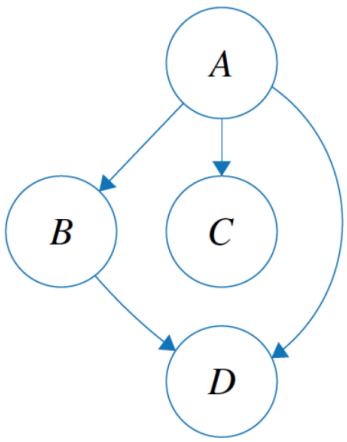
Graphs Worksheet

Representation



Draw the adjacency list and the adjacency matrix representation of the above graph.

DFS Traversal

Idea: We must look through all the values of our graph. So, given some starting point, we do a DFS traversal with the caveat that we now track what vertices we've visited before (to avoid cycles)

```
public void dfs() {
    Stack fringe = new Stack();
    Set visited = new Set();
    fringe.push(startVertex);
    while (!fringe.isEmpty()) {
        Vertex v = fringe.pop();
        if (!visited.contains(v)) {
            process(v); //Do something with v
            for (Vertex neighbor: v.neighbors) {
                fringe.push(neighbor);
            }
            visited.add(v);
        }
    }
}
```

Topological Sort

Idea: Given a directed, acyclic graph, how do we 'sort' the vertices based on their dependencies on one another?

```
public void topologicalSort() {
    Stack fringe = new Stack();
    Map currentInDegree = new Map<Vertex, Integer>();

    while (!fringe.isEmpty()) {
      Vertex v = fringe.pop();
        process(v); //Do something with v
        for (Vertex neighbor: v.neighbors) {
            currentInDegree(neighbor) -= 1; //Not actual Java code
            if (currentInDegree(neighbor) == 0) {
                fringe.push(neighbor);
            }
        }
    }

}
```

Exercise: Topologically sort the graph given in class. Draw out the `currentInDegree` map as well as the `fringe`.

Practice Problems

1. Provide a brief description of how to solve each of the following graph problems efficiently. Provide a worst case runtime bound in Θ notation in terms of V and E.

   (a) Find a path from node s to node t in a strongly connected, directed graph.

   (b) Determine if a cycle exists in a directed graph.

2. Consider the following implementation of DFS, which contains a crucial error:

```
public void dfs() {
    Stack fringe = new Stack();
    Set visited = new Set();

    fringe.add(startVertex);
    visited.add(startVertex);
    while (!fringe.isEmpty()) {
        v = fringe.pop();
        process(v);
        for (Vertex n: v.neighbors()) {
            if (!visited.contains(neighbor)) {
                fringe.push(neighbor);
                visited.add(neighbor);
            }
        }
    }
}
```

   Give an example of a graph where this algorithm may not traverse in DFS order.

Answers to #1 at http://datastructur.es/sp17/materials/discussion/discussion11epsol.pdf
Answers to #2 at http://datastructur.es/sp17/materials/discussion/discussion11sol.pdf