Data Structures Lab 12 Graphs Name:_____

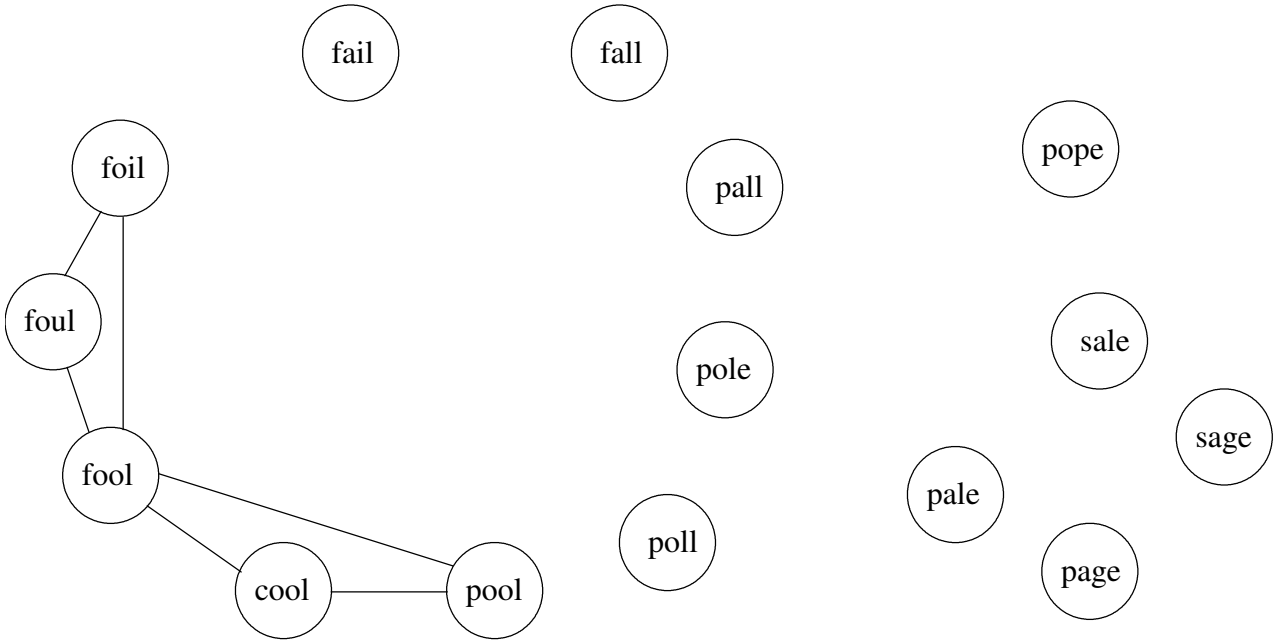
Objectives: To understand how a graph can be represented and traversed.

To start the lab: Download and unzip the file lab12.zip

Part A: In a word ladder puzzle you tranform one word into another by changing one letter at a time, e.g., transform FOOL into SAGE by FOOL → FOIL → FAIL → FALL → PALL → PALE → SALE → SAGE.

a) We can use a graph algorithm to solve this problem by constructing a graph such that
- words are represented by the vertices, and
- an edge represents?

b) For the words listed below, complete the graph by adding edges as defined above.



b) List a **different** transformation from FOOL to SAGE then the one given above

c) If we wanted to find the shortest transformation from FOOL to SAGE, what does that represent in the graph?

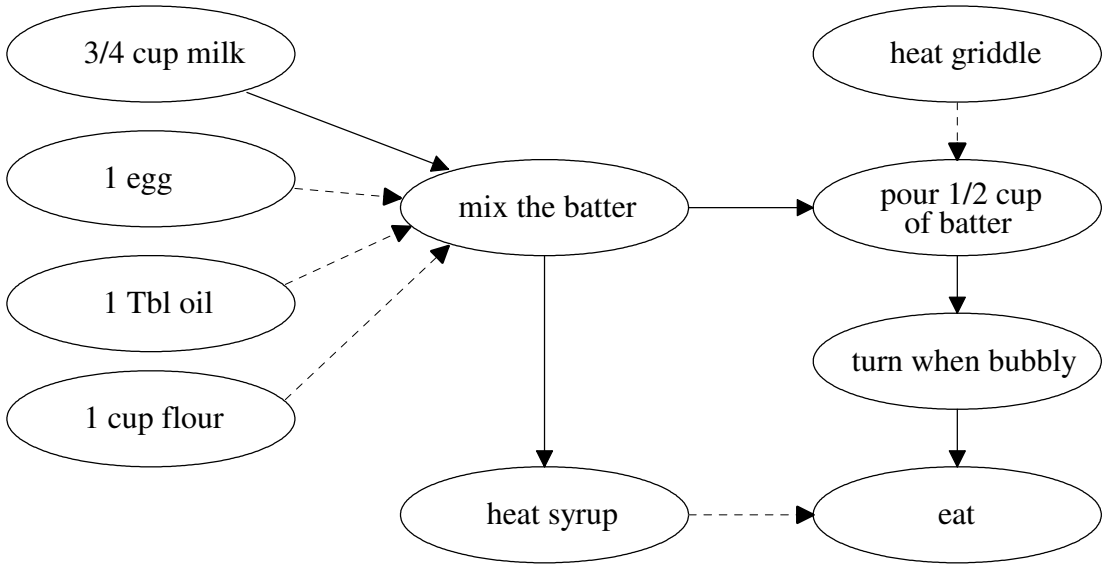
d) There are two general approaches for traversing a graph from some starting vertex *s*:

- Breadth First Search (BFS) where you find all vertices a distance 1 (directly connected) from s, before finding all vertices a distance 2 from s, etc.
- Depth First Search (DFS) where you explore as deeply into the graph as possible. If you reach a "dead end," we backtrack to the deepest vertex that allows us to try a different path.

Which of these traversals would be helpful for finding the shortest solution to a word ladder puzzle?

After you have answered the above questions, raise your hand and explain your answers.

Part B: The case study in section 20.9, discusses a menu-driven graph-testing program which uses the LinkedDirectedGraph class. The graph is input by specifying the directed edges as strings of the form: "p>q:0", where p and q are vertice labels with a directed edge from p to q with a weight of 0. Vertices are inferred from the edges, except for disconnected vertices that are strings of just vertex labels. Consider the graph for making pancakes. Vertices are ingredients or steps, and edges represents the partial order among the steps.



The file pancakes.txt has two lines, where the first describes the edges and the second containing a start vertex (needed for some algorithms).

```
milk>mix:0 egg>mix:0 oil>mix:0 flour>mix:0 mix>syrup:0 mix>pour:0 heat>pour:0 syrup>eat:0 pour>turn:0 turn>eat:0
milk
```

A topological sort algorithm (in the `algorithms.py` module) uses a recursive dfs to determine a proper order to avoid putting the "cart before the horse." For example, we don't want to "pour ½ cup of batter" before we "mix the batter", and we don't want to "mix the batter" until all the ingredients have been added. I've modified the `algorithms.py` module to include some additional print statements in the `topoSort` and `dfs` functions.

a) Run the `view.py` program, load the graph defined in the `pancakes.txt` file (option 2), view the graph (option 3), and then perform a topological sort (option 6).

b) Study the output and the code. Draw the recursion tree(s) for all calls to the `dfs` function performed during the `topoSort` function.

After you have answered the above questions, raise your hand and explain your answers.

EXTRA CREDIT: Part C: Define a function `bfs`, which performs a breadth-first search traversal of a graph, given a starting vertex. This function should return a list of vertex labels in the order that they are visited. Test the function thoroughly with the case study program.