

1. Perform matrix multiplication using Strassen's method.

```
def add(A, B):
    n = len(A)
    result = [[0] * n for _ in range(n)]
    for i in range(n):
        for j in range(n):
            result[i][j] = A[i][j] + B[i][j]
    return result

def sub(A, B):
    n = len(A)
    result = [[0] * n for _ in range(n)]
    for i in range(n):
        for j in range(n):
            result[i][j] = A[i][j] - B[i][j]
    return result

def split(matrix):
    n = len(matrix)
    mid = n // 2
    A11 = [[matrix[i][j] for j in range(mid)] for i in range(mid)]
    A12 = [[matrix[i][j] for j in range(mid, n)] for i in range(mid)]
    A21 = [[matrix[i][j] for j in range(mid)] for i in range(mid, n)]
    A22 = [[matrix[i][j] for j in range(mid, n)] for i in range(mid, n)]
    return A11, A12, A21, A22

def combine(C11, C12, C21, C22):
    n = len(C11) * 2
    result = [[0] * n for _ in range(n)]
    mid = n // 2
    for i in range(mid):
        for j in range(mid):
            result[i][j] = C11[i][j]
            result[i][j + mid] = C12[i][j]
            result[i + mid][j] = C21[i][j]
            result[i + mid][j + mid] = C22[i][j]
    return result

def strassen(A, B):
    n = len(A)
    if n == 1:
        return [[A[0][0] * B[0][0]]]
    A11, A12, A21, A22 = split(A)
    B11, B12, B21, B22 = split(B)
    M1 = strassen(add(A11, A22), add(B11, B22))
    M2 = strassen(add(A21, A22), B11)
    M3 = strassen(A11, sub(B12, B22))
    M4 = strassen(A22, sub(B21, B11))
    M5 = strassen(add(A11, A12), B22)
    M6 = strassen(sub(A21, A11), add(B11, B12))
    M7 = strassen(sub(A12, A22), add(B21, B22))
    C11 = add(sub(add(M1, M4), M5), M7)
    C12 = add(M3, M5)
    C21 = add(M2, M4)
    C22 = add(sub(add(M1, M3), M2), M6)
    return combine(C11, C12, C21, C22)

A = [
    [1, 2, 3, 4],
    [5, 6, 7, 8],
    [1, 2, 3, 4],
    [5, 6, 7, 8]
]
B = [
    [1, 2, 1, 3],
    [1, 4, 1, 5],
    [1, 6, 1, 7],
    [1, 8, 1, 9]
]
C = strassen(A, B)
for row in C:
    print(row)
```

2. 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.

```

6.2.py - C:\Users\balas\OneDrive\Documents\6.2.py (3.12.1)
File Edit Format Run Options Window Help
import ctypes
class Node:
    def __init__(self, data=None, next=None):
        self.data = data
        self.next = next
head = None
tail = None
n = None
def ins_end(num):
    global head, tail, n
    n = Node()
    n.data = num
    n.next = None
    if head is None:
        head = n
        tail = n
    else:
        tail.next = n
        tail = n
def sort():
    global head
    t = head
    while t is not None:
        s = t.next
        while s is not None:
            if t.data > s.data:
                t.data, s.data = s.data, t.data
            s = s.next
        t = t.next
def display():
    global head
    t = head
    while t is not None:
        print(f"{t.data}, ", end="")
        t = t.next
    print()
ins_end(1)
ins_end(2)
ins_end(4)
ins_end(1)
ins_end(3)
ins_end(4)
sort()
display()

```

```

IDLE Shell 3.12.1
File Edit Shell Debug Options Window Help
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\balas\OneDrive\Documents\6.2.py =====
1,1,2,3,4,4,
>>>

```

3. 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].

```
6.3.py - C:\Users\balas\OneDrive\Documents\6.3.py (3.12.1)
File Edit Format Run Options Window Help
lists = [[1,4,5],[1,3,4],[2,6]]
lst2=[]
for i in range(len(lists)):
    lst2+=lists[i]
for i in range(len(lst2)):
    for j in range(len(lst2)):
        if lst2[i]<lst2[j]:
            temp=lst2[i]
            lst2[i]=lst2[j]
            lst2[j]=temp
print(lst2)

IDLE Shell 3.12.1
File Edit Shell Debug Options Window Help
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\balas\OneDrive\Documents\6.3.py =====
>>> [1, 1, 2, 3, 4, 4, 5, 6]
```

4. 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. Since it is impossible to change the length of the array in some languages, you must instead have the result be placed in the first part of the array `nums`. More formally, if there are `k` elements after removing the duplicates, then the first `k` elements of `nums` should hold the final result. It does not matter what you leave beyond the first `k` elements. Return `k` after placing the final result in the first `k` slots of `nums`. Do not allocate extra space for another array. You must do this by modifying the input array in-place with $O(1)$ extra memory.

```
6.4.py - C:\Users\balas\OneDrive\Documents\6.4.py (3.12.1)
File Edit Format Run Options Window Help
nums=[0,1,1,1,2,2,2,3,3,4,4]
nums2=[]
for i in range(len(nums)):
    if nums[i] not in nums2:
        nums2.append(nums[i])
print(nums2)

IDLE Shell 3.12.1
File Edit Shell Debug Options Window Help
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\balas\OneDrive\Documents\6.4.py =====
>>> [0, 1, 2, 3, 4]
```

5. 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` ($1 \leq k < \text{nums.length}$) 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`.

```
6.5.py - C:\Users\balas\OneDrive\Documents\6.5.py (3.12.1)
File Edit Format Run Options Window Help
def search(nums, target):
    low = 0
    high = len(nums) - 1
    while low <= high:
        mid = (low + high) // 2
        if nums[mid] < target:
            low = mid + 1
        elif nums[mid] > target:
            high = mid - 1
        else:
            return mid
    return -1
nums = [4,5,6,7,0,1,2]
nums2=sorted(nums)
target = 0
if search(nums2, target)>=-1:
    print(nums.index(target))
else:
    print(search(nums, target))

IDLE Shell 3.12.1
File Edit Shell Debug Options Window Help
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\balas\OneDrive\Documents\6.5.py =====
>>> 4
```

6. Find First and Last Position of Element in Sorted Array 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].

```

6.6.py - C:\Users\balas\OneDrive\Documents\6.6.py (3.12.1)
File Edit Format Run Options Window Help
def search(nums, target):
    def Left(nums, target):
        l, r = 0, len(nums)
        while l < r:
            mid = l + (r - l) // 2
            if nums[mid] < target:
                l = mid + 1
            else:
                r = mid
        return l

    def Right(nums, target):
        l, r = 0, len(nums)
        while l < r:
            mid = l + (r - l) // 2
            if nums[mid] <= target:
                l = mid + 1
            else:
                r = mid
        return l

    lidx = Left(nums, target)
    ridx = Right(nums, target) - 1

    if lidx <= ridx:
        return [lidx, ridx]
    else:
        return [-1, -1]

nums = [5,7,7,8,8,10]
target = 8
print(search(nums, target))

```

```

IDLE Shell 3.12.1
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\balas\OneDrive\Documents\6.6.py =====
>>> [3, 4]
>>>

```

7. Sort Colors Given an array nums with n objects colored red, white, or blue, sort them in-place so that objects of the same color are adjacent, with the colors in the order red, white, and blue. We will use the integers 0, 1, and 2 to represent the color red, white, and blue, respectively. You must solve this problem without using the library's sort function.

```

6.7.py - C:\Users\balas\OneDrive\Documents\6.7.py (3.12.1)
File Edit Format Run Options Window Help
nums = [2,0,2,1,1,0]
for i in range(len(nums)):
    for j in range(len(nums)):
        if nums[i]<nums[j]:
            temp=nums[i]
            nums[i]=nums[j]
            nums[j]=temp
print(nums)

```

```

IDLE Shell 3.12.1
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\balas\OneDrive\Documents\6.7.py =====
>>> [0, 0, 1, 1, 2, 2]
>>>

```

8. 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.

```

6.8.py - C:\Users\balas\OneDrive\Documents\6.8.py (3.12.1)
File Edit Format Run Options Window Help
class Node:
    def __init__(self, val):
        self.val = val
        self.next = None

class LList:
    def __init__(self):
        self.head = None

    def push(self, new_data):
        new_node = Node(new_data)
        new_node.next = self.head
        self.head = new_node
        return new_node

    def remove(self, temp):
        curr = temp
        head = prev = Node(None)
        head.next = curr
        while curr and curr.next:
            if curr.val == curr.next.val:
                while(curr and curr.next and curr.val == curr.next.val):
                    curr = curr.next
                curr = curr.next
                prev.next = curr
            else:
                prev = prev.next
                curr = curr.next
        return head.next

    def printList(self):
        templ = self.head
        while templ is not None:
            print(templ.val, end = " ")
            templ = templ.next

    def get_head(self):
        return self.head

if __name__ == '__main__':
    llist = LList()
    llist.push(3)
    llist.push(3)
    llist.push(2)
    llist.push(1)
    llist.push(0)
    print('Created linked list is:')
    llist.printList()
    print('\nLinked list after deletion of duplicates:')
    head1 = llist.get_head()
    llist.remove(head1)

```

```

IDLE Shell 3.12.1
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\balas\OneDrive\Documents\6.8.py =====
>>> Created linked list is:
0 1 2 3 3
Linked list after deletion of duplicates:
0 1 2
>>>

```

```

f __name__ == '__main__':
    llist = LList()
    llist.push(3)
    llist.push(3)
    llist.push(2)
    llist.push(1)
    llist.push(0)
    print('Created linked list is:')
    llist.printList()
    print('\nLinked list after deletion of duplicates:')
    head1 = llist.get_head()
    llist.remove(head1)
    llist.printList()

```

9. 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.

```

6.9.py - C:\Users\balas\OneDrive\Documents\6.9.py (3.12.1)
File Edit Format Run Options Window Help
def merge(nums1, m, nums2, n):
    p1, p2, p = m - 1, n - 1, m + n - 1

    while p1 >= 0 and p2 >= 0:
        if nums1[p1] > nums2[p2]:
            nums1[p] = nums1[p1]
            p1 -= 1
        else:
            nums1[p] = nums2[p2]
            p2 -= 1
        p -= 1

    while p2 >= 0:
        nums1[p] = nums2[p2]
        p2 -= 1
        p -= 1

    nums1 = [1, 2, 3, 0, 0, 0]
    m = 3
    nums2 = [2, 5, 6]
    n = 3
    merge(nums1, m, nums2, n)
    print(nums1)

```

```

IDLE Shell 3.12.1
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\balas\OneDrive\Documents\6.9.py =====
>>> [1, 2, 2, 3, 5, 6]

```

10. 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.

```

6.10.py - C:\Users\balas\OneDrive\Documents\6.10.py (3.12.1)
File Edit Format Run Options Window Help
class Node:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

def bst(nums):
    if not nums:
        return None
    mid = len(nums) // 2
    root = Node(nums[mid])
    root.left = bst(nums[:mid])
    root.right = bst(nums[mid+1:])

    return root

def levelOrder(root):
    if not root:
        return []

    result = []
    queue = [root]

    while queue:
        node = queue.pop(0)
        if node:
            result.append(node.val)
            queue.append(node.left)
            queue.append(node.right)
        else:
            result.append(None)
    while result and result[-1] is None:
        result.pop()

    return result

nums = [-10, -3, 0, 5, 9]
bst_root = bst(nums)
print(levelOrder(bst_root))

```

```

IDLE Shell 3.12.1
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937 AMD64] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\balas\OneDrive\Documents\6.10.py =====
>>> [0, -3, 9, -10, None, 5]

```

11. Insertion Sort List Given the head of a singly linked list, sort the list using insertion sort, and return the sorted list's head.

The screenshot shows a Python IDE with a file named `6.11.py` and an IDLE Shell. The code defines a `ListNode` class and a function `ins(head)` to insert a new node into a sorted linked list. It also includes a `printList` function to display the list. The main execution creates a linked list with values 4, 2, 1, 3 and sorts it in ascending order, resulting in 1, 2, 3, 4.

```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next
def ins(head):
    if not head or not head.next:
        return head
    sorted_head = ListNode(0)
    current = head
    while current:
        prev = sorted_head
        next_node = sorted_head.next
        while next_node:
            if next_node.val > current.val:
                break
            prev = next_node
            next_node = next_node.next
        next_iter = current.next
        current.next = next_node
        prev.next = current
        current = next_iter
    return sorted_head.next
def printList(head):
    while head:
        print(head.val, end=" -> ")
        head = head.next
    print("None")
head = ListNode(4)
head.next = ListNode(2)
head.next.next = ListNode(1)
head.next.next.next = ListNode(3)
print("Original list:")
printList(head)
sorted_head = ins(head)
print("Sorted list:")
printList(sorted_head)
```

12. Sort Characters By Frequency Given a string `s`, sort it in decreasing order based on the frequency of the characters. The frequency of a character is the number of times it appears in the string. Return the sorted string. If there are multiple answers, return any of them.

The screenshot shows a Python IDE with a file named `6.12.py` and an IDLE Shell. The code uses `collections.Counter` to count the frequency of characters in a string `s`. It then sorts the characters by frequency in descending order and reconstructs the string. Examples shown are `"tree"` becoming `"eert"`, `"cccaaa"` becoming `aaaccc`, and `"Aabb"` becoming `bbAa`.

```
from collections import Counter
def sorting(s):
    freq = Counter(s)
    sorted_chars = sorted(freq, key=lambda x: (-freq[x], x))
    result = ''.join([char * freq[char] for char in sorted_chars])
    return result
s = "tree"
print(sorting(s))
s = "cccaaa"
print(sorting(s))
s = "Aabb"
print(sorting(s))
```

13. . Max Chunks To Make Sorted You are given an integer array `arr` of length `n` that represents a permutation of the integers in the range `[0, n - 1]`. We split `arr` into some number of chunks (i.e., partitions), and individually sort each chunk. After concatenating them, the result should equal the sorted array. Return the largest number of chunks we can make to sort the array.

The screenshot shows a Python IDE with a file named `6.13.py` and an IDLE Shell. The code defines a function `maxchunks(arr)` that iterates through the array, keeping track of the maximum value seen so far. Whenever the current index equals the maximum value, a new chunk is identified. Examples shown are `[4, 3, 2, 1, 0]` resulting in 1 chunk and `[1, 0, 2, 3, 4]` resulting in 2 chunks.

```
def maxchunks(arr):
    max_so_far = 0
    chunks = 0
    for i, num in enumerate(arr):
        max_so_far = max(max_so_far, num)
        if max_so_far == i:
            chunks += 1
    return chunks
arr = [4, 3, 2, 1, 0]
print(maxchunks(arr))
arr = [1, 0, 2, 3, 4]
print(maxchunks(arr))
```

14. Intersection of Three Sorted Arrays Given three integer arrays `arr1`, `arr2` and `arr3` sorted in strictly increasing order, return a sorted array of only the integers that appeared in all three arrays.

```
6.14.py - C:\Users\balas\OneDrive\Documents\6.14.py (3.12.1)
File Edit Format Run Options Window Help

arr1 = [1, 2, 3, 4, 5]
arr2 = [1, 2, 5, 7, 9]
arr3 = [1, 3, 4, 5, 8]
i, j, k = 0, 0, 0
result = []
while i < len(arr1) and j < len(arr2) and k < len(arr3):
    if arr1[i] == arr2[j] == arr3[k]:
        result.append(arr1[i])
        i += 1
        j += 1
        k += 1
    else:
        if arr1[i] < arr2[j]:
            i += 1
        elif arr2[j] < arr3[k]:
            j += 1
        else:
            k += 1
print(result)

IDLE Shell 3.12.1
File Edit Shell Debug Options Window Help
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\balas\OneDrive\Documents\6.14.py =====
>>> [1, 5]
```

15. 4. Sort the Matrix Diagonally A matrix diagonal is a diagonal line of cells starting from some cell in either the topmost row or leftmost column and going in the bottom-right direction until reaching the matrix's end. For example, the matrix diagonal starting from mat[2][0], where mat is a 6 x 3 matrix, includes cells mat[2][0], mat[3][1], and mat[4][2]

```
6.15.py - C:\Users\balas\OneDrive\Documents\6.15.py (3.12.1)
File Edit Format Run Options Window Help

def diagonal(mat):
    from collections import defaultdict
    import heapq
    diagonals = defaultdict(list)
    for i in range(len(mat)):
        for j in range(len(mat[0])):
            diagonals[i - j].append(mat[i][j])
    for key in diagonals:
        diagonals[key].sort()
    for i in range(len(mat)):
        for j in range(len(mat[0])):
            mat[i][j] = heapq.heappop(diagonals[i - j])
    return mat

mat = [
    [3, 3, 1, 1],
    [2, 2, 1, 2],
    [1, 1, 1, 2]
]

sorted_mat = diagonal(mat)
print(sorted_mat)

IDLE Shell 3.12.1
File Edit Shell Debug Options Window Help
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec 7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\balas\OneDrive\Documents\6.15.py =====
>>> [[1, 1, 1, 1], [1, 2, 2, 2], [1, 2, 3, 3]]
```