COMPSCI 2C03 – Week 11 Exercises

Sample solutions and notes on sample solutions for this week's exercises.

Lecture 1: Digraphs

1. Exercise 4.2.1: What is the maximum number of edges in a digraph with V vertices and no parallel edges? What does this tell us about the complexity of DFS and BFS?

   Note that no parallel edges does not rule out the possibility of a single self loop per vertex. So it's the max number of edges in undirected graph * 2 + V for the self loops.

   $V-1 + V-2 + \ldots + 1 = V(V-1)/2$. $V(V-1)/2*2+V = V^2-V+V = V^2$

   This means that if no parallel edges, DFS and BFS are both $O(V^2)$ in the worst case when there are no parallel edges.

2. Exercise 4.2.1: What is the minimum number of edges in a digraph with V vertices, none of which are isolated (i.e., indegree and outdegree are 0)?

   Each pair can have one edge connecting. Plus one edge for the odd one out if V is not even. So it's ceilng(v/2).

3. Exercise 4.2.2: Draw an adjacency list for the graph shown in Figure 1 shown at right.

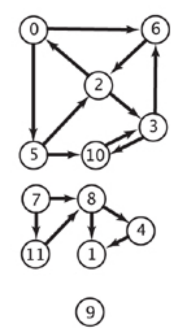


Figure 1

```
0: 5, 6
1: empty
2: 0, 3
3: 6, 10
4: 1
5: 2, 10
6: 2
7: 8, 11
8: 1, 4
9: empty
10: 3
11: 8
```

4. Exercise 4.2.7: A sink is a vertex with outdegree 0. A source is a vertex with indegree 0. Write pseudocode or code for two graph operations named **sinks** and **sources** that take a graph as a parameter (represented as an adjacency list). The operations return a collection of sinks and sources, respectively, for the given digraph. Your choice of ADT for the collection returned.

```
sinks(g):
      sinks ← create_list()
      for v ← 0 to num_vertices(g) - 1:
            if size(adj(g, v)) == 0:
                  add(sinks, v)
      return sinks

sources(g):
      source_array ← array (num_vertices(g)) of True
      for v ← 0 to num_vertices(g) - 1:
            for w in adj(g, v):
                  source_array[w] ← False
      sources ← create_list()
      for v ← 0 to num_vertices(g) - 1:
            if source_array[v]:
                  add(sources, v)
      return sources
```

Lecture 2: Topological Sort

5. Consider the graph G in Figure 2 below. Compute the topological sort for G (the list of vertices in topological order).

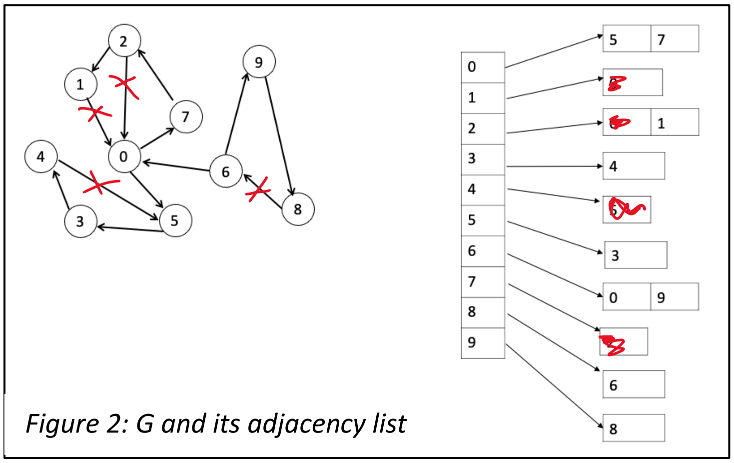


Figure 2: G and its adjacency list

```
Perform a post-order DFS:
Dfs(0)
      Dfs(5)
            Dfs(3)
                  Dfs(4)
                  DONE                  4
            DONE                        3
      DONE                              5
      Dfs(7)
            Dfs(2)
                  Dfs(1)
                  DONE                  1
            DONE                        2
      DONE                              7
DONE                                    0
Dfs(6)
      Dfs(9)
            Dfs(8)
            DONE                        8
      DONE                              9
DONE                                    6

Topological Sort: 6-9-8-0-7-2-1-5-3-4
```

Lecture 3: Strong Connectivity

6. How can the number of strongly connected components of a graph change if a new edge is added?

   Adding an edge either keep the number of strongly connected components unchanged if

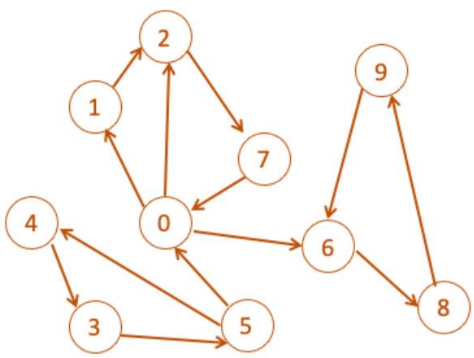
   - it is an intra-component edge (between the nodes of one component) or
   - it is an edge from component a to component b but there is no edge from b to a

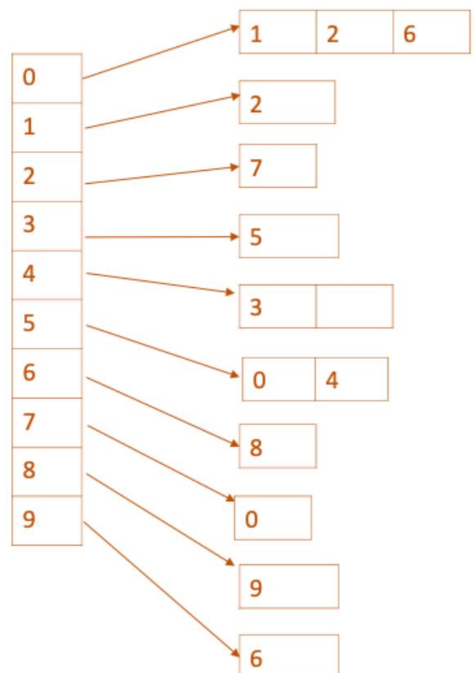
   It can decrease the number of connected components by one if

   - it is an edge from component a to component b and there is already an edge from b to a

7. Compute the strongly connected components of the digraph G given in Figure 2 above, using the Kosaraju-Sharir algorithm.

   Figure below represents G^R (the reverse of the graph shown in Figure 2).





- The post order of G^R starting from node 0: 7-2-1-9-8-6-0-4-5-3
- The reverse post order of G^R: 3-5-4-0-6-8-9-1-2-7
- Labeled nodes by DFS on G from the source 3: 3-4-5, next unlabeled node in above list is 0
- Labeled nodes by DFS on G from the source 0: 0-7-2-1, next unlabeled node in above list is 6
- Labeled nodes by DFS on G from the source 6: 6-9-8
- So there are three strong components: “3-4-5”, “0-7-2-1”, and “6-9-8”

8. True or False: If we modify the Kosaraju-Sharir algorithm to run the first depth-first search in the digraph G (instead of the reverse digraph G^R) and the second depth-first search in G^R (instead of G), then it will still find the strong components.

   True, the strong components of a digraph are the same as the strong components of its reverse.