This assignment is **due on September 22** and should be submitted on Gradescope. All submitted work must be *done individually* without consulting someone else's solutions in accordance with the University's "Academic Dishonesty and Plagiarism" policies.

Before you read any further, go to the last page of this document and read the Written Assignment Guidelines section.

Problem 1. (10 points) In this question we're going to analyse a response generated by an AI, in this case ChatGPT. It was asked to design an algorithm for the following problem that runs in $O(n^2 \log n)$ (expected) time, argue its correctness, and analyse its running time.

Problem Description: You and your best friend just got out of the movies and are very hungry; unfortunately, it is now very late at night and all restaurants are closed. You manage to find, by chance, some vending machine still full of very... nutritious foods (bags of chips, and the occasional cereal bar). Looking into your pockets, you look what change you have, in order to try and buy something (anything!) to eat.

You have n coins of various values. So does your friend! As for the vending machine... it contains n different "food items" each with its own price. Looks like you may eat tonight... except for two things:

- the vending machine is old and somewhat broken: it only accepts at most two coins, and does not return change: you must put **exactly** the right price to get the item you ask for.
- out of fairness, you and your friend refuse to pay for the whole thing alone. So each of you has to contribute (no matter how little): to buy the food, each of you has to contribute at least one coin.

Which means that, if you want to eat tonight, you must figure out if there is an item in the vending machine whose price is exactly equal to the sum of two coins, one from you and one from your friend. And you are very hungry, so you want to figure that out **fast**.

Your task: given three arrays Y, F, and V (You, Friend, Vending machine) each containing n positive integers, decide if there exist $0 \leq i, j, k < n$ such that $Y[i] + F[j] = V[k]$. You can assume that all integers in all arrays are distinct (also between different arrays). Since you want to eat soon, you want an algorithm for this task which solves your problem fast: running in (expected) time $O(n^2 \log n)$.

Example:
$Y = [3, 2, 1]$, $F = [4, 5, 6]$, $V = [50, 8, 13]$.
We need to return true, since $Y[1] + F[2] = V[1]$ (i.e., $2 + 6 + 8$).
For the same Y and F, but with $V = [50, 2123, 9123]$, we'd return false, as there are no i, j, and k such that $Y[i] + F[j] = V[k]$.

AI Output:
Algorithm Description:

1. Hash Map Initialization: Start by creating a hash map (‘map_v‘) that maps each price in array ‘V‘ to its index. This allows us to quickly lookup whether a specific price exists in ‘V‘.
2. Iterate Over Pairs:
   - Use two nested loops to iterate over all pairs ‘(i, j)‘ where ‘i‘ is an index in array ‘Y‘ and ‘j‘ is an index in array ‘F‘.
   - For each pair ‘(i, j)‘, calculate the sum ‘sum = Y[i] + F[j]‘.
3. Binary Search for Matching Price:
   - For each calculated sum ‘sum‘, check if ‘sum‘ exists in ‘map_v‘.
   - If ‘sum‘ exists, retrieve its index from ‘map_v‘.
4. Check Validity: Ensure that the retrieved index from ‘map_v‘ is different from ‘i‘ and ‘j‘ to satisfy the condition that you and your friend are both contributing at least one coin.
5. Return Result:
   - If a valid index ‘k‘ is found that satisfies the conditions, return ‘true‘.
   - If no such triplet ‘(i, j, k)‘ is found after checking all pairs, return ‘false‘.

Correctness Proof:

- Hash Map Usage: By using a hash map for ‘V‘, we achieve $O(1)$ average-time complexity for lookups, ensuring efficiency in checking whether a sum exists.
- Binary Search: The binary search operation (using ‘lower_bound‘ or similar) for finding the index in ‘V‘ ensures that each lookup operation is efficient within the nested loop structure.

Time Complexity Analysis:

- Hash Map Construction: Building ‘map_v‘ takes $O(n)$ time.
- Nested Loops: There are $O(n^2)$ pairs ‘(i, j)‘ to consider.
- Binary Search: Each binary search operation within the nested loop is $O(\log n)$.

Therefore, the overall time complexity is $O(n^2 \log n)$, dominated by the nested loops and the binary search operations.

Your task: To analyse the above response consider each of the following questions and briefly explain your answer to each.

a) Does the AI response answer the question, i.e., does the algorithm solve the stated problem and does it run in the required time?

b) Is the algorithm described in sufficient detail, i.e., is all information present or are any parts unclear or missing?

c) Is the correctness argument complete or are there leaps or errors in the logic?

d) Is the running time analysis correct for the provided algorithm?

Problem 2. (25 points)
The company Versatile Routing Over Outstanding Metal or VROOM produces routers and prides itself in the fact that at no point during the lifetime of any of its routers the average delay is larger than a certain value X. As one of their Quality Control Engineers, you're tasked with testing a new range of their routers to ensure these too satisfy this requirement. We're testing $k > 0$ routers, each identified using a distinct integer. Because we're using the ultimate black box testing techniques, we don't know beforehand what these integers are and k isn't known either (and k is *not* a constant). Whenever a router exceeds the maximum allowed average delay X, it's removed from the testing suite and sent to R&D for analysis, never to be seen again. You're tasked with designing a data structure that supports the testing process, including the following operations:

- INITIALIZE(): creates the (empty) data structure in $O(1)$ time.
- ADD-DATA($ID, delay$): updates the average delay of router ID using a new data point $delay$ in $O(\log k)$ time.
- CURRENT-DELAY(ID): returns the current average delay of router ID in $O(\log k)$ time, if it's part of the test suite, and returns null otherwise.
- MAX-DELAY(): returns the ID of the router with the current highest average delay in $O(1)$ time, or null if no routers remain in the test suite.

You can assume that the delay is always a non-negative integer. You should ensure that after every operation only routers with an average delay of at most X are stored in the data structure (routers with average delay more than X are sent to *R&D*). Your data structure should take $O(k)$ space, regardless of how much data is added for any specific router. Remember to

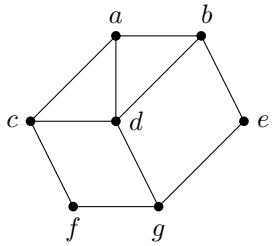
a) Design a data structure that supports the required operations in the required time and space.

b) Briefly argue the correctness of your data structure and operations.

c) Analyse the running time of your operations and space of your data structure.

Problem 3. (25 points)
We're volunteering in the Bushfire Prevention Club and we're tasked with designing an algorithm to predict the spread of bushfires. Our input is given as a simple undirected graph $G = (V, E)$ ($|V| = n$ vertices and $|E| = m$ edges) in adjacency list representation and an array of vertices $F \subseteq V$ that are on fire. We're told that a vertex $v \notin F$ catches fire if at least 3 of its neighbours are already on fire. Your task is to compute all vertices of V that are on fire if fires are started at all vertices in F (and we make no attempt to prevent the fire from spreading).

Example:



If $F = \{b, c, g\}$ then we need to report $\{a, b, c, d, g\}$: $\{b, c, g\}$ are initially on fire and together set d on fire, and in turn $\{b, c, d\}$ set a on fire as well. Note that e and f don't catch fire, since fewer than 3 of their neighbours are on fire. If $F = \{d, e, f\}$, we report $\{d, e, f, g\}$ since $\{d, e, f\}$ make g catch fire and no other vertex has 3 burning neighbours after that.

Design an algorithm that given an initial set of burning vertices F, computes all burning vertices of V. For full marks, your algorithm should run in $O(n + m)$ time. Remember to:

a) describe your algorithm in plain English,

b) argue its correctness, and

c) analyze its time complexity.

Written Assignment Guidelines

- Assignments should be typed and submitted as pdf (no pdf containing text as images, no handwriting).
- Start by typing your student ID at the top of the first page of your submission. Do **not** type your name.
- Submit only your answers to the questions. Do **not** copy the questions.
- When asked to give a plain English description, describe your algorithm as you would to a friend over the phone, such that you completely and unambiguously describe your algorithm, including all the important (i.e., non-trivial) details. It often helps to give a very short (1-2 sentence) description of the overall idea, then to describe each step in detail. At the end you can also include pseudocode, but this is optional.
- In particular, when designing an algorithm or data structure, it might help you (and us) if you briefly describe your general idea, and after that you might want to develop and elaborate on details. If we don't see/understand your general idea, we cannot give you marks for it.
- Be careful with giving multiple or alternative answers. If you give multiple answers, then we will give you marks only for "your worst answer", as this indicates how well you understood the question.
- Some of the questions are very easy (with the help of the slides or book). You can use the material presented in the lecture or book without proving it. You do not need to write more than necessary (see comment above).
- When giving answers to questions, always prove/explain/motivate your answers.
- When giving an algorithm as an answer, the algorithm does not have to be given as (pseudo-)code.
- If you do give (pseudo-)code, then you still have to explain your code and your ideas in plain English.
- Unless otherwise stated, we always ask about worst-case analysis, worst-case running times, etc.
- As done in the lecture, and as it is typical for an algorithms course, we are interested in the most efficient algorithms and data structures, though slower solutions may receive partial marks.
- If you use further resources (books, scientific papers, the internet,...) to formulate your answers, then add references to your sources and explain it in your own words. Only citing a source doesn't show your understanding and will thus get you very few (if any) marks. Copying from any source without reference is considered plagiarism.