Algorithms Illuminated Part 2: Ch. 8 Andrew Goss

8.1 Which of the following statements hold?

a) True. BFS can be used to compute the connected components of an undirected graph in $O(m+n)$ time. As explained in the chapter, if UCC i has $|V| = n_i$ and $|E| = m_i$, the search through UCC i is $O(n_i + m_i)$ and $\sum n_i = n$, $\sum m_i = m$.

b) True. The key qualifier here is that "shortest" is defined as the minimum number of edges. This statement is not true in the general case of non-unit length edge "distances" or "weights." All that is required is incrementing and recording an index with each layer (transition from explored to unexplored vertex) in the search.

c). True. This is possible through Kosaraju's algorithm which consists of a single DFS to order the SCCs ($O(m+n)$) and a DFS per SCC which in total is also $O(m+n)$ using a similar explanation to part a) for running BFS per UCC.

d) True. If a graph is directed and acyclic, the algorithm will only "backtrack" beyond a node if all descendents have been explored. If order is assigned upon completing the exploration of (back tracking from) a node, all orders will meet the property that any vertices connected by a edge are ordered correctly. This (loosely) explains correctness. Only one DFS is required, and we take for granted that a single DFS is $O(m+n)$.

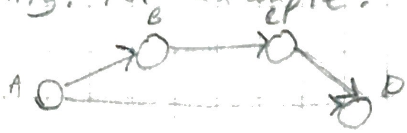
2.2 DFS is $O(m+n)$ with an adjacency list. What about with an adjacency matrix?

I will assume that the rows and columns are indexed such that, given a vertex, accessing its row or column is $O(1)$. In this case, the main difference from an adjacency list is the time required to get all edges for a given vertex. With an adjacency list, vertex n_i can be accessed in $O(1)$ and the associated edges m_i can be accessed in $O(m_i)$. With an adjacency matrix, associated edges can be accessed in $O(n)$ (by a linear scan over the appropriate column or row). Therefore, the overall running time for DFS with an adjacency matrix is $O(n^2)$. Note that for a dense graph, $O(m)$ is $\Theta(n^2)$, and the performance is similar, but for a sparse graph, the difference is significant.

✓

8.7

a) False, BFS can not be used for topological sorting. For example:



If we start from A, we get the following layers:

0: A

1: B, D

2: C

So layer indices do not provide a topological ordering, as C and D have an invalid order w.r.t. their edge.

b) False, for the reason explained above.

c) True, the second step reduces to the general search problem because BFS cannot travel "up" to a lower-ordered SCC nor can it travel down to a previously explored SCC.

d) False, per the answer to problem c).

8.8

a) True, this is a trivial reversal of the numbers assigned to indices but the second step of repeated DFS proceeds in the same order as the original algorithm.

b) True, the first step now orders the SCCs as sources while the second step operates on a reversed graph where the sources are valid sink orderings. The SCCs are the same for original and reversed graphs.

c) False, this is a trivial reversal of indices and as shown in the chapter this does not correctly order sinks of the input graph.

d) False, this is equivalent to C.