Homework for Introduction to Algorithms (Spring of 2021)
Instructor: Kyuseok Shim

1. (Greedy Algorithm) Given a set $\{x_1 \leq x_2 \leq \ldots \leq x_n\}$ of points on the real line, determine the smallest set of unit-length closed intervals (e.g. the interval $[1.25, 2.25]$ includes all x_i such that $1.25 \leq x_i \leq 2.25$) that contains all of the points. Give the most efficient algorithm you can to solve this problem, prove it is correct and analyze the time complexity. [20 pts.]

2. **(Minimum Spanning Tree):** In design of electronic circuitry, it is often necessary to make the pins of several components electrically equivalent by wiring them together. To interconnect a set of n pins, we can use an arrangement of $n-1$ wires, each connecting two pins. of all such arrangements, the one that uses the least amount of wire is usually the most desirable.

   We can model this wiring problem with a connected, undirected graph $G = (V, E)$, where V is the set of pins, E is the set of possible interconnections between pairs of pins, and for each edge $(u, v) \in E$, we have a weight $w(u, v)$ specifying the cost (amount of wire needed) to connect u and v. We then wish to find an acyclic subset $T \subset E$ that connects all of the vertices and whose total weight

   $$w(T) = \sum_{(u,v)\in T} w(u, v)$$

   is minimized. Since T is acyclic and connects all the vertices, it must form a tree, which we call a spanning tree since it "spans" the graph G. We call the problem of determining the tree T the *minimum-spanning-tree problem.*

   Assume that we have a connected, undirected graph $G = (V, E)$ with a weight function $w : E \to R$, and we wish to find a minimum spanning tree for G. The two algorithms we consider in this chapter use a greedy approach to the problem, although they differ in how they apply this approach.

   This greedy strategy is captured by the following "generic" algorithm, which grows the minimum spanning tree one edge at a time.

   Generic-MST(G, w)
   begin
   1. $A = \emptyset$
   2. **while** A does not form a spanning tree **do**
   3. find an edge (u, v) that is safe for A
   4. $A = A \cup \{(u, v)\}$
   5. **return** A

   end

The algorithm manages a set of edges A, maintaining the following loop invariant:

Prior to each iteration, A is a subset of some minimum spanning tree.

At each step, we determine an edge (u, v) that can be added to A without violating this invariant, in the sense that $A \cup \{(u, v)\}$ is also a subset of a minimum spanning tree. We call such an edge a *safe edge* for A, since it can be safely added to A while maintaining the invariant.

A *cut(S,V-S)* of an undirected graph G = (V, E) is a partition of V. We say that an edge $(u, v) \in E$ *crosses* the cut$(S, V - S)$ if one of its endpoints is in S and the other is in $V - S$. We say that a cut *respects* a set A of edges if no edges in A crosses the cut. Note that there can be more than one light edge crossing a cut in the case of ties. More generally, we say that an edge is a *light edge* satisfying a given property if its weight is the minimum of any edge satisfying the property. Our rule for recognizing safe edges is given by the following theorem.

Theorem 0.1: *Let $G = (V, E)$ be a connected undirected graph with a real-valued weight function w defined on E. Let A be a subset of E that is included in some minimum spanning tree for G, let $(S, V - S)$ be any cut of G that respects A, and let (u, v) be a light edge crossing $(S, V - S)$. Then, edge (u, v) is safe for A.* ∎

(a) Prove the above Theorem. [4 pts.]

(b) Write the pseudo-code of Prim's algorithm. Show why Prim's algorithm is correct? What is the time complexity of Prim's algorithm? [3 pts.]

(c) For the graph shown in Figure 1, show the stages of Prim's algorithm with v_1 as the root to show how each edge is selected in each step. [3 pts.]

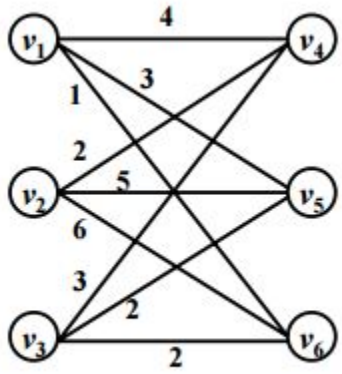

Figure 1: A sample graph

(d) Let G =(V,E) be a connected, undirected graph with edge-weight function $w : E \to R$, and assume all edge weights are distinct. Consider a cycle $< v_1, v_2, ..., v_k, v_{k+1} >$ in G, where $v_{k+1} = v_1$, and let (v_i, v_{i+1}) be the edge in the cycle with the largest edge weight. Prove that (v_i, v_{i+1}) does not belong to the minimum spanning tree T of G. [10 pts.]

(e) A graph G has edges which are colored either red or blue. Give the fastest algorithm that you can to compute a spanning tree with as few blue edges as possible. What is the run time of your algorithm? [5 pts.]

3. (Bellman-Ford algorithm) Consider the following properties.

Lemma 0.2: (Triangle inequality):
Let $G = (V, E)$ be a weighted, directed graph with weight function $w : E \to R$ and source vertex s. Then, for all edges $(u, v) \in E$, we have

$$\delta(s, v) \leq \delta(s, u) + w(u, v)$$

Lemma 0.3: (Upper-bound property):
Let $G = (V, E)$ be a weighted, directed graph with weight function $w : E \to R$. Let $s \in V$ be the source vertex, and let the graph be initialized by INITIALIZE-SINGLE-SOURCE(G,s). Then, $d[v] \geq \delta(s, v)$ for all $v \in V$, and this invariant is maintained over any sequence of relaxation steps on the edges of G. Moreover, once $d[v]$ achieves its lower bound $\delta(s, v)$, it never changes.

Corollary 0.4: (No-path property):
Suppose that in a weighted, directed graph $G = (V, E)$ with weight function $w : E \to R$, no path connects a source vertex $s \in V$ to a given vertex $v \in V$. Then, after the graph is initialized by INITIALIZE-SINGLE-SOURCE(G,s), we have $d[v] = \delta(s, v) = \infty$, and this equality is maintained as an invariant over any sequence of relaxation steps on the edges of G.

Lemma 0.5: (Convergence property):
Let $G = (V, E)$ be a weighted, directed graph with weight function $w : E \to R$, let $s \in V$ be a source vertex, and let $s \rightsquigarrow u \to v$ be a shortest path in G for some vertices $u, v \in V$. Suppose that G is initialized by INITIALIZE-SINGLE-SOURCE(G,s) and then a sequence of relaxation steps that includes the call $RELAX(u, v, w)$ is executed on the edges of G. If $d[u] = \delta(s, u)$ at any time prior to the call, then $d[v] = \delta(s, v)$ at all times after the call.

Lemma 0.6: (Path-relaxation property):
Let $G = (V, E)$ be a weighted, directed graph with weight function $w : E \to R$ and let $s \in V$ be a source vertex. Consider any shortest path $p =< v_0, v_1, \ldots, v_k >$ from $s = v_0$ to v_k. If G is initialized by INITIALIZE-SINGLE-SOURCE(G,s) and then a sequence of relaxation steps occurs that includes, in order, relaxations of edges (v_0, v_1), (v_1, v_2), $\ldots$, (v_{k-1}, v_k), then $d[v_k] = \delta(s, v_k)$ after these relaxations and at all times afterward. This property holds no matter what other edge relaxations occur, including relaxations that are intermixed with relaxations of the edges of p.

- For the graph shown in Figure 2, show the step-by-step change of the $d[]$ values after each iteration of the outer-loop of the pseudo code. (The source vertex is 1.) [5 pts.]
- Let $G = (V, E)$ be a weighted, directed graph with source s and weight function $w : E \to R$, and assume that G contains no negative-weight cycles that are reachable from s. Then, after the iterations of applying RELAX operations with for-loop in Bellman-Ford algorithm, show that $d[v] = \delta(s, v)$ for all vertices that are reachable from s. [5 pts.]

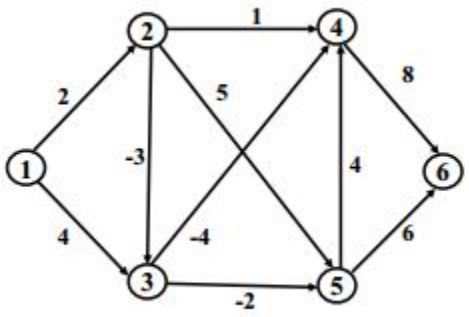

Figure 2: A sample graph

4. (Dijkstra's algorithm)

   - Show the correctness of Dijkstra's algorithm. In other words, prove that Dijkstra's algorithm, run on a weighted, directed graph $G = (V, E)$ with non-negative weight function w and source s, terminates with $d[u] = \delta(s, u)$ for all vertices $u \in V$. [5 pts.]
   - You are given a strongly connected directed graph G = (V, E) with positive edge weights along with a particular node $v_0 \in V$. Describe an $O((|V| + |E|) \lg |V|)$ time algorithm for finding shortest paths between all pairs of nodes, with the one restriction that these paths must all pass through v_0. [10 pts.]

5. (NP-complete) Consider the following NP-complete problems.

   - HAMILTONIAN PATH PROBLEM:
     Given a graph $G = (V, E)$, does G contain a Hamiltonian path? For a graph $G = (V, E)$, a simple path in G is a sequence $\langle v_1, v_2, \ldots, v_k \rangle$ of distinct vertices from V such that $(v_i, v_{i+1}) \in E$ with $1 \leq i < k$. A Hamiltonian path in G is a simple path that includes all the vertices of G.
   - CLIQUE PROBLEM:
     Given a graph $G = (V, E)$ and a positive integer K, does G contain a clique of size K? For a graph $G = (V, E)$, a clique of size K is defined as a subset of vertices $V' \subset V$, each pair of which is connected by an edge in E and $|V'| = K$.

   Answer the following questions:

   - Show DEGREE-CONSTRAINED SPANNING TREE PROBLEM is NP-complete.
     DEGREE-CONSTRAINED SPANNING TREE PROBLEM: Let $G = (V, E)$ be a graph and K be a positive integer. Does G have a spanning tree in which no node has degree greater than K? [20 pts.]
   - Show MAXIMUM 2-CNF-SAT PROBLEM is NP-complete.
     MAXIMUM 2-CNF-SAT PROBLEM: A boolean formula is called 2-conjunctive normal form (2-CNF), if each clause has exactly two distinct literals. Given a 2-CNF ϕ and a positive integer K, is there any assignment which satisfies at least K of the clauses? [20 pts.]