College of Computer & Information Science
Northeastern University
Spring 2019

CS5800
Algorithms
15 March 2019

Sample Solution to Quiz 2

1. (2 + 2 + 6 = 10 points) Identifying powerbrokers

You work for a public relations firm and have just acquired complete information on the Washington lobbying network. In particular, you have a list V of n persons (congressmen, bureaucrats, lobbyists, businessmen, etc.) and for each person i in V, a list of persons in V that i can *influence*. (Note that it is possible that i can influence j but j cannot influence i.)

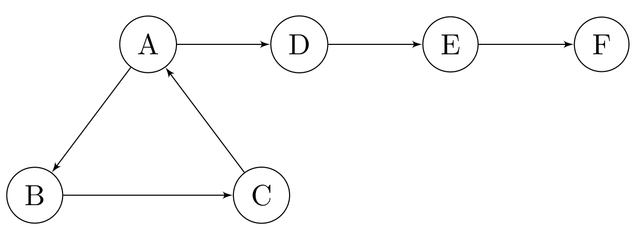
Having just studied graph algorithms, you immediately capture the above information by a directed graph G with V as the set of vertices and the set E of edges defined as follows.

$$E = \{(i, j) : i \text{ can influence } j\}.$$

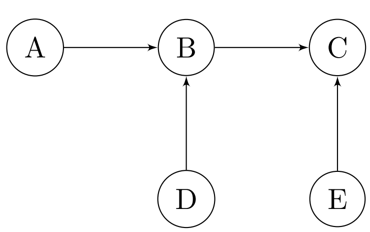
We call a person i a *powerbroker* if for every other person $j \in V$, there is a path from i to j in G.

(a) (2 points) Draw an example lobbying network with ***six*** persons in which there are exactly ***three*** powerbrokers.

Answer: There are many graphs that will satisfy the requirement. The key is to have a strongly connected component with 3 nodes, from which all other nodes can be reached. The following is one such graph. Nodes A, B, and C are *powerbrokers*, while nodes nodes D, E, and F are not.



(b) (2 points) Draw an example lobbying network with ***five*** persons in which there are ***no*** powerbrokers.
Answer: Any directed graph whose SCC component DAG has more than one source will satisfy this requirement. The following example is a DAG, whose SCC DAG is itself. There are 3 nodes, A, D, and E which are source nodes and cannot be reached by any of the other nodes. C is a sink node, and B can only reach C. Thus, there are no *powerbrokers*.



(c) (6 points) Give a polynomial-time algorithm to determine *all* powerbrokers in the given lobbying network. If there are no powerbrokers, then your algorithm must indicate so. State the running time of your algorithm in terms of the number of vertices and edges of G. The more efficient your algorithm is in terms of its worst case running time, the more credit you will get.

Note: You do not need to prove the correctness or time complexity of your algorithm. Be sure to spell out your overall strategy. Your algorithm can reference any of the algorithms covered in class such as BFS, DFS, Topological Sort, Strongly Connected Components, etc. without going into the details.

Answer: In order for there to be any *powerbrokers* in a lobbying network, they will all need to be members of the source strongly connected component of the corresponding directed graph G. Furthermore, the source SCC should have a path to the remaining SCCs in G (therefore, it would be the only source).

Algorithm 1: Finding Powebrokers

---

1 Find the strongly connected components of graph G.
2 Define the SCC metagraph G_{SCC} each node of which is a strongly connected component of G.
3 Topologically sort the nodes of G_{SCC}
4 Let s denote the first node in this topological ordering.
5 Perform a DFS search starting at node s on the graph G_{SCC}
6 **if** *all of the nodes of G_{SCC} are present in the tree rooted at s* **then**
7   **return** *the members of the lobbying network that correspond to nodes in s as powerbrokers*
8 **else**
9   **return** *There are no powerbrokers in this lobbying network*
10 **end**

The SCCs of G (line 2) can be determined in time $\Theta(V + E)$. Each of steps 3 and 4 can be achieved in $\Theta(E)$ and $\Theta(V + E)$, respectively, as the number of nodes and edges in G_{SCC} are bounded by the number of nodes and edges in G. [Note: The algorithm in CLRS for line 2 returns the SCCs in topologically sorted order, so step in line 4 can be circumvented.] Checking for the existence of a path from s to every other node in G_{SCC} is achieved through a DFS search, which is accomplished in time $\Theta(V + E)$. As each step in the algorithm outlined above is bounded by $\Theta(V + E)$, the overall complexity of the algorithm is also $\Theta(V + E)$.