Array assignment questions

GitHub location to update the solution:

https://github.com/dracha-sf/j2e-fy24

1. Please clone the above repo.
2. 2.Create a branch as student/<collegeName>/studentName
3. For example student/bansasthali/riyaz

Problem 1:
There are `n` kids with candies. You are given an integer array `candies`, where each `candies[i]` represents the number of candies the `i`th kid has, and an integer `extraCandies`, denoting the number of extra candies that you have.
Return *a boolean array* `result` *of length* `n`*, where* `result[i]` *is* `true` *if, after giving the* `i`th *kid all the* `extraCandies`*, they will have the* ***greatest*** *number of candies among all the kids, or* `false` *otherwise*.
Note that **multiple** kids can have the **greatest** number of candies.

Example 1

```
Input: candies = [2,3,5,1,3], extraCandies = 3
Output: [true,true,true,false,true]
Explanation: If you give all extraCandies to:
- Kid 1, they will have 2 + 3 = 5 candies, which is the greatest among the kids.
- Kid 2, they will have 3 + 3 = 6 candies, which is the greatest among the kids.
- Kid 3, they will have 5 + 3 = 8 candies, which is the greatest among the kids.
- Kid 4, they will have 1 + 3 = 4 candies, which is not the greatest among the kids.
- Kid 5, they will have 3 + 3 = 6 candies, which is the greatest among the kids.
```

Example 2

```
Input: candies = [4,2,1,1,2], extraCandies = 1
Output: [true,false,false,false,false]
Explanation: There is only 1 extra candy.
Kid 1 will always have the greatest number of candies, even if a different kid is give
```

Problem 2:

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

Example 1:

```
Input: height = [1,8,6,2,5,4,8,3,7]
Output: 49
```

```
Explanation: The above vertical lines are represented by array [1,8,6,2,5,4,8,3,7]. I
```

Example 2:

```
Input: height = [1,1]
Output: 1
```

Problem 3:

Given an integer array `nums` of length `n` and an integer `target`, find three integers in `nums` such that the sum is closest to `target`.
Return *the sum of the three integers*.
You may assume that each input would have exactly one solution.

Example 1:

```
Input: nums = [-1,2,1,-4],
target = 1
Output: 2
Explanation: The sum that is closest to the target is 2. (-1 + 2 + 1 = 2).
```

Example 2:

```
Input: nums = [0,0,0],
target = 1
Output: 0
Explanation: The sum that is closest to the target is 0. (0 + 0 + 0 = 0).
```

Problem 4:

A **permutation** of an array of integers is an arrangement of its members into a sequence or linear order.

- For example, for `arr = [1,2,3]`, the following are all the permutations of `arr`: `[1,2,3], [1,3,2], [2, 1, 3], [2, 3, 1], [3,1,2], [3,2,1]`.

The **next permutation** of an array of integers is the next lexicographically greater permutation of its integer. More formally, if all the permutations of the array are sorted in one container according to their lexicographical order, then the **next permutation** of that array is the permutation that follows it in the sorted container. If such arrangement is not possible, the array must be rearranged as the lowest possible order (i.e., sorted in ascending order).

- For example, the next permutation of `arr = [1,2,3]` is `[1,3,2]`.
- Similarly, the next permutation of `arr = [2,3,1]` is `[3,1,2]`.
- While the next permutation of `arr = [3,2,1]` is `[1,2,3]` because `[3,2,1]` does not have a lexicographical larger rearrangement.

Given an array of integers `nums`, *find the next permutation of* `nums`.
The replacement must be **in place** and use only constant extra memory.
Example 1:

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

Example 2:

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

Example 3:

```
Input: nums = [1,1,5]
Output: [1,5,1]
```

Constraints:

- `1 <= nums.length <= 100`
- `0 <= nums[i] <= 100`

Problem 5:

There is an integer array `nums` sorted in ascending order (with **distinct** values).
Prior to being passed to your function, `nums` is **possibly rotated** at an unknown pivot index `k` (`1 <= k < nums.length`) such that the resulting array is `[nums[k], nums[k+1], ..., nums[n-1], nums[0], nums[1], ..., nums[k-1]]` (**0-indexed**). For example, `[0,1,2,4,5,6,7]` might be rotated at pivot index `3` and become `[4,5,6,7,0,1,2]`.
Given the array `nums` **after** the possible rotation and an integer `target`, return *the index of* `target` *if it is in* `nums`*, or* `-1` *if it is not in* `nums`.
You must write an algorithm with `O(log n)` runtime complexity.
Example 1:

```
Input: nums = [4,5,6,7,0,1,2], target = 0
Output: 4
```

Example 2:

```
Input: nums = [4,5,6,7,0,1,2], target = 3
Output: -1
```

Example 3:

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

Problem 6:

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]]
```

Problem 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 <= 10^5`
- `-10^4 <= nums[i] <= 10^4`

Problem 8

Given an array of `intervals` where `intervals[i] = [start_i, end_i]`, merge all overlapping intervals, and return *an array of the non-overlapping intervals that cover all the intervals in the input*.

Example 1:

```
Input: intervals = [[1,3],[2,6],[8,10],[15,18]]
Output: [[1,6],[8,10],[15,18]]
Explanation: Since intervals [1,3] and [2,6] overlap, merge them into [1,6].
```

Example 2:

```
Input: intervals = [[1,4],[4,5]]
Output: [[1,5]]
Explanation: Intervals [1,4] and [4,5] are considered overlapping.
```

Problem 9:

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

```
Input: nums = [2,0,2,1,1,0]Output: [0,0,1,1,2,2]
```

Example 2:

```
Input: nums = [2,0,1]Output: [0,1,2]
```

Constraints:

- `n == nums.length`
- `1 <= n <= 300`
- `nums[i]` is either `0`, `1`, or `2`

Problem 10

You are given an integer array `prices` where `prices[i]` is the price of a given stock on the i^{th} day.
On each day, you may decide to buy and/or sell the stock. You can only hold **at most one** share of the stock at any time.
However, you can buy it then immediately sell it on the **same day**.
Find and return *the **maximum** profit you can achieve*.
Example 1:

```
Input: prices = [7,1,5,3,6,4]Output: 7Explanation: Buy on day 2 (price = 1) and sell
Then buy on day 4 (price = 3) and sell on day 5 (price = 6), profit = 6-3 = 3.
Total profit is 4 + 3 = 7.
```

Example 2:

```
Input: prices = [1,2,3,4,5]Output: 4Explanation: Buy on day 1 (price = 1) and sell o
Total profit is 4.
```

Example 3:

```
Input: prices = [7,6,4,3,1]Output: 0Explanation: There is no way to make a positive
```

Constraints:

- `1 <= prices.length <= 3 * 10^4`
- `0 <= prices[i] <= 10^4`

Problem 11

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**.

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
```

Constraints

- `1 <= nums.length <= 1000`
- `-2^31 <= nums[i] <= 2^31 - 1`
- `nums[i] != nums[i + 1]` for all valid `i`.

Problem 12

Given two sorted arrays nums1 and nums2 of size m and n respectively, return **the median** of the two sorted arrays.

Example 1:

```
Input: nums1 = [1,3], nums2 = [2]
Output: 2.00000
Explanation: merged array = [1,2,3] and median is 2.
```

Example 2:

```
Input: nums1 = [1,2], nums2 = [3,4]
Output: 2.50000
Explanation: merged array = [1,2,3,4] and median is (2 + 3) / 2 = 2.5.
```

Constraints

- `nums1.length == m`
- `nums2.length == n`
- `0 <= m <= 1000`
- `0 <= n <= 1000`
- `1 <= m + n <= 2000`
- `-10^6 <= nums1[i], nums2[i] <= 10^6`

Problem 13

You are given an integer array height of length n. There are n vertical lines drawn such that the two endpoints of the i^{th} line are (i, 0) and (i, height[i]).
Find two lines that together with the x-axis form a container, such that the container contains the most water.
Return *the maximum amount of water a container can store*.

Example 1:

```
Input: height = [1,8,6,2,5,4,8,3,7]
Output: 49
Explanation: The above vertical lines are represented by array [1,8,6,2,5,4,8,3,7]. I
```

Example 2:

```
Input: height = [1,1]
Output: 1
```

Constraints:

- `n == height.length`
- `2 <= n <= 10^5`
- `0 <= height[i] <= 10^4`

Problem 14

Given a binary array nums, return *the maximum number of consecutive* 1*'s in the array*.

Example 1:

```
Input: nums = [1,1,0,1,1,1]
Output: 3
Explanation: The first two digits or the last three digits are consecutive 1s. The ma
```

Example 2:

```
Input: nums = [1,0,1,1,0,1]
Output: 2
```

Constraints:

- `1 <= nums.length <= 10^5`
- `nums[i]` is either `0` or `1`.

Problem 15

Given an integer array nums and an integer k, return *the* kth *largest element in the array*.
Note that it is the kth largest element in the sorted order, not the kth distinct element.
You must solve it in O(n) time complexity.

Example 1:

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

Example 2:

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

Constraints:

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