These are design problems, not research problems — use what has been discussed in class and/or in the readings to design the data structures or algorithms called for rather than looking up a specialized data structure or algorithm.

Level of detail —

- Overview before detail — identify the key idea behind the data structure or algorithm to give context before getting into the more detailed description. You don't need full details in the overview, just the main idea.
- Don't describe details of known algorithms. #2 says to use BFS — just explain how you use and/or modify BFS rather than giving the whole algorithm from scratch. This saves time and effort in the writing, but also provides greater clarity in expressing the solution — imagine someone is telling you how to do a somewhat complicated thing. What is going to be easier for you to understand and do, step-by-step instructions for the entire process or "it's just like ... except you do X instead of Y"?

If you add information to a standard data structure, make sure you fully address how various operations use or impact that information. For example, remember that insert and remove in a balanced search tree (such as an AVL tree) may involve restructuring after the insert/remove operation in order to restore/maintain the balance — if you add subtree sizes (for example) to nodes, how does restructuring affect those stored values? How can you recompute the new value, and how many values do you have to recompute? This can affect the running time if too much needs updating (or if it is too expensive to update) so it is necessary to address why it doesn't (if that is in fact the case).

BFS is $O(n+m)$ in general, but #2 asked you to explain why the variation of BFS to find a removable edge is $O(n)$. Think about how many edges you have to look at before finding a removable edge...