Exam 3 Practice Problems

HONOR CODE: You are allowed to work in groups on these problems, and also to talk to the TAs (the TAs have not seen these problems before and they do not know the solutions but they will cover any background information that might come up as you try to solve the problems). If you work with a group, all communication has to be oral or on the board — you cannot pass solutions in written form from one to another. Naturally, it is not allowed to search the internet trying to find solutions to these problems.

1. All problems in labs 7, 8, 9 and 10.

2. (CLRS 24.2-4) Suppose that all edge weights in a graph are integers in the range from 1 to $|V|$. How can you take advantage of this in Kruskal's algorithm, and how fast can you make it run? What if the edge weights are integers from 1 to W for some constant W?

3. Consider a weighted DAG and let u, v be two vertices in G. Some of the edges may be negative. Describe an algorithm to compute the shortest path from u to v, and analyze it. For full points you should aim for the fastest possible algorithm.

4. Suppose you had a bunch of edges (given as adjacency lists) and you want to figure out if they form a tree. How would you do it, and how fast?

5. Prove that the following claim is false by showing a counterexample:

   Claim: Let $G = (V, E)$ be a directed graph with negative-weight edges, but no negative-weight cycles. Let $w, w < 0$, be the smallest weight in G. Then one can compute SSSP in the following way: transform G into a graph with all positive weights by adding $-w$ to all edges, run Dijkstra, and subtract from each shortest path the corresponding number of edges times w. Thus, SSSP can be solved by Dijkstra's algorithm even on graph with negative weights.

6. Suppose the degree requirements for a computer science major are organized as a *DAG* (directed acyclic graph), where vertices are required courses and an edge (x, y) means course x must be completed prior to beginning course y. Make the following assumptions:

   - All prerequisites must be obeyed.

- There is a course, CPS1, that must be taken before any other course.
- Every course is offered every semester.
- There is no limit to the number of courses you can take in one semester.

Describe an efficient algorithm to compute the minimum number of semesters required to complete the degree and analyze its running time.

7. An independent set of an undirected graph $G = (V, E)$ is a subset I of V, such that no two vertices in I are adjacent (ie connected by an edge). A **maximal independent set** M is an independent set such that , if we were to add any additional vertex to M, it would not be independent any longer.

   (a) Draw a graph of 5 vertices and show a maximal IS.

   (b) Draw a graph of n vertices that has a maximal IS of 1 vertex.

   (c) Draw a graph of n vertices that has a maximal IS of $n - 1$ vertices.

   (d) Draw a graph of n vertices that has a maximal IS of 1 vertex, and another maximal IS of $n - 1$ vertices.

   (e) Give an efficient algorithm that computes a maximal independent set for a graph G. What is it's running time?

   Comment: a graph may have many maximal independent sets. The largest possible IS of a graph is called the *maximum* IS. Note the difference between *maximum* and *maximal* (not any maximal set is maximum). Finding a maximal IS can be done in polynomial time (part e), but finding the maximum IS (the largest possible maximal IS) is known to be hard (NPC). If your algorithm in (e) finds a *maximum* IS in polynomial time..... then it's either wrong, or you'll win the Millenium Prize!!! Hint: Think greedily.

8. Suppose $G(V, E)$ is a directed, weighted graph whose weights are known to be integers in the set $\{1, 2, 3, ..., k\}$, where k is some non-negative constant.

   (a) What is the size of the longest possible shortest path in this graph?

   (b) Assume we are computing SSSP from a vertex s using Dijkstra's algorithm. Consider the vertices in the graph, in the order in which they come out of the priority queue, and their distances: $s, v1, v2, v3, ...$ The first vertex is the source vertex s, and its distance is $d[s] = 0$.

   State a relation between $d[s], d[v1], d[v2], ...$ No need to justify.

   (c) Using this knowledge (about the integer weights) and the relation between the distances of the vertices that come out of the priority queue, describe a modification to Dijkstra's algorithm. Hint: Change the priority queue in a way similar to the linear time sorting algorithms. For full credit, keep your algorithm as simple and elegant as possible.

(d) What is the running time of your algorithm?

9. Would you use the adjacency list or the adjacency matrix in each of the following cases? Briefly justify.

   (a) The graph has 10,000 vertices and 20,000 edges and it is important to use as little space as possible.

   (b) The graph has 10,000 vertices and 20,000,000 edges and it is important to use as little space as possible

   (c) You need to answer adjacency queries as fast as possible, no matter how much space you use.

10. Let $G = (V, E)$ be a weighted graph and assume that some weights are negative. What happens if you run Dijkstra's algorithm on G? (brief answer)

11. Consider a directed graph with all edge weights equal to 1, and let u, v be two vertices in G. How can you compute the shortest path from u to v, and how fast? (brief answer)

12. Consider a weighted DAG and let u, v be two vertices in G. Some of the edges can be negative. How can you compute the shortest path from u to v, and how fast? (brief answer)

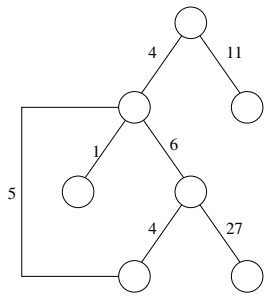
13. Draw a graph with 5 vertices that is connected and has the smallest possible number of edges. What is the smallest possible number of edges in a connected graph? (express it as function of $|V|$).

14. How fast can you figure out if a graph is connected and how? (brief answer)

15. Suppose you had a bunch of edges () given as adjacency lists) and you want to figure out if they form a tree. How would you do it, and how fast? (brief answer)

16. Draw a graph with at least one negative-weight edge for which Dijkstra's algorithm calculates incorrectly the shortest path from s for at least one vertex. Your example should be fairly simple, but not less than 4 vertices. Clearly indicate which vertex is s. Also indicate what distance labels Dijkstra would assign, and indicate the graph's shortest path value.

17. Consider an undirected weighted graph which is formed by taking a binary tree and adding an edge from *exactly one* of the leaves to another node in the tree. We call such a graph a *loop-tree*. An example of a loop-tree could be the following:



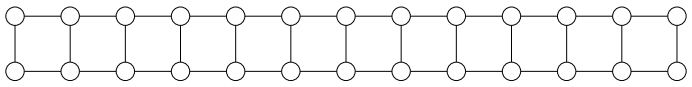
Let n be the number of vertices in a loop-tree and assume that the graph is given in the normal edge-list representation without any extra information. In particular, the representation does not contain information about which vertex is the root.

(a) How many edges are in a graph of n vertices?

(b) How long time would it take Prim's or Kruskal's algorithms to find the minimal spanning tree of a loop-tree?

(c) Describe and analyze a more efficient algorithm for finding the minimal spanning tree of a loop-tree. Remember to argue that the algorithm is correct.

18. A *rail-track graph* $G = (V, E)$ of length n is an undirected graph consisting of n connected "squares". For example, the following is the rail-track graph of length $n = 12$:



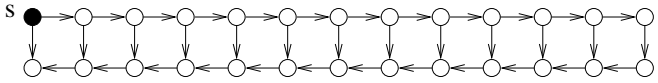
a) What is the number of edges $|E|$ and vertices $|V|$ in a rail-track graph as a function of n?

(b) We want to compute the minimum spanning tree of a weighted rail-track graph given in adjacency list representation. The following can be shown about minimum spanning trees:

> Let $G = (V, E)$ be a weighted undirected graph and let $K \subseteq E$ be a set of edges containing a minimum spanning tree for G. If K contains a cycle C and e is the maximal weight edge in C, then $K - \{e\}$ also contains a minimum spanning tree for G.

Describe an $O(n)$ algorithm for finding the minimum spanning tree of a rail-track graph with positive integer edge-weights. Remember to argue for both running time and correctness.

(c) Assume now that all top horizontal edges of a rail-track graph are oriented right, all bottom horizontal edges left, and all vertical edges are directed down. Call such a graph a *directed rail-track graph.*



We define the *directed spanning tree* of a directed rail-track graph $G_d = (V, E)$ to be a set $T \subseteq E$ of $|V| - 1$ edges, such there is a directed path from the topmost left vertex s (marked in the above example) to all other vertices in G_d. When G_d is weighted, the *minimum directed spanning tree* is defined as the spanning tree with minimal total weight.

c) Describe an $O(n)$ algorithm for finding the minimum directed spanning tree of a directed rail-track graph with positive integer edge-weights. Remember to argue for both running time and correctness.

19. Is the path between a pair of nodes u and v in a minimum spanning tree (i.e. the path composed of edges in the MST) necessarily the minimum weight path between u and v if we consider all the edges in the graph G? If yes, prove it. If no, provide a counterexample.

20. Argue that if all the edge weights of an undirected graph are positive, then any subset of edges that connects all vertices and has minimum total weight must be a tree. (A tree is an undirected, connected, acyclic graph, so you need to show all three of these.) Give an example to show that the same conclusion does not hold if we allow some weights in the graph to be nonpositive.