COMS 4231: Analysis of Algorithms I, Fall 2018

Problem Set 5, due Wednesday November 21, 11:59pm on Courseworks.

Please follow the homework submission guidelines posted on Courseworks.

- *As usual, for each of the algorithms that you give, include an explanation of how the algorithm works and show its correctness.*
- *Make your algorithms as efficient as you can, state their running time, and justify why they have the claimed running time. All time bounds below refer to worst-case complexity, unless specified otherwise.*

Problem 1. An undirected graph *G*=(*N*,*E*) is called *bipartite* if its set *N* of nodes can be partitioned into two subsets N_1, N_2 ($N_1 \cap N_2 = \emptyset$, $N_1 \cup N_2 = N$) so that every edge connects a node of N_1 with a node of N_2.

a. Prove that if a graph contains a cycle of odd length then it is not bipartite.

b. Give a O(*n*+*e*)-time algorithm that determines whether a given graph is bipartite, where *n* is the number of nodes and *e* is the number of edges; the graph is given by its adjacency list representation.

If the graph is bipartite, then the algorithm should compute a bipartition of the nodes according to the above definition.

If the graph is not bipartite then the algorithm should output a cycle of odd length.

c. We are given a set of *non-equality* constraints of the form $x_i \neq x_j$ over a set of Boolean variables $x_1, x_2, ..., x_n$. We wish to determine if there is an assignment of Boolean values 0,1 to the variables that satisfies all the constraints, and compute such a satisfying assignment if there is one. Show that this problem can be solved in time O(*n*+*m*), where *n* is the number of variables and *m* is the number of constraints.

Problem 2. A weighted graph may have many different shortest (minimum-weight) paths between two nodes. In this case we may prefer among them one that has the minimum number of edges. Modify Dijkstra's algorithm to solve the following problem:

Input: A weighted directed graph G=(N,E) with positive weights w: E→R_+ on its edges, and two nodes *s*, *t*.

Output: A minimum-weight path from *s* to *t* that contains as few edges as possible.

Problem 3. We are given a set *V* of *n* variables $\{x_1, x_2, \ldots, x_n\}$ and a set *C* of *m* weak and strict inequalities between the variables, i.e., inequalities of the form $x_i \leq x_j$ or $x_i < x_j$. The set *C* of inequalities is called *consistent* over the positive integers Z^+ iff there is an assignment of positive integer values to the variables that satisfies all the inequalities.

For example, the set $\{ x_1 \leq x_3,\ x_2 < x_1 \}$ is consistent, whereas $\{ x_1 \leq x_3,\ x_2 < x_1,\ x_3 < x_2 \}$ is not consistent.

a. Given a set C of m inequalities between n variables as above, let G be the directed graph with node set $V = \{x_1, x_2, \ldots, x_n\}$ and edge set $E = \{(x_i, x_j) \mid C \text{ contains } x_i \le x_j \text{ or } x_i < x_j\}$, i.e., G has one node for every variable and one edge for every inequality of C. Show that the set C is not consistent if and only if it contains a strict inequality $x_u < x_v$ such that the graph G has a path from x_v to x_u.

b. Give an O(n+m)-time algorithm to determine whether a given set C of inequalities is consistent over the positive integers.

c. If the set of inequalities has a solution over the positive integers, then it has a unique minimum solution, i.e. a solution in which every variable has the minimum value among all possible solutions. For example the minimum solution for the set $\{x_1 \le x_3,\ x_2 < x_1\}$ is $x_1=2$, $x_2=1$, $x_3=2$. Give an O(n+m)-time algorithm to compute the minimum solution of a given consistent set C.

Problem 4. Do Problem 22-2, parts a-d in CLRS (page 622) on articulation points. (Do for yourself the remaining parts also (e-h), but you do not have to turn them in.) We reproduce parts a-d below for convenience.

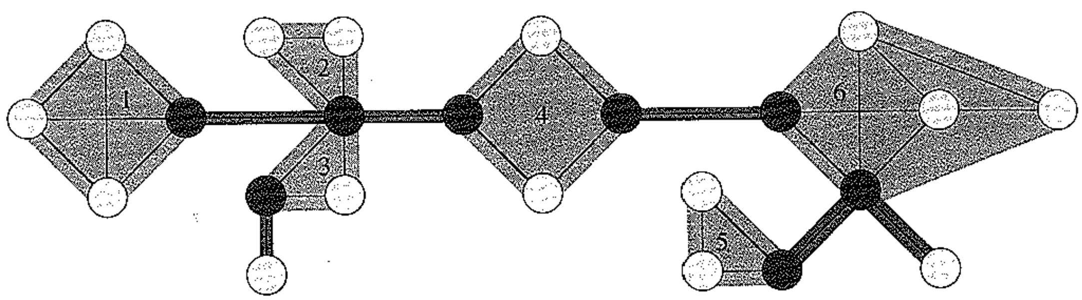

Figure 22.10 The articulation points, bridges, and biconnected components of a connected, undirected graph for use in Problem 22-2. The articulation points are the heavily shaded vertices, the bridges are the heavily shaded edges, and the biconnected components are the edges in the shaded regions, with a *bcc* numbering shown.

22-2 *Articulation points, bridges, and biconnected components*

Let $G = (V, E)$ be a connected, undirected graph. An ***articulation point*** of G is a vertex whose removal disconnects G. A ***bridge*** of G is an edge whose removal disconnects G. A ***biconnected component*** of G is a maximal set of edges such that any two edges in the set lie on a common simple cycle. Figure 22.10 illustrates these definitions. We can determine articulation points, bridges, and biconnected components using depth-first search. Let $G_\pi = (V, E_\pi)$ be a depth-first tree of G.

a. Prove that the root of G_π is an articulation point of G if and only if it has at least two children in G_π.

b. Let v be a nonroot vertex of G_π. Prove that v is an articulation point of G if and only if v has a child s such that there is no back edge from s or any descendant of s to a proper ancestor of v.

c. Let

$$v.low = \min \begin{cases} v.d\,, \\ w.d : (u, w) \text{ is a back edge for some descendant } u \text{ of } v\,. \end{cases}$$

Show how to compute $v.low$ for all vertices $v \in V$ in $O(E)$ time.

d. Show how to compute all articulation points in $O(E)$ time.

Problem 5. a. We showed in class that every minimum spanning tree T of a (weighted undirected, connected) graph G has the following property: the weight of every nontree edge (u,v) of the graph is at least as large as the weight of all the edges on the u-v path of the tree T. In this part you will use Kruskal's algorithm to show the converse: If a spanning tree T has this property then T is a minimum spanning tree.
For this purpose, suppose that T is a spanning tree that has this property, i.e. the weight of every nontree edge (u,v) is at least as large as the weight of all the edges on the u-v path of the tree T. Let $L=\langle e_1, e_2, \ldots, e_m \rangle$ be a sorted listing of the edges of G in non-decreasing order of weight with ties broken in favor of the edges of T, i.e., if several edges have the same weight, then L lists first the edges with this weight that are in T (if any) and then the edges that are not in T. Suppose that we run Kruskal's algorithm with the edges processed in the order of L, and let T_K be the tree computed by Kruskal's algorithm. Use induction to show that for all i=1,…,m, the edge $e_i \in T_K$ if and only if $e_i \in T$. Conclude that $T_K = T$, and therefore, by the correctness of Kruskal's algorithm, T is a minimum spanning tree.

b. We are given a weighted graph G and a minimum spanning tree T of G. Both G and T are given by adjacency list representations. Suppose that we decrease the weight of one of the edges that is not in T. Give an O(n)-time algorithm that computes the minimum spanning tree of the modified graph, where n is the number of nodes.

Exercises for your practice (do not turn in):

Practice the graph algorithms that we learn on some graphs of your choice.

For example, draw an arbitrary directed or undirected graph, pick a source node and apply Breadth-First Search. Compute the BFS tree, the distances, and the partition of nodes into layers. Does the tree depend on the order in which nodes appear in the adjacency lists? (Recall that the adjacency list of a node contains the adjacent nodes in arbitrary order.) How about the distances and the layers?

Similarly with Depth-First Search. Compute the DFS tree, the discovery and finish times of the nodes, classify the edges (tree, forward, back, cross). Compute the strongly connected components for a directed graph. For example, do exercises 22.3-2 and 22.5-2.

Draw a weighted undirected graph, and apply Prim's and Kruskal's algorithm to it.

Similarly, practice Dijkstra's algorithm and all the other graph algorithms that we do in class.