CSci 231 Homework 9 *

Graph Algorithms

CLRS Chapter 22, 24

1. [22.1-5] The *square* of a directed graph $G = (V, E)$ is a graph $G^2 = (V, E^2)$ such that $(u, w) \in E^2$ if and only if, for some $v \in V$, both $(u, v) \in E$ and $(v, w) \in E$. Thus, G^2 contains an edge between u and w whenever G contains a path with exactly two edges between u and w. Describe efficient algorithms for computing G^2 from G from both the adjacency-list and the adjacency-matrix representatiosn of G. Analyze the running time of your algorithms.

2. [22.2-8] Let $G = (V, E)$ be a connected, undirected graph. Give an $O(V + E)$-time algorithm to compute a path in G that traverses each edge in E exactly once in each direction. Describe how you can find your way out of a maze if you are given a large supply of pennies.

3. [22.4-3] Give an algorithm that determines whether or not an undirected graph $G = (V, E)$ contains a cycle. Your algorithm should run in $O(V)$ time, independent of $|E|$.

4. [22.4-5] One way to perform topological sorting on a DAG is to repeatedly find a vertex of in-degree 0, output it, and remove it and all of its outgoing edges from the graph. Explain how to implement this idea so that it runs in $O(V + E)$ time. What happens to this algorithm of G has cycles?

5. [22-3] *Euler tour.* An Euler tour of a connected, directed graph $G = (V, E)$ is a cycle that traverses each edge of G exactly once, although it may visit a vertex more than once.

   (a) Show that G has an Euler tour if and only of in-degree(v) = out-degree(v) for each vertex $v \in V$.

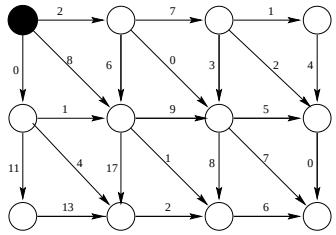
   (b) Describe an $O(E)$-time algorithm to find an Euler tour if one exists. (Hint: merge edge-disjoint cycles).

   Dijkstra'a SSSP algorithm works on general graphs with non-negative weights. The running time of Dijkstra's algorithm is $O(|E|+|V|\times$ INSERT $+|V|\times$ DELETE-MIN $+|E|\times$ CHANGE-KEY). Assuming the graph is connected and the priority queue is implemented as a heap the running time is $O(|E| \log |V|)$. The running time can be improved to $O(|E|+|V| \log |V|)$ using improved versions of priority queue (for instance the Fibonacci heap, which supports INSERT and CHANGE-KEY in $O(1)$ time amortized, and DELETE-MIN in $O(\lg n)$ amortized). While Dijkstra's algorithms gives the best known upper bounds for general SSSP with general non-negative weights and linear space, improved algorithms are known for special classes of graphs. In the following problems you will investigate several examples and derive improved bounds for computing SSSP.

6. *Shortest path for Directed Acyclic Graphs (DAGs):* Let $G = (V, E)$ be a DAG and let s be a vertex in G. Find a linear time $O(|V| + |E|)$ algorithm for computing SSSP(s). What vertices are reachable from s? Sketch a proof that your algorithm is correct. Does your algorithm need the constraint that the edge weights are non-negative?

*Collaboration is allowed and encouraged, if it is constructive and helps you study better. Remember, exams will be individual. Write the solutions individually, and list the names of the collaborators along with the solutions.

7. Consider a directed weighted graph with non-negative weights and V vertices arranged on a rectangular grid. Each vertex has an edge to its southern, eastern and southeastern neighbours (if existing). The northwest-most vertex is called the root. The figure below shows an example graph with V=12 vertices and the root drawn in black:



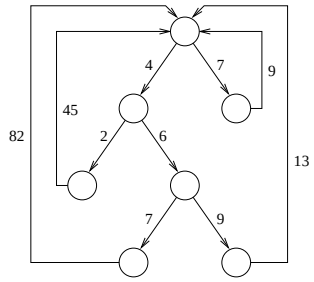
Assume that the graph is represented such that each vertex can access **all** its neighbours in constant time.

(a) How long would it take Dijkstra's algorithm to find the length of the shortest path from the root to all other vertices?

(b) Describe an algorithm that finds the length of the shortest paths from the root to all other vertices in $O(V)$ time.

(c) Describe an efficient algorithm for solving the all-pair-shortest-paths problem on the graph (it is enough to find the length of each shortest path).

8. Consider a directed weighted graph with non-negative weights which is formed by adding an edge from every leaf in a binary tree to the root of the tree. Let the graph/tree have n vertices. An example of such a graph with $n = 7$ could be the following:



We want to design an algorithm for finding the shortest path between two vertices in such a graph.

(a) How long time would it take Dijkstra's algorithm to solve the problem?

(b) Describe and analyze a more efficient algorithm for the problem.