SABUDH

1. Inserting at the Beginning

This involves pointing the next pointer of the new data node to the current head of the linked list. So the current head of the linked list becomes the second data element, and the new node becomes the head of the linked list.

TESTCASE 1:

Input:
list: 2->3->4->5
value: 1
Output:
1->2->3->4->5

TEST CASE 2:

Input:
list: 3->2->5->7->1->2
value: 9
Output:
9->3->2->5->7->1->2

2. Maximum Twin Sum of A Linked List

Given a linked list of length N, where N is even. The task is to maximize the sum of two equidistant nodes from the front and back ends of the given linked list.

Note: As explained in the lectures, two nodes (i and j) are equidistant from both ends if the distance of ith node from front is same as distance of the jth node from back.

Input: lis = {5, 4, 2, 1}
Output: 6

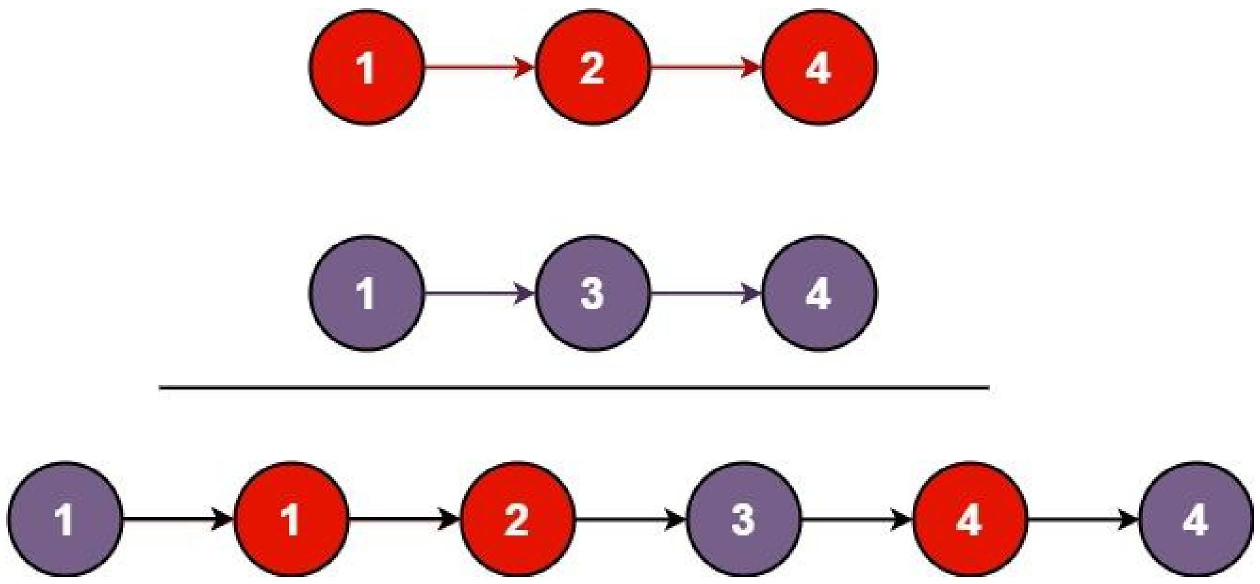
Input: lis = {4, 2, 2, 3}
Output: 7

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

Merge the two lists into 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.

4. Given an integer array nums, in which exactly two elements appear only once and all the other elements appear exactly twice. Find the two elements that appear only once. You can return the answer in any order.

You must write an algorithm that runs in linear runtime complexity and uses only constant extra space.

Example 1

Input: nums= [1,2,1,3,2,5]
Output: [3,5]
Explanation: [5,3] is also a valid answer

Example 2:

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

Example 3:

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

Constraints:

- <= nums.length <= 3 * 104
- -231 <= nums[i] <= 231 – 1
- Each integer in nums will appear twice; only two integers will appear once.

5. Given two sparse vectors, compute their dot product.

Implement class SparseVector:

- SparseVector(nums) Initialises the object with the vector nums
- dotProduct(vec) Compute the dot product between the instance of *SparseVector* and vec

A sparse vector is a vector that has mostly zero values; you should store the sparse vector efficiently and compute the dot product between two *SparseVector*.

Follow-up: What if only one of the vectors is sparse?

Example 1:
Input: nums1 = [1,0,0,2,3], nums2 = [0,3,0,4,0]
Output: 8
Explanation: v1 = SparseVector(nums1) , v2 = SparseVector(nums2)
v1.dotProduct(v2) = 1*0 + 0*3 + 0*0 + 2*4 + 3*0 = 8

Example 2:
Input: nums1 = [0,1,0,0,0], nums2 = [0,0,0,0,2]
Output: 0
Explanation: v1 = SparseVector(nums1) , v2 = SparseVector(nums2)
v1.dotProduct(v2) = 0*0 + 1*0 + 0*0 + 0*0 + 0*2 = 0

Example 3:
Input: nums1 = [0,1,0,0,2,0,0], nums2 = [1,0,0,0,3,0,4]
Output: 6

Constraints:

- n == nums1.length == nums2.length
- 1 <= n <= 10^5
- 0 <= nums1[i], nums2[i] <= 100

6. Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it can trap after raining.

Example 1:

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

Explanation: The above elevation map (black section) is represented by array [0,1,0,2,1,0,1,3,2,1,2,1]. In this case, 6 units of rain water (blue section) are being trapped.

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

Constraints:

- n == height.length
- 1 <= n <= 2 * 104
- 0 <= height[i] <= 105

7. Given an integer array nums, find the subarray with the largest sum, and return *its sum.*

Example 1:
Input: nums = [-2,1,-3,4,-1,2,1,-5,4]
Output: 6
Explanation: The subarray [4,-1,2,1] has the largest sum 6.

Example 2:
Input: nums = [1]
Output: 1
Explanation: The subarray [1] has the largest sum 1.

Example 3:

Input: nums = [5,4,-1,7,8]
Output: 23
Explanation: The subarray [5,4,-1,7,8] has the largest sum 23.

Constraints:

- 1 <= nums.length <= 105
- -104 <= nums[i] <= 104

Follow up: If you have figured out the O(n) solution, try coding another solution using the divide and conquer approach, which is more subtle.