Drill questions — an error in the solution for one of the topological sort questions has been fixed. If it was previously marked correct, it no longer is and you should correct it.

Sometimes graphs may not be inherently simple vs not simple, sparse vs dense, cyclic vs acyclic — the nature may depend on the specific instance of the problem being modeled. If one alternative is the typical or most likely case, you can give that as the main answer but still acknowledge that both alternatives are possible and that it depends on the specific problem instance.

Labeled vs unlabeled graph — "labeled" refers to vertex identifiers, not other sorts of information that may be associated with vertices or edges. In addition, "labeled" typically means that the vertex identifiers are meaningful beyond distinguishing one vertex from another. A way to test whether the identifiers are meaningful is to consider whether changing the identifiers without changing anything about the graph structure itself results in a different graph. Any application where you need to refer to specific vertices (e.g. to find the shortest path from A to B) requires a labeled graph — changing the identifiers around means that "A" and "B" no longer refer to the same vertices, and the result may be a completely different path. On the other hand, MST doesn't rely on the vertex labels — the MST will connect the same vertices regardless of how they are identified.

Implicit vs explicit graph — can you determine adjacent vertices without storing edges? "Implicit" means you have a way of computing adjacent vertices e.g. for the four-digit configuration graph, you can compute the configurations reachable by a button press by adding and subtracting one from each digit of the current configuration. "Explicit" means that you need to explicitly store information (edges) in order to determine which vertices are adjacent — you have to look up what vertices are connected rather than being able to compute it.

Level of detail — you can (and should) reference known algorithms without further description of the algorithm, but don't only half-describe the details and expect the reader to recognize it. For example, "2-coloring" is an known algorithm that was discussed in class. If 2-coloring solves your problem, explain how the graph is built (what the vertices and edges represent), say you 2-color it, and then explain how the colors assigned translate into the solution for that problem. That's enough information for someone who is familiar with 2-coloring to solve the problem. By contrast, explaining how the graph is built and then saying "use BFS to assign colors" or "use BFS to assign colors so edges connect vertices of different colors" is *not* enough information unless the reader is familiar enough with 2-coloring to recognize that what is being described and then fill in the gaps with their knowledge.