CS-UY 2413: Design&Analysis of Algorithms—Spring 2017 Prof. Boris Aronov
Department of Computer Science and Engineering Prof. Lisa Hellerstein

Homework 7

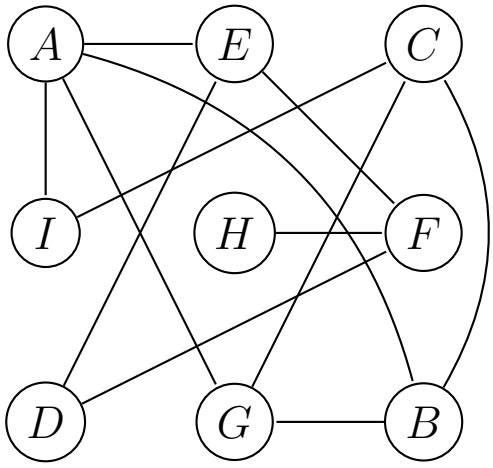
Due April 12 at the start of lecture (Aronov's sections).
Due April 13 at the start of lecture (Hellerstein's section).
This homework should be handed in on paper. No late homeworks accepted. Contact your professor for special circumstances.

Policy on collaboration on this homework: The policy for collaboration on this homework is the same as in previous homeworks. By handing in this homework, you accept that policy. Remember: A maximum of 3 people per group.

Notes: (i) Every answer has to be justified unless otherwise stated. Show your work! All performance estimates (running times, etc) should be in asymptotic notation unless otherwise noted.

(ii) You may use any theorem/property/fact proven in class or in the textbook. You do not need to re-prove any of them.

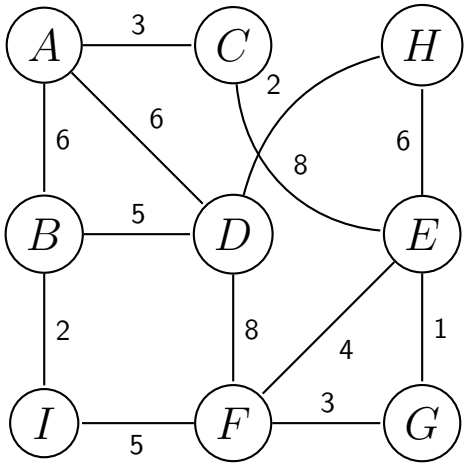
1. Run BFS on the given connected undirected graph, starting with vertex A. Show BFS tree and distance from A. Mark which edges are tree (T) edges and which are cross (X) edges.



2. Use Dijkstra's Algorithm to find the shortest distance from A to all other vertices of the graph given below. Draw a table showing the values in the `dist` and `prev` arrays after each iteration of the while loop in the pseudocode for Dijkstra's algorithm in Figure 4.8 of the textbook; use the format below.

Set	A	B	C	D	E	F	G	H
{}	0/nil	∞/nil	∞/nil	∞/nil	∞/nil	∞/nil	∞/nil	∞/nil
A	0/nil	6/A	3/A	6/A	∞/nil	∞/nil	∞/nil	∞/nil

Use alphabetical ordering to break any ties (favoring the letter that is closer to A). (Note: the weight of CE is 8 and that of DH is 2.)



3. For each of the following situations, state whether it is possible. If not, explain why not. If yes, give a (small) example when it happens:

   (a) We run BFS on a connected undirected graph starting with vertex s. We remove an edge of the BFS tree and re-run BFS from s. The graph remains connected and all the distances to s remain the same.

   (b) We run BFS on a connected undirected graph starting with vertex s. We remove an edge that does not belong to the BFS tree. Some distances to s change.

   (c) We run BFS on a connected undirected graph with 6 vertices, starting with vertex s. We remove an edge of the BFS tree and re-run BFS from s. The graph remains connected, but there is a vertex v whose distance from s increases by 4.

4. Suppose we are given a directed graph G in adjacency list form: We have an array of list headers. Each list header $A[v]$ contains a pointer to the first element of a singly-linked list containing the vertices adjacent to vertex v and nothing else. G has n vertices and m edges. Assume vertices of G are numbers 1 through n.

   (a) For each of the following operations, explain how to implement it and give its worst-case running time, as a function of n and m.
   - Given a pair (v, u), determine if it is an edge of G.
   - Given a vertex v, determine if it is a *source*: its *in-degree* is 0.
   - Given a vertex v, determine if it is a *sink*: its *out-degree* is 0.

   (b) Now design a new way to represent the graph, that still takes $O(n + m)$ space (so no adjacency matrices!) that speeds up as many of the above operations as you can. Describe the new representation, and the new implementation of each operation, and its worst-case running time.

   No hashing, please.

5. Modify the pseudocode for Dijkstra's algorithm in Figure 4.8 so that for every vertex v, it computes a Boolean value mulitple(v) which is True if there is more than one shortest path from s to v, and False otherwise. Write out the entire pseudocode, including your modification. (You do not need to include the descriptions of the Input and Output.)

   Give a brief explanation, in English, of how and why your modification works.

6. **Hellerstein's Section Only:**

   You did not do the Strongly Connected Components problems in Homework 6. To give you practice on this subject, you should do Problems 5c and 5d from Homework 6, but you don't need to hand them in. Check the solution set to Homework 6 to see if your answers are correct.

   Then, do the following problem, and HAND IT IN with your solutions to the rest of the problems on this homework.

   On the given directed graph, run the strongly connected components (SCC) algorithm described in the textbook. Show the first DFS forest, give the list of vertices in decreasing order of their post times, and finally show the strongly connected components identified by the algorithm. (You do not need to show the second DFS forest that is computed.)