TOPIC 1 : INTRODUCTION

1.Given an array of strings words, return the first palindromic string in the array. If there is no such string, return an empty string "". A string is palindromic if it reads the same forward and backward.

Example 1:

Input: words = ["abc","car","ada","racecar","cool"]

Output: "ada"

Explanation: The first string that is palindromic is "ada".

Note that "racecar" is also palindromic, but it is not the first.

Example 2:

Input: words = ["notapalindrome","racecar"]

Output: "racecar"

Explanation: The first and only string that is palindromic is "racecar".

2.You are given two integer arrays nums1 and nums2 of sizes n and m, respectively. Calculate the following values: answer1 : the number of indices i such that nums1[i] exists in nums2. answer2 : the number of indices i such that nums2[i] exists in nums1 Return [answer1,answer2].

Example 1:

Input: nums1 = [2,3,2], nums2 = [1,2]

Output: [2,1]

Explanation:

Example 2:

Input: nums1 = [4,3,2,3,1], nums2 = [2,2,5,2,3,6]

Output: [3,4]

Explanation:

The elements at indices 1, 2, and 3 in nums1 exist in nums2 as well. So answer1 is 3.

The elements at indices 0, 1, 3, and 4 in nums2 exist in nums1. So answer2 is 4.

3.You are given a 0-indexed integer array nums. The distinct count of a subarray of nums is defined as: Let nums[i..j] be a subarray of nums consisting of all the indices from i to j such that 0 <= i <= j < nums.length. Then the number of distinct values in nums[i..j] is called the distinct count of nums[i..j]. Return the sum of the squares of distinct counts of all subarrays of nums. A subarray is a contiguous non-empty sequence of elements within an array.

Example 1:

Input: nums = [1,2,1]

Output: 15

Explanation: Six possible subarrays are:

[1]: 1 distinct value

[2]: 1 distinct value

[1]: 1 distinct value

[1,2]: 2 distinct values

[2,1]: 2 distinct values

[1,2,1]: 2 distinct values

The sum of the squares of the distinct counts in all subarrays is equal to 12 + 12 + 12 + 22 + 22 + 22 = 15.

Example 2:

Input: nums = [1,1]

Output: 3

Explanation: Three possible subarrays are:

[1]: 1 distinct value

[1]: 1 distinct value

[1,1]: 1 distinct value

The sum of the squares of the distinct counts in all subarrays is equal to 12 + 12 + 12 = 3.

4. Given a 0-indexed integer array nums of length n and an integer k, return *the number of pairs* (i, j) *where* 0 <= i < j < n, *such that* nums[i] == nums[j] *and* (i * j) *is divisible by* k.

   Example 1:

   Input: nums = [3,1,2,2,2,1,3], k = 2

   Output: 4

   Explanation:

   There are 4 pairs that meet all the requirements:

   - nums[0] == nums[6], and 0 * 6 == 0, which is divisible by 2.

   - nums[2] == nums[3], and 2 * 3 == 6, which is divisible by 2.

   - nums[2] == nums[4], and 2 * 4 == 8, which is divisible by 2.

   - nums[3] == nums[4], and 3 * 4 == 12, which is divisible by 2.

   Example 2:

   Input: nums = [1,2,3,4], k = 1

   Output: 0

   Explanation: Since no value in nums is repeated, there are no pairs (i,j) that meet all the requirements.

5. Write a program FOR THE BELOW TEST CASES with least time complexity
   Test Cases: -

   Input: {1, 2, 3, 4, 5} Expected Output: 5

   Input: {7, 7, 7, 7, 7} Expected Output: 7

   Input: {-10, 2, 3, -4, 5} Expected Output: 5

6. You have an algorithm that process a list of numbers. It firsts sorts the list using an efficient sorting algorithm and then finds the maximum element in sorted list. Write the code for the same.

Test Cases

1. Empty List
    1. Input: []
    2. Expected Output: None or an appropriate message indicating that the list is empty.
2. Single Element List
    1. Input: [5]
    2. Expected Output: 5
3. All Elements are the Same
    1. Input: [3, 3, 3, 3, 3]
    2. Expected Output: 3

7. Write a program that takes an input list of n numbers and creates a new list containing only the unique elements from the original list. What is the space complexity of the algorithm?

    Test Cases

    Some Duplicate Elements

    Input: [3, 7, 3, 5, 2, 5, 9, 2]

    Expected Output: [3, 7, 5, 2, 9] (Order may vary based on the algorithm used)

    Negative and Positive Numbers

    Input: [-1, 2, -1, 3, 2, -2]

    Expected Output: [-1, 2, 3, -2] (Order may vary)

    List with Large Numbers

    Input: [1000000, 999999, 1000000]

    Expected Output: [1000000, 999999]

8. Sort an array of integers using the bubble sort technique. Analyze its time complexity using Big-O notation. Write the code

9. Checks if a given number x exists in a sorted array arr using binary search. Analyze its time complexity using Big-O notation.

   Test Case:

   Example X={ 3,4,6,-9,10,8,9,30} KEY=10

   Output: Element 10 is found at position 5

   Example X={ 3,4,6,-9,10,8,9,30} KEY=100

   Output : Element 100 is not found

10. Given an array of integers nums, sort the array in ascending order and return it. You must solve the problem without using any built-in functions in O(nlog(n)) time complexity and with the smallest space complexity possible.

11. Given an m x n grid and a ball at a starting cell, find the number of ways to move the ball out of the grid boundary in exactly N steps.

Example:

- Input: m=2,n=2,N=2,i=0,j=0 · Output: 6
- Input: m=1,n=3,N=3,i=0,j=1 · Output: 12

12. You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed. All houses at this place are arranged in a circle. That means the first house is the neighbor of the last one. Meanwhile, adjacent houses have security systems connected, and it will automatically contact the police if two adjacent houses were broken into on the same night.

    Examples:

Input : nums = [2, 3, 2]

Output : The maximum money you can rob without alerting the

police is 3(robbing house 1).

(ii) Input : nums = [1, 2, 3, 1]

Output : The maximum money you can rob without alerting the

police is 4 (robbing house 1 and house 3).

13. 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?

Examples:

Input: n=4 Output: 5

Input: n=3 Output: 3

14. A robot is located at the top-left corner of a m×n grid .The robot can only move either down or right at any point in time. The robot is trying to reach the bottom-right corner of the grid. How many possible unique paths are there?

Examples:

Input: m=7,n=3 Output: 28

Input: m=3,n=2 Output: 3

15. In a string S of lowercase letters, these letters form consecutive groups of the same character. For example, a string like s = "abbxxxxzyy" has the groups "a", "bb", "xxxx", "z", and "yy". A group is identified by an interval [start, end], where start and end denote the start and end indices (inclusive) of the group. In the above example, "xxxx" has the interval [3,6]. A group is considered large if it has 3 or more characters. Return the intervals of every large group sorted in increasing order by start index.

Example 1:

Input: s = "abbxxxxzzy"

Output: [[3,6]]

Explanation: "xxxx" is the only large group with start index 3 and end index 6.

Example 2:

Input: s = "abc"

Output: []

Explanation: We have groups "a", "b", and "c", none of which are large groups.

15. "The Game of Life, also known simply as Life, is a cellular automaton devised by the British mathematician John Horton Conway in 1970." The board is made up of an m x n grid of cells, where each cell has an initial state: live (represented by a 1) or dead (represented by a 0). Each cell interacts with its eight neighbors (horizontal, vertical, diagonal) using the following four rules

    Any live cell with fewer than two live neighbors dies as if caused by under-population.

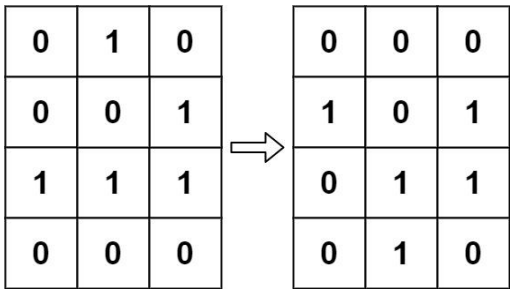
    Any live cell with two or three live neighbors lives on to the next generation.

    Any live cell with more than three live neighbors dies, as if by over-population.

    Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.

    The next state is created by applying the above rules simultaneously to every cell in the current state, where births and deaths occur simultaneously. Given the current state of the m x n grid board, return *the next state*.

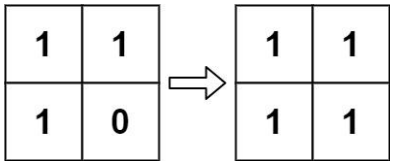
    Example 1:



Input: board = [[0,1,0],[0,0,1],[1,1,1],[0,0,0]]

Output: [[0,0,0],[1,0,1],[0,1,1],[0,1,0]]

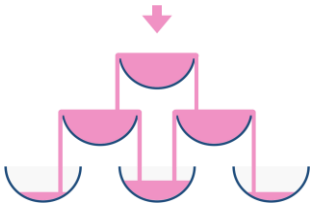
Example 2:



Input: board = [[1,1],[1,0]]

Output: [[1,1],[1,1]]

16. We stack glasses in a pyramid, where the first row has 1 glass, the second row has 2 glasses, and so on until the 100th row. Each glass holds one cup of champagne. Then, some champagne is poured into the first glass at the top. When the topmost glass is full, any excess liquid poured will fall equally to the glass immediately to the left and right of it. When those glasses become full, any excess champagne will fall equally to the left and right of those glasses, and so on. (A glass at the bottom row has its excess champagne fall on the floor.) For example, after one cup of champagne is poured, the top most glass is full. After two cups of champagne are poured, the two glasses on the second row are half full. After three cups of champagne are poured, those two cups become full - there are 3 full glasses total now. After four cups of champagne are poured, the third row has the middle glass half full, and the two outside glasses are a quarter full, as pictured below.



Now after pouring some non-negative integer cups of champagne, return how full the j^{th} glass in the i^{th} row is (both i and j are 0-indexed.)

Example 1:

Input: poured = 1, query_row = 1, query_glass = 1

Output: 0.00000

Explanation: We poured 1 cup of champange to the top glass of the tower (which is indexed as (0, 0)). There will be no excess liquid so all the glasses under the top glass will remain empty.

Example 2:

Input: poured = 2, query_row = 1, query_glass = 1

Output: 0.50000

Explanation: We poured 2 cups of champange to the top glass of the tower (which is indexed as (0, 0)). There is one cup of excess liquid. The glass indexed as (1, 0) and the glass indexed as (1, 1) will share the excess liquid equally, and each will get half cup of champange.

TOPIC 2 : BRUTE FORCE

1. Write a program to perform the following

   An empty list

   A list with one element

   A list with all identical elements

   A list with negative numbers

   Test Cases:

   1. **Input:** []
      - **Expected Output:** []
   1. **Input:** [1]
      - **Expected Output:** [1]
   2. **Input:** [7, 7, 7, 7]
      - **Expected Output:** [7, 7, 7, 7]
   3. **Input:** [-5, -1, -3, -2, -4]
      - **Expected Output:** [-5, -4, -3, -2, -1]

2. Describe the Selection Sort algorithm's process of sorting an array. Selection Sort works by dividing the array into a sorted and an unsorted region. Initially, the sorted region is empty, and the unsorted region contains all elements. The algorithm repeatedly selects the smallest element from the unsorted region and swaps it with the leftmost unsorted element, then moves the boundary of the sorted region one element to the right. Explain

why Selection Sort is simple to understand and implement but is inefficient for large datasets. Provide examples to illustrate step-by-step how Selection Sort rearranges the elements into ascending order, ensuring clarity in your explanation of the algorithm's mechanics and effectiveness.

Sorting a Random Array:

Input: [5, 2, 9, 1, 5, 6]

Output: [1, 2, 5, 5, 6, 9]

Sorting a Reverse Sorted Array:

Input: [10, 8, 6, 4, 2]

Output: [2, 4, 6, 8, 10]

Sorting an Already Sorted Array:

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

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

3. Write code to modify bubble_sort function to stop early if the list becomes sorted before all passes are completed.
4. **Test Cases:**
    - Test your optimized function with the following lists:
        1. **Input:** [64, 25, 12, 22, 11]
            - **Expected Output:** [11, 12, 22, 25, 64]
        2. **Input:** [29, 10, 14, 37, 13]
            - **Expected Output:** [10, 13, 14, 29, 37]
        3. **Input:** [3, 5, 2, 1, 4]
            - **Expected Output:** [1, 2, 3, 4, 5]
        4. **Input:** [1, 2, 3, 4, 5] (Already sorted list)
            - **Expected Output:** [1, 2, 3, 4, 5]
        5. **Input:** [5, 4, 3, 2, 1] (Reverse sorted list)
            - **Expected Output:** [1, 2, 3, 4, 5]

Write code for Insertion Sort that manages arrays with duplicate elements during the sorting process. Ensure the algorithm's behavior when encountering duplicate values, including whether it preserves the relative order of duplicates and how it affects the overall sorting outcome.

Examples:

1. **Array with Duplicates**:
    - **Input**: [3, 1, 4, 1, 5, 9, 2, 6, 5, 3]
    - **Output**: [1, 1, 2, 3, 3, 4, 5, 5, 6, 9]
2. **All Identical Elements**:
    - **Input**: [5, 5, 5, 5, 5]
    - **Output**: [5, 5, 5, 5, 5]
3. **Mixed Duplicates**:
    - **Input**: [2, 3, 1, 3, 2, 1, 1, 3]
    - **Output**: [1, 1, 1, 2, 2, 3, 3, 3]

5. Given an array arr of positive integers sorted in a strictly increasing order, and an integer k. return the kth positive integer that is missing from this array.

    Example 1:

    Input: arr = [2,3,4,7,11], k = 5

    Output: 9

    Explanation: The missing positive integers are [1,5,6,8,9,10,12,13,...]. The 5th missing positive integer is 9.

    Example 2:

    Input: arr = [1,2,3,4], k = 2

    Output: 6

Explanation: The missing positive integers are [5,6,7,...]. The 2nd missing positive integer is 6.

6. A peak element is an element that is strictly greater than its neighbors. Given a 0-indexed integer array nums, find a peak element, and return its index. If the array contains multiple peaks, return the index to any of the peaks. You may imagine that nums[-1] = nums[n] = -∞. In other words, an element is always considered to be strictly greater than a neighbor that is outside the array. You must write an algorithm that runs in O(log n) time.

   Example 1:

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

   Output: 2

   Explanation: 3 is a peak element and your function should return the index number 2.

   Example 2:

   Input: nums = [1,2,1,3,5,6,4]

   Output: 5

   Explanation: Your function can return either index number 1 where the peak element is 2, or index number 5 where the peak element is 6.

7. 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.

8. Given an array of string words, return all strings in words that is a substring of another word. You can return the answer in any order. A substring is a contiguous sequence of characters within a string

    Example 1:

    Input: words = ["mass","as","hero","superhero"]

    Output: ["as","hero"]

    Explanation: "as" is substring of "mass" and "hero" is substring of "superhero".

    ["hero","as"] is also a valid answer.

    Example 2:

    Input: words = ["leetcode","et","code"]

    Output: ["et","code"]

    Explanation: "et", "code" are substring of "leetcode".

    Example 3:

    Input: words = ["blue","green","bu"]

    Output: []

    Explanation: No string of words is substring of another string.

9. Write a program that finds the closest pair of points in a set of 2D points using the brute force approach.

    Input:

    A list or array of points represented by coordinates (x, y).

    Points: [(1, 2), (4, 5), (7, 8), (3, 1)]

    Output:

    The two points with the minimum distance between them.

    The minimum distance itself.

    Closest pair: (1, 2) - (3, 1) Minimum distance: 1.4142135623730951

10. Write a program to find the closest pair of points in a given set using the brute force approach. Analyze the time complexity of your implementation. Define a function to calculate the Euclidean distance between two points. Implement a function to find the closest pair of points using the brute force method. Test your program with a sample set of points and verify the correctness of your results. Analyze the time complexity of your implementation. Write a brute-force algorithm to solve the convex hull problem for the following set S of points? P1 (10,0)P2 (11,5)P3 (5, 3)P4 (9, 3.5)P5 (15, 3)P6 (12.5, 7)P7 (6, 6.5)P8 (7.5, 4.5).How do you modify your brute force algorithm to handle multiple points that are lying on the sameline?

    Given points: P1 (10,0), P2 (11,5), P3 (5, 3), P4 (9, 3.5), P5 (15, 3), P6 (12.5, 7), P7 (6, 6.5), P8 (7.5, 4.5).

    output: P3, P4, P6, P5, P7, P1

11. Write a program that finds the convex hull of a set of 2D points using the brute force approach.

    Input:

    A list or array of points represented by coordinates (x, y).

    Points: [(1, 1), (4, 6), (8, 1), (0, 0), (3, 3)]

    Output:

    The list of points that form the convex hull in counter-clockwise order.

    Convex Hull: [(0, 0), (1, 1), (8, 1), (4, 6)]

12. You are given a list of cities represented by their coordinates. Develop a program that utilizes exhaustive search to solve the TSP. The program should:

    1. Define a function distance(city1, city2) to calculate the distance between two cities (e.g., Euclidean distance).
    2. Implement a function tsp(cities) that takes a list of cities as input and performs the following:
        - Generate all possible permutations of the cities (excluding the starting city) using itertools.permutations.

- For each permutation (representing a potential route):
    - Calculate the total distance traveled by iterating through the path and summing the distances between consecutive cities.
    - Keep track of the shortest distance encountered and the corresponding path.
- Return the minimum distance and the shortest path (including the starting city at the beginning and end).

3. Include test cases with different city configurations to demonstrate the program's functionality. Print the shortest distance and the corresponding path for each test case.

Test Cases:

Simple Case: Four cities with basic coordinates (e.g., [(1, 2), (4, 5), (7, 1), (3, 6)])

More Complex Case: Five cities with more intricate coordinates (e.g., [(2, 4), (8, 1), (1, 7), (6, 3), (5, 9)])

Output:

Test Case 1:

Shortest Distance: 7.0710678118654755

Shortest Path: [(1, 2), (4, 5), (7, 1), (3, 6), (1, 2)]

Test Case 2:

Shortest Distance: 14.142135623730951

Shortest Path: [(2, 4), (1, 7), (6, 3), (5, 9), (8, 1), (2, 4)]

13. You are given a cost matrix where each element cost[i][j] represents the cost of assigning worker i to task j. Develop a program that utilizes exhaustive search to solve the assignment problem. The program should Define a function total_cost(assignment, cost_matrix) that takes an assignment (list representing worker-task pairings) and the cost matrix as input. It iterates through the assignment and calculates the total cost by summing the corresponding costs from the cost matrix Implement a function assignment_problem(cost_matrix) that takes the cost matrix as input and

performs the following Generate all possible permutations of worker indices (excluding repetitions).

Test Cases:

Input

Simple Case: Cost Matrix:

[[3, 10, 7],

[8, 5, 12],

[4, 6, 9]]

More Complex Case: Cost Matrix:

[[15, 9, 4],

[8, 7, 18],

[6, 12, 11]]

Output:

Test Case 1:

Optimal Assignment: [(worker 1, task 2), (worker 2, task 1), (worker 3, task 3)]

Total Cost: 19

Test Case 2:

Optimal Assignment: [(worker 1, task 3), (worker 2, task 1), (worker 3, task 2)]

Total Cost: 24

14. You are given a list of items with their weights and values. Develop a program that utilizes exhaustive search to solve the 0-1 Knapsack Problem. The program should:

Define a function total_value(items, values) that takes a list of selected items (represented by their indices) and the value list as input. It iterates through the selected items and calculates the total value by summing the corresponding values from the value list.

Define a function is_feasible(items, weights, capacity) that takes a list of selected items (represented by their indices), the weight list, and the

knapsack capacity as input. It checks if the total weight of the selected items exceeds the capacity.

Test Cases:

Simple Case:

Items: 3 (represented by indices 0, 1, 2)

Weights: [2, 3, 1]

Values: [4, 5, 3]

Capacity: 4

More Complex Case:

Items: 4 (represented by indices 0, 1, 2, 3)

Weights: [1, 2, 3, 4]

Values: [2, 4, 6, 3]

Capacity: 6

Output:

Test Case 1:

Optimal Selection: [0, 2] (Items with indices 0 and 2)

Total Value: 7

Test Case 2:

Optimal Selection: [0, 1, 2] (Items with indices 0, 1, and 2)

Total Value: 10

TOPIC 3 : DIVIDE AND CONQUER

Write a Program to find both the maximum and minimum values in the array. Implement using any programming language of your choice. Execute your code and provide the maximum and minimum values found.

Input : N= 8, a[] = {5,7,3,4,9,12,6,2}

Output : Min = 2, Max = 12

Test Cases :

Input : N= 9, a[] = {1,3,5,7,9,11,13,15,17}

Output : Min = 1, Max = 17

Test Cases :

Input : N= 10, a[] = {22,34,35,36,43,67, 12,13,15,17}

Output : Min 12, Max 67

2. Consider an array of integers sorted in ascending order: 2,4,6,8,10,12,14,18. Write a Program to find both the maximum and minimum values in the array. Implement using any programming language of your choice. Execute your code and provide the maximum and minimum values found.

   Input : N=8, 2,4,6,8,10,12,14,18.

   Output : Min = 2, Max =18

   Test Cases :

   Input : N= 9, a[] = {11,13,15,17,19,21,23,35,37}

   Output : Min = 11, Max = 37

   Test Cases :

   Input : N= 10, a[] = {22,34,35,36,43,67, 12,13,15,17}

   Output : Min 12, Max 67

3. You are given an unsorted array 31,23,35,27,11,21,15,28. Write a program for Merge Sort and implement using any programming language of your choice.

   Test Cases :

   Input : N= 8, a[] = {31,23,35,27,11,21,15,28}

   Output : 11,15,21,23,27,28,31,35

   Test Cases :

   Input : N= 10, a[] = {22,34,25,36,43,67, 52,13,65,17}

Output : 13,17,22,25,34,36,43,52,65,67

4. Implement the Merge Sort algorithm in a programming language of your choice and test it on the array 12,4,78,23,45,67,89,1. Modify your implementation to count the number of comparisons made during the sorting process. Print this count along with the sorted array.

   Test Cases :

   Input : N= 8, a[] = {12,4,78,23,45,67,89,1}

   Output : 1,4,12,23,45,67,78,89

   Test Cases :

   Input : N= 7, a[] = {38,27,43,3,9,82,10}

   Output : 3,9,10,27,38,43,82,

5. Given an unsorted array 10,16,8,12,15,6,3,9,5 Write a program to perform Quick Sort. Choose the first element as the pivot and partition the array accordingly. Show the array after this partition. Recursively apply Quick Sort on the sub-arrays formed. Display the array after each recursive call until the entire array is sorted.

   Input : N= 9, a[]= {10,16,8,12,15,6,3,9,5}

   Output : 3,5,6,8,9,10,12,15,16

   Test Cases :

   Input : N= 8, a[] = {12,4,78,23,45,67,89,1}

   Output : 1,4,12,23,45,67,78,89

   Test Cases :

   Input : N= 7, a[] = {38,27,43,3,9,82,10}

   Output : 3,9,10,27,38,43,82,

6. Implement the Quick Sort algorithm in a programming language of your choice and test it on the array 19,72,35,46,58,91,22,31. Choose the middle element as the pivot and partition the array accordingly. Show the array after this partition. Recursively apply Quick Sort on the sub-arrays formed. Display the

array after each recursive call until the entire array is sorted. Execute your code and show the sorted array.

Input : N= 8, a[] = {19,72,35,46,58,91,22,31}

Output : 19,22,31,35,46,58,72,91

Test Cases :

Input : N= 8, a[] = {31,23,35,27,11,21,15,28}

Output : 11,15,21,23,27,28,31,35

Test Cases :

Input : N= 10, a[] = {22,34,25,36,43,67, 52,13,65,17}

Output : 13,17,22,25,34,36,43,52,65,67

7. Implement the Binary Search algorithm in a programming language of your choice and test it on the array 5,10,15,20,25,30,35,40,45 to find the position of the element 20. Execute your code and provide the index of the element 20. Modify your implementation to count the number of comparisons made during the search process. Print this count along with the result.

   Input : N= 9, a[] = {5,10,15,20,25,30,35,40,45}, search key = 20

   Output : 4

   Test cases

   Input : N= 6, a[] = {10,20,30,40,50,60}, search key = 50

   Output : 5

   Input : N= 7, a[] = {21,32,40,54,65,76,87}, search key = 32

   Output : 2

8. You are given a sorted array 3,9,14,19,25,31,42,47,53 and asked to find the position of the element 31 using Binary Search. Show the mid-point calculations and the steps involved in finding the element. Display, what would happen if the array was not sorted, how would this impact the performance and correctness of the Binary Search algorithm?

   Input : N= 9, a[] = {3,9,14,19,25,31,42,47,53}, search key = 31

   Output : 6

Test cases

Input : N= 7, a[] = {13,19,24,29,35,41,42}, search key = 42

Output : 7

Test cases

Input : N= 6, a[] = {20,40,60,80,100,120}, search key = 60

Output : 3

9. Given an array of points where points[i] = [xi, yi] represents a point on the X-Y plane and an integer k, return the k closest points to the origin (0, 0).

   Input : points = [[**1**,**3**],[-**2**,**2**],[**5**,**8**],[**0**,**1**]],k=2

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

   (ii) **Input**: points = [[1, 3], [-2, 2]], k = 1

   Output: [[-2, 2]]

   (iii) **Input**: points = [[3, 3], [5, -1], [-2, 4]], k = 2

   Output: [[3, 3], [-2, 4]]

10. Given four lists A, B, C, D of integer values,Write a program to compute how many tuples n(i, j, k, l) there are such that A[i] + B[j] + C[k] + D[l] is zero.

    Input: A = [1, 2], B = [-2, -1], C = [-1, 2], D = [0, 2]

    Output: 2

    (ii) **Input**: A = [0], B = [0], C = [0], D = [0]

    Output: 1

11. To Implement the Median of Medians algorithm ensures that you handle the worst-case time complexity efficiently while finding the k-th smallest element in an unsorted array.

    arr = [12, 3, 5, 7, 19] k = 2 Expected Output:5

    arr = [12, 3, 5, 7, 4, 19, 26] k = 3 Expected Output:5

arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] k = 6 Expected Output:6

12. To Implement a function median_of_medians(arr, k) that takes an unsorted array arr and an integer k, and returns the k-th smallest element in the array.

    arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] k = 6

    arr = [23, 17, 31, 44, 55, 21, 20, 18, 19, 27] k = 5

    Output: An integer representing the k-th smallest element in the array.

13. Write a program to implement Meet in the Middle Technique. Given an array of integers and a target sum, find the subset whose sum is closest to the target. You will use the Meet in the Middle technique to efficiently find this subset.

    a) Set[] = {45, 34, 4, 12, 5, 2} Target Sum : 42

    b) Set[]= {1, 3, 2, 7, 4, 6} Target sum = 10:

14. Write a program to implement Meet in the Middle Technique. Given a large array of integers and an exact sum E, determine if there is any subset that sums exactly to E. Utilize the Meet in the Middle technique to handle the potentially large size of the array. Return true if there is a subset that sums exactly to E, otherwise return false.

    a) E = {1, 3, 9, 2, 7, 12} exact Sum = 15

    b) E = {3, 34, 4, 12, 5, 2} exact Sum = 15

15 Given two 2×2 Matrices A and B

A=(1 7 B=(1 3

3 5) 7 5)

Use Strassen's matrix multiplication algorithm to compute the product matrix C such that C=A×B.

Test Cases:

Consider the following matrices for testing your implementation:

Test Case 1:

A=(1 7 B=(6 8

3 5), 4 2)

Expected Output:

C=(18 14

62 66)

16 Given two integers X=1234 and Y=5678: Use the Karatsuba algorithm to compute the product Z=X x Y

Test Case 1:

Input: x=1234,y=5678

Expected Output: z=1234×5678=7016652

TOPIC 4 : DYNAMIC PROGRAMMING

1. You are given the number of sides on a die (num_sides), the number of dice to throw (num_dice), and a target sum (target). Develop a program that utilizes dynamic programming to solve the Dice Throw Problem.

Test Cases:

1.Simple Case:

- Number of sides: 6
- Number of dice: 2
- Target sum: 7

2.More Complex Case:

- Number of sides: 4
- Number of dice: 3
- Target sum: 10

Output

Test Case 1:

Number of ways to reach sum 7: 6

Test Case 2:

Number of ways to reach sum 10: 27

2. In a factory, there are two assembly lines, each with n stations. Each station performs a specific task and takes a certain amount of time to complete. The task must go through each station in order, and there is also a transfer time for switching from one line to another. Given the time taken at each station on both lines and the transfer time between the lines, the goal is to find the minimum time required to process a product from start to end.

Input

n: Number of stations on each line.

a1[i]: Time taken at station i on assembly line 1.

a2[i]: Time taken at station i on assembly line 2.

t1[i]: Transfer time from assembly line 1 to assembly line 2 after station i.

t2[i]: Transfer time from assembly line 2 to assembly line 1 after station i.

e1: Entry time to assembly line 1.

e2: Entry time to assembly line 2.

x1: Exit time from assembly line 1.

x2: Exit time from assembly line 2.

Output

The minimum time required to process the product.

3. An automotive company has three assembly lines (Line 1, Line 2, Line 3) to produce different car models. Each line has a series of stations, and each station takes a certain amount of time to complete its task. Additionally, there are transfer times between lines, and certain dependencies must be respected due to the sequential nature of some tasks. Your goal is to minimize the total production time by determining the optimal scheduling of tasks across these lines, considering the transfer times and dependencies.

   Number of stations: 3

   - Station times:
   - Line 1: [5, 9, 3]
   - Line 2: [6, 8, 4]
   - Line 3: [7, 6, 5]
   - Transfer times:

   [

   [0, 2, 3],

   [2, 0, 4],

   [3, 4, 0]

   ]

   Dependencies: [(0, 1), (1, 2)] (i.e., the output of the first station is needed for the second, and the second for the third, regardless of the line).

4. Write a c program to find the minimum path distance by using matrix form.

   Test Cases:

   1)

   {0,10,15,20}

   {10,0,35,25}

   {15,35,0,30}

   {20,25,30,0}

   Output: 80

2)

{0,10,10,10}

{10,0,10,10}

{10,10,0,10}

{10,10,10,0}

Output: 40

3)

{0,1,2,3}

{1,0,4,5}

{2,4,0,6}

{3,5,6,0}

Output: 12

5. Assume you are solving the Traveling Salesperson Problem for 4 cities (A, B, C, D) with known distances between each pair of cities. Now, you need to add a fifth city (E) to the problem.

Test Cases

1. Symmetric Distances

- Description: All distances are symmetric (distance from A to B is the same as B to A).

Distances:

A-B: 10, A-C: 15, A-D: 20, A-E: 25 B-C: 35, B-D: 25, B-E: 30 C-D: 30, C-E: 20 D-E: 15

Expected Output: The shortest route and its total distance. For example, A -> B -> D -> E -> C -> A might be the shortest route depending on the given distances.

6. 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"

   Constraints: ● 1 <= s.length <= 1000 ● s consist of only digits and English letters.

7. 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. Constraints: ● 0 <= s.length <= 5 * 104 ● s consists of English letters, digits, symbols and spaces.

8. Given a string s and a dictionary of strings wordDict, return true if s can be segmented into a space-separated sequence of one or more dictionary words.

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

   Example 1:

   Input: s = "leetcode", wordDict = ["leet","code"]

   Output: true

Explanation: Return true because "leetcode" can be segmented as "leet code".

Example 2:

Input: s = "applepenapple", wordDict = ["apple","pen"]

Output: true

Explanation: Return true because "applepenapple" can be segmented as "apple pen apple".

Note that you are allowed to reuse a dictionary word.

Example 3:

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

Output: false

9. Given an input string and a dictionary of words, find out if the input string can be segmented into a space-separated sequence of dictionary words.Consider the following dictionary { i, like, sam, sung, samsung, mobile, ice, cream, icecream, man, go, mango}

   Input: ilike

   Output: Yes

   The string can be segmented as "i like".

   Input: ilikesamsung

   Output: Yes The string can be segmented as "i like samsung" or "i like sam sung".

10. 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. 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.

11. Design a special dictionary that searches the words in it by a prefix and a suffix. Implement the WordFilter class: WordFilter(string[] words) Initializes the object with the words in the dictionary.f(string pref, string suff) Returns the index of the word in the dictionary, which has the prefix pref and the suffix suff. If there is more than one valid index, return the largest of them. If there is no such word in the dictionary, return -1.

    Example 1:

    Input

    ["WordFilter", "f"]

    [[["apple"]], ["a", "e"]]

Output

[null, 0]

Explanation

WordFilter wordFilter = new WordFilter(["apple"]);

wordFilter.f("a", "e"); // return 0, because the word at index 0 has prefix = "a" and suffix = "e".

12. Implement Floyd's Algorithm to find the shortest path between all pairs of cities. Display the distance matrix before and after applying the algorithm. Identify and print the shortest path

Input: n = 4, edges = [[0,1,3],[1,2,1],[1,3,4],[2,3,1]], distanceThreshold = 4

Output: 3

Explanation: The figure above describes the graph.

The neighboring cities at a distanceThreshold = 4 for each city are:

City 0 -> [City 1, City 2]

City 1 -> [City 0, City 2, City 3]

City 2 -> [City 0, City 1, City 3]

City 3 -> [City 1, City 2]

Cities 0 and 3 have 2 neighboring cities at a distanceThreshold = 4, but we have to return city 3 since it has the greatest number.

Test cases :

a) You are given a small network of 4 cities connected by roads with the following distances:

City 1 to City 2: 3

City 1 to City 3: 8

City 1 to City 4: -4

City 2 to City 4: 1

City 2 to City 3: 4

City 3 to City 1: 2

City 4 to City 3: -5

City 4 to City 2: 6

Implement Floyd's Algorithm to find the shortest path between all pairs of cities. Display the distance matrix before and after applying the algorithm. Identify and print the shortest path from City 1 to City 3.

Input as above

Output : City 1 to City 3 = -9

b. Consider a network with 6 routers. The initial routing table is as follows:

Router A to Router B: 1

Router A to Router C: 5

Router B to Router C: 2

Router B to Router D: 1

Router C to Router E: 3

Router D to Router E: 1

Router D to Router F: 6

Router E to Router F: 2

13. Write a Program to implement Floyd's Algorithm to calculate the shortest paths between all pairs of routers. Simulate a change where the link between Router B and Router D fails. Update the distance matrix accordingly. Display the shortest path from Router A to Router F before and after the link failure.

    Input as above

    Output : Router A to Router F = 5

14. Implement Floyd's Algorithm to find the shortest path between all pairs of cities. Display the distance matrix before and after applying the algorithm. Identify and print the shortest path

Input: n = 5, edges = [[0,1,2],[0,4,8],[1,2,3],[1,4,2],[2,3,1],[3,4,1]], distanceThreshold = 2

Output: 0

Explanation: The figure above describes the graph.

The neighboring cities at a distanceThreshold = 2 for each city are:

City 0 -> [City 1]

City 1 -> [City 0, City 4]

City 2 -> [City 3, City 4]

City 3 -> [City 2, City 4]

City 4 -> [City 1, City 2, City 3]

The city 0 has 1 neighboring city at a distanceThreshold = 2.

a) Test cases :

B to A 2

A TO C 3

C TO D 1

D TO A 6

C TO B 7

Find shortest path from C to A

Output = 7

b) Find shortest path from E to C

C TO A 2

A TO B 4

B TO C 1

B TO E 6

E TO A 1

A TO D 5

D TO E 2

E TO D 4

D TO C 1

C TO D 3

Output : E to C = 5

15. Implement the Optimal Binary Search Tree algorithm for the keys A,B,C,D with frequencies 0.1,0.2,0.4,0.3 Write the code using any programming language to construct the OBST for the given keys and frequencies. Execute your code and display the resulting OBST and its cost. Print the cost and root matrix.

    Input N =4, Keys = {A,B,C,D} Frequencies = {01.02.,0.3,0.4}

    Output : 1.7

    Cost Table

	0	1	2	3	4
1	0	0.1	0.4	1.1	1.7
2		0	0.2	0.8	0.4
3			0	0.4	1.0
4				0	0.3
5					0

    Root table

	1	2	3	4
1	1	2	3	3
2		2	3	3
3			3	3
4				4

    a) Test cases

Input: keys[] = {10, 12}, freq[] = {34, 50}

Output = 118

b) Input: keys[] = {10, 12, 20}, freq[] = {34, 8, 50}

Output = 142

16. Consider a set of keys 10,12,16,21 with frequencies 4,2,6,3 and the respective probabilities. Write a Program to construct an OBST in a programming language of your choice. Execute your code and display the resulting OBST, its cost and root matrix.

Input N =4, Keys = {10,12,16,21} Frequencies = {4,2,6,3}

Output : 26

	0	1	2	3
0	4	80	202	262
1		2	102	162
2			6	12
3				3

a) Test cases

Input: keys[] = {10, 12}, freq[] = {34, 50}

Output = 118

b) Input: keys[] = {10, 12, 20}, freq[] = {34, 8, 50}

Output = 142

17. A game on an undirected graph is played by two players, Mouse and Cat, who alternate turns. The graph is given as follows: graph[a] is a list of all nodes b such that ab is an edge of the graph. The mouse starts at node 1 and goes first, the cat starts at node 2 and goes second, and there is a hole at node 0. During each player's turn, they must travel along one edge of the graph that meets where they are. For example, if the Mouse is at node 1, it must travel to any node in graph[1]. Additionally, it is not allowed for the Cat to travel to the Hole (node 0).Then, the game can end in three ways:

If ever the Cat occupies the same node as the Mouse, the Cat wins.

If ever the Mouse reaches the Hole, the Mouse wins.

If ever a position is repeated (i.e., the players are in the same position as a previous turn, and it is the same player's turn to move), the game is a draw.

Given a graph, and assuming both players play optimally, return

1 if the mouse wins the game,

2 if the cat wins the game, or

0 if the game is a draw.

Example 1:

Input: graph = [[2,5],[3],[0,4,5],[1,4,5],[2,3],[0,2,3]]

Output: 0

Example 2:

Input: graph = [[1,3],[0],[3],[0,2]]

Output: 1

18. You are given an undirected weighted graph of n nodes (0-indexed), represented by an edge list where edges[i] = [a, b] is an undirected edge connecting the nodes a and b with a probability of success of traversing that edge succProb[i]. Given two nodes start and end, find the path with the maximum probability of success to go from start to end and return its success probability. If there is no path from start to end, return 0. Your answer will be accepted if it differs from the correct answer by at most 1e-5.

    Example 1:

    Input: n = 3, edges = [[0,1],[1,2],[0,2]], succProb = [0.5,0.5,0.2], start = 0, end = 2

    Output: 0.25000

    Explanation: There are two paths from start to end, one having a probability of success = 0.2 and the other has 0.5 * 0.5 = 0.25.

    Example 2:

    Input: n = 3, edges = [[0,1],[1,2],[0,2]], succProb = [0.5,0.5,0.3], start = 0, end = 2

Output: 0.30000

19. There is a robot on an m x n grid. The robot is initially located at the top-left corner (i.e., grid[0][0]). The robot tries to move to the bottom-right corner (i.e., grid[m - 1][n - 1]). The robot can only move either down or right at any point in time. Given the two integers m and n, return the number of possible unique paths that the robot can take to reach the bottom-right corner. The test cases are generated so that the answer will be less than or equal to 2 * 10 9.

    Example 1:

    START

    FINISH

    Input: m = 3, n = 7

    Output: 28

        Example 2:

        Input: m = 3, n = 2

        Output: 3

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

        1. Right -> Down -> Down

        2. Down -> Down -> Right

        3. Down -> Right -> Down

20. Given an array of integers nums, return the number of good pairs. A pair (i, j) is called good if nums[i] == nums[j] and i < j.

    Example 1:

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

    Output: 4

    Explanation: There are 4 good pairs (0,3), (0,4), (3,4), (2,5) 0-indexed.

    Example 2:

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

Output: 6

Explanation: Each pair in the array are good.

21. There are n cities numbered from 0 to n-1. Given the array edges where edges[i] = [fromi, toi, weighti] represents a bidirectional and weighted edge between cities fromi and toi, and given the integer distanceThreshold. Return the city with the smallest number of cities that are reachable through some path and whose distance is at most distanceThreshold, If there are multiple such cities, return the city with the greatest number. Notice that the distance of a path connecting cities i and j is equal to the sum of the edges' weights along that path.

    Example 1:

    Input: n = 4, edges = [[0,1,3],[1,2,1],[1,3,4],[2,3,1]], distanceThreshold = 4

    Output: 3

    Explanation: The figure above describes the graph.

    The neighboring cities at a distanceThreshold = 4 for each city are:

    City 0 -> [City 1, City 2]

    City 1 -> [City 0, City 2, City 3]

    City 2 -> [City 0, City 1, City 3]

    City 3 -> [City 1, City 2]

    Cities 0 and 3 have 2 neighboring cities at a distance Threshold = 4, but we have to return city 3 since it has the greatest number.

    Example 2:

    Input: n = 5, edges = [[0,1,2],[0,4,8],[1,2,3],[1,4,2],[2,3,1],[3,4,1]], distance Threshold = 2

    Output: 0

    Explanation: The figure above describes the graph.

    The neighboring cities at a distance Threshold = 2 for each city are:

    City 0 -> [City 1]

    City 1 -> [City 0, City 4]

    City 2 -> [City 3, City 4]

City 3 -> [City 2, City 4]

City 4 -> [City 1, City 2, City 3]

The city 0 has 1 neighboring city at a distanceThreshold = 2.

22. You are given a network of n nodes, labeled from 1 to n. You are also given times, a list of travel times as directed edges times[i] = (ui, vi, wi), where ui is the source node, vi is the target node, and wi is the time it takes for a signal to travel from source to target. We will send a signal from a given node k. Return the minimum time it takes for all the n nodes to receive the signal. If it is impossible for all the n nodes to receive the signal, return -1.

    Example 1:

    Input: times = [[2,1,1],[2,3,1],[3,4,1]], n = 4, k = 2

    Output: 2

    Example 2:

    Input: times = [[1,2,1]], n = 2, k = 1

    Output: 1

    Example 3:

    Input: times = [[1,2,1]], n = 2, k = 2

    Output: -1

TOPIC 5 : GREEDY

1. There are 3n piles of coins of varying size, you and your friends will take piles of coins as follows: In each step, you will choose any 3 piles of coins (not necessarily consecutive). Of your choice, Alice will pick the pile with the maximum number of coins. You will pick the next pile with the maximum number of coins. Your friend Bob will pick the last pile. Repeat until there are no more piles of coins. Given an array of integers piles where piles[i] is the number of coins in the ith pile. Return the maximum number of coins that you can have.

    Example 1:

    Input: piles = [2,4,1,2,7,8]

Output: 9

Explanation: Choose the triplet (2, 7, 8), Alice Pick the pile with 8 coins, you the pile with 7 coins and Bob the last one.

Choose the triplet (1, 2, 4), Alice Pick the pile with 4 coins, you the pile with 2 coins and Bob the last one.

The maximum number of coins which you can have is: 7 + 2 = 9.

On the other hand if we choose this arrangement (1, 2, 8), (2, 4, 7) you only get 2 + 4 = 6 coins which is not optimal.

Example 2:

Input: piles = [2,4,5]

Output: 4

2. You are given a 0-indexed integer array coins, representing the values of the coins available, and an integer target. An integer x is obtainable if there exists a subsequence of coins that sums to x. Return the minimum number of coins of any value that need to be added to the array so that every integer in the range [1, target] is obtainable. A subsequence of an array is a new non-empty array that is formed from the original array by deleting some (possibly none) of the elements without disturbing the relative positions of the remaining elements.

Example 1:

Input: coins = [1,4,10], target = 19

Output: 2

Explanation: We need to add coins 2 and 8. The resulting array will be [1, 2, 4, 8, 10].

It can be shown that all integers from 1 to 19 are obtainable from the resulting array, and that 2 is the minimum number of coins that need to be added to the array.

Example 2:

Input: coins = [1, 4, 10, 5, 7, 19], target = 19

Output: 1

Explanation: We only need to add the coin 2. The resulting array will be [1,2, 4, 5, 7, 10, 19].

It can be shown that all integers from 1 to 19 are obtainable from the resulting array, and that 1 is the minimum number of coins that need to be added to the array

.

3. You are given an integer array jobs, where jobs[i] is the amount of time it takes to complete the ith job. There are k workers that you can assign jobs to. Each job should be assigned to exactly one worker. The working time of a worker is the sum of the time it takes to complete all jobs assigned to them. Your goal is to devise an optimal assignment such that the maximum working time of any worker is minimized. Return the minimum possible maximum working time of any assignment.

   Example 1:

   Input: jobs = [3,2,3], k = 3

   Output: 3

   Explanation: By assigning each person one job, the maximum time is 3.

   Example 2:

   Input: jobs = [1,2,4,7,8], k = 2

   Output: 11

   Explanation: Assign the jobs the following way:

   Worker 1: 1, 2, 8 (working time = 1 + 2 + 8 = 11)

   Worker 2: 4, 7 (working time = 4 + 7 = 11)

   The maximum working time is 11.

4. We have n jobs, where every job is scheduled to be done from startTime[i] to endTime[i], obtaining a profit of profit[i]. You're given the startTime,

endTime and profit arrays, return the maximum profit you can take such that there are no two jobs in the subset with overlapping time range. If you choose a job that ends at time X you will be able to start another job that starts at time X.

Example 1:

Input: startTime = [1,2,3,3], endTime = [3,4,5,6], profit = [50,10,40,70]

Output: 120

Explanation: The subset chosen is the first and fourth job.

Time range [1-3]+[3-6] , we get profit of 120 = 50 + 70.

Example 2:

Input: startTime = [1,2,3,4,6], endTime = [3,5,10,6,9], profit = [20,20,100,70,60]

Output: 150

Explanation: The subset chosen is the first, fourth and fifth job. Profit obtained 150 = 20 + 70 + 60.

5. Given a graph represented by an adjacency matrix, implement Dijkstra's Algorithm to find the shortest path from a given source vertex to all other vertices in the graph. The graph is represented as an adjacency matrix where graph[i][j] denote the weight of the edge from vertex i to vertex j. If there is no edge between vertices i and j, the value is Infinity (or a very large number).

   Test Case 1:

   Input:

   n = 5

   graph = [[0, 10, 3, Infinity, Infinity], [Infinity, 0, 1, 2, Infinity], [Infinity, 4, 0, 8, 2],

   [Infinity, Infinity, Infinity, 0, 7], [Infinity, Infinity, Infinity, 9, 0]]

   source = 0

   Output: [0, 7, 3, 9, 5]

Test Case 2:

Input:

n = 4

graph = [[0, 5, Infinity, 10], [Infinity, 0, 3, Infinity], [Infinity, Infinity, 0, 1],

[Infinity, Infinity, Infinity, 0]]

source = 0

Output: [0, 5, 8, 9]

6. Given a graph represented by an edge list, implement Dijkstra's Algorithm to find the shortest path from a given source vertex to a target vertex. The graph is represented as a list of edges where each edge is a tuple (u, v, w) representing an edge from vertex u to vertex v with weight w.

   Test Case 1:

   Input:

   n = 6

   edges = [(0, 1, 7), (0, 2, 9), (0, 5, 14), (1, 2, 10), (1, 3, 15),

   (2, 3, 11), (2, 5, 2), (3, 4, 6), (4, 5, 9)]

   source = 0

   target = 4

   Output: 20

   Test Case 2:

   Input:

   n = 5

   edges = [(0, 1, 10), (0, 4, 3), (1, 2, 2), (1, 4, 4), (2, 3, 9), (3, 2, 7), (4, 1, 1), (4, 2, 8), (4, 3, 2)]

   source = 0

   target = 3

   Output: 8

7. Given a set of characters and their corresponding frequencies, construct the Huffman Tree and generate the Huffman Codes for each character.

   Test Case 1:

   Input:

   n = 4

   characters = ['a', 'b', 'c', 'd']

   frequencies = [5, 9, 12, 13]

   Output: [('a', '110'), ('b', '10'), ('c', '0'), ('d', '111')]

   Test Case 2:

   Input:

   n = 6

   characters = ['f', 'e', 'd', 'c', 'b', 'a']

   frequencies = [5, 9, 12, 13, 16, 45]

   Output: [('a', '0'), ('b', '101'), ('c', '100'), ('d', '111'), ('e', '1101'), ('f', '1100')]

8. Given a Huffman Tree and a Huffman encoded string, decode the string to get the original message.

   Test Case 1:

   Input:

   n = 4

   characters = ['a', 'b', 'c', 'd']

   frequencies = [5, 9, 12, 13]

   encoded_string = '1101100111110'

   Output: "abacd"

   Test Case 2:

   Input:

   n = 6

   characters = ['f', 'e', 'd', 'c', 'b', 'a']

frequencies = [5, 9, 12, 13, 16, 45]

encoded_string = '1100110111001011110010 11'

Output: "fcbade"

9. Given a list of item weights and the maximum capacity of a container, determine the maximum weight that can be loaded into the container using a greedy approach. The greedy approach should prioritize loading heavier items first until the container reaches its capacity.

   Test Case 1:

   Input:

   n = 5

   weights = [10, 20, 30, 40, 50]

   max_capacity = 60

   Output: 50

   Test Case 2:

   Input:

   n = 6

   weights = [5, 10, 15, 20, 25, 30]

   max_capacity = 50

   Output: 50

10. Given a list of item weights and a maximum capacity for each container, determine the minimum number of containers required to load all items using a greedy approach. The greedy approach should prioritize loading items into the current container until it is full before moving to the next container.

    Test Case 1:

    Input:

    n = 7

    weights = [5, 10, 15, 20, 25, 30, 35]

max_capacity = 50

Output: 4

Test Case 2:

Input:

n = 8

weights = [10, 20, 30, 40, 50, 60, 70, 80]

max_capacity = 100

Output: 6

11. Given a graph represented by an edge list, implement Kruskal's Algorithm to find the Minimum Spanning Tree (MST) and its total weight.

    Test Case 1:

    Input:

    n = 4

    m = 5

    edges = [(0, 1, 10), (0, 2, 6), (0, 3, 5), (1, 3, 15), (2, 3, 4)]

    Output:

    Edges in MST: [(2, 3, 4), (0, 3, 5), (0, 1, 10)]

    Total weight of MST: 19

    Test Case 2:

    Input:

    n = 5

    m = 7

    edges = [(0, 1, 2), (0, 3, 6), (1, 2, 3), (1, 3, 8), (1, 4, 5), (2, 4, 7), (3, 4, 9)]

    Output:

    Edges in MST: [(0, 1, 2), (1, 2, 3), (1, 4, 5), (0, 3, 6)]

    Total weight of MST: 16

12. Given a graph with weights and a potential Minimum Spanning Tree (MST), verify if the given MST is unique. If it is not unique, provide another possible MST.

    Test Case 1:

    Input:

    n = 4

    m = 5

    edges = [(0, 1, 10), (0, 2, 6), (0, 3, 5), (1, 3, 15), (2, 3, 4)]

    given_mst = [(2, 3, 4), (0, 3, 5), (0, 1, 10)]

    Output: Is the given MST unique? True

    Test Case 2:

    Input:

    n = 5

    m = 6

    edges = [(0, 1, 1), (0, 2, 1), (1, 3, 2), (2, 3, 2), (3, 4, 3), (4, 2, 3)]

    given_mst = [(0, 1, 1), (0, 2, 1), (1, 3, 2), (3, 4, 3)]

    Output: Is the given MST unique? False

    Another possible MST: [(0, 1, 1), (0, 2, 1), (2, 3, 2), (3, 4, 3)]

    Total weight of MST: 7

TOPIC 6 : BACKTRACKING

1. Discuss the importance of visualizing the solutions of the N-Queens Problem to understand the placement of queens better. Use a graphical representation to show how queens are placed on the board for different values of N. Explain how visual tools can help in debugging the algorithm and gaining insights into the problem's complexity. Provide examples of visual representations for N = 4, N = 5, and N = 8, showing different valid solutions.

    Visualization for 4-Queens:

Input: N = 4

Output:

Explanation: Each 'Q' represents a queen, and '.' represents an empty space.

b. Visualization for 5-Queens:

Input: N = 5

Output:

c. Visualization for 8-Queens:

Input: N = 8

Output:

2. Discuss the generalization of the N-Queens Problem to other board sizes and shapes, such as rectangular boards or boards with obstacles. Explain how the algorithm can be adapted to handle these variations and the additional constraints they introduce. Provide examples of solving generalized N-Queens Problems for different board configurations, such as an 8×10 board, a 5×5 board with obstacles, and a 6×6 board with restricted positions.

8×10 Board:

8 rows and 10 columns

Output: Possible solution [1, 3, 5, 7, 9, 2, 4, 6]

Explanation: Adapt the algorithm to place 8 queens on an 8×10 board, ensuring no two queens threaten each other.

b. 5×5 Board with Obstacles:

Input: N = 5, Obstacles at positions [(2, 2), (4, 4)]

Output: Possible solution [1, 3, 5, 2, 4]

Explanation: Modify the algorithm to avoid placing queens on obstacle positions, ensuring a valid solution that respects the constraints.

c. 6×6 Board with Restricted Positions:

Input: N = 6, Restricted positions at columns 2 and 4 for the first queen

Output: Possible solution [1, 3, 5, 2, 4, 6]

Explanation: Adjust the algorithm to handle restricted positions, ensuring the queens are placed without conflicts and within allowed columns.

3. 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"]]

4. 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"]]

5. You are given an integer array nums and an integer target. You want to build an expression out of nums by adding one of the symbols '+' and '-' before each integer in nums and then concatenate all the integers.For example, if nums = [2, 1], you can add a '+' before 2 and a '-' before 1 and concatenate them to build the expression "+2-1" Return the number of different expressions that you can build, which evaluates to target.

   Example 1:

   Input: nums = [1,1,1,1,1], target = 3

   Output: 5

   Explanation: There are 5 ways to assign symbols to make the sum of nums be target 3.

   -1 + 1 + 1 + 1 + 1 = 3

   +1 - 1 + 1 + 1 + 1 = 3

   +1 + 1 - 1 + 1 + 1 = 3

   +1 + 1 + 1 - 1 + 1 = 3

   +1 + 1 + 1 + 1 - 1 = 3

   Example 2:

   Input: nums = [1], target = 1

   Output: 1

6. Given an array of integers arr, find the sum of min(b), where b ranges over every (contiguous) subarray of arr. Since the answer may be large, return the answer modulo 109 + 7.

   Example 1:

   Input: arr = [3,1,2,4]

   Output: 17

   Explanation:

   Subarrays are [3], [1], [2], [4], [3,1], [1,2], [2,4], [3,1,2], [1,2,4], [3,1,2,4].

   Minimums are 3, 1, 2, 4, 1, 1, 2, 1, 1, 1.

   Sum is 17.

Example 2:

Input: arr = [11,81,94,43,3]

Output: 444

7. 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]]

   4. COMBINATION SUM 2:

8. Given a collection of candidate numbers (candidates) and a target number (target), find all unique combinations in candidates where the candidate numbers sum to target. Each number in candidates may only be used once in the combination. The solution set must not contain duplicate combinations.

   Example 1:

   Input: candidates = [10,1,2,7,6,1,5], target = 8

   Output:

[

[1,1,6],

[1,2,5],

[1,7],

[2,6]

]

Example 2:

Input: candidates = [2,5,2,1,2], target = 5

Output:

[

[1,2,2],

[5]

]

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

   Example 1:

   Input: nums = [1,2,3]

   Output: [[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]

   Example 2:

   Input: nums = [0,1]

   Output: [[0,1],[1,0]]

   Example 3:

   Input: nums = [1]

   Output: [[1]]

10. Given a collection of numbers, nums, that might contain duplicates, return all possible unique permutations in any order.

Example 1:

Input: nums = [1,1,2]

Output:

[[1,1,2],

[1,2,1],

[2,1,1]]

Example 2:

Input: nums = [1,2,3]

Output: [[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]

11. You and your friends are assigned the task of coloring a map with a limited number of colors. The map is represented as a list of regions and their adjacency relationships. The rules are as follows: At each step, you can choose any uncolored region and color it with any available color. Your friend Alice follows the same strategy immediately after you, and then your friend Bob follows suit. You want to maximize the number of regions you personally color. Write a function that takes the map's adjacency list representation and returns the maximum number of regions you can color before all regions are colored. Write a program to implement the Graph coloring technique for an undirected graph. Implement an algorithm with minimum number of colors. edges = [(0, 1), (1, 2), (2, 3), (3, 0), (0, 2)] No. of vertices, n = 4

Input:

- Number of vertices: `n = 4`
- Edges: `[(0, 1), (1, 2), (2, 3), (3, 0), (0, 2)]`
- Number of colors: `k = 3`

Output:

- Maximum number of regions you can color: `2`

12. You are given an undirected graph represented by a list of edges and the number of vertices n. Your task is to determine if there exists a Hamiltonian cycle in the graph. A Hamiltonian cycle is a cycle that visits each vertex exactly once and returns to the starting vertex. Write a function that takes the list of edges and the number of vertices as input and returns true if there exists a Hamiltonian cycle

in the graph, otherwise return false. Example: Given edges = [(0, 1), (1, 2), (2, 3), (3, 0), (0, 2), (2, 4), (4, 0)] and n = 5

Input:

- Number of vertices: `n = 5`
- Edges: `[(0, 1), (1, 2), (2, 3), (3, 0), (0, 2), (2, 4), (4, 0)]`

Output:

- Hamiltonian Cycle Exists: `True` (Example cycle: `0 -> 1 -> 2 -> 4 -> 3 -> 0`)

14. You are given an undirected graph represented by a list of edges and the number of vertices n. Your task is to determine if there exists a Hamiltonian cycle in the graph. A Hamiltonian cycle is a cycle that visits each vertex exactly once and returns to the starting vertex. Write a function that takes the list of edges and the number of vertices as input and returns true if there exists a Hamiltonian cycle in the graph, otherwise return false. Example:edges = [(0, 1), (1, 2), (2, 3), (3, 0), (0, 2)] and n = 4

Input:

- Number of vertices: `n = 4`
- Edges: `[(0, 1), (1, 2), (2, 3), (3, 0), (0, 2)]`

Output:

- Hamiltonian Cycle Exists: `True` (Example cycle: `0 -> 1 -> 2 -> 3 -> 0`)

15. You are tasked with designing an efficient coading to generate all subsets of a given set S containing n elements. Each subset should be outputted in lexicographical order. Return a list of lists where each inner list is a subset of the given set. Additionally, find out how your coading handles duplicate elements in S. A = [1, 2, 3] The subsets of [1, 2, 3] are: [], [1], [2], [3], [1, 2], [1, 3], [2, 3], [1, 2, 3]

Input:

- Set: `A = [1, 2, 3]`

Output:

- Subsets: `[[], [1], [2], [3], [1, 2], [1, 3], [2, 3], [1, 2, 3]]`
- Handling of duplicates: If `A` contained duplicates (e.g., `[1, 2, 2]`), subsets would include duplicates unless duplicates are removed.

16. Write a program to implement the concept of subset generation. Given a set of unique integers and a specific integer 3, generate all subsets that contain the element 3. Return a list of lists where each inner list is a subset containing the element 3 E = [2, 3, 4, 5], x = 3, The subsets containing 3 : [3], [2, 3], [3, 4], [3,5], [2, 3, 4], [2, 3, 5], [3, 4, 5], [2, 3, 4, 5] Given an integer array nums of unique elements, return all possible subsets(the power set). The solution set must not contain duplicate subsets. Return the solution in any order.

    Example 1:

    Input: nums = [1,2,3]

    Output: [[],[1],[2],[1,2],[3],[1,3],[2,3],[1,2,3]]

    Example 2:

    Input: nums = [0]

    Output: [[],[0]]

17. You are given two string arrays words1 and words2. A string b is a subset of string a if every letter in b occurs in a including multiplicity. For example, "wrr" is a subset of "warrior" but is not a subset of "world". A string a from words1 is universal if for every string b in words2, b is a subset of a. Return an array of all the universal strings in words1. You may return the answer in any order.

    Example 1:

    Input: words1 = ["amazon","apple","facebook","google","leetcode"], words2 = ["e","o"]

    Output: ["facebook","google","leetcode"]

    Example 2:

    Input: words1 = ["amazon","apple","facebook","google","leetcode"], words2 = ["l","e"]

    Output: ["apple","google","leetcode"]

TOPIC 7:TRACTABILITY AND APPROXIMATION ALGORITHM

1.Implement a program to verify if a given problem is in class P or NP. Choose a specific decision problem (e.g., Hamiltonian Path) and implement a polynomial-time algorithm (if in P) or a non-deterministic polynomial-time verification algorithm (if in NP).

Input:

- Graph G with vertices V = {A, B, C, D} and edges E = {(A, B), (B, C), (C, D), (D, A)}

Output:

- Hamiltonian Path Exists: True (Path: A -> B -> C -> D)

2.Implement a solution to the 3-SAT problem and verify its NP-Completeness. Use a known NP-Complete problem (e.g., Vertex Cover) to reduce it to the 3-SAT problem.

Input:

- 3-SAT Formula: (x1 ∨ x2 ∨ ¬x3) ∧ (¬x1 ∨ x2 ∨ x4) ∧ (x3 ∨ ¬x4 ∨ x5)
- Reduction from Vertex Cover: Vertex Cover instance with V = {1, 2, 3, 4, 5}, E = {(1,2), (1,3), (2,3), (3,4), (4,5)}

Output:

- Satisfiability: True (Example satisfying assignment: x1 = True, x2 = True, x3 = False, x4 = True, x5 = False)
- NP-Completeness Verification: Reduction successful from Vertex Cover to 3-SAT

3.Implement an approximation algorithm for the Vertex Cover problem. Compare the performance of the approximation algorithm with the exact solution obtained through brute-force. Consider the following graph G=(V,E) where V={1,2,3,4,5} and E={(1,2),(1,3),(2,3),(3,4),(4,5).

Input:

- Graph G = (V, E) with V = {1, 2, 3, 4, 5}, E = {(1,2), (1,3), (2,3), (3,4), (4,5)}

Output:

- Approximation Vertex Cover: {2, 3, 4}
- Exact Vertex Cover (Brute-Force): {2, 4}
- Performance Comparison: Approximation solution is within a factor of 1.5 of the optimal solution.

4.Implement a greedy approximation algorithm for the Set Cover problem. Analyze its performance on different input sizes and compare it with the optimal solution. Consider the following universe U={1,2,3,4,5,6,7} and sets ={{1,2,3},{2,4},{3,4,5,6},{4,5},{5,6,7},{6,7}}

Input:

- Universe U = {1, 2, 3, 4, 5, 6, 7}
- Sets S = {{1, 2, 3}, {2, 4}, {3, 4, 5, 6}, {4, 5}, {5, 6, 7}, {6, 7}}

Output:

- Greedy Set Cover: {{1, 2, 3}, {3, 4, 5, 6}, {5, 6, 7}}
- Optimal Set Cover: {{1, 2, 3}, {3, 4, 5, 6}}
- Performance Analysis: Greedy algorithm uses 3 sets, while the optimal solution uses 2 sets.

5.Implement a heuristic algorithm (e.g., First-Fit, Best-Fit) for the Bin Packing problem. Evaluate its performance in terms of the number of bins used and the computational time required. Consider a list of item weights {4,8,1,4,2,1}and a bin capacity of 10.

Input:

- List of item weights: {4, 8, 1, 4, 2, 1}
- Bin capacity: 10

Output:

- Number of Bins Used: 3
- Bin Packing: Bin 1: [4, 4, 2], Bin 2: [8, 1, 1], Bin 3: [1]
- Computational Time: O(n)

6.Implement an approximation algorithm for the Maximum Cut problem using a greedy or randomized approach. Compare the results with the optimal solution obtained through an exhaustive search for small graph instances.

Input:

- Graph G = (V, E) with V = {1, 2, 3, 4}, E = {(1,2), (1,3), (2,3), (2,4), (3,4)}
- Edge Weights: w(1,2) = 2, w(1,3) = 1, w(2,3) = 3, w(2,4) = 4, w(3,4) = 2

Output:

- Greedy Maximum Cut: Cut = {(1,2), (2,4)}, Weight = 6
- Optimal Maximum Cut (Exhaustive Search): Cut = {(1,2), (2,4), (3,4)}, Weight = 8
- Performance Comparison: Greedy solution achieves 75% of the optimal weight.