Due date: November 16 at 7:30 AM. You will need to submit this via GradeScope.

For this assignment, each answer must be contained within a single piece of paper. When you submit to GradeScope, you will need to inform the system which page of your scanned PDF contains the answer. *Do this even if your submission is a single page.*

Please review the syllabus and course reference for the expectations of assignments in this class. Remember that problem sets are not online treasure hunts. You are welcome to discuss matters with classmates, but remember the Kenny Loggins rule. Remember that you may not seek help from any source where not all respondents are subject to UC Irvine's academic honesty policy.

1. Suppose we want to implement a PriorityQueue and I am considering storing the underlying data as an AVL Tree.

   (a) For each of the fundamental operations of a PriorityQueue, explain briefly how you would implement it with an AVL Tree. Each should take you 1-2 sentences at most; you do not need to write full code. If something is a standard AVL tree operation, such as a re-balance, you need only refer to it. You do not need to re-write the description.

   Then, state how long these will take to run.

   The last one is done for you as an example (only because last instead of first will fit better into a Gradescope outline).

      i. `extractMin()`:

      ii. `insert`

      iii. `min()` :
      We could follow left pointers from the root until a node does not have a left pointer. That is the min. This takes $\mathcal{O}(\log n)$ time because the AVL Tree has $\mathcal{O}(\log n)$ depth.

   (b) Is this better or worse than a heap, as shown in lecture? Why or why not? You need only a brief explanation for why you said yes or no.

2. There exists a haunted maze which contains n scare stations, with a designated starting station s and a final station t. To model the haunted maze as a graph, there is a vertex for each scare station and a directed edge from one station to another if it is easy to walk between the two directly (note: because the owners of the haunted house place a restriction on which houses you can visit, the edge between the two scare stations is not bidirectional). Each scare station v has a scare factor of $c(v) > 0$, where the higher $c(v)$ is, the scarier the scare stations are. Thus the graph has costs on the vertices rather than the edges. Shindler is a scaredy-cat, and wants to complete the maze by minimizing the total scare factor of the houses; in other words, he wants to find a path P from s to t such that $\sum_{v \in P} c(v)$ is minimum.

   Suppose you already have Dijkstra's Algorithm implemented for directed graphs. Describe how you can use that implementation to solve this problem. For credit, your answer must involve creating a set of edge weights for the same set of vertices and edges such that a call to Dijkstra's algorithm will produce the desired tree. Do not re-create a "Dijkstra-like" algorithm and do not modify the code for Dijkstra's algorithm. Your answer must make a function call to Dijkstra's as a meaningful part of your solution.

3. We saw in lecture that Dijkstra's Algorithm can find a single-source **shortest** path tree for a graph with positive edge weights. Suppose I have a graph with positive edge weights and I want a single-source **longest** path tree: that is, for each vertex, I want the longest (simple) path[1] to the other vertices.

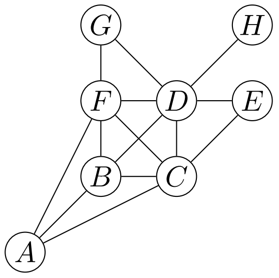
   I propose the following algorithm: for each edge e, I change the weight w'_e to $1/w_e$. That is, I take the reciprocal of each edge weight. Then I run Dijkstra's Algorithm on this modified graph.

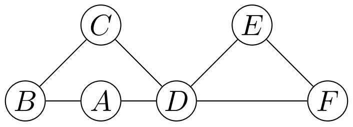
   Does this give me the single-source longest path tree? If it does, explain briefly (2-3 sentences) why it does. If it does not, give a counter-example for why it does not. If you are giving a counter-example, you may either draw a graph or describe it. Be sure to demonstrate why this is a counter-example if you do so by showing what the algorithm would find and what a better solution on the same input is.

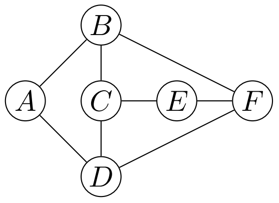
4. We say a graph is *amazing* if its chromatic number is equal to the size of the largest clique found in any subgraph of it (i.e., you can find the complete graph $K_{\chi(G)}$ as a subgraph, but you cannot find $K_{\chi(G)+1}$).

   A quick note on terminology: $\chi(G)$ is the abbreviation for the chromatic number of a graph. K_n is a complete [simple] graph with n vertices: that is, it has n vertices and all pairs of distinct vertices have an undirected edge between them.

   (a) For each of the following graphs, is it amazing? For each one, either show that it is or show why it is not.

   i.


   ii.


   iii.


   (b) The graph $K_{m,n}$ is the "complete bipartite graph." It has one group of m vertices and one group of n vertices. All pairs of vertices such that exactly one is in each group has an undirected edge between them.

   For what value(s) of m, n is $K_{m,n}$ an amazing graph? Explain your answer.

   (c) Draw a graph that (i) is **not** amazing, (ii) has five or more vertices, (iii) has a chromatic number of at least three, and (iv) is not isomorphic to any from part (a). Explain why your graph fits the first three criteria.

[1]That is, a path with no repeated vertices within it.