CS 161: Homework 5

Due by November 11, 2016 at noon

Instructions: Please answer the following questions to the best of your ability. Provide full and rigorous proofs and include all relevant calculations. When writing proofs, please strive for clarity and brevity (in that order). Please see the course website for submission instructions and collaboration policy. Cite any sources that you use.

Remember that when you design an algorithm, in addition to an appropriate description of your algorithm, you must provide sufficient explanation for its correctness and analyze its running time. See the Homework Advice document on the course website for details.

In your submission, please start each question on a new page.

Question 1 (20 points)

On Monday afternoon an explosion in Shelby County, Alabama severed the Colonial Pipeline which transports more than 100 million gallons of refined petroleum products from Houston to the southeast every day. The incident caused gasoline prices to jump 15% and might cause jet fuel and gasoline shortages in the region. This event has made you concerned about fuel supply/price stability in the US.

As a student of graph theory, you would like to analyze this system. Here we will consider the robustness of the American fuel distribution system under a few simplifications. Let our system be a graph such that:

- Each edge in the graph is a pipe.
- Each edge is directed (fuel flow is not bidirectional).
- Each node in the graph is a combination refinery/distribution terminal (nodes both produce and consume fuel products).
- Every refinery produces the same fuel that every distribution terminal accepts. In other words, we will not worry about dedicated pipelines for different products.

You feel that our nation should have a more robust system by two measures:

1. There is a way to transport fuel from any node in the graph to any other node in the graph.
2. No single pipe failure can isolate any node from either distributing fuel to the rest of the system or consuming fuel from the rest of the system. That is, removing a single edge cannot cause any node to be unreachable from any other node.

(a) (10 points) Let the number of edges (pipes) in the graph be m and the number of nodes (refineries/distribution terminals) be n. Design an algorithm to test property 1 in $O(n + m)$ time.

(b) (10 points) Design an algorithm to test property 2 which uses the algorithm from part (a) as a black box. (Note: there are more efficient algorithms to test property 2.)

Question 2 (30 points)

Suppose we have n boolean variables

```
boolean b1
boolean b2
. . .
boolean bn
```

and m conditionals

```
if a1 or a2: // conditional 1
  // do task 1
if a3 or a4: // conditional 2
  // do task 2
. . .
```

where each conditional is the disjunction of two literals `ai` which are each either a boolean variable `(bj)` or some negation of a boolean variable `(not bj)`. For example, one such list of conditionals is

```
if b1 or (not b2):
  // do task 1
if b2 or b3:
  // do task 2
if (not b1) or (not b2):
  // do task 3
```

In this problem, we will devise a polynomial time algorithm to assign all n boolean variables such that every conditional evaluates to true, and every task is executed (if possible). For example, in the above example, the algorithm might output

```
b1 = True
b2 = False
b3 = True
```

Consider a set of $2n$ vertices V where n vertices correspond to the n boolean variables $\{\texttt{b1}, \ldots, \texttt{bn}\}$ and the other n correspond to their negations $\{\texttt{not b1}, \ldots, \texttt{not bn}\}$. Denote the vertex that corresponds to boolean variable `bi` as $v_{\texttt{bi}}$, and the vertex that corresponds to its negation as $v_{\texttt{(not bi)}}$. Construct a directed graph on these vertices such that for every conditional `if c or d`, we add the two directed edges $(v_{\texttt{not c}}, v_{\texttt{d}})$ and $(v_{\texttt{not d}}, v_{\texttt{c}})$. (Note that `c` is any literal and can be a negation of a variable, and `not not c = c`.)

(a) (3 points) Show that all tasks are executed if and only if for every edge $(\texttt{a}, \texttt{b})$ in the graph, if $\texttt{a} = \texttt{True}$, then $\texttt{b} = \texttt{True}$. Hence, we can consider edges of the graphs as "implications": `a` implies `b`.

(b) (8 points) Show that if any boolean variable `bi` lies in the same strongly connected component as its negation `(not bi)` in this graph, then there is no assignment of the n boolean variables that satisfies all m clauses.

(c) (3 points) Show that if there is a path in the graph $(\texttt{a} \to \texttt{b})$ then there is a path $(\texttt{not b} \to \texttt{not a})$.

(d) (3 points) In the following parts, we assume that no literal is in the same strongly connected component as its negation. Consider one strongly connected component $S \subset V$ of this graph. Show that the negations of every literal $a \in S$ are contained in a single other strongly connected component $\bar{S} \subset V$.

(e) (3 points) If a particular strongly connected component S is a sink node of the SCC meta-graph (i.e., there are no edges from any vertex in S to any vertex outside S), what do we know about $\bar{S}$?

(f) (10 points) Design an algorithm that finds an assignment that leads to an execution of all tasks (assuming no literal is in the same strongly connected component as its negation). Prove that your algorithm gives a valid truth assignment to each literal that satisfies every conditional, and show that it has running time polynomial in n and m.

Question 3 (30 points)

Dijkstra's algorithm will in general fail on graphs with negative-weight edges. In this problem, we will explore an interesting special case with negative edges for which we can still solve the shortest paths problem as quickly as Dijkstra's algorithm.

Recall that any directed graph $G = (V, E)$ can be broken up into its *strongly connected components* (SCCs)—in other words, a partition of V into disjoint vertex sets C_i such that within each C_i, any two vertices have a path to one another.

Suppose that in addition to the graph $G = (V, E)$, we are given an *ordered* set of its SCCs, $V = C_1 \cup C_2 \cup \cdots \cup C_k$, with the promise that:

- Within each component C_i, all edges have nonnegative weight.
- Edges between different components may have negative weight. However, they all obey the following rule: for any edge (u, v) such that $u \in C_i$ and $v \in C_j$ for $i \neq j$, we are promised that $i < j$. In other words, an edge out of one component can only point to a component "to the right".

It is helpful to draw some examples of graphs that obey this structure.

Devise an algorithm that, given G, an ordered set $\{C_1, C_2, \ldots, C_k\}$ of its SCCs obeying the above constraints, and a starting node s, finds the shortest path lengths from s to all other nodes $v \in V$. It should have running time $O((|E| + |V|) \log |V|)$.

Question 4 (20 points)

Alice and Bob are planning a road trip from San Francisco to Los Angeles. Assume that there are a set of roads that connect the cities between San Francisco and Los Angeles. We model this as a graph of n nodes (cities) connected by m edges (roads). Each edge weight corresponds to the time is takes to travel along that road, so the edge weight for the road between city u and city v is $w(u, v) = t_{u,v}$.

There are two types of roads: regular roads and toll roads. All toll roads cost the same amount and are generally (although not always) faster than regular roads. Alice and Bob would like to reach Los Angeles as fast as possible, however they only have enough money to take at most one toll road.

(a) (10 points) Explain how to modify the graph of roads and cities to represent this problem using a weighted, directed graph with $2n$ vertices.

(b) (10 points) Provide an efficient algorithm that uses the graph from part (a) to find the shortest path between San Francisco and Los Angeles that uses at most one toll road.