#Q1:

A screenshot of a computer

AI-generated content may be incorrect.

#Q2:

a)

A screenshot of a computer program

AI-generated content may be incorrect.

First, we set up a union-find structure to merge variables that have to be equal. Then we just check every inequality to see if the two variables end up in the same group. If they do, it's not possible to satisfy everything. Otherwise, it works.

b)

The initialization takes O(n) time. Processing m₁ equalities and m₂ inequalities takes O(m₁ + m₂) time (amortized nearly constant per operation), so the overall running time is O(n + m₁ + m₂).

Q3

a)

Beginning with the initial item, the greedy algorithm seeks to fit as many things as it can into a value-restricted box while staying within the upper limit of the permitted value. It then determines how many objects can be placed inside a weight-restricted box without going over the weight limit. The algorithm chooses the box type that can accommodate more items by comparing the amount of items that can be packed in each type of box. After then, it advances to the following set of items and raises the shipping cost. In order to minimize the amount of boxes utilized, this process is repeated until all things are packed into boxes.

b)

A screenshot of a computer program

AI-generated content may be incorrect.

Each item is considered only once because the inner loops move the index forward without rechecking items. This means the overall running time grows linearly with the number of items, i.e., O(n).

Q4:

a)

A screenshot of a computer

AI-generated content may be incorrect.

b)

When I originally started working on the "Same Tree" problem, I thought about utilizing a queue or stack to solve it iteratively. Comparing similar nodes while performing a level-order or depth-first traversal was the concept. But it became difficult to handle the situations where one node was null while the other wasn't. The approach was overly complicated due to the additional bookkeeping required for pushing and popping nodes. Although the iterative method was effective, it was more difficult to troubleshoot and felt counterintuitive.

After struggling with the iterative approach, I realized that recursion might be a better fit. Trees have a naturally recursive structure, and the problem itself is inherently self-similar—each subtree should also be identical. I rewrote the solution using recursion, checking the base cases first: if both nodes were null, they were equal; if one was null while the other wasn't, or if their values differed, they were not. Then, I recursively compared their left and right subtrees.

Compared to my iterative attempt, recursive method was more cleaner and functioned effectively. Refining the base cases helped me better comprehend the edge cases, which I had first missed, such as when both subtrees were empty.

After giving this procedure some thought, I saw that I was struggling because I was attempting to apply an iterative technique to an issue where recursion was a better fit. This experience improved my ability to choose the best strategy for various data structures and reaffirmed the significance of identifying when recursion simplifies an issue.