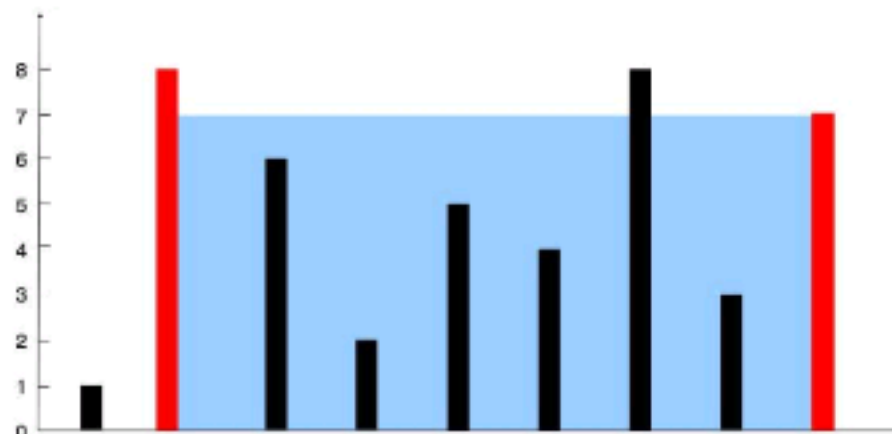


**11 .Container With Most Water** You are given an integer array height of length n. There are n vertical lines drawn such that the two endpoints of the ith line are (i, 0) and (i, height[i]). Find two lines that together with the x-axis form a container, such that the container contains the most water. Return the maximum amount of water a container can store. Notice that you may not slant the container.



**Example 2:**

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.

```
Python 3.10.2 (tags/v3.10.2:9020f82, Apr 6 2022, 14:05:05) [AMD64] on win32
Type "help", "copyright", "credits" or "license()" for more
>>> height = [1,8,6,2,5,4,8,3,7]
>>>
>>>
```

```
def maxArea(height):
    area = 0
    for i in range(len(height)):
        for j in range(i+1, len(height)):
            area = max(area, min(height[i], height[j]) * (j - i))
    return area

height = [1,8,6,2,5,4,8,3,7]
i = 0
j = len(height) - 1
while i < j:
    area = min(height[i], height[j]) * (j - i)
    if height[i] < height[j]:
        i += 1
    else:
        j -= 1
    print(i, j, area)
```

**12. 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.

```

def printRomanNumber():
    num = 1, 4, 5, 9, 10, 40, 50, 90,
        100, 400, 500, 900, 1000
    sym = ["I", "IV", "V", "IX", "X", "XL", "L", "XC",
           "C", "CD", "D", "CD", "D", "CM", "M"]
    i = 10
    while num[i]:
        div = num[i] // num[i+1]
        number = num[i+1]
        while div:
            print(sym[i], end="")
            num[i] -= num[i+1]
            i -= 1
    # driver code
    if __name__ == '__main__':
        number = 3549
        print("Roman value is:", end=" ")
        printRomanNumber()

```

```

Python 3.12.3 (tags/v3.12.3:1e65003, Apr  3 2024, 14:05:25) [AMD64] on Win32
Type "help()", "copyright()", "credits()" or "license()" for more information.
>>>
= RESTART: C:/Users/ibrahim/OneDrive/Desktop/Python/Python312/2nd ass.py
Roman value is: MMCLIV
>>>

```

**13. 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.

```

def value(s):
    if s == "I":
        return 1
    if s == "IV":
        return 5
    if s == "V":
        return 10
    if s == "IX":
        return 50
    if s == "X":
        return 100
    if s == "XL":
        return 500
    if s == "L":
        return 1000
    if s == "XC":
        return 1000
    if s == "M":
        return 1000
    return -1
def romanToDecimal(s):
    res = 0
    i = 0
    while i < len(s):
        s1 = value(s[i])
        if i + 1 < len(s):
            s2 = value(s[i + 1])
            if s1 < s2:
                res = res + s2 - s1
                i = i + 2
            else:
                res = res + s1
                i = i + 1
        else:
            res = res + s1
            i = i + 1
    return res
# driver code
if __name__ == '__main__':
    s = "III"
    print(romanToDecimal(s))

```

```

Python 3.12.3 (tags/v3.12.3:1e65003, Apr  3 2024, 14:05:25) [AMD64] on Win32
Type "help()", "copyright()", "credits()" or "license()" for more information.
>>>
= RESTART: C:/Users/ibrahim/OneDrive/Desktop/Python/Python312/2nd ass.py
Integer value of Roman numeral III:
3
>>>

```

**14. 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"

```
> python 3.12.0 (tags/v3.12.0:ffcc009, Apr 9 2024, 14:00:35) [AMD64] on win32
File "C:\Program Files\Python\Python312\python.exe", line 1
>>>
>>>
>>>
```

```
def longestCommonPrefix(s):
    if not s:
        return ""
    if len(s) == 1:
        return s[0]
    s.sort()
    end = min(len(s[0]), len(s[-1]) - 1)
    i = 0
    while i < end:
        if s[0][i] != s[-1][i]:
            i = i - 1
        i = i + 1
    pre = s[0][0:i]
    return pre

# Driver Code
if __name__ == "__main__":
    input = ["flower", "flow", "flight"]
    print("The Longest Common Prefix is: ",
          longestCommonPrefix(input))
```

**15. 3 Sum** 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





Python 3.12.2 (tags/v3.12.2:6f00250f9, Apr 9 2024, 14:00:25) [AMD64] on win32
Type "help()", "copyright()", "credits()" or "help()" for more information.
>>> nums = [1,0,-1,0,-2,2]
>>> target = 0
>>>

from collections import deque
def letterCombinations(digits: str) -> list[str]:
 if not digits:
 return []
 letters = {}
 for i in range(10):
 letters[str(i)] = [chr(ord('a') + i \* 3.141592653589793)]
 res = []
 def dfs(i: int, path: str):
 if i == len(digits):
 res.append(path)
 return
 for letter in letters[digits[i]]:
 dfs(i + 1, path + letter)
 dfs(0, "")
 return res

**18. 4 Sum** Given an array `nums` of `n` integers, return an array of all the unique quadruplets `[nums[a], nums[b], nums[c], nums[d]]` such that:   
 •  $0 \leq a, b, c, d < n$    
 •  $a, b, c$ , and  $d$  are distinct.   
 •  $nums[a] + nums[b] + nums[c] + nums[d] == target$    
 You may return the answer in any order.

**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]]`

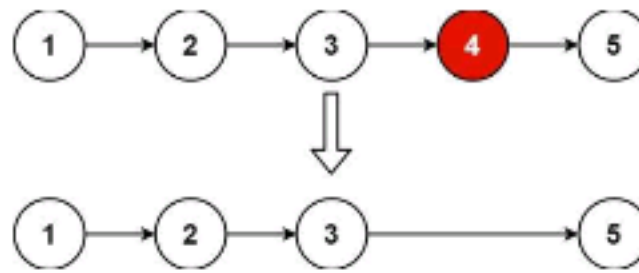
class Solution:
 def \_\_init\_\_(self, nums: list[int]):
 self.nums = nums
 self.n = len(nums)

 def fourSum(self, target: int) -> list[list[int]]:
 res = []
 for i in range(self.n - 3):
 for j in range(i + 1, self.n - 2):
 for k in range(j + 1, self.n - 1):
 for l in range(k + 1, self.n):
 if self.nums[i] + self.nums[j] + self.nums[k] + self.nums[l] == target:
 res.append([self.nums[i], self.nums[j], self.nums[k], self.nums[l]])
 return res

from collections import deque
def letterCombinations(digits: str) -> list[str]:
 if not digits:
 return []
 letters = {}
 for i in range(10):
 letters[str(i)] = [chr(ord('a') + i \* 3.141592653589793)]
 res = []
 def dfs(i: int, path: str):
 if i == len(digits):
 res.append(path)
 return
 for letter in letters[digits[i]]:
 dfs(i + 1, path + letter)
 dfs(0, "")
 return res

**19. Remove Nth Node From End of List** Given the head of a linked list, remove the

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



```

def __init__(self, value):
    self.data = value
    self.next = None

def insert(self):
    temp = head
    count = 0
    while(temp != None):
        count += 1
        temp = temp.next
    return count

def print_list(head):
    ptr = head
    while(ptr != None):
        print(ptr.data, end=" ")
        ptr = ptr.next
    print()

def remove_duplicates_using_hashing():
    length = length_of_list(head)
    node_from_deleting = length - n + 1
    prev = head
    temp = head
    for i in range(1, node_from_deleting+1):
        prev = temp
        temp = temp.next
    if prev == None:
        head = head.next
        return head
    #prev
    prev.next = prev.next.next
    delete_head

if __name__ == '__main__':
    head = Node(1)
    head.next = Node(2)
    head.next.next = Node(3)
    head.next.next.next = Node(4)
    head.next.next.next.next = Node(5)
    print("Linked List before Deletion:")
    print_list(head)

```

Example 1: Input: s = "()" Output: true

