COP 3530: Fall 2016: Final Exam Review

For each algorithmic question, think about why it is correct and analyze its worst-case time complexity. For graphs assume that it is represented as an adjacency list (not an adjacency matrix). When not specified, asume the graph is simple, unweighted and undirected.

1. Make a big table whose rows correspond to the major data structures you have learnt in class (`ArrayList, LinkedList, SortedArrayList, Stack, Queue, PriorityQueue, Tree, BST, AVLTree, HashTable, Suffix Tree, Suffix Array, KDTree`) and the columns correspond to the main methods (insert, delete, and search – or their equivalents), and the entries correspond to the worst-case time complexities of the methods for that data structure.
2. Given an `AVLTree` data structure, what are the time complexities of `Rank` and `Select` operations?
3. Make a table with the time complexities of each of the major graph algorithms (`DFS, BFS, MST, Dijkstra, TopologicalSort`).
4. Make a table where the rows correspond to every sorting algorithm we have discussed or mentioned in class (`selectionSort, insertionSort, bubbleSort, shakerSort, shellSort, quickSort, mergeSort, heapSort, treeSort, bucketSort, radixSort, countingSort`), and the columns correspond to worst-case time complexity, space complexity (extra space), average-case time complexity, best-case time complexity, and worst-case input description.
5. Make sure you go over the definitions of $f(n) = O(g(n); f(n) = \Omega(g(n)); f(n) = o(g(n)); f(n) = \omega(g(n)); f(n) = \Theta(g(n))$.
6. Explain the change in output if the MST algorithm (either Prim's or Kruskal's) is run on the same graph but with the weight of each edge increased by a positive number D? Negative number?
7. Answer the previous questions for Dijkstra's single-source shortest path algorithm?
8. Design an algorithm to compute the degree of each vertex
9. Design an algorithm to check if the graph has a cycle.
10. Design an algorithm to check if the graph is directed.
11. Given an unweighted directed graph, and a start vertex s, design an algorithm to compute all vertices reachable from s.
12. Design an algorithm to sort each adjacency list.
13. An *Euler circuit* for an undirected graph is a path which starts and ends at the same vertex and uses every edge exactly once. It is known that a graph G has an Euler circuit if and only if every vertex has even degree. Design an algorithm to compute an Euler circuit of a graph, if one exists.

14. Modify Dijkstra's algorithm to output the shortest path (not just the length of the path) from the start vertex s to a specified vertex v.
15. Design an algorithm to run Dijkstra's algorithm simultaneously from two vertices, s and t, using one priority queue. How can you use this to find the shortest path from s to t.
16. Design an algorithm to compute the shortest cycle in an undirected graph.