CS344: Design and Analysis of Computer Algorithms

Homework 3

Group Members: Stephen Kuo, Derek Mui

3.4) Run the strongly connected components algorithm on the following directed graphs G. When doing DFS on G^R: whenever there is a choice of vertices to explore, always pick the one that is alphabetically first.

In each case, answer the following questions:

(a) in what order are the strongly connected components found?

(b) Which are source SCC's and which are sink SCC's?

(c) Draw the "metagraph"

(d) What is the minimum number of edges you must add to this graph to make it strongly connected?

Solution:

DRAWINGS GO HERE

i) a) For graph i the ordering is
SCC 1 $\{C, J, F, H, I, G, D\}$
SCC 2 $\{A, E, B\}$

i) b) SCC 1 the node with the highest post number, C, so it is a source SCC in G^R and a sink SCC in G. SCC 2 is a source SCC in G^R and a sink SCC in G

i) a) For graph ii
SCC 1$\{D, G, H, I, F\}$
SCC 2 $\{C\}$
SCC 3 $\{A, E, B\}$

i) b) SCC 1 the node with the highest post number is D, so it s a source SCC in G^R and a sink SCC in G. SCC 3 is the source SCC in G since A has the highest post number in a DFS of G

i) c)

i) d) Since the meta graphs reveal a DAG, adding any edge from the sink to source will create a cycle , and make the entire graph strongly connected

3.5) The reverse of a directed graph G = (V, E) is another directed graph $G^R = (V, E^R)$ on the same vertex set, but with all edges reversed. Give a linear-time algorithm for computing the reverse of a graph in adjacency list format.

Solution:
*(Assume the graph vertices have been named 0,1,2...and so forth. Same as the array indices)

```
reverse(adjacencyList[][])
    Array reverseAdjacencyList[][] //array storing adj list of reverse graph G^R
    for each vertex in adjacency List
        for each neighbor in AdjacencyList[vertex]
            reverseAdjacencyList[neighbor].append(vertex)
        end
    end
    return reverseAdjacencyList
end of reverse
```

3.22) Give an efficient algorithm which takes as input a directed graph G = (V, E0 and determines if there is a vertex s in V from which all other vertices are reachable

Solution:
Perform a DFS. The node with the highest post number will be a vertex in a source SCC. Then perform a DFS from that vertex to see if every other vertex is reachable

3.25)