BREADTH FIRST SEARCH

Section Overview

- Overview of Breadth First Search
- Example built off of Word Ladder Problem

Breadth First Search

- How can we find the shortest solution to the word ladder problem?
- The graph algorithm we are going to use is called the “breadth first search” algorithm.
- **Breadth first search** (**BFS**) is one of the easiest algorithms for searching a graph.
- It also serves as a prototype for several other important graph algorithms that we will study later.

Breadth First Search

- Extra Resource:
  - MIT Algorithms and Data Structure Course!
  - https://www.youtube.com/watch?v=s-CYnVz-uh4
- You'll be surprised how well you can follow along!

Breadth First Search

- Given a graph **G** and a starting vertex **s**, a breadth first search proceeds by exploring edges in the graph to find all the vertices in **G** for which there is a path from **s**.
- The remarkable thing about a breadth first search is that it finds *all* the vertices that are a distance **k** from **s** before it finds *any* vertices that are a distance **k+1**.

Breadth First Search

- One good way to visualize what the breadth first search algorithm does is to imagine that it is building a tree, one level of the tree at a time.
- A breadth first search adds all children of the starting vertex before it begins to discover any of the grandchildren.

Breadth First Search

- To keep track of its progress, BFS colors each of the vertices white, gray, or black.
- All the vertices are initialized to white when they are constructed.
- A white vertex is an undiscovered vertex.

Breadth First Search

- When a vertex is initially discovered it is colored gray, and when BFS has completely explored a vertex it is colored black.
- This means that once a vertex is colored black, it has no white vertices adjacent to it.
- A gray node, on the other hand, may have some white vertices adjacent to it, indicating that there are still additional vertices to explore.

Breadth First Search

- BFS begins at the starting vertex s and colors start gray to show that it is currently being explored.
- Two other values, the distance and the predecessor, are initialized to 0 and None respectively for the starting vertex.
- Finally, start is placed on a Queue.
- The next step is to begin to systematically explore vertices at the front of the queue.

Breadth First Search

- We explore each new node at the front of the queue by iterating over its adjacency list. As each node on the adjacency list is examined its color is checked.
- If it is white, the vertex is unexplored, and four things happen:

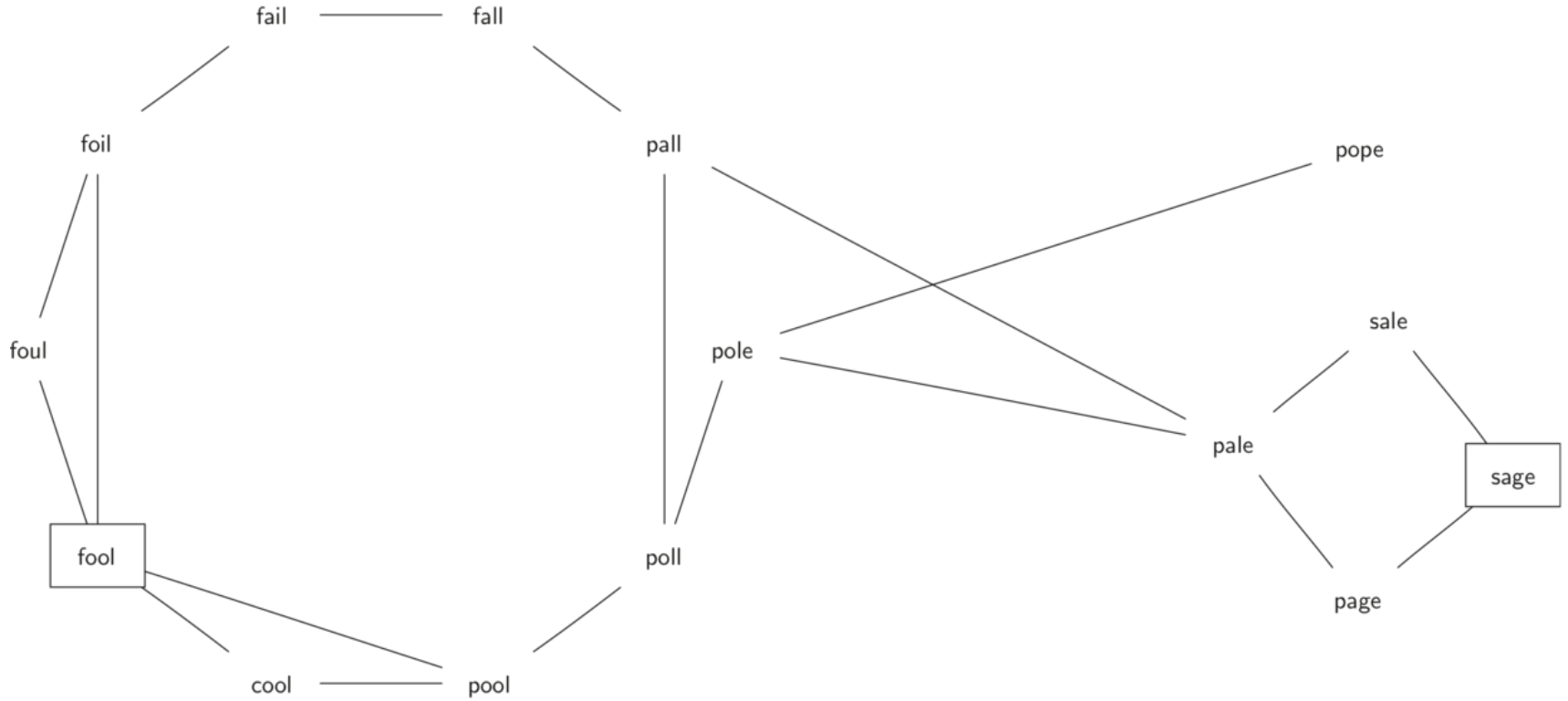
Breadth First Search

- If it is white, the vertex is unexplored, and four things happen:
  - The new, unexplored vertex nbr, is colored gray.
  - The predecessor of nbr is set to the current node currentVert
  - The distance to nbr is set to the distance to currentVert + 1
  - nbr is added to the end of a queue. Adding nbr to the end of the queue effectively schedules this node for further exploration, but not until all the other vertices on the adjacency list of currentVert have been explored.

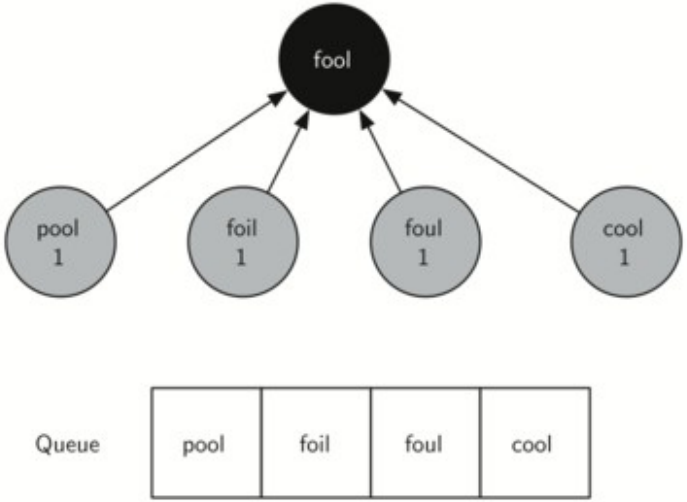
Breadth First Search

```
def bfs(g,start):
  start.setDistance(0)
  start.setPred(None)
  vertQueue = Queue()
  vertQueue.enqueue(start)
  while (vertQueue.size() > 0):
    currentVert = vertQueue.dequeue()
    for nbr in currentVert.getConnections():
      if (nbr.getColor() == 'white'):
        nbr.setColor('gray')
        nbr.setDistance(currentVert.getDistance() + 1)
        nbr.setPred(currentVert)
        vertQueue.enqueue(nbr)
    currentVert.setColor('black')
```

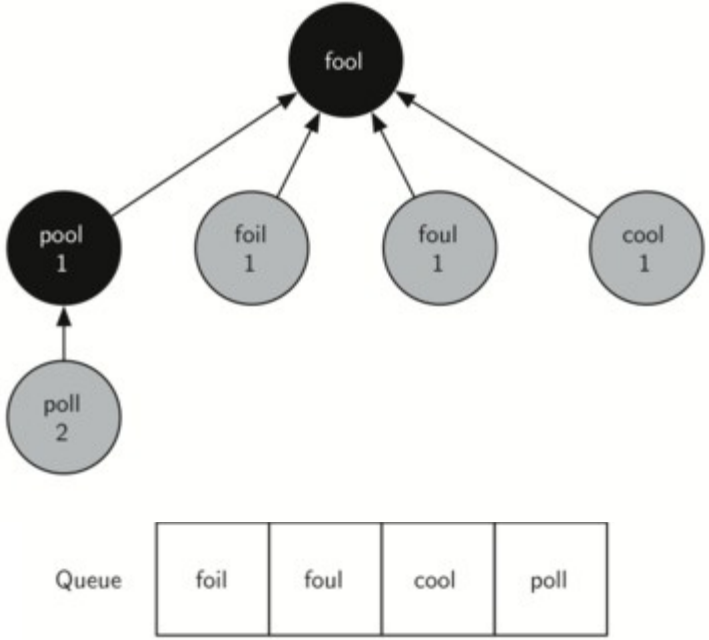
Breadth First Search



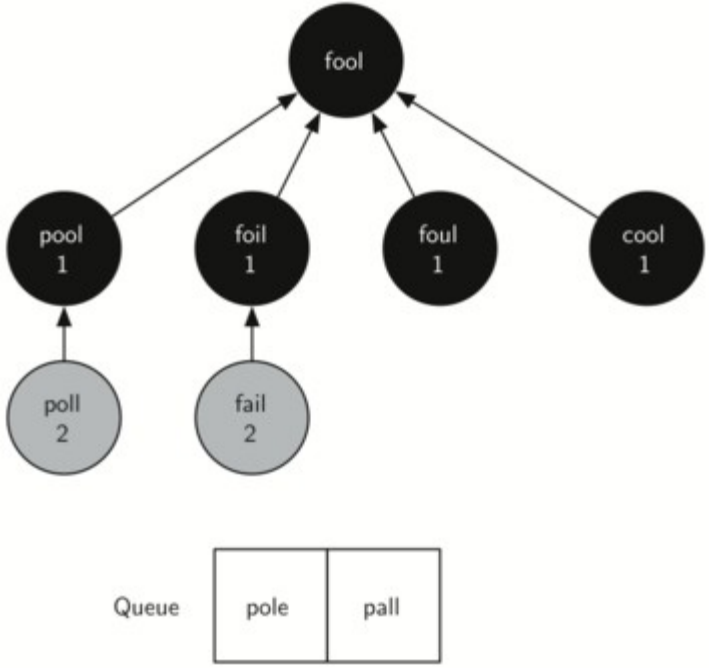
Breadth First Search



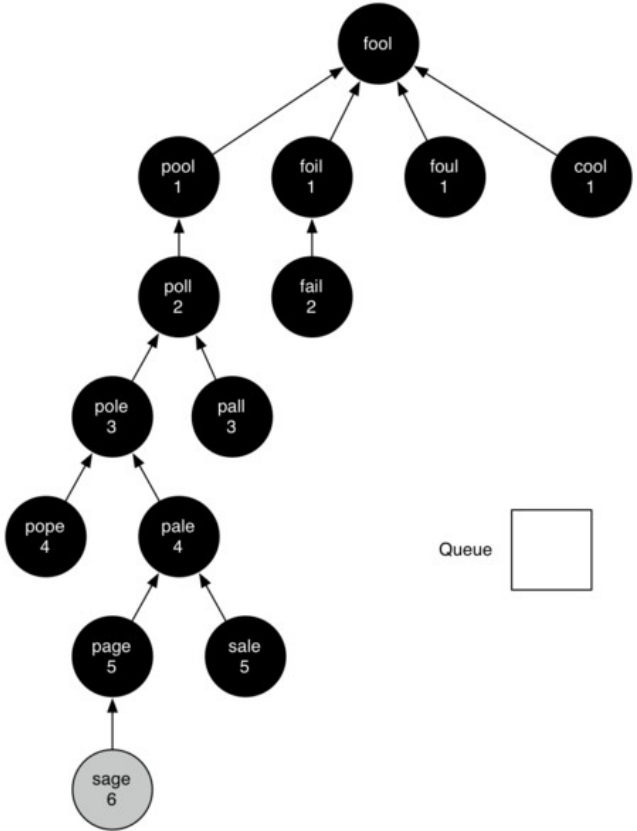
Breadth First Search



Breadth First Search



Breadth First Search



Breadth First Search

- The amazing thing about the breadth first search solution is that we have not only solved the FOOL–SAGE problem we started out with, but we have solved many other problems along the way.
- We can start at any vertex in the breadth first search tree and follow the predecessor arrows back to the root to find the shortest word ladder from any word back to fool.

Breadth First Search

- Up next… the Knight’s Path Problem as an introduction to Depth First Search