Instructions

Instructions

1. The question is an algorithms problem on graphs, though it might not explicitly talk about graphs.

2. **For each answer, your solution must have three clearly demarcated parts:**
    1. **The main idea:** give a few sentences explaining the main idea of your approach and give a short summary of your algorithm. If the problem does not explicitly mention a graph, then you will likely have to construct in (in your algorithm). Clearly state what the vertices and edges of the graph are.
    2. **The pseudocode***
    3. **The running time analysis**

    ***Some guidelines for pseudocode:**

        1. Number each line of pseudocode. Clearly demarcate loops with indentation. For clarity, also demarcate loops either using parentheses (like C/C++) or numbering with Roman numerals, letters, etc., to differentiate from the main pseudocode.
        2. You do not need to write pseudocode for any algorithm/data structure taught in class. For example, you can write: "Run BFS on graph G to get the pred and visited arrays", and use the latter arrays. Or, you can "Run Dijkstra's shortest path algorithm on G with sources vertex s and weights (whatever)".
        3. It's ok to get indexing wrong and have off-by-one errors.
2. For maximum clarity, it is best to write out the big-Oh running time of each Step of your algorithm (which is conveniently numbered!). If you have used some algorithm taught in class, you can simply state the running time. For example: "Mergesort runs in O(n log n), as explained in class."
3. Your running time will typically depend on n (the number of vertices) and m (the number of edges). If these quantities are not explicitly defined in your question, you will need to explain how they relate to the size of the input.
4. Try to use subscript-superscripts for math notation, so it is easier for us to read.
5. As long as your algorithm is correct and the running time analysis is correct, you will get full credit.
6. We heavily consider precision of your writing and pseudocode in our grading. If we cannot understand your algorithm (or there is too much ambiguity in how the steps work), we cannot give you full credit.
7. The question might not specify how to output (either print, or output as an object). Make reasonable choices, and as long as you get the algorithm correct, you will get credit. You are free to define objects that might be convenient, such as pairs or triples of integers. (You don't need to write out code for them.) Again, we value precision of language.