Algorithms

Sorting Homework 2

Mostafa S. Ibrahim
Teaching, Training and Coaching for more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

Tips During the course

- For any homework in this course, compute **time and memory complexity** when applicable
  - Any time I mention complexity without specifying ⇒ it is **time complexity**
- See the **online judge section** to use the given online judge
- Some problems might not have an online judge. Just test locally
  - It is hard to provide test cases
- Some problems might be private (for premium subscription)
- The goal of homework is to think in an efficient way. So, don’t just stop with the first easy solution that comes to your mind

Terminologies

- **Brute Force**: It means just try everything to get the solution.
  - Find how many triplets each in range [1-100] and their sum is 125
  - Brute: let's try 3 nested loops
  - With some skills: we can do 2 nested loops only
- **Ad-hoc**: It means no specific well-known algorithm for a problem. Usually harder than other problems, as there are few patterns for such problems
- **Greedy**: A well-known algorithmic topic, with many such problems being solvable using basic logic and thinking due to clear choices. Many of these problems are actually greedy problems but you don't notice. When you see it, replace the greedily word with **'logically**'
  - Many of the greedy solutions start with a **sorting** step!

In this section

- Please use the **built-in sort** as long as it is convenient
- Assume the following
  - Its time complexity is O(nlogn)
  - Its space complexity is O(1), e.g. assuming *the iterative heapsort implementation*
- #include<algorithm>
  - For sort and other built in algorithms
- #include<climits>
  - For constants like INT_MAX

Problem #1: LeetCode 1200 - Minimum Absolute Difference

Given an array of **distinct** integers `arr`, find all pairs of elements with the minimum absolute difference of any two elements.

Return a list of pairs in ascending order(with respect to pairs), each pair `[a, b]` follows

- `a, b` are from `arr`
- `a < b`
- `b - a` equals to the minimum absolute difference of any two elements in `arr`

- vector<vector<int>> minimumAbsDifference(vector<int>& nums)
- Develop an O(nlogn) time solution

Examples

Example 1:

```
Input: arr = [4,2,1,3]
Output: [[1,2],[2,3],[3,4]]
Explanation: The minimum absolute difference is 1. List all pairs
with difference equal to 1 in ascending order.
```

Example 2:

```
Input: arr = [1,3,6,10,15]
Output: [[1,3]]
```

Example 3:

```
Input: arr = [3,8,-10,23,19,-4,-14,27]
Output: [[-14,-10],[19,23],[23,27]]
```

Problem #2: LeetCode 976 - Largest Perimeter Triangle

- Given an integer array nums, return the largest perimeter of a triangle with a non-zero area, formed from three of these lengths.
  - If it is impossible to form any triangle of a non-zero area, return 0.
- Review what makes a triangle a valid one
- int largestPerimeter(vector<int> &nums)
  - Array size >= 3
- Develop O(nlogn) time solution

Example 1:

```
Input: nums = [2,1,2]
Output: 5
```

Example 2:

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

Example 3:

```
Input: nums = [3,2,3,4]
Output: 10
```

Example 4:

```
Input: nums = [3,6,2,3]
Output: 8
```

Problem #3: LeetCode 561 - Array Partition I

- Given an integer array nums of **2n** integers:
  - group these integers into n pairs (a1, b1), (a2, b2), ..., (an, bn)
  - such that the sum of min(ai, bi) for all i is maximized.
  - return the **maximized sum**.
- int arrayPairSum(vector<int> &nums)
- Develop O(nlogn) time solution
- **Terminology**: An **optimization problem** is the problem of finding the **best** solution from all feasible solutions. It usually asks for the best **minimum or maximum**
- Tip: Given an array of 2n numbers ⇒ implies the array has an **even** size
  - [1, 2, 3, 4, 5, 6, 7, 8] ⇒ n = 4

Examples

Example 1:

```
Input: nums = [1,4,3,2]
Output: 4
Explanation: All possible pairings (ignoring the ordering of elements) are:
1. (1, 4), (2, 3) -> min(1, 4) + min(2, 3) = 1 + 2 = 3
2. (1, 3), (2, 4) -> min(1, 3) + min(2, 4) = 1 + 2 = 3
3. (1, 2), (3, 4) -> min(1, 2) + min(3, 4) = 1 + 3 = 4
So the maximum possible sum is 4.
```

Example 2:

```
Input: nums = [6,2,6,5,1,2]
Output: 9
Explanation: The optimal pairing is (2, 1), (2, 5), (6, 6). min(2, 1) + min(2, 5) + min(6, 6)
= 1 + 2 + 6 = 9.
```

Problem #4: LeetCode 280 - Wiggle Sort

- Given an integer array nums, reorder it such that
- nums[0] <= nums[1] >= nums[2] <= nums[3]
- void wiggleSort(vector<int> &nums)
  - You may assume the input array always has a valid answer.
  - Develop O(nlogn) time solution.
    - Optionally: Develop O(n) solution
- Input ⇒ Output
  - [3,5,2,1,6,4] ⇒ [3,5,1,6,2,4] or [1,6,2,5,3,4]
  - [6,6,5,6,3,8] ⇒ [6,6,5,6,3,8]
- It is guaranteed that there will be an answer for the given input nums.
  - So don't build an invalid test case for yourself

Problem #5: LeetCode 1921 - Eliminate Maximum Number of Monsters

You are playing a video game where you are defending your city from a group of `n` monsters. You are given a **0-indexed** integer array `dist` of size `n`, where `dist[i]` is the **initial distance** in kilometers of the i^{th} monster from the city.

The monsters walk toward the city at a **constant** speed. The speed of each monster is given to you in an integer array `speed` of size `n`, where `speed[i]` is the speed of the i^{th} monster in kilometers per minute.

You have a weapon that, once fully charged, can eliminate a **single** monster. However, the weapon takes **one minute** to charge.The weapon is fully charged at the very start.

You lose when any monster reaches your city. If a monster reaches the city at the exact moment the weapon is fully charged, it counts as a **loss**, and the game ends before you can use your weapon.

Return *the **maximum** number of monsters that you can eliminate before you lose, or* `n` *if you can eliminate all the monsters before they reach the city.*

- int eliminateMaximum(vector<int>& dist, vector<int>& speed)

Example 1:

```
Input: dist = [1,3,4], speed = [1,1,1]
Output: 3
Explanation:
In the beginning, the distances of the monsters are [1,3,4]. You eliminate the first monster.
After a minute, the distances of the monsters are [X,2,3]. You eliminate the second monster.
After a minute, the distances of the monsters are [X,X,2]. You eliminate the thrid monster.
All 3 monsters can be eliminated.
```

Example 2:

```
Input: dist = [1,1,2,3], speed = [1,1,1,1]
Output: 1
Explanation:
In the beginning, the distances of the monsters are [1,1,2,3]. You eliminate the first monster.
After a minute, the distances of the monsters are [X,0,1,2], so you lose.
You can only eliminate 1 monster.
```

Example 3:

```
Input: dist = [3,2,4], speed = [5,3,2]
Output: 1
Explanation:
In the beginning, the distances of the monsters are [3,2,4]. You eliminate the first monster.
After a minute, the distances of the monsters are [X,0,2], so you lose.
You can only eliminate 1 monster.
```

Can you code such that there are no double conversions (e.g. division) or modulus operator?

Problem #6: LeetCode 1005 - Maximize Sum Of Array After K Negations

- Given an integer array nums and an integer k, modify the array in the following way:
  - choose an index i and replace nums[i] with -nums[i].
  - You should apply this process exactly k times.
- You may choose the same index i multiple times.
- Return the **largest possible sum** of the array after modifying it in this way.
- int largestSumAfterKNegations(vector<int>& nums, int k)

Example 1:

```
Input: nums = [4,2,3], k = 1
Output: 5
Explanation: Choose index 1 and nums becomes [4,-2,3].
```

Example 2:

```
Input: nums = [3,-1,0,2], k = 3
Output: 6
Explanation: Choose indices (1, 2, 2) and nums becomes [3,1,0,2].
```

Example 3:

```
Input: nums = [2,-3,-1,5,-4], k = 2
Output: 13
Explanation: Choose indices (1, 4) and nums becomes [2,3,-1,5,4].
```

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."