

Input: nums1 = [1,2], nums2 = [3,4]
Output: 2.50000
Explanation: merged array = [1,2,3,4] and median is (2 + 3) / 2 = 2.5.
```

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).



Example 1:

Input: s = "aa", p = "a"
Output: false
Explanation: "a" does not match the entire string "aa".
Example 2:
```

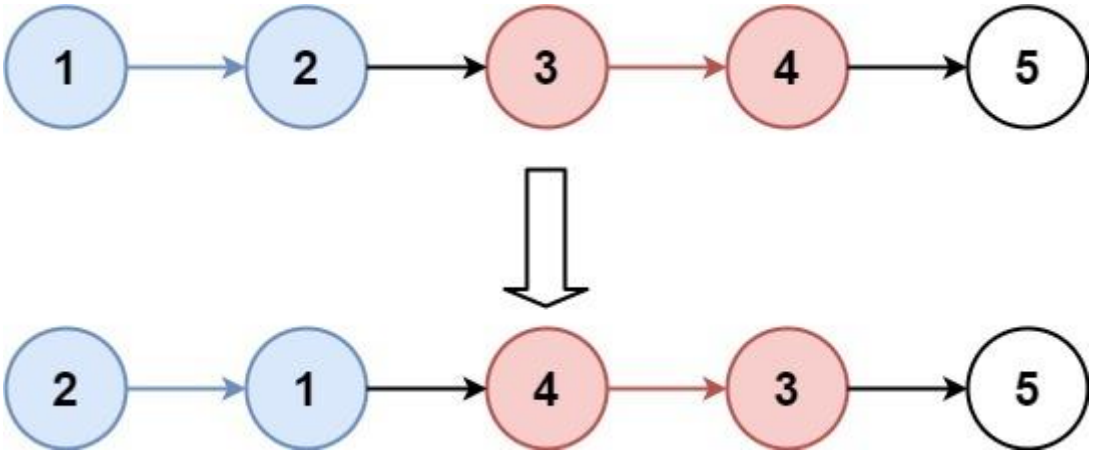
```
Input: s = "aa", p = "a*"
Output: true
Explanation: '*' means zero or more of the preceding element, 'a'.
Therefore, by repeating 'a' once, it becomes "aa".
Example 3:

Input: s = "ab", p = ".*"
Output: true
Explanation: ".*" means "zero or more (*) of any character (.)".
```

Merge k Sorted Lists

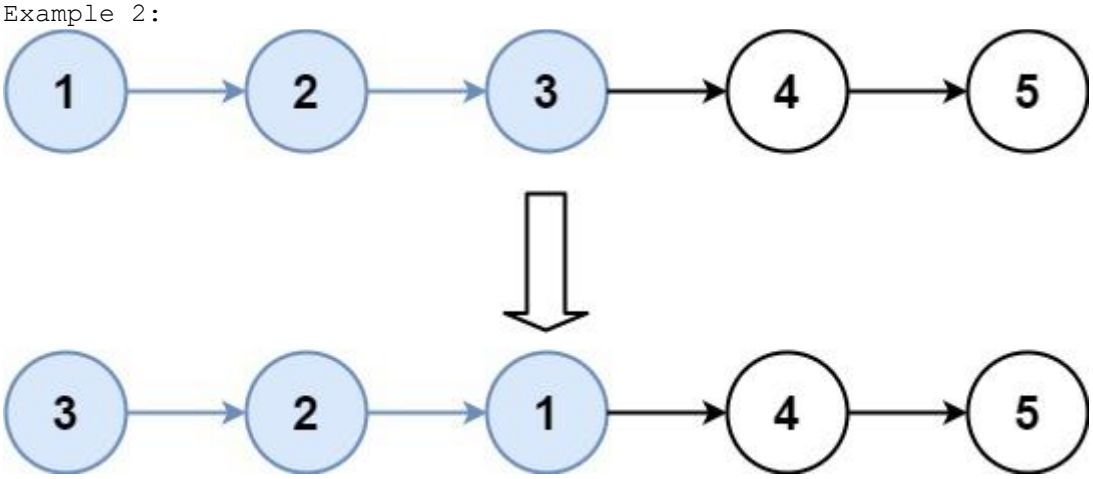
```
You are given an array of k linked-lists lists, each linked-list is sorted
in ascending order.

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



Example 1:

Input: lists = [[1,4,5],[1,3,4],[2,6]]
Output: [1,1,2,3,4,4,5,6]
Explanation: The linked-lists are:
[
  1->4->5,
  1->3->4,
  2->6
]
merging them into one sorted list:
1->1->2->3->4->4->5->6
Example 2:

Input: lists = []
Output: []
Example 3:

Input: lists = [[]]
Output: []
```

Reverse Nodes in k-Group

```
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.

You may not alter the values in the list's nodes, only nodes themselves may
be changed.



Example 1:
```




```
Input: head = [1,2,3,4,5], k = 2
Output: [2,1,4,3,5]
Example 2:
```



Substring with Concatenation of All Words

```
You are given a string s and an array of strings words. All the strings of
words are of the same length.

A concatenated substring in s is a substring that contains all the strings
of any permutation of words concatenated.

For example, if words = ["ab","cd","ef"], then "abcdef", "abefcd",
"cdabef", "cdefab", "efabcd", and "efcdab" are all concatenated strings.
"acdbef" is not a concatenated substring because it is not the
concatenation of any permutation of words.
Return the starting indices of all the concatenated substrings in s. You
can return the answer in any order.



Example 1:

Input: s = "barfoothefoobarman", words = ["foo","bar"]
Output: [0,9]
Explanation: Since words.length == 2 and words[i].length == 3, the
concatenated substring has to be of length 6.
The substring starting at 0 is "barfoo". It is the concatenation of
["bar","foo"] which is a permutation of words.
```

The substring starting at 9 is "foobar". It is the concatenation of ["foo","bar"] which is a permutation of words.
The output order does not matter. Returning [9,0] is fine too.
Example 2:

Input: s = "wordgoodgoodgoodbestword", words = ["word","good","best","word"]
Output: []
Explanation: Since words.length == 4 and words[i].length == 4, the concatenated substring has to be of length 16.
There is no substring of length 16 is s that is equal to the concatenation of any permutation of words.
We return an empty array.
Example 3:

Input: s = "barfoofoobarthefoobarman", words = ["bar","foo","the"]
Output: [6,9,12]
Explanation: Since words.length == 3 and words[i].length == 3, the concatenated substring has to be of length 9.
The substring starting at 6 is "foobarthe". It is the concatenation of ["foo","bar","the"] which is a permutation of words.
The substring starting at 9 is "barthefoo". It is the concatenation of ["bar","the","foo"] which is a permutation of words.
The substring starting at 12 is "thefoobar". It is the concatenation of ["the","foo","bar"] which is a permutation of

Longest Valid Parentheses

Given a string containing just the characters '(' and ')', return the length of the longest valid (well-formed) parentheses substring
.

Example 1:

Input: s = "(()"
Output: 2
Explanation: The longest valid parentheses substring is "()".
Example 2:

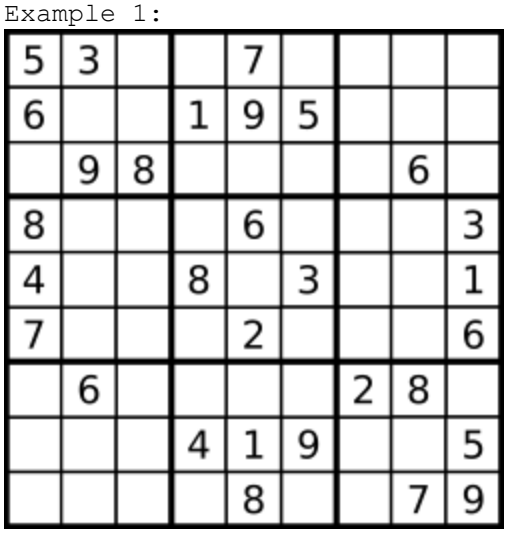
Input: s = ")()())"
Output: 4
Explanation: The longest valid parentheses substring is "()()".
Example 3:

Input: s = ""
Output: 0
Write a program to solve a Sudoku puzzle by filling the empty cells.

A sudoku solution must satisfy all of the following rules:

Each of the digits 1-9 must occur exactly once in each row.
Each of the digits 1-9 must occur exactly once in each column.
Each of the digits 1-9 must occur exactly once in each of the 9 3x3 sub-boxes of the grid.
The '.' character indicates empty cells.

Example 1:



Input: board =
[["5","3",".",".","7",".",".",".","."],["6",".",".","1","9","5",".",".","."],[".","9","8",".",".",".",".","6","."],["8",".",".",".","6",".",".",".","3"],["4",".",".","8",".","3",".",".","1"],["7",".",".",".","2",".",".",".","6"],[".","6",".",".",".",".",".","2","8","."],[".",".",".","4","1","9",".",".","5"],[".",".",".",".","8",".",".","7","9"]]
Output:
[["5","3","4","6","7","8","9","1","2"],["6","7","2","1","9","5","3","4","8"],["1","9","8","3","4","2","5","6","7"],["8","5","9","7","6","1","4","2","3"],["4","2","6","8","5","3","7","9","1"],["7","1","3","9","2","4","8","5","6"],["9","6","1","5","3","7","2","8","4"],["2","8","7","4","1","9","6","3","5"],["3","4","5","2","8","6","1","7","9"]]
Explanation: The input board is shown above and the only valid solution is shown below:

First Missing Positive

Given an unsorted integer array nums, return the smallest missing positive integer.

You must implement an algorithm that runs in O(n) time and uses constant extra space.

Example 1:

Input: nums = [1,2,0]
Output: 3
Explanation: The numbers in the range [1,2] are all in the array.
Example 2:

Input: nums = [3,4,-1,1]
Output: 2
Explanation: 1 is in the array but 2 is missing.
Example 3:

Input: nums = [7,8,9,11,12]
Output: 1
Explanation: The smallest positive integer 1 is missing.

Trapping Rain Water

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.

Example 1:

Input: height = [0,1,0,2,1,0,1,3,2,1,2,1]
Output: 6
Explanation: 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.
Example 2:

Input: height = [4,2,0,3,2,5]
Output: 9

Wildcard Matching

Given an input string (s) and a pattern (p), implement wildcard pattern matching with support for '?' and '*' where:

'?' Matches any single character.
'*' Matches any sequence of characters (including the empty sequence).
The matching should cover the entire input string (not partial).

Example 1:

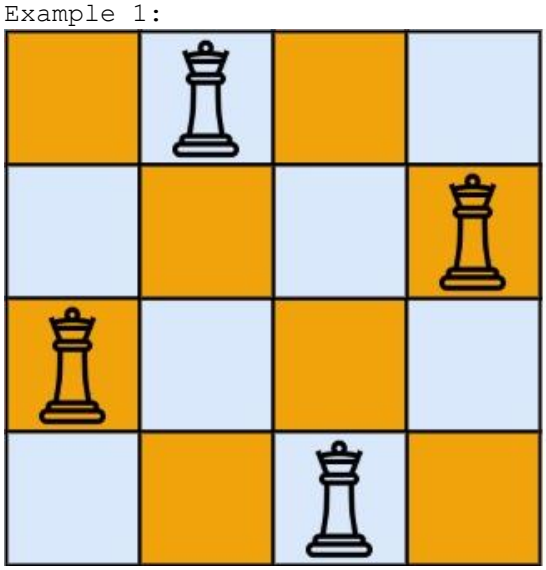
Input: s = "aa", p = "a"

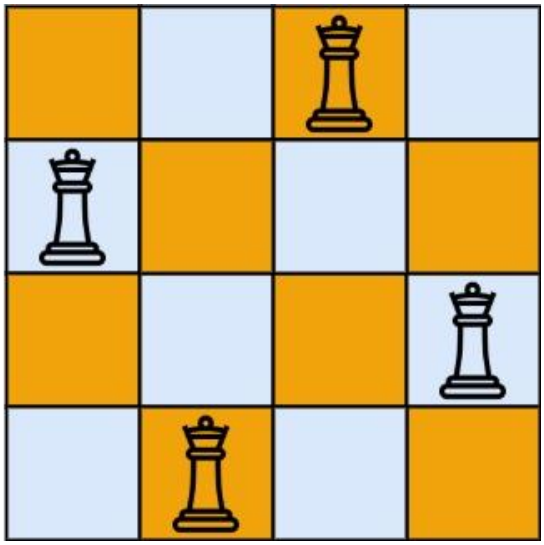
```
Output: false
Explanation: "a" does not match the entire string "aa".
Example 2:

Input: s = "aa", p = "*"
Output: true
Explanation: '*' matches any sequence.
Example 3:

Input: s = "cb", p = "?a"
Output: false
Explanation: '?' matches 'c', but the second letter is 'a', which does not
match 'b'.
```

N-Queens

```
The n-queens puzzle is the problem of placing n queens on an n x 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.
```

Example 1:





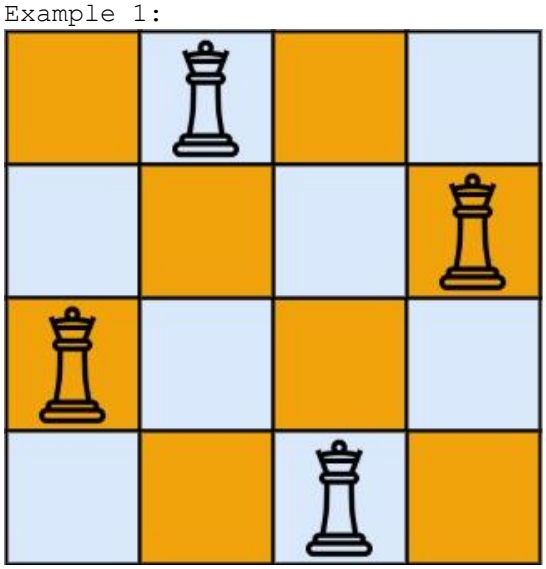
```
Input: n = 4
Output: [[".Q..","...Q","Q...","..Q."],["..Q.","Q...","...Q",".Q.."]]
Explanation: There exist two distinct solutions to the 4-queens puzzle as
shown above
Example 2:

Input: n = 1
Output: [["Q"]]
```

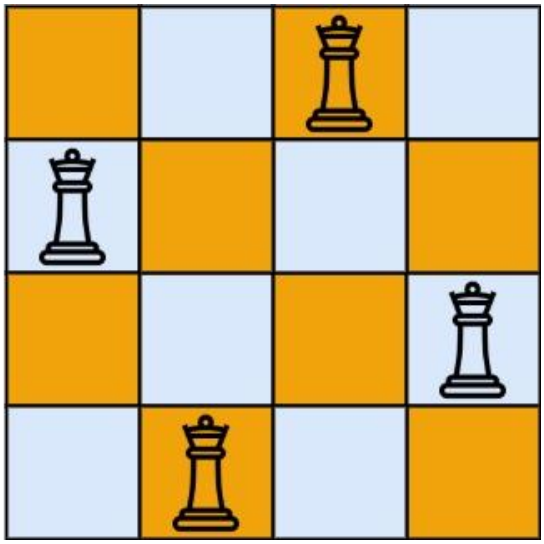
N-Queens II

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

Given an integer n, return the number of distinct solutions to the n-queens puzzle.

Example 1:





Input: n = 4
Output: 2
Explanation: There are two distinct solutions to the 4-queens puzzle as shown.
Example 2:

Input: n = 1
Output: 1

Permutation Sequence

The set [1, 2, 3, ..., n] contains a total of n! unique permutations.

By listing and labeling all of the permutations in order, we get the following sequence for n = 3:

"123"
"132"
"213"
"231"
"312"
"321"
Given n and k, return the kth permutation sequence.

Example 1:

Input: n = 3, k = 3
Output: "213"
Example 2:

Input: n = 4, k = 9

Output: "2314"
Example 3:

Input: n = 3, k = 1
Output: "123"

Valid Number

A valid number can be split up into these components (in order):

A decimal number or an integer.
(Optional) An 'e' or 'E', followed by an integer.
A decimal number can be split up into these components (in order):

(Optional) A sign character (either '+' or '-').
One of the following formats:
One or more digits, followed by a dot '.'.
One or more digits, followed by a dot '.', followed by one or more digits.
A dot '.', followed by one or more digits.
An integer can be split up into these components (in order):

(Optional) A sign character (either '+' or '-').
One or more digits.
For example, all the following are valid numbers: ["2", "0089", "-0.1", "+3.14", "4.", "-.9", "2e10", "-90E3", "3e+7", "+6e-1", "53.5e93", "-123.456e789"], while the following are not valid numbers: ["abc", "1a", "1e", "e3", "99e2.5", "--6", "-+3", "95a54e53"].

Given a string s, return true if s is a valid number.

Example 1:

Input: s = "0"
Output: true
Example 2:

Input: s = "e"
Output: false
Example 3:

Input: s = "."
Output: false

Text Justification

Given an array of strings words and a width maxWidth, format the text such that each line has exactly maxWidth characters and is fully (left and right) justified.

You should pack your words in a greedy approach; that is, pack as many words as you can in each line. Pad extra spaces ' ' when necessary so that each line has exactly maxWidth characters.

Extra spaces between words should be distributed as evenly as possible. If the number of spaces on a line does not divide evenly between words, the empty slots on the left will be assigned more spaces than the slots on the right.

For the last line of text, it should be left-justified, and no extra space is inserted between words.

Note:

A word is defined as a character sequence consisting of non-space characters only.
Each word's length is guaranteed to be greater than 0 and not exceed maxWidth.
The input array words contains at least one word.

Example 1:

Input: words = ["This", "is", "an", "example", "of", "text", "justification."], maxWidth = 16
Output:
[
   "This   is   an",
   "example of text",
   "justification. "
]
Example 2:

Input: words = ["What","must","be","acknowledgment","shall","be"], maxWidth = 16
Output:
[
  "What   must   be",
  "acknowledgment ",
  "shall be   "
]
Explanation: Note that the last line is "shall be   " instead of "shall be", because the last line must be left-justified instead of fully-justified.
Note that the second line is also left-justified because it contains only one word.
Example 3:

Input: words = ["Science","is","what","we","understand","well","enough","to","explain","to","a","computer.","Art","is","everything","else","we","do"], maxWidth = 20
Output:
[
  "Science is what we",
  "understand   well",
  "enough to explain to"

Edit Distance

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

Example 1:

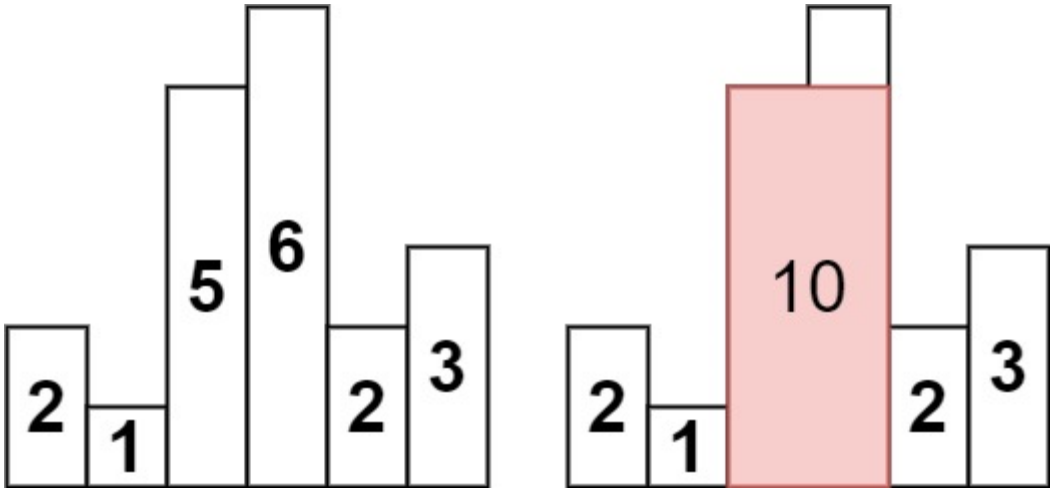
```
Input: word1 = "horse", word2 = "ros"
Output: 3
Explanation:
horse -> rorse (replace 'h' with 'r')
rorse -> rose (remove 'r')
rose -> ros (remove 'e')
Example 2:

Input: word1 = "intention", word2 = "execution"
Output: 5
Explanation:
intention -> inention (remove 't')
inention -> enention (replace 'i' with 'e')
enention -> exention (replace 'n' with 'x')
exention -> exection (replace 'n' with 'c')
exection -> execution (insert 'u')
```

Minimum Window Substring

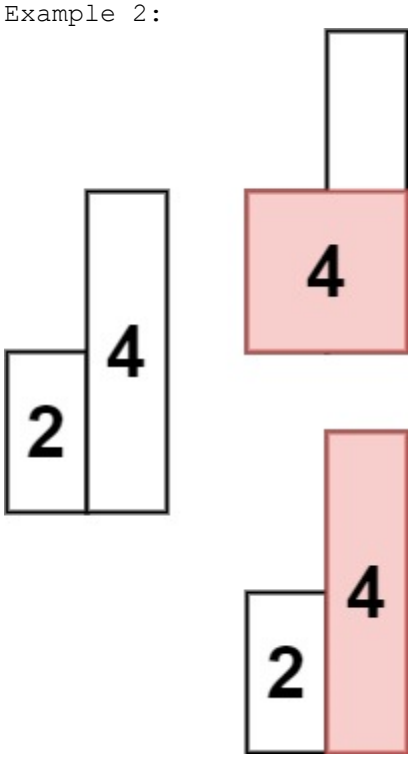
```
Given two strings s and t of lengths m and n respectively, return the
minimum window
substring
 of s such that every character in t (including duplicates) is included in
the window. If there is no such substring, return the empty string "".

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



Example 1:

Input: s = "ADOBECODEBANC", t = "ABC"
Output: "BANC"
Explanation: The minimum window substring "BANC" includes 'A', 'B', and 'C'
from string t.
Example 2:

Input: s = "a", t = "a"
Output: "a"
Explanation: The entire string s is the minimum window.
Example 3:

Input: s = "a", t = "aa"
Output: ""
Explanation: Both 'a's from t must be included in the window.
Since the largest window of s only has one 'a', return empty string.
```

Largest Rectangle in Histogram

```
Given an array of integers heights representing the histogram's bar height
where the width of each bar is 1, return the area of the largest rectangle
in the histogram.



Example 1:
```




```
Input: heights = [2,1,5,6,2,3]
Output: 10
Explanation: The above is a histogram where width of each bar is 1.
The largest rectangle is shown in the red area, which has an area = 10
units.
Example 2:
```



Scramble String

```
We can scramble a string s to get a string t using the following algorithm:

If the length of the string is 1, stop.
If the length of the string is > 1, do the following:
Split the string into two non-empty substrings at a random index, i.e., if
the string is s, divide it to x and y where s = x + y.
Randomly decide to swap the two substrings or to keep them in the same
order. i.e., after this step, s may become s = x + y or s = y + x.
Apply step 1 recursively on each of the two substrings x and y.
Given two strings s1 and s2 of the same length, return true if s2 is a
scrambled string of s1, otherwise, return false.
```

```
Example 1:

Input: s1 = "great", s2 = "rgeat"
Output: true
Explanation: One possible scenario applied on s1 is:
"great" --> "gr/eat" // divide at random index.
"gr/eat" --> "gr/eat" // random decision is not to swap the two substrings
and keep them in order.
"gr/eat" --> "g/r / e/at" // apply the same algorithm recursively on both
substrings. divide at random index each of them.
"g/r / e/at" --> "r/g / e/at" // random decision was to swap the first
substring and to keep the second substring in the same order.
"r/g / e/at" --> "r/g / e/ a/t" // again apply the algorithm recursively,
divide "at" to "a/t".
"r/g / e/ a/t" --> "r/g / e/ a/t" // random decision is to keep both
substrings in the same order.
The algorithm stops now, and the result string is "rgeat" which is s2.
As one possible scenario led s1 to be scrambled to s2, we return true.
Example 2:

Input: s1 = "abcde", s2 = "caebd"
Output: false
Example 3:

Input: s1 = "a", s2 = "a"
Output: true
```

Distinct Subsequences

```
Given two strings s and t, return the number of distinct
subsequences
 of s which equals t.

The test cases are generated so that the answer fits on a 32-bit signed
integer.



Example 1:

Input: s = "rabbbit", t = "rabbit"
Output: 3
Explanation:
As shown below, there are 3 ways you can generate "rabbit" from s.
rabbbit
rabbbit
rabbbit
Example 2:

Input: s = "babgbag", t = "bag"
Output: 5
Explanation:
As shown below, there are 5 ways you can generate "bag" from s.
babgbag
babgbag
babgbag
babgbag
babgbag
```

Best Time to Buy and Sell Stock III

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

Find the maximum profit you can achieve. You may complete at most two transactions.

Note: You may not engage in multiple transactions simultaneously (i.e., you must sell the stock before you buy again).

Example 1:

Input: prices = [3,3,5,0,0,3,1,4]
Output: 6
Explanation: Buy on day 4 (price = 0) and sell on day 6 (price = 3), profit = 3-0 = 3.
Then buy on day 7 (price = 1) and sell on day 8 (price = 4), profit = 4-1 = 3.
Example 2:

Input: prices = [1,2,3,4,5]
Output: 4
Explanation: Buy on day 1 (price = 1) and sell on day 5 (price = 5), profit = 5-1 = 4.
Note that you cannot buy on day 1, buy on day 2 and sell them later, as you are engaging multiple transactions at the same time. You must sell before buying again.
Example 3:

Input: prices = [7,6,4,3,1]
Output: 0
Explanation: In this case, no transaction is done, i.e. max profit = 0.

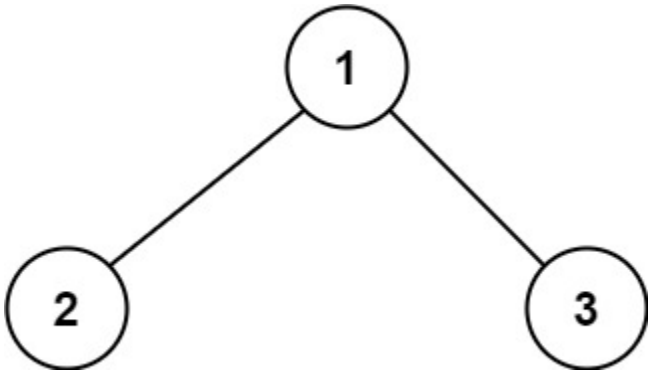
Binary Tree Maximum Path Sum

A path in a binary tree is a sequence of nodes where each pair of adjacent nodes in the sequence has an edge connecting them. A node can only appear in the sequence at most once. Note that the path does not need to pass through the root.

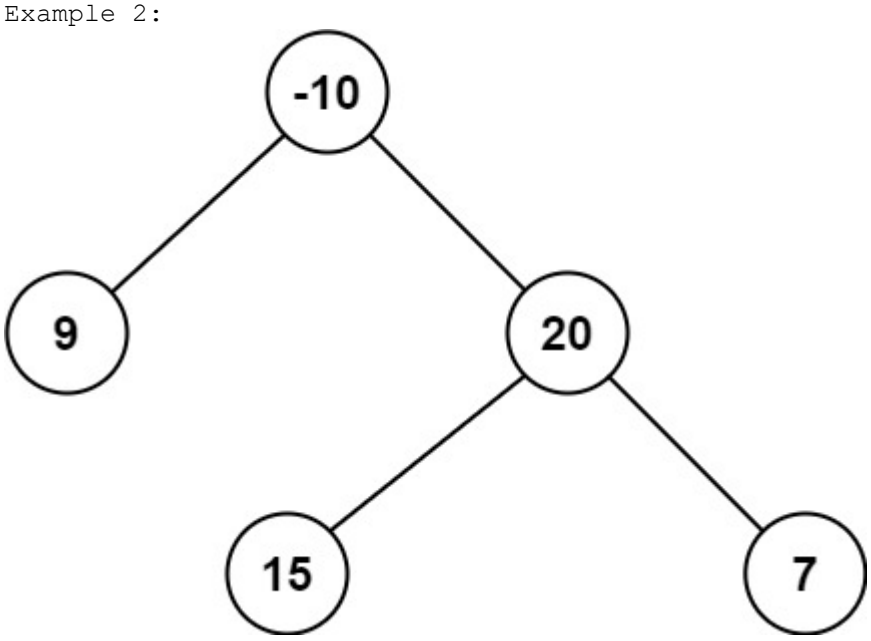
The path sum of a path is the sum of the node's values in the path.

Given the root of a binary tree, return the maximum path sum of any non-empty path.

Example 1:




```
Input: root = [1,2,3]
Output: 6
Explanation: The optimal path is 2 -> 1 -> 3 with a path sum of 2 + 1 + 3 =
6.
Example 2:
```



The Skyline Problem

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] = [lefti, righti, heighti]:

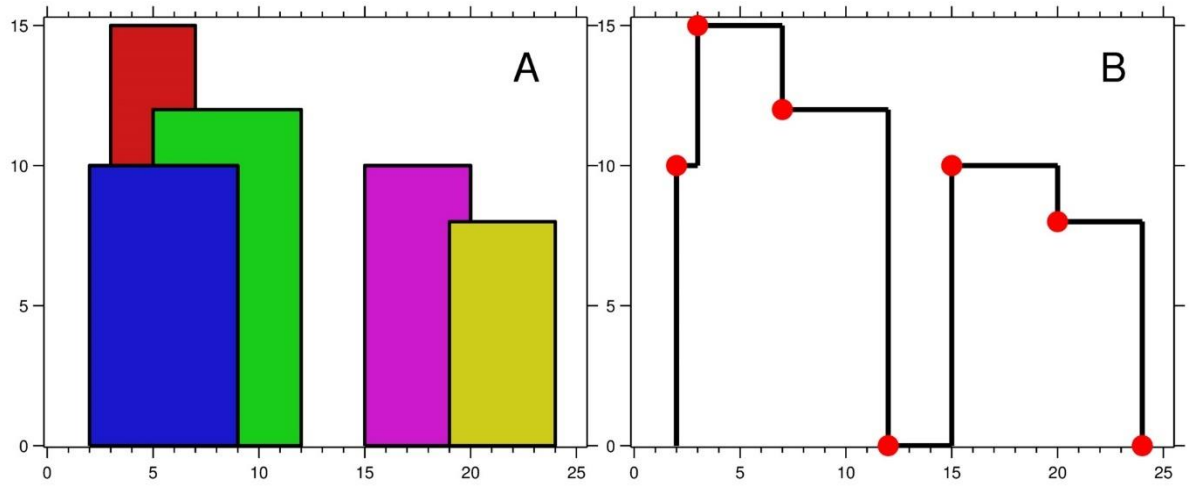
lefti is the x coordinate of the left edge of the ith building.
righti is the x coordinate of the right edge of the ith building.
heighti is the height of the ith building.
You may assume all buildings are perfect rectangles grounded on an absolutely flat surface at height 0.

The skyline should be represented as a list of "key points" sorted by their x-coordinate in the form [[x1,y1],[x2,y2],...]. Each key point is the left endpoint of some horizontal segment in the skyline except the last point in

the list, which always has a y-coordinate 0 and is used to mark the skyline's termination where the rightmost building ends. Any ground between the leftmost and rightmost buildings should be part of the skyline's contour.

Note: There must be no consecutive horizontal lines of equal height in the output skyline. For instance, [...,[2 3],[4 5],[7 5],[11 5],[12 7],...] is not acceptable; the three lines of height 5 should be merged into one in the final output as such: [...,[2 3],[4 5],[12 7],...]

Example 1:



Input: buildings = [[2,9,10],[3,7,15],[5,12,12],[15,20,10],[19,24,8]]
Output: [[2,10],[3,15],[7,12],[12,0],[15,10],[20,8],[24,0]]
Explanation:
Figure A shows the buildings of the input.
Figure B shows the skyline formed by those buildings. The red points in figure B represent the key points in the output list.

Contains Duplicate III

You are given an integer array nums and two integers indexDiff and valueDiff.

Find a pair of indices (i, j) such that:

i != j,
abs(i - j) <= indexDiff.
abs(nums[i] - nums[j]) <= valueDiff, and
Return true if such pair exists or false otherwise.

Example 1:

Input: nums = [1,2,3,1], indexDiff = 3, valueDiff = 0
Output: true
Explanation: We can choose (i, j) = (0, 3).
We satisfy the three conditions:
i != j --> 0 != 3
abs(i - j) <= indexDiff --> abs(0 - 3) <= 3
abs(nums[i] - nums[j]) <= valueDiff --> abs(1 - 1) <= 0

Example 2:

Input: nums = [1,5,9,1,5,9], indexDiff = 2, valueDiff = 3
Output: false
Explanation: After trying all the possible pairs (i, j), we cannot satisfy the three conditions, so we return false.

Basic Calculator

Given a string s representing a valid expression, implement a basic calculator to evaluate it, and return the result of the evaluation.

Note: You are not allowed to use any built-in function which evaluates strings as mathematical expressions, such as eval().

Example 1:

Input: s = "1 + 1"
Output: 2
Example 2:

Input: s = " 2-1 + 2 "
Output: 3
Example 3:

Input: s = "(1+(4+5+2)-3)+(6+8)"
Output: 23

Find Median from Data Stream

The median is the middle value in an ordered integer list. If the size of the list is even, there is no middle value, and the median is the mean of the two middle values.

For example, for arr = [2,3,4], the median is 3.
For example, for arr = [2,3], the median is (2 + 3) / 2 = 2.5.
Implement the MedianFinder class:

MedianFinder() initializes the MedianFinder object.
void addNum(int num) adds the integer num from the data stream to the data structure.
double findMedian() returns the median of all elements so far. Answers within 10-5 of the actual answer will be accepted.

Example 1:

Input
["MedianFinder", "addNum", "addNum", "findMedian", "addNum", "findMedian"]
[[], [1], [2], [], [3], []]
Output
[null, null, null, 1.5, null, 2.0]

Explanation
MedianFinder medianFinder = new MedianFinder();
medianFinder.addNum(1);   // arr = [1]
medianFinder.addNum(2);   // arr = [1, 2]
medianFinder.findMedian(); // return 1.5 (i.e., (1 + 2) / 2)

```
medianFinder.addNum(3);    // arr[1, 2, 3]
medianFinder.findMedian(); // return 2.0
```

Number of Digit One

Given an integer n, count the total number of digit 1 appearing in all non-negative integers less than or equal to n.

Example 1:

```
Input: n = 13
Output: 6
```

Example 2:

```
Input: n = 0
Output: 0
```

Sliding Window Maximum

You are given an array of integers nums, there is a sliding window of size k which is moving from the very left of the array to the very right. You can only see the k numbers in the window. Each time the sliding window moves right by one position.

Return the max sliding window.

Example 1:

```
Input: nums = [1,3,-1,-3,5,3,6,7], k = 3
Output: [3,3,5,5,6,7]
Explanation:
Window position                Max
---------------               -----
[1  3  -1] -3  5  3  6  7       3
 1 [3  -1  -3] 5  3  6  7       3
 1  3 [-1  -3  5] 3  6  7       5
 1  3  -1 [-3  5  3] 6  7       5
 1  3  -1  -3 [5  3  6] 7       6
 1  3  -1  -3  5 [3  6  7]      7
```

Example 2:

```
Input: nums = [1], k = 1
Output: [1]
```

Strobogrammatic Number III

Given two strings low and high that represent two integers low and high where low <= high, return the number of strobogrammatic numbers in the range [low, high].
A strobogrammatic number is a number that looks the same when rotated 180 degrees (looked at upside down).

Example 1:

```
Input: low = "50", high = "100"
Output: 3
```

```
Example 2:
Input: low = "0", high = "0"
Output: 1
```

Trips and Users

```
SQL Schema
Table: Trips

+-------------+----------+
| Column Name | Type     |
+-------------+----------+
| id          | int      |
| client_id   | int      |
| driver_id   | int      |
| city_id     | int      |
| status      | enum     |
| request_at  | date     |
+-------------+----------+
id is the primary key for this table.
The table holds all taxi trips. Each trip has a unique id, while client_id
and driver_id are foreign keys to the users_id at the Users table.
Status is an ENUM type of ('completed', 'cancelled_by_driver',
'cancelled_by_client').


Table: Users

+-------------+----------+
| Column Name | Type     |
+-------------+----------+
| users_id    | int      |
| banned      | enum     |
| role        | enum     |
+-------------+----------+
users_id is the primary key for this table.
The table holds all users. Each user has a unique users_id, and role is an
ENUM type of ('client', 'driver', 'partner').
banned is an ENUM type of ('Yes', 'No').


The cancellation rate is computed by dividing the number of canceled (by
client or driver) requests with unbanned users by the total number of
requests with unbanned users on that day.

Write a SQL query to find the cancellation rate of requests with unbanned
users (both client and driver must not be banned) each day between "2013-
10-01" and "2013-10-03". Round Cancellation Rate to two decimal points.

Return the result table in any order.

The query result format is in the following example.



Example 1:
```

Paint House II

There is a row of m houses in a small city, each house must be painted with one of the n colors (labeled from 1 to n), some houses that have been painted last summer should not be painted again.
A neighborhood is a maximal group of continuous houses that are painted with the same color.

- For example: houses = [1,2,2,3,3,2,1,1] contains 5 neighborhoods [{1}, {2,2}, {3,3}, {2}, {1,1}].

Given an array houses, an m x n matrix cost and an integer target where:

- houses[i]: is the color of the house i, and 0 if the house is not painted yet.
- cost[i][j]: is the cost of paint the house i with the color j + 1.

Return the minimum cost of painting all the remaining houses in such a way that there are exactly target neighborhoods. If it is not possible, return -1.

Example 1:
Input: houses = [0,0,0,0,0], cost = [[1,10],[10,1],[10,1],[1,10],[5,1]], m = 5, n = 2, target = 3
Output: 9
Explanation: Paint houses of this way [1,2,2,1,1]
This array contains target = 3 neighborhoods, [{1}, {2,2}, {1,1}].
Cost of paint all houses (1 + 1 + 1 + 1 + 5) = 9.

Example 2:
Input: houses = [0,2,1,2,0], cost = [[1,10],[10,1],[10,1],[1,10],[5,1]], m = 5, n = 2, target = 3
Output: 11
Explanation: Some houses are already painted, Paint the houses of this way [2,2,1,2,2]
This array contains target = 3 neighborhoods, [{2,2}, {1}, {2,2}].
Cost of paint the first and last house (10 + 1) = 11.

Example 3:
Input: houses = [3,1,2,3], cost = [[1,1,1],[1,1,1],[1,1,1],[1,1,1]], m = 4, n = 3, target = 3
Output: -1
Explanation: Houses are already painted with a total of 4 neighborhoods [{3},{1},{2},{3}] different of target = 3.

Alien Dictionary

In an alien language, surprisingly, they also use English lowercase letters, but possibly in a different order. The order of the alphabet is some permutation of lowercase letters.
Given a sequence of words written in the alien language, and the order of the alphabet, return true if and only if the given words are sorted lexicographically in this alien language.

Example 1:
Input: words = ["hello","leetcode"], order = "hlabcdefgijkmnopqrstuvwxyz"
Output: true
Explanation: As 'h' comes before 'l' in this language, then the sequence is sorted.

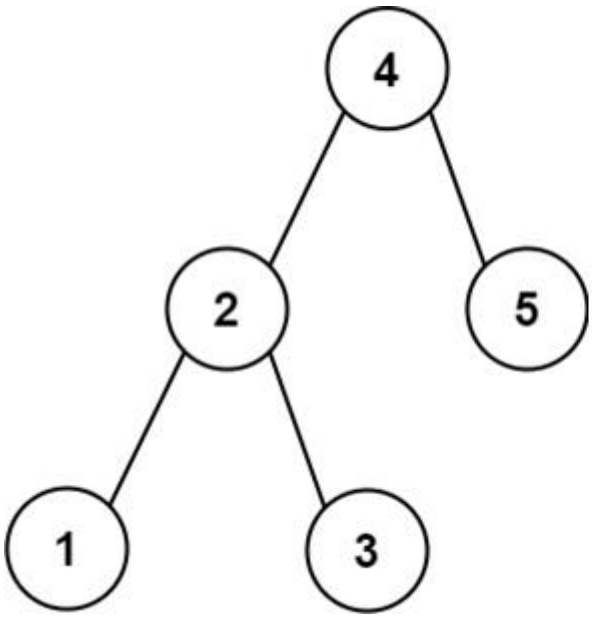
Example 2:
Input: words = ["word","world","row"], order = "worldabcefghijkmnpqstuvxyz"
Output: false
Explanation: As 'd' comes after 'l' in this language, then words[0] > words[1], hence the sequence is unsorted.

Example 3:

```
Input: words = ["apple","app"], order = "abcdefghijklmnopqrstuvwxyz"
Output: false
Explanation: The first three characters "app" match, and the second string
is shorter (in size.) According to lexicographical rules "apple" > "app",
because 'l' > 'Ø', where 'Ø' is defined as the blank character which is
less than any other character (More info).
```

Closest Binary Search Tree Value II

```
Given the root of a binary search tree, a target value, and an integer k,
return the k values in the BST that are closest to the target. You may
return the answer in any order.
You are guaranteed to have only one unique set of k values in the BST that
are closest to the target.

Example 1:
```




```
Input: root = [4,2,5,1,3], target = 3.714286, k = 2
Output: [4,3]

Example 2:
Input: root = [1], target = 0.000000, k = 1
Output: [1]
```

Integer to English Words

```
Convert a non-negative integer num to its English words representation.



Example 1:

Input: num = 123
Output: "One Hundred Twenty Three"
Example 2:

Input: num = 12345
Output: "Twelve Thousand Three Hundred Forty Five"
Example 3:
```

```
Input: num = 1234567
Output: "One Million Two Hundred Thirty Four Thousand Five Hundred Sixty
Seven"
```

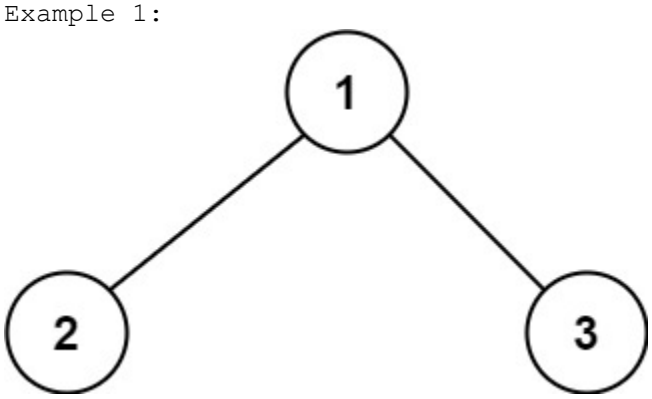
Binary Tree Maximum Path Sum

A path in a binary tree is a sequence of nodes where each pair of adjacent nodes in the sequence has an edge connecting them. A node can only appear in the sequence at most once. Note that the path does not need to pass through the root.

The path sum of a path is the sum of the node's values in the path.

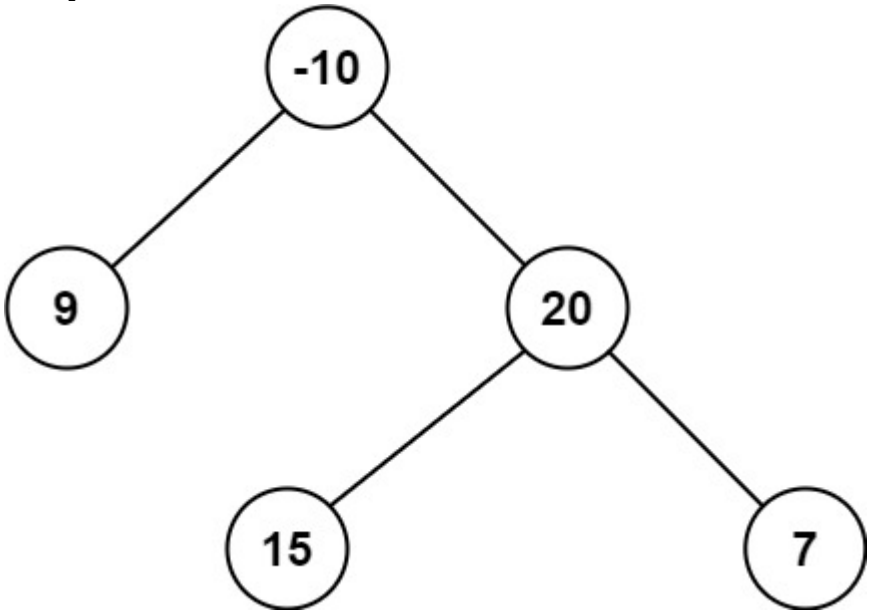
Given the root of a binary tree, return the maximum path sum of any non-empty path.

Example 1:




```
Input: root = [1,2,3]
Output: 6
Explanation: The optimal path is 2 -> 1 -> 3 with a path sum of 2 + 1 + 3 =
6.
```

Example 2:



Word Ladder II

A transformation sequence from word beginWord to word endWord using a dictionary wordList is a sequence of words beginWord -> s1 -> s2 -> ... -> sk such that:

Every adjacent pair of words differs by a single letter.
Every si for 1 <= i <= k is in wordList. Note that beginWord does not need to be in wordList.
sk == endWord
Given two words, beginWord and endWord, and a dictionary wordList, return all the shortest transformation sequences from beginWord to endWord, or an empty list if no such sequence exists. Each sequence should be returned as a list of the words [beginWord, s1, s2, ..., sk].

Example 1:

Input: beginWord = "hit", endWord = "cog", wordList = ["hot","dot","dog","lot","log","cog"]
Output: [["hit","hot","dot","dog","cog"],["hit","hot","lot","log","cog"]]
Explanation: There are 2 shortest transformation sequences:
"hit" -> "hot" -> "dot" -> "dog" -> "cog"
"hit" -> "hot" -> "lot" -> "log" -> "cog"
Example 2:

Input: beginWord = "hit", endWord = "cog", wordList = ["hot","dot","dog","lot","log"]
Output: []
Explanation: The endWord "cog" is not in wordList, therefore there is no valid transformation sequence.

Word Ladder

A transformation sequence from word beginWord to word endWord using a dictionary wordList is a sequence of words beginWord -> s1 -> s2 -> ... -> sk such that:

Every adjacent pair of words differs by a single letter.
Every si for 1 <= i <= k is in wordList. Note that beginWord does not need to be in wordList.
sk == endWord
Given two words, beginWord and endWord, and a dictionary wordList, return the number of words in the shortest transformation sequence from beginWord to endWord, or 0 if no such sequence exists.

Example 1:

Input: beginWord = "hit", endWord = "cog", wordList = ["hot","dot","dog","lot","log","cog"]
Output: 5
Explanation: One shortest transformation sequence is "hit" -> "hot" -> "dot" -> "dog" -> cog", which is 5 words long.
Example 2:

Input: beginWord = "hit", endWord = "cog", wordList = ["hot","dot","dog","lot","log"]
Output: 0
Explanation: The endWord "cog" is not in wordList, therefore there is no valid transformation sequence.

Palindrome Partitioning II

Given a string s, partition s such that every
substring
 of the partition is a
palindrome
.

Return the minimum cuts needed for a palindrome partitioning of s.

Example 1:

Input: s = "aab"
Output: 1
Explanation: The palindrome partitioning ["aa","b"] could be produced using
1 cut.
Example 2:

Input: s = "a"
Output: 0
Example 3:

Input: s = "ab"
Output: 1

Candy

There are n children standing in a line. Each child is assigned a rating value given in the integer array ratings.

You are giving candies to these children subjected to the following requirements:

Each child must have at least one candy.
Children with a higher rating get more candies than their neighbors.
Return the minimum number of candies you need to have to distribute the candies to the children.

Example 1:

Input: ratings = [1,0,2]
Output: 5
Explanation: You can allocate to the first, second and third child with 2,
1, 2 candies respectively.
Example 2:

Input: ratings = [1,2,2]
Output: 4
Explanation: You can allocate to the first, second and third child with 1,
2, 1 candies respectively.
The third child gets 1 candy because it satisfies the above two conditions.

Word Break II

Given a string s and a dictionary of strings wordDict, add spaces in s to construct a sentence where each word is a valid dictionary word. Return all such possible sentences in any order.

Note that the same word in the dictionary may be reused multiple times in the segmentation.

Example 1:

Input: s = "catsanddog", wordDict = ["cat","cats","and","sand","dog"]
Output: ["cats and dog","cat sand dog"]
Example 2:

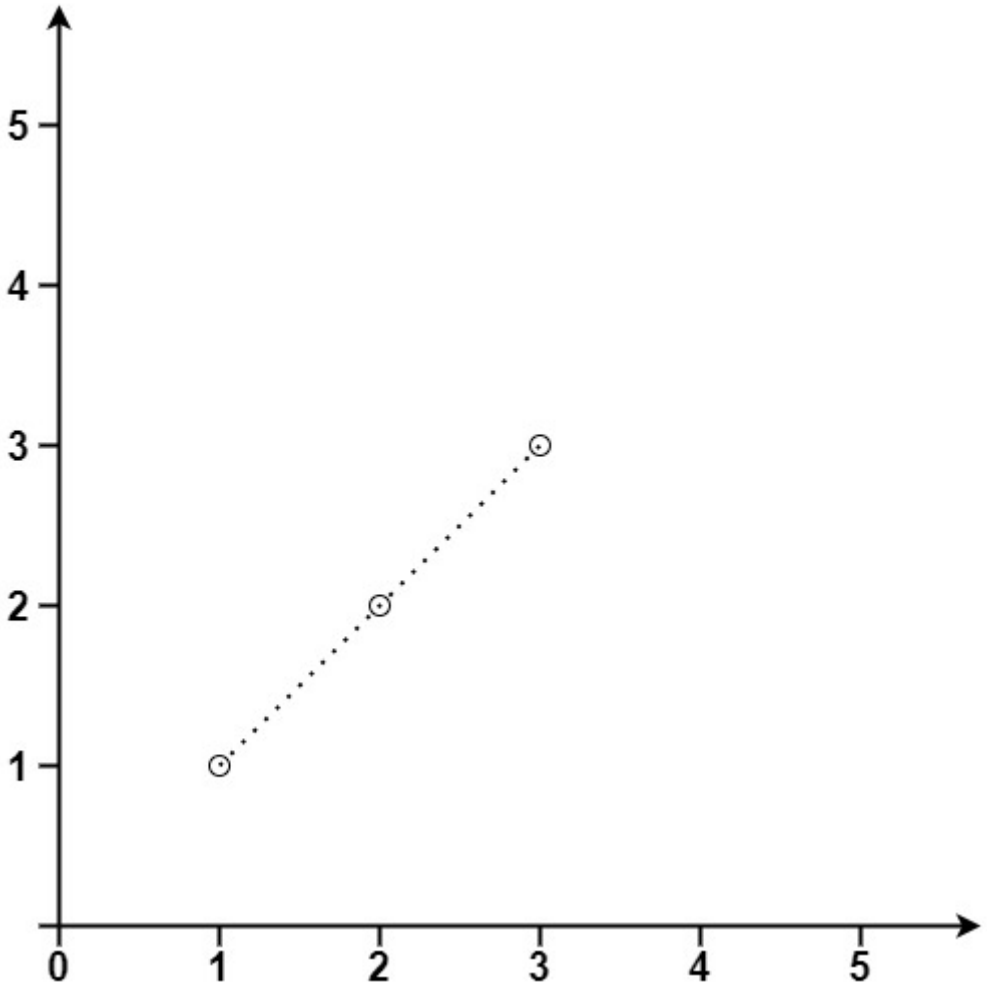
Input: s = "pineapplepenapple", wordDict =
["apple","pen","applepen","pine","pineapple"]
Output: ["pine apple pen apple","pineapple pen apple","pine applepen apple"]
Explanation: Note that you are allowed to reuse a dictionary word.
Example 3:

Input: s = "catsandog", wordDict = ["cats","dog","sand","and","cat"]
Output: []

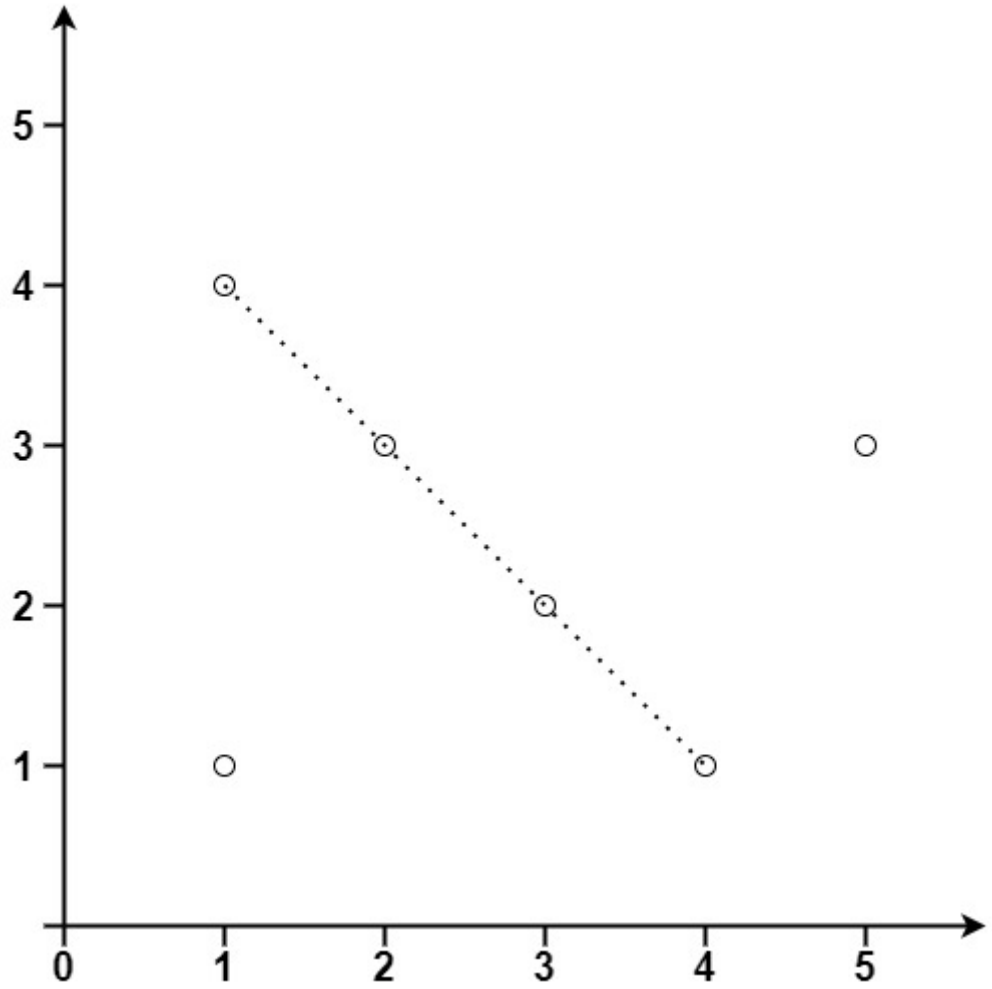
Max Points on a Line

Given an array of points where points[i] = [xi, yi] represents a point on the X-Y plane, return the maximum number of points that lie on the same straight line.

Example 1:




```
Input: points = [[1,1],[2,2],[3,3]]
Output: 3
Example 2:
```



Find Minimum in Rotated Sorted Array II

Suppose an array of length n sorted in ascending order is rotated between 1 and n times. For example, the array nums = [0,1,4,4,5,6,7] might become:

[4,5,6,7,0,1,4] if it was rotated 4 times.
[0,1,4,4,5,6,7] if it was rotated 7 times.
Notice that rotating an array [a[0], a[1], a[2], ..., a[n-1]] 1 time results in the array [a[n-1], a[0], a[1], a[2], ..., a[n-2]].

Given the sorted rotated array nums that may contain duplicates, return the minimum element of this array.

You must decrease the overall operation steps as much as possible.

Example 1:

Input: nums = [1,3,5]
Output: 1
Example 2:

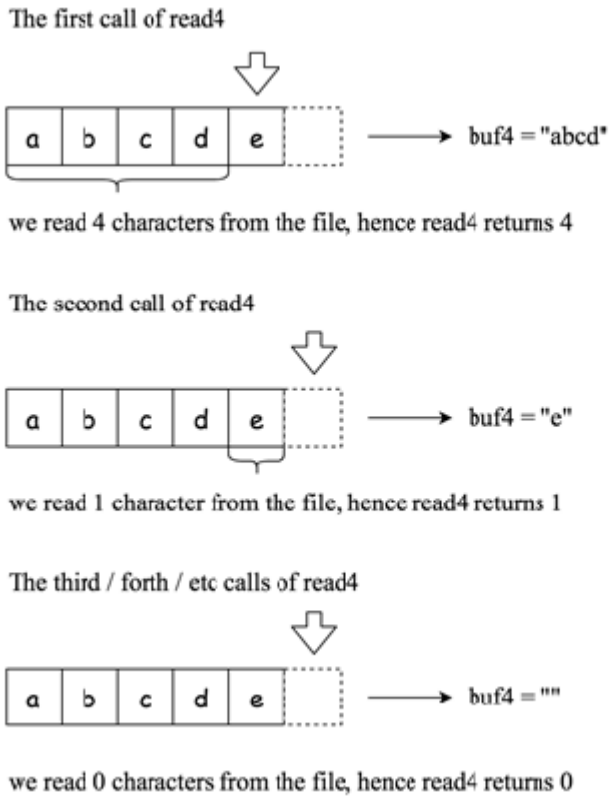
Input: nums = [2,2,2,0,1]
Output: 0

Read N Characters Given read4 II - Call Multiple T

Given a file and assume that you can only read the file using a given method read4, implement a method read to read n characters. Your method read may be called multiple times.
Method read4:
The API read4 reads four consecutive characters from file, then writes those characters into the buffer array buf4.
The return value is the number of actual characters read.
Note that read4() has its own file pointer, much like FILE *fp in C.
Definition of read4:
   Parameter: char[] buf4
    Returns: int

buf4[] is a destination, not a source. The results from read4 will be copied to buf4[].

Below is a high-level example of how read4 works:




```
File file("abcde"); // File is "abcde", initially file pointer (fp) points to 'a'
char[] buf4 = new char[4]; // Create buffer with enough space to store characters
read4(buf4); // read4 returns 4. Now buf4 = "abcd", fp points to 'e'
read4(buf4); // read4 returns 1. Now buf4 = "e", fp points to end of file
read4(buf4); // read4 returns 0. Now buf4 = "", fp points to end of file
```

Method read:
By using the read4 method, implement the method read that reads n characters from file and store it in the buffer array buf. Consider that you cannot manipulate file directly.
The return value is the number of actual characters read.

Definition of read:

Maximum Gap

Given an integer array nums, return the maximum difference between two successive elements in its sorted form. If the array contains less than two elements, return 0.

You must write an algorithm that runs in linear time and uses linear extra space.

Example 1:

Input: nums = [3,6,9,1]
Output: 3
Explanation: The sorted form of the array is [1,3,6,9], either (3,6) or (6,9) has the maximum difference 3.
Example 2:

Input: nums = [10]
Output: 0
Explanation: The array contains less than 2 elements, therefore return 0.

Dungeon Game

The demons had captured the princess and imprisoned her in the bottom-right corner of a dungeon. The dungeon consists of m x n rooms laid out in a 2D grid. Our valiant knight was initially positioned in the top-left room and must fight his way through dungeon to rescue the princess.

The knight has an initial health point represented by a positive integer. If at any point his health point drops to 0 or below, he dies immediately.

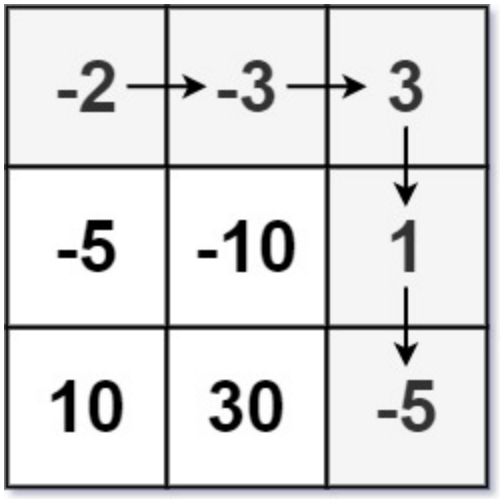
Some of the rooms are guarded by demons (represented by negative integers), so the knight loses health upon entering these rooms; other rooms are either empty (represented as 0) or contain magic orbs that increase the knight's health (represented by positive integers).

To reach the princess as quickly as possible, the knight decides to move only rightward or downward in each step.

Return the knight's minimum initial health so that he can rescue the princess.

Note that any room can contain threats or power-ups, even the first room the knight enters and the bottom-right room where the princess is imprisoned.

Example 1:




```
Input: dungeon = [[-2,-3,3],[-5,-10,1],[10,30,-5]]
Output: 7
Explanation: The initial health of the knight must be at least 7 if he
follows the optimal path: RIGHT-> RIGHT -> DOWN -> DOWN.
Example 2:

Input: dungeon = [[0]]
Output: 1
```

Department Top Three Salaries

```
SQL Schema
Table: Employee

+--------------+---------+
| Column Name  | Type    |
+--------------+---------+
| id           | int     |
| name         | varchar |
| salary       | int     |
| departmentId | int     |
+--------------+---------+
id is the primary key column for this table.
departmentId is a foreign key of the ID from the Department table.
Each row of this table indicates the ID, name, and salary of an employee.
It also contains the ID of their department.


Table: Department

+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| id          | int     |
| name        | varchar |
+-------------+---------+
id is the primary key column for this table.
Each row of this table indicates the ID of a department and its name.


A company's executives are interested in seeing who earns the most money in
each of the company's departments. A high earner in a department is an
```

employee who has a salary in the top three unique salaries for that department.

Write an SQL query to find the employees who are high earners in each of the departments.

Return the result table in any order.

The query result format is in the following example.

Example 1:

Input:
Employee table:
+----+-------+--------+--------------+

Best Time to Buy and Sell Stock IV

You are given an integer array prices where prices[i] is the price of a given stock on the ith day, and an integer k.

Find the maximum profit you can achieve. You may complete at most k transactions: i.e. you may buy at most k times and sell at most k times.

Note: You may not engage in multiple transactions simultaneously (i.e., you must sell the stock before you buy again).

Example 1:

Input: k = 2, prices = [2,4,1]
Output: 2
Explanation: Buy on day 1 (price = 2) and sell on day 2 (price = 4), profit = 4-2 = 2.
Example 2:

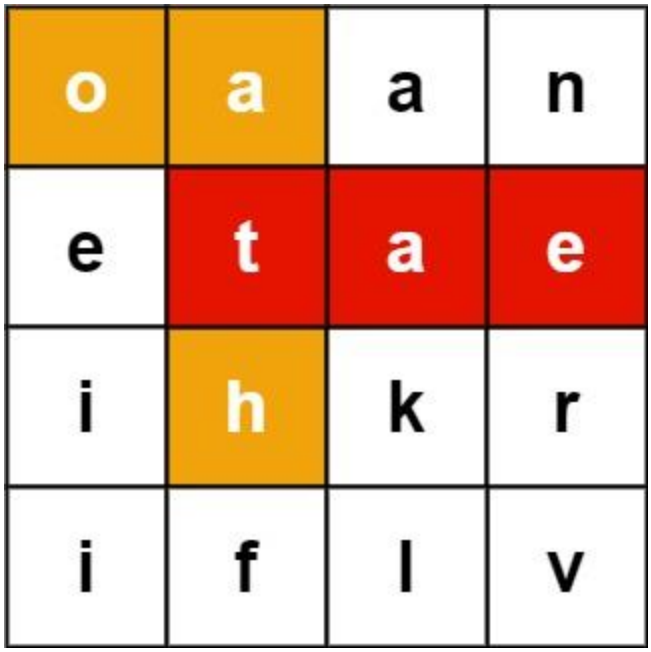
Input: k = 2, prices = [3,2,6,5,0,3]
Output: 7
Explanation: Buy on day 2 (price = 2) and sell on day 3 (price = 6), profit = 6-2 = 4. Then buy on day 5 (price = 0) and sell on day 6 (price = 3), pr

Word Search II

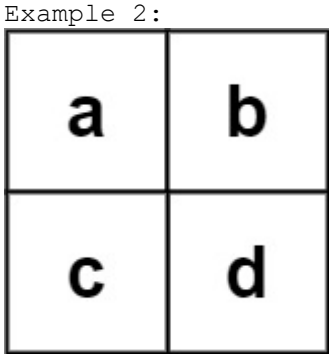
Given an m x n board of characters and a list of strings words, return all words on the board.

Each word must be constructed from letters of sequentially adjacent cells, where adjacent cells are horizontally or vertically neighboring. The same letter cell may not be used more than once in a word.

Example 1:




```
Input: board =
[["o","a","a","n"],["e","t","a","e"],["i","h","k","r"],["i","f","l","v"]],
words = ["oath","pea","eat","rain"]
Output: ["eat","oath"]
Example 2:
```



Expression Add Operators

Given a string num that contains only digits and an integer target, return all possibilities to insert the binary operators '+', '-', and/or '*' between the digits of num so that the resultant expression evaluates to the target value.

Note that operands in the returned expressions should not contain leading zeros.

Example 1:

```
Input: num = "123", target = 6
Output: ["1*2*3","1+2+3"]
Explanation: Both "1*2*3" and "1+2+3" evaluate to 6.
Example 2:

Input: num = "232", target = 8
Output: ["2*3+2","2+3*2"]
```

```
Explanation: Both "2*3+2" and "2+3*2" evaluate to 8.
Example 3:

Input: num = "3456237490", target = 9191
Output: []
Explanation: There are no expressions that can be created from "3456237490"
to evaluate to 9191.
```

Shortest Palindrome

```
You are given a string s. You can convert s to a
palindrome
 by adding characters in front of it.

Return the shortest palindrome you can find by performing this
transformation.



Example 1:

Input: s = "aacecaaa"
Output: "aaacecaaa"
Example 2:

Input: s = "abcd"
Output: "dcbabcd"
```

Medium Questions

Count Ways To Build Good Strings

```
Given the integers zero, one, low, and high, we can construct a string by
starting with an empty string, and then at each step perform either of the
following:

Append the character '0' zero times.
Append the character '1' one times.
This can be performed any number of times.

A good string is a string constructed by the above process having a length
between low and high (inclusive).

Return the number of different good strings that can be constructed
satisfying these properties. Since the answer can be large, return it
modulo 109 + 7.



Example 1:

Input: low = 3, high = 3, zero = 1, one = 1
Output: 8
Explanation:
One possible valid good string is "011".
It can be constructed as follows: "" -> "0" -> "01" -> "011".
All binary strings from "000" to "111" are good strings in this example.
Example 2:
```

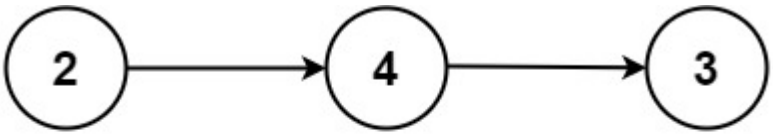
```
Input: low = 2, high = 3, zero = 1, one = 2
Output: 5
Explanation: The good strings are "00", "11", "000", "110", and "011".
```

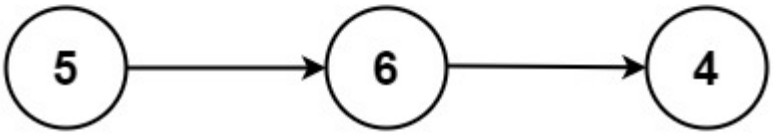
Add Two Numbers

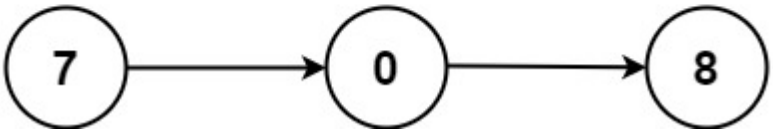
You are given two non-empty linked lists representing two non-negative integers. The digits are stored in reverse order, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

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

Example 1:








```
Input: l1 = [2,4,3], l2 = [5,6,4]
Output: [7,0,8]
Explanation: 342 + 465 = 807.
Example 2:

Input: l1 = [0], l2 = [0]
Output: [0]
Example 3:

Input: l1 = [9,9,9,9,9,9,9], l2 = [9,9,9,9]
Output: [8,9,9,9,0,0,0,1]
```

Longest Substring Without Repeating Characters

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

```
Example 1:

Input: s = "abcabcbb"
Output: 3
Explanation: The answer is "abc", with the length of 3.
Example 2:

Input: s = "bbbbb"
Output: 1
Explanation: The answer is "b", with the length of 1.
Example 3:

Input: s = "pwwkew"
Output: 3
Explanation: 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.
```

Longest Palindromic Substring

```
Given a string s, return the longest
palindromic

substring
 in s.



Example 1:

Input: s = "babad"
Output: "bab"
Explanation: "aba" is also a valid answer.
Example 2:

Input: s = "cbbd"
Output: "bb"
```

Zigzag Conversion

```
The string "PAYPALISHIRING" is written in a zigzag pattern on a given
number of rows like this: (you may want to display this pattern in a fixed
font for better legibility)

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:

string convert(string s, int numRows);


Example 1:

Input: s = "PAYPALISHIRING", numRows = 3
Output: "PAHNAPLSIIGYIR"
Example 2:

Input: s = "PAYPALISHIRING", numRows = 4
```

```
Output: "PINALSIGYAHRPI"
Explanation:
P     I    N
A   L S  I G
Y A   H R
P     I
Example 3:

Input: s = "A", numRows = 1
Output: "A"
```

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
[-231, 231 - 1], then return 0.

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



Example 1:

Input: x = 123
Output: 321
Example 2:

Input: x = -123
Output: -321
Example 3:

Input: x = 120
Output: 21
```

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:

Read in and ignore any leading whitespace.
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.
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.
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).
If the integer is out of the 32-bit signed integer range [-231, 231 - 1],
then clamp the integer so that it remains in the range. Specifically,
integers less than -231 should be clamped to -231, and integers greater
than 231 - 1 should be clamped to 231 - 1.
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.

Example 1:

```
Input: s = "42"
Output: 42
Explanation: The underlined characters are what is read in, the caret is the current reader position.
Step 1: "42" (no characters read because there is no leading whitespace)
         ^
Step 2: "42" (no characters read because there is neither a '-' nor '+')
         ^
Step 3: "42" ("42" is read in)
           ^
The parsed integer is 42.
Since 42 is in the range [-231, 231 - 1], the final result is 42.
```

Example 2:

```
Input: s = "   -42"
Output: -42
Explanation:
Step 1: "   -42" (leading whitespace is read and ignored)
            ^
Step 2: "   -42" ('-' is read, so the result should be negative)
             ^
Step 3: "   -42" ("42" is read in)
               ^
The parsed integer is -42.
Since -42 is in the range [-231, 231 - 1], the final result is -42.
```

Example 3:

Container With Most Water

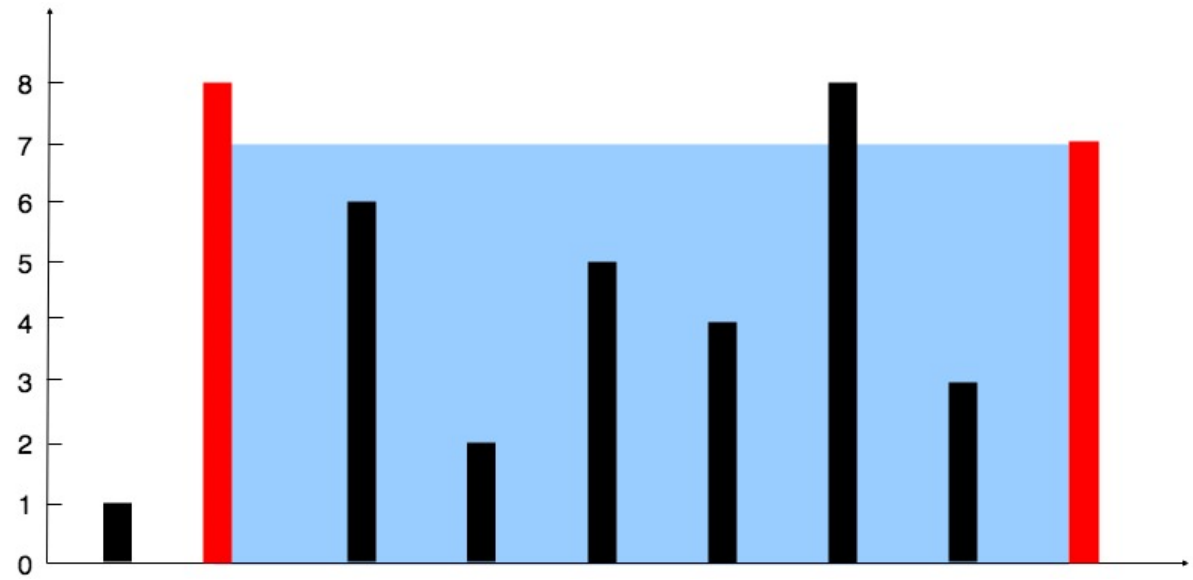
You are given an integer array height of length n. There are n vertical lines drawn such that the two endpoints of the ith line are (i, 0) and (i, height[i]).

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.

Example 1:




```
Input: height = [1,8,6,2,5,4,8,3,7]
Output: 49
Explanation: 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.
Example 2:

Input: height = [1,1]
Output: 1
```

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.



Example 1:
```

```
Input: num = 3
Output: "III"
Explanation: 3 is represented as 3 ones.
Example 2:

Input: num = 58
Output: "LVIII"
Explanation: L = 50, V = 5, III = 3.
Example 3:

Input: num = 1994
Output: "MCMXCIV"
Explanation: M = 1000, CM = 900, XC = 90 and IV = 4.
```

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.



Example 1:

Input: nums = [-1,0,1,2,-1,-4]
Output: [[-1,-1,2],[-1,0,1]]
Explanation:
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].
Notice that the order of the output and the order of the triplets does not
matter.
Example 2:

Input: nums = [0,1,1]
Output: []
Explanation: The only possible triplet does not sum up to 0.
Example 3:

Input: nums = [0,0,0]
Output: [[0,0,0]]
Explanation: The only possible triplet sums up to 0.
```

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.



Example 1:
```

```
Input: nums = [-1,2,1,-4], target = 1
Output: 2
Explanation: The sum that is closest to the target is 2. (-1 + 2 + 1 = 2).
Example 2:

Input: nums = [0,0,0], target = 1
Output: 0
Explanation: The sum that is closest to the target is 0. (0 + 0 + 0 = 0).
```

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.


```
Example 1:

Input: digits = "23"
Output: ["ad","ae","af","bd","be","bf","cd","ce","cf"]
Example 2:

Input: digits = ""
Output: []
```

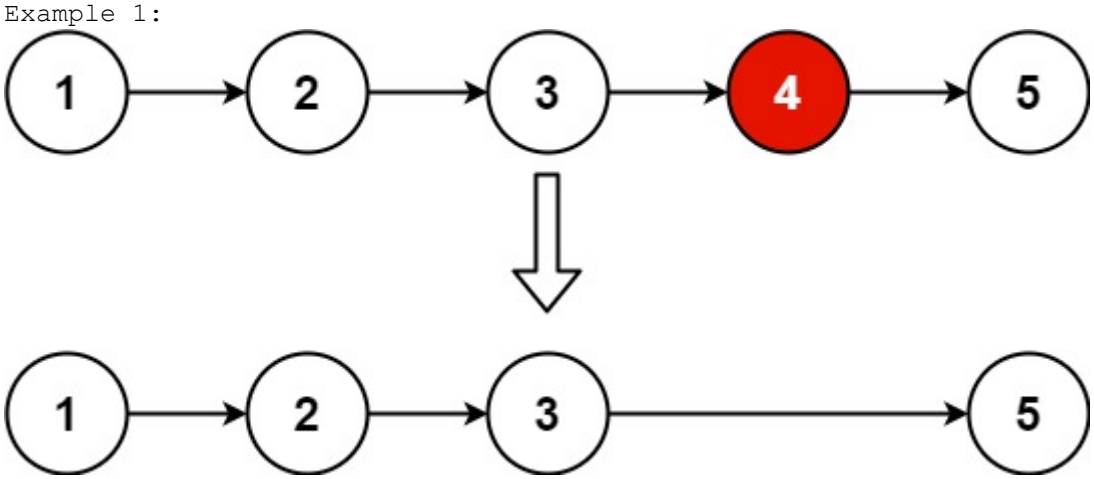
```
Example 3:

Input: digits = "2"
Output: ["a","b","c"]
```

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.



Example 1:

Input: nums = [1,0,-1,0,-2,2], target = 0
Output: [[-2,-1,1,2],[-2,0,0,2],[-1,0,0,1]]
Example 2:

Input: nums = [2,2,2,2,2], target = 8
Output: [[2,2,2,2]]
```

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.



Example 1:
```




```
Input: head = [1,2,3,4,5], n = 2
Output: [1,2,3,5]
Example 2:

Input: head = [1], n = 1
Output: []
Example 3:

Input: head = [1,2], n = 1
Output: [1]
```

Generate Parentheses

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

Example 1:

```
Input: n = 3
Output: ["((()))","(()())","(())()","()(())","()()()"]
```

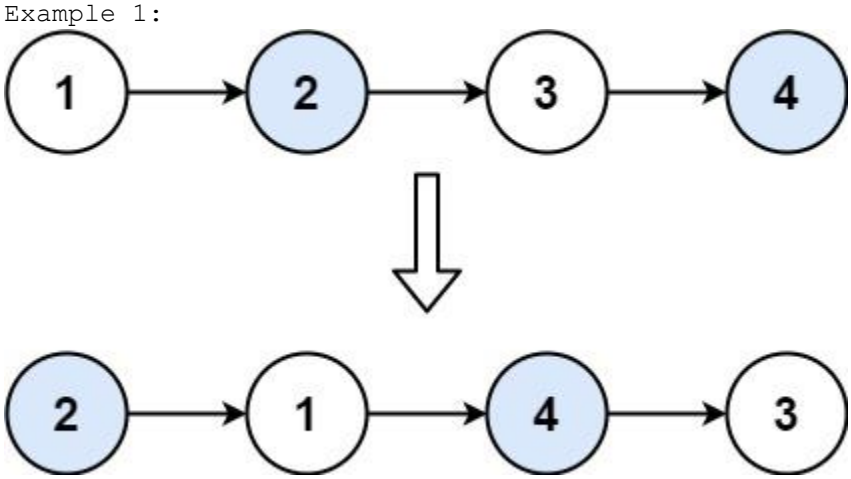
Example 2:

```
Input: n = 1
Output: ["()"]
```

Swap Nodes in Pairs

Given a linked list, swap every two adjacent nodes and return its head. You must solve the problem without modifying the values in the list's nodes (i.e., only nodes themselves may be changed.)

Example 1:




```
Input: head = [1,2,3,4]
Output: [2,1,4,3]
```

Example 2:

```
Input: head = []
Output: []
```

Example 3:

```
Input: head = [1]
Output: [1]
```

Divide Two Integers

Given two integers dividend and divisor, divide two integers without using multiplication, division, and mod operator.

The integer division should truncate toward zero, which means losing its fractional part. For example, 8.345 would be truncated to 8, and -2.7335 would be truncated to -2.

Return the quotient after dividing dividend by divisor.

Note: Assume we are dealing with an environment that could only store integers within the 32-bit signed integer range: [−231, 231 − 1]. For this problem, if the quotient is strictly greater than 231 - 1, then return 231 - 1, and if the quotient is strictly less than -231, then return -231.

Example 1:

Input: dividend = 10, divisor = 3
Output: 3
Explanation: 10/3 = 3.33333.. which is truncated to 3.
Example 2:

Input: dividend = 7, divisor = -3
Output: -2
Explanation: 7/-3 = -2.33333.. which is truncated to -2.

Next Permutation

A permutation of an array of integers is an arrangement of its members into a sequence or linear order.

For example, for arr = [1,2,3], the following are all the permutations of arr: [1,2,3], [1,3,2], [2, 1, 3], [2, 3, 1], [3,1,2], [3,2,1].
The next permutation of an array of integers is the next lexicographically greater permutation of its integer. More formally, if all the permutations of the array are sorted in one container according to their lexicographical order, then the next permutation of that array is the permutation that follows it in the sorted container. If such arrangement is not possible, the array must be rearranged as the lowest possible order (i.e., sorted in ascending order).

For example, the next permutation of arr = [1,2,3] is [1,3,2].
Similarly, the next permutation of arr = [2,3,1] is [3,1,2].
While the next permutation of arr = [3,2,1] is [1,2,3] because [3,2,1] does not have a lexicographical larger rearrangement.
Given an array of integers nums, find the next permutation of nums.

The replacement must be in place and use only constant extra memory.

Example 1:

Input: nums = [1,2,3]
Output: [1,3,2]
Example 2:

Input: nums = [3,2,1]
Output: [1,2,3]
Example 3:

Input: nums = [1,1,5]
Output: [1,5,1]

Search in Rotated Sorted Array

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

Prior to being passed to your function, nums is possibly rotated at an unknown pivot index k (1 <= k < nums.length) 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 rotated at pivot index 3 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 O(log n) runtime complexity.

Example 1:

Input: nums = [4,5,6,7,0,1,2], target = 0
Output: 4
Example 2:

Input: nums = [4,5,6,7,0,1,2], target = 3
Output: -1
Example 3:

Input: nums = [1], target = 0
Output: -1

Find First and Last Position of Element in Sorted

Given an array of integers nums sorted in non-decreasing order, find the starting and ending position of a given target value.

If target is not found in the array, return [-1, -1].

You must write an algorithm with O(log n) runtime complexity.

Example 1:

Input: nums = [5,7,7,8,8,10], target = 8
Output: [3,4]
Example 2:

Input: nums = [5,7,7,8,8,10], target = 6
Output: [-1,-1]
Example 3:

Input: nums = [], target = 0
Output: [-1,-1]

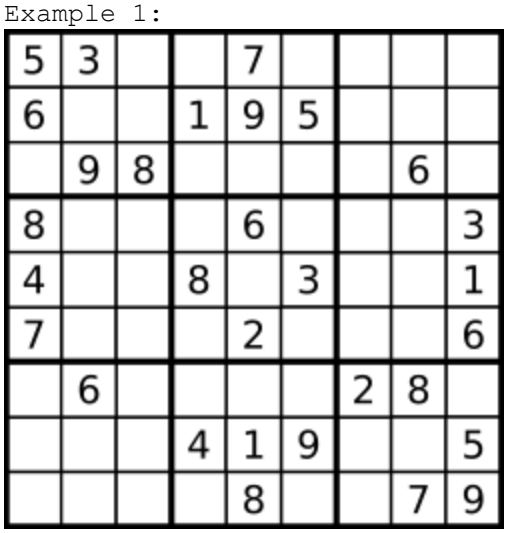
Valid Sudoku

Given an array of integers nums sorted in non-decreasing order, find the starting and ending position of a given target value.

If target is not found in the array, return [-1, -1].

You must write an algorithm with O(log n) runtime complexity.

Example 1:



Input: nums = [5,7,7,8,8,10], target = 8
Output: [3,4]
Example 2:

Input: nums = [5,7,7,8,8,10], target = 6
Output: [-1,-1]
Example 3:

Input: nums = [], target = 0
Output: [-1,-1]

Count and Say

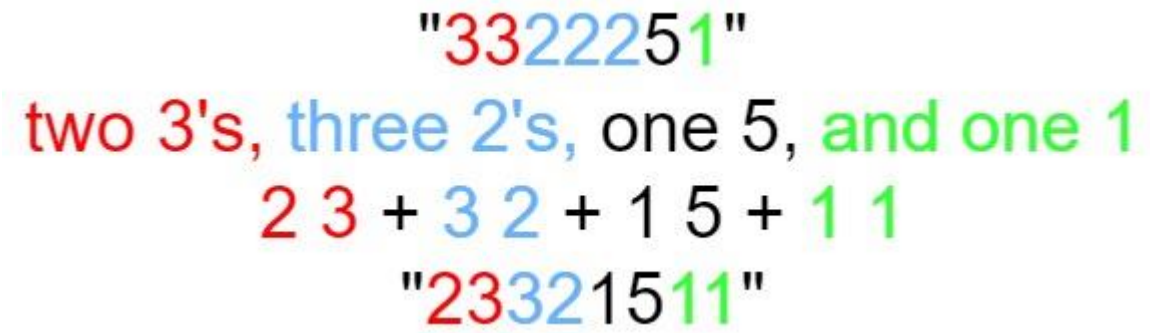
The count-and-say sequence is a sequence of digit strings defined by the recursive formula:

countAndSay(1) = "1"
countAndSay(n) is the way you would "say" the digit string from countAndSay(n-1), which is then converted into a different digit string.
To determine how you "say" a digit string, split it into the minimal number of substrings such that each substring contains exactly one unique digit. Then for each substring, say the number of digits, then say the digit. Finally, concatenate every said digit.

For example, the saying and conversion for digit string "3322251":

Given a positive integer n, return the nth term of the count-and-say sequence.

Example 1:




```
Input: n = 1
Output: "1"
Explanation: This is the base case.
Example 2:

Input: n = 4
Output: "1211"
Explanation:
countAndSay(1) = "1"
```

Combination Sum

```
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.



Example 1:

Input: candidates = [2,3,6,7], target = 7
Output: [[2,2,3],[7]]
Explanation:
2 and 3 are candidates, and 2 + 2 + 3 = 7. Note that 2 can be used multiple
times.
7 is a candidate, and 7 = 7.
These are the only two combinations.
Example 2:

Input: candidates = [2,3,5], target = 8
Output: [[2,2,2,2],[2,3,3],[3,5]]
Example 3:

Input: candidates = [2], target = 1
Output: []
```