CS 351 – DAA Lab Problems

Lab Cycle - I

Divide and Conquer

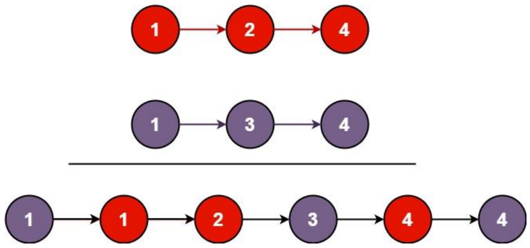
1. Merge Two Sorted Lists

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

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

Return *the head of the merged linked list*.

Example 1:



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

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

Example 2:

Input: list1 = [], list2 = []

Output: []

Example 3:

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

Output: [0]

Constraints:

- The number of nodes in both lists is in the range [0, 50].
- -100 <= Node.val <= 100
- Both list1 and list2 are sorted in **non-decreasing** order.

Link: https://leetcode.com/problems/merge-two-sorted-lists/

2. Merge Sorted Array

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

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

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

Example 1:

Input: nums1 = [1,2,3,0,0,0], m = 3, nums2 = [2,5,6], n = 3

Output: [1,2,2,3,5,6]

Explanation: The arrays we are merging are [1,2,3] and [2,5,6].

The result of the merge is [1,2,2,3,5,6] with the underlined elements coming from nums1.

Example 2:

Input: nums1 = [1], m = 1, nums2 = [], n = 0

Output: [1]

Explanation: The arrays we are merging are [1] and [].

The result of the merge is [1].

Example 3:

Input: nums1 = [0], m = 0, nums2 = [1], n = 1
Output: [1]

Explanation: The arrays we are merging are [] and [1].

The result of the merge is [1].

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

Constraints:

- nums1.length == m + n
- nums2.length == n
- 0 <= m, n <= 200
- 1 <= m + n <= 200
- -10^9 <= nums1[i], nums2[j] <= 10^9

Link: https://leetcode.com/problems/merge-sorted-array/

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]

Explanation: The linked-lists are:

[

1->4->5,

1->3->4,

2->6

]

merging them into one sorted list:

1->1->2->3->4->4->5->6

Example 2:

Input: lists = []

Output: []

Example 3:

Input: lists = [[]]

Output: []

Constraints:

- k == lists.length
- 0 <= k <= 10^4
- 0 <= lists[i].length <= 500
- -10^4 <= lists[i][j] <= 10^4
- lists[i] is sorted in **ascending order**.
- The sum of lists[i].length will not exceed 10^4.

Link: https://leetcode.com/problems/merge-k-sorted-lists/

4. Sort an Array

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.

Example 1:

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

Output: [1,2,3,5]

Explanation: After sorting the array, the positions of some numbers are not changed (for example, 2 and 3), while the positions of other numbers are changed (for example, 1 and 5).

Example 2:

Input: nums = [5,1,1,2,0,0]

Output: [0,0,1,1,2,5]

Explanation: Note that the values of nums are not necessairly unique.

Constraints:

- 1 <= nums.length <= 5 * 10^4
- -5 * 10^4 <= nums[i] <= 5 * 10^4

Link: https://leetcode.com/problems/sort-an-array/

Lab Cycle - II

Greedy Programming

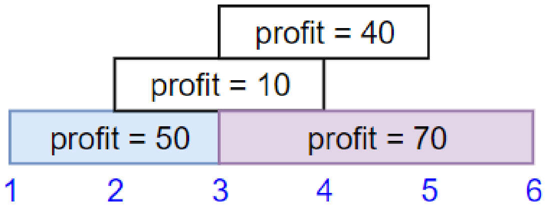
1. Maximum Profit in Job Scheduling

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.

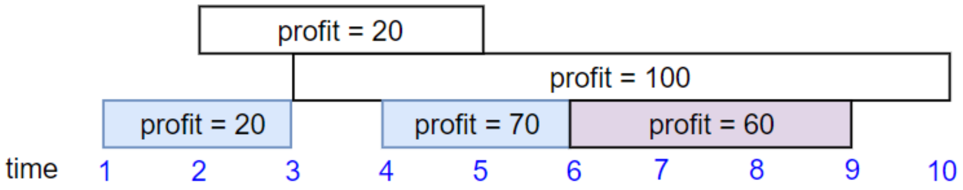
Example 1:



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.

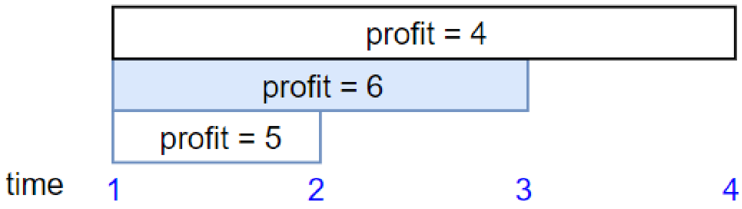
Example 2:



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.

Example 3:



Input: startTime = [1,1,1], endTime = [2,3,4], profit = [5,6,4]

Output: 6

Constraints:

- 1 <= startTime.length == endTime.length == profit.length <= 5 * 10^4
- 1 <= startTime[i] < endTime[i] <= 10^9
- 1 <= profit[i] <= 10^4

Link: https://leetcode.com/problems/maximum-profit-in-job-scheduling/

2. Coin Change

You are given an integer array coin representing coins of different denominations and an integer amount representing a total amount of money.

Return *the fewest number of coins that you need to make up that amount*. If that amount of money cannot be made up by any combination of the coins, return -1.

You may assume that you have an infinite number of each kind of coin.

Example 1:

Input: coins = [1,2,5], amount = 11

Output: 3

Explanation: 11 = 5 + 5 + 1

Example 2:

Input: coins = [2], amount = 3

Output: -1

Example 3:

Input: coins = [1], amount = 0

Output: 0

Constraints:

- 1 <= coins.length <= 12
- 1 <= coins[i] <= 2^{31} - 1
- 0 <= amount <= 10^4

Link: https://leetcode.com/problems/coin-change/

3. Maximum Units on a Truck

You are assigned to put some amount of boxes onto **one truck**. You are given a 2D array boxTypes, where boxTypes[i] = [numberOfBoxes_i, $\text{numberOfUnitsPerBox}_i$]:

- numberOfBoxes_i is the number of boxes of type i.
- $\text{numberOfUnitsPerBox}_i$ is the number of units in each box of the type i.

You are also given an integer truckSize, which is the **maximum** number of **boxes** that can be put on the truck. You can choose any boxes to put on the truck as long as the number of boxes does not exceed truckSize.

Return *the **maximum** total number of **units** that can be put on the truck.*

Example 1:

Input: boxTypes = [[1,3],[2,2],[3,1]], truckSize = 4

Output: 8

Explanation: There are:

- 1 box of the first type that contains 3 units.
- 2 boxes of the second type that contain 2 units each.
- 3 boxes of the third type that contain 1 unit each.

You can take all the boxes of the first and second types, and one box of the third type.

The total number of units will be = (1 * 3) + (2 * 2) + (1 * 1) = 8.

Example 2:

Input: boxTypes = [[5,10],[2,5],[4,7],[3,9]], truckSize = 10

Output: 91

Constraints:

- 1 <= boxTypes.length <= 1000
- 1 <= numberOfBoxes_i, $\text{numberOfUnitsPerBox}_i$ <= 1000
- 1 <= truckSize <= 10^6

Link: https://leetcode.com/problems/maximum-units-on-a-truck/

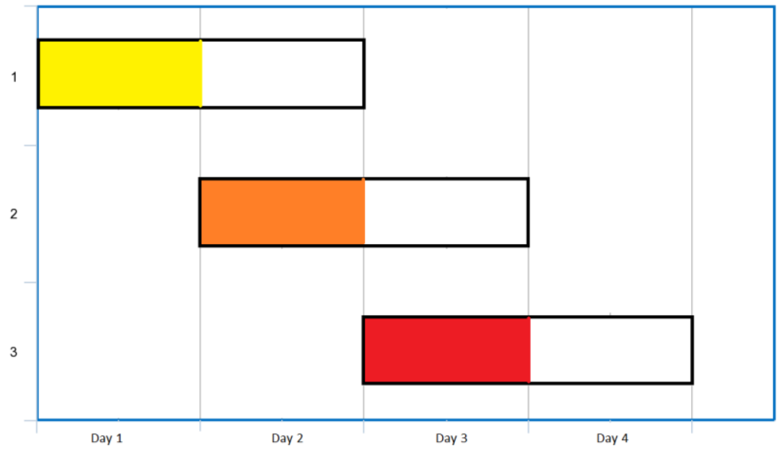
4. Maximum Number of Events That Can Be Attended

You are given an array of events where events[i] = [startDay$_i$, endDay$_i$]. Every event i starts at startDay$_i$ and ends at endDay$_i$.

You can attend an event i at any day d where startTime$_i$ <= d <= endTime$_i$. You can only attend one event at any time d.

Return *the maximum number of events you can attend*.

Example 1:



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

Output: 3

Explanation: You can attend all the three events.

One way to attend them all is as shown.

Attend the first event on day 1.
Attend the second event on day 2.
Attend the third event on day 3.

Example 2:

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

Output: 4

Constraints:

- 1 <= events.length <= 10^5
- events[i].length == 2
- 1 <= startDay$_i$ <= endDay$_i$ <= 10^5

Link: https://leetcode.com/problems/maximum-number-of-events-that-can-be-attended/

Lab Cycle - III

Dynamic Programming & Graphs

1. Reverse Words in a String

Given an input string s, reverse the order of the **words**.

A **word** is defined as a sequence of non-space characters. The **words** in s will be separated by at least one space.

Return *a string of the words in reverse order concatenated by a single space.*

Note that s may contain leading or trailing spaces or multiple spaces between two words. The returned string should only have a single space separating the words. Do not include any extra spaces.

Example 1:

Input: s = "the sky is blue"

Output: "blue is sky the"

Example 2:

Input: s = " hello world "

Output: "world hello"

Explanation: Your reversed string should not contain leading or trailing spaces.

Example 3:

Input: s = "a good   example"

Output: "example good a"

Explanation: You need to reduce multiple spaces between two words to a single space in the reversed string.

Constraints:

- $1 <= s.length <= 10^4$
- s contains English letters (upper-case and lower-case), digits, and spaces ' '.
- There is **at least one** word in s.

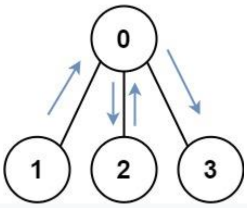
Link: https://leetcode.com/problems/reverse-words-in-a-string/

2. Shortest Path Visiting All Nodes

You have an undirected, connected graph of n nodes labeled from 0 to n - 1. You are given an array graph where graph[i] is a list of all the nodes connected with node i by an edge.

Return *the length of the shortest path that visits every node*. You may start and stop at any node, you may revisit nodes multiple times, and you may reuse edges.

Example 1:

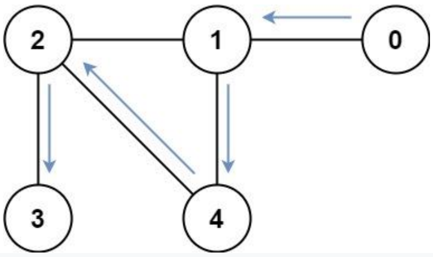


Input: graph = [[1,2,3],[0],[0],[0]]

Output: 4

Explanation: One possible path is [1,0,2,0,3]

Example 2:



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

Output: 4

Explanation: One possible path is [0,1,4,2,3]

Constraints:

- n == graph.length
- 1 <= n <= 12
- 0 <= graph[i].length < n
- graph[i] does not contain i.
- If graph[a] contains b, then graph[b] contains a.
- The input graph is always connected.

Link: https://leetcode.com/problems/shortest-path-visiting-all-nodes/

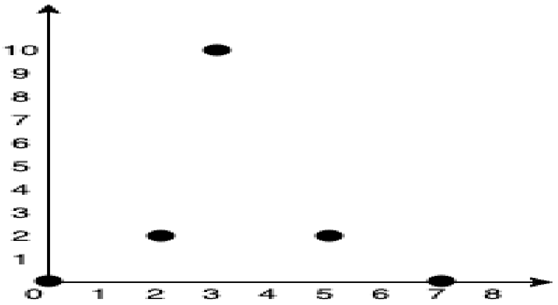
3. Min Cost to Connect All Points

You are given an array points representing integer coordinates of some points on a 2D-plane, where points[i] = [x_i, y_i].

The cost of connecting two points [x_i, y_i] and [x_j, y_j] is the **manhattan distance** between them: $|x_i - x_j| + |y_i - y_j|$, where |val| denotes the absolute value of val.

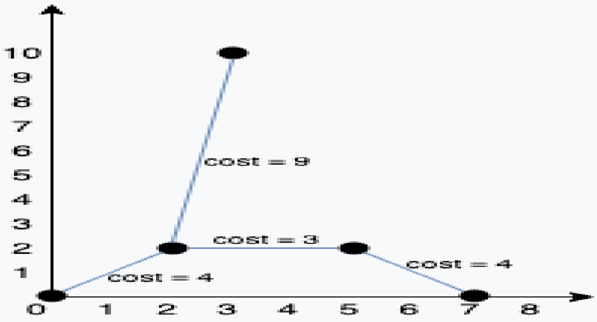
Return *the minimum cost to make all points connected.* All points are connected if there is **exactly one** simple path between any two points.

Example 1:



Input: points = [[0,0],[2,2],[3,10],[5,2],[7,0]]
Output: 20

Explanation:



We can connect the points as shown above to get the minimum cost of 20.
Notice that there is a unique path between every pair of points.

Example 2:

Input: points = [[3,12],[-2,5],[-4,1]]
Output: 18

Constraints:

- 1 <= points.length <= 1000
- $-10^6 <= x_i, y_i <= 10^6$
- All pairs (x_i, y_i) are distinct.

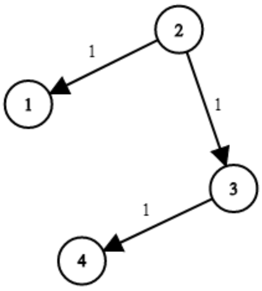
Link: https://leetcode.com/problems/min-cost-to-connect-all-points/

4. Network Delay Time

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] = (u_i, v_i, w_i), where u_i is the source node, v_i is the target node, and w_i 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

Constraints:

- 1 <= k <= n <= 100
- 1 <= times.length <= 6000
- times[i].length == 3
- 1 <= u_i, v_i <= n
- u_i != v_i
- 0 <= w_i <= 100
- All the pairs (u_i, v_i) are **unique**. (i.e., no multiple edges.)

Link: https://leetcode.com/problems/network-delay-time/

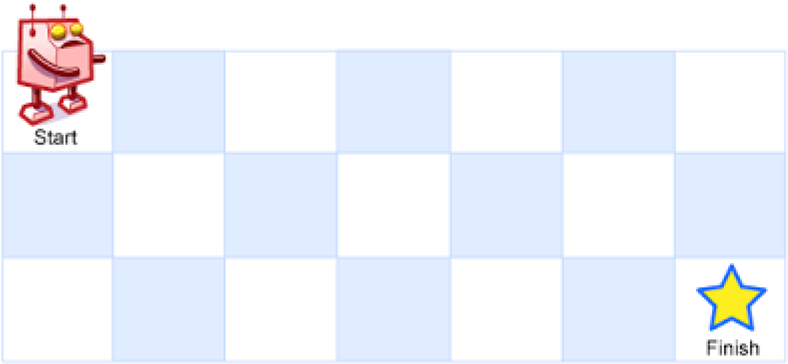
5. Unique Paths

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:



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

Constraints:

- 1 <= m, n <= 100

Link: https://leetcode.com/problems/unique-paths/

Lab Cycle - IV

Backtracking & String Matching

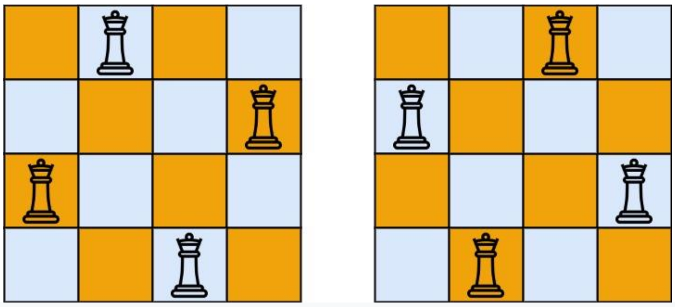
1. N-Queens

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

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

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

Example 1:



Input: n = 4

Output: [[".Q..","...Q","Q...","..Q."],["..Q.","Q...","...Q",".Q.."]]

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

Example 2:

Input: n = 1

Output: [["Q"]]

Constraints:

- 1 <= n <= 9

Link: https://leetcode.com/problems/n-queens/

2. Longest Substring Without Repeating Characters

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

Example 1:

Input: s = "abcabcbb"

Output: 3

Explanation: The answer is "abc", with the length of 3.

Example 2:

Input: s = "bbbbb"

Output: 1

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

Example 3:

Input: s = "pwwkew"

Output: 3

Explanation: The answer is "wke", with the length of 3.

Notice that the answer must be a substring, "pwke" is a subsequence and not a substring.

Constraints:

- $0 <= \text{s.length} <= 5 * 10^4$
- s consists of English letters, digits, symbols and spaces.

Link: https://leetcode.com/problems/longest-substring-without-repeating-characters/

3. Longest Common Prefix

Write a function to find the longest common prefix string amongst an array of strings.

If there is no common prefix, return an empty string "".

Example 1:

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

Example 2:

Input: strs = ["dog","racecar","car"]
Output: ""

Explanation: There is no common prefix among the input strings.

Constraints:

- `1 <= strs.length <= 200`
- `0 <= strs[i].length <= 200`
- `strs[i]` consists of only lowercase English letters.

Link: https://leetcode.com/problems/longest-common-prefix/