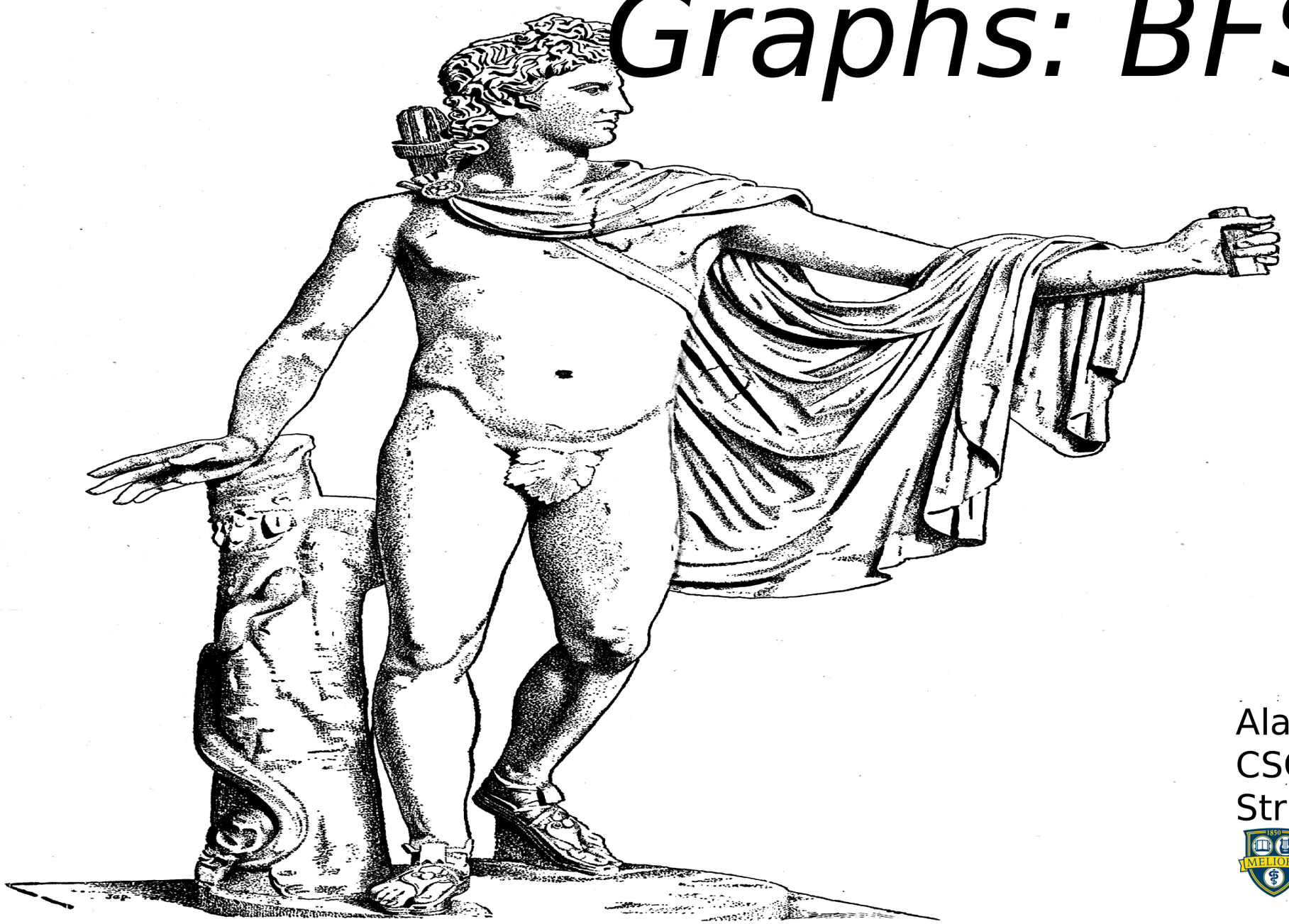
The Art of Data Structures

Graphs: BFS



Alan Beadle
CSC 162: The Art of Data Structures
UNIVERSITY of ROCHESTER

Agenda

- To learn what a breadth first search of graph is and how it is used

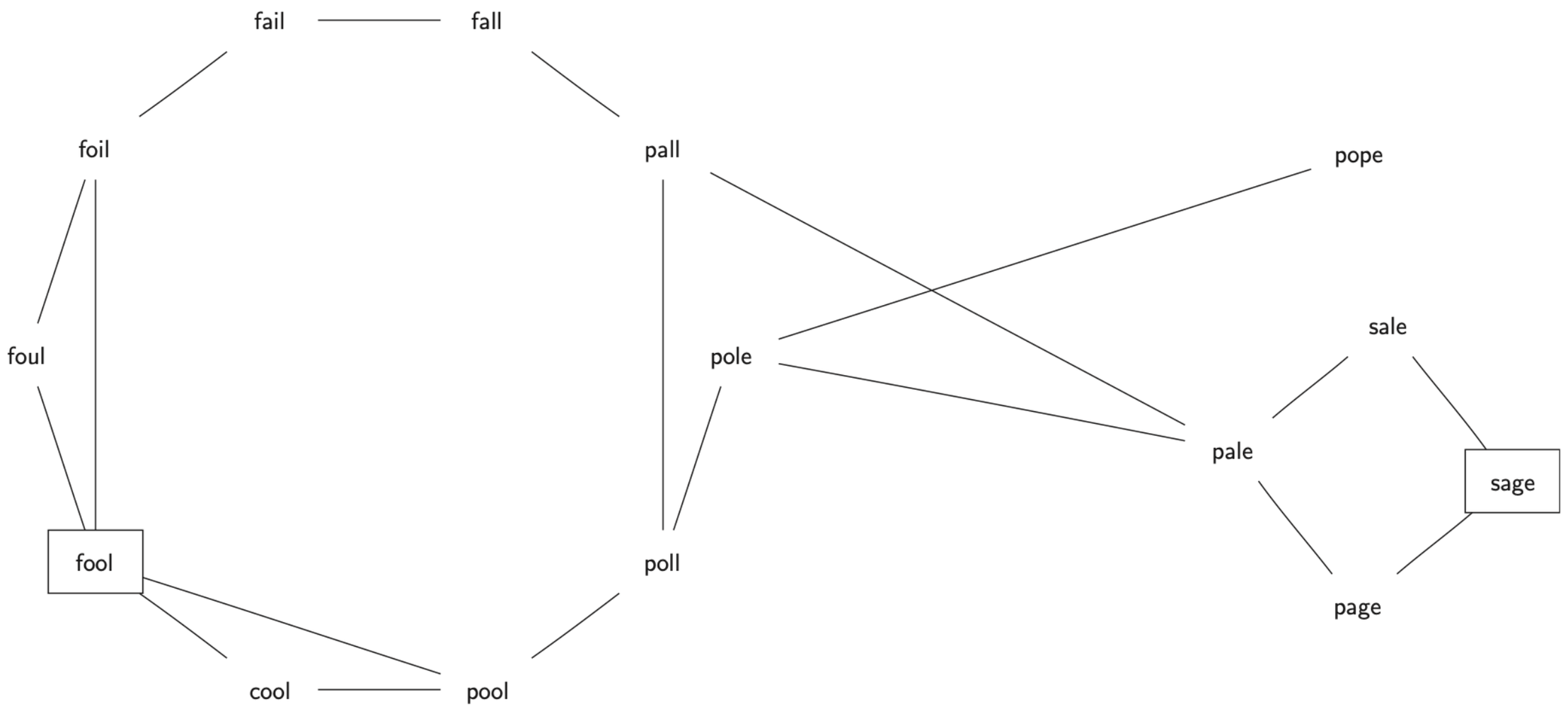
Breadth First Search

Breadth First Search

- Represent the relationships between the words as a graph
- Use the graph algorithm known as breadth first search to find an efficient path from the starting word to the ending word

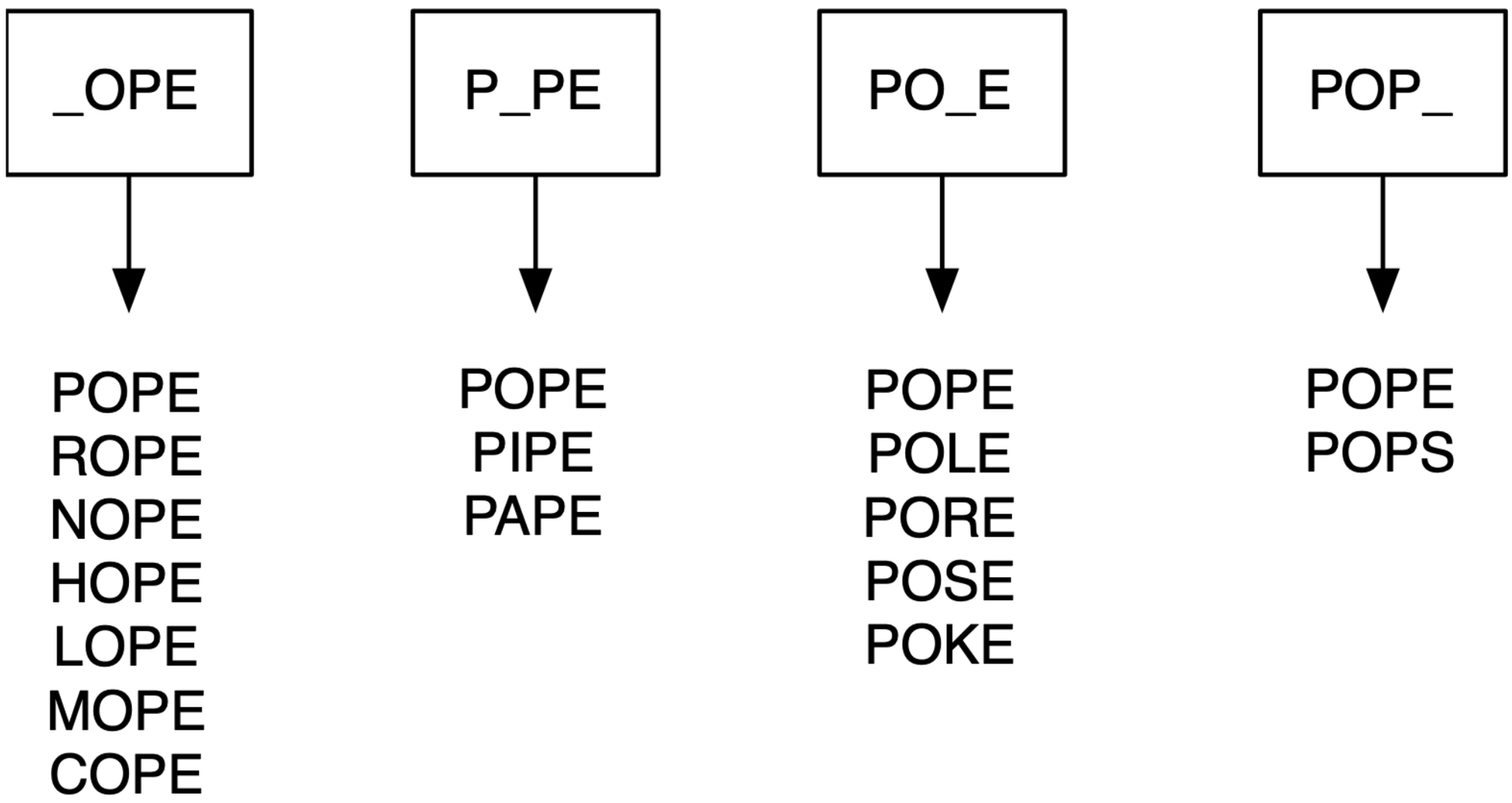
Breadth First Search

A Small Word Ladder Graph



Breadth First Search

A Small Word Ladder Graph



Breadth First Search

Building a Graph of Words for the Word Ladder Problem

```
from pythonds.graphs import Graph, Vertex
from pythonds.basic import Queue

def buildGraph(wordFile):
    d = {}
    g = Graph()
    wfile = open(wordFile,'r')
    # create buckets of words that differ by one letter
    for line in wfile:
        word = line[:-1]
        for i in range(len(word)):
            bucket = word[:i] + '_' + word[i+1:]
            if bucket in d:
                d[bucket].append(word)
            else:
                d[bucket] = [word]
```

Breadth First Search

Building a Graph of Words for the Word Ladder Problem (cont.)

```
    # add vertices and edges for words in the same bucket
    for bucket in d.keys():
        for word1 in d[bucket]:
            for word2 in d[bucket]:
                if word1 != word2:
                    g.addEdge(word1,word2)
    return g
```

Breadth First Search

Building a Graph of Words for the Word Ladder Problem

```
def traverse(y):
    x = y
    while (x.getPred()):
        print(x.getId())
        x = x.getPred()
    print(x.getId())
```

Breadth First Search

Implementation

```
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

Building a Graph of Words for the Word Ladder Problem

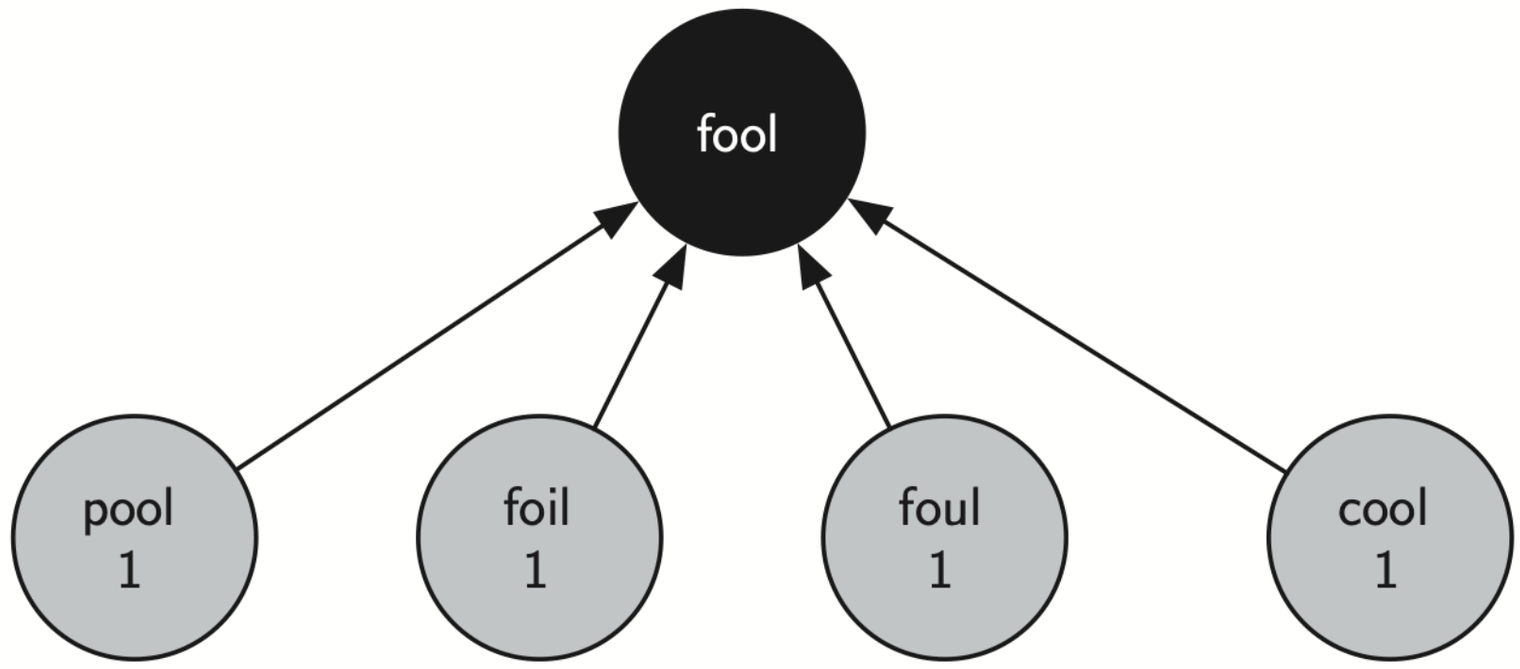
```
wordgraph = buildGraph("fourletterwords.txt")

bfs(wordgraph, wordgraph.getVertex('FOOL'))

traverse(wordgraph.getVertex('SAGE'))
```

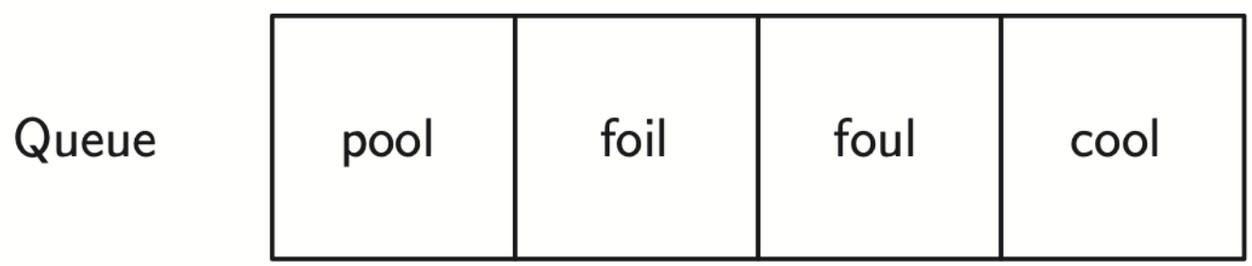
Breadth First Search

- The new, unexplored vertex `v`, is colored gray.
- The predecessor of `v` is set to the current node `w`
- The distance to `v` is set to the distance to `w + 1`
- `v` is added to the end of a queue
- Adding `v` 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 w have been explored

Breadth First Search

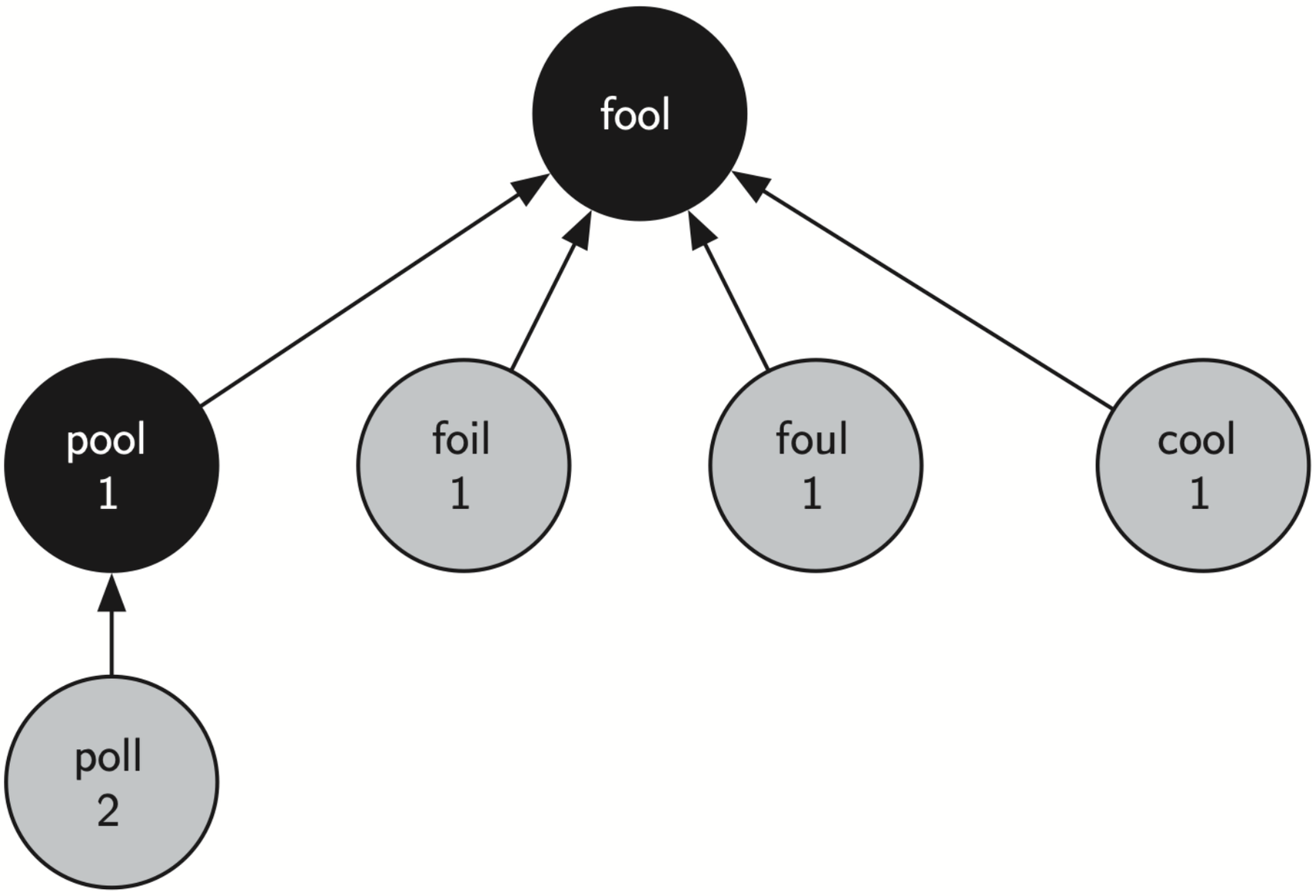
First Step in the Breadth First Search





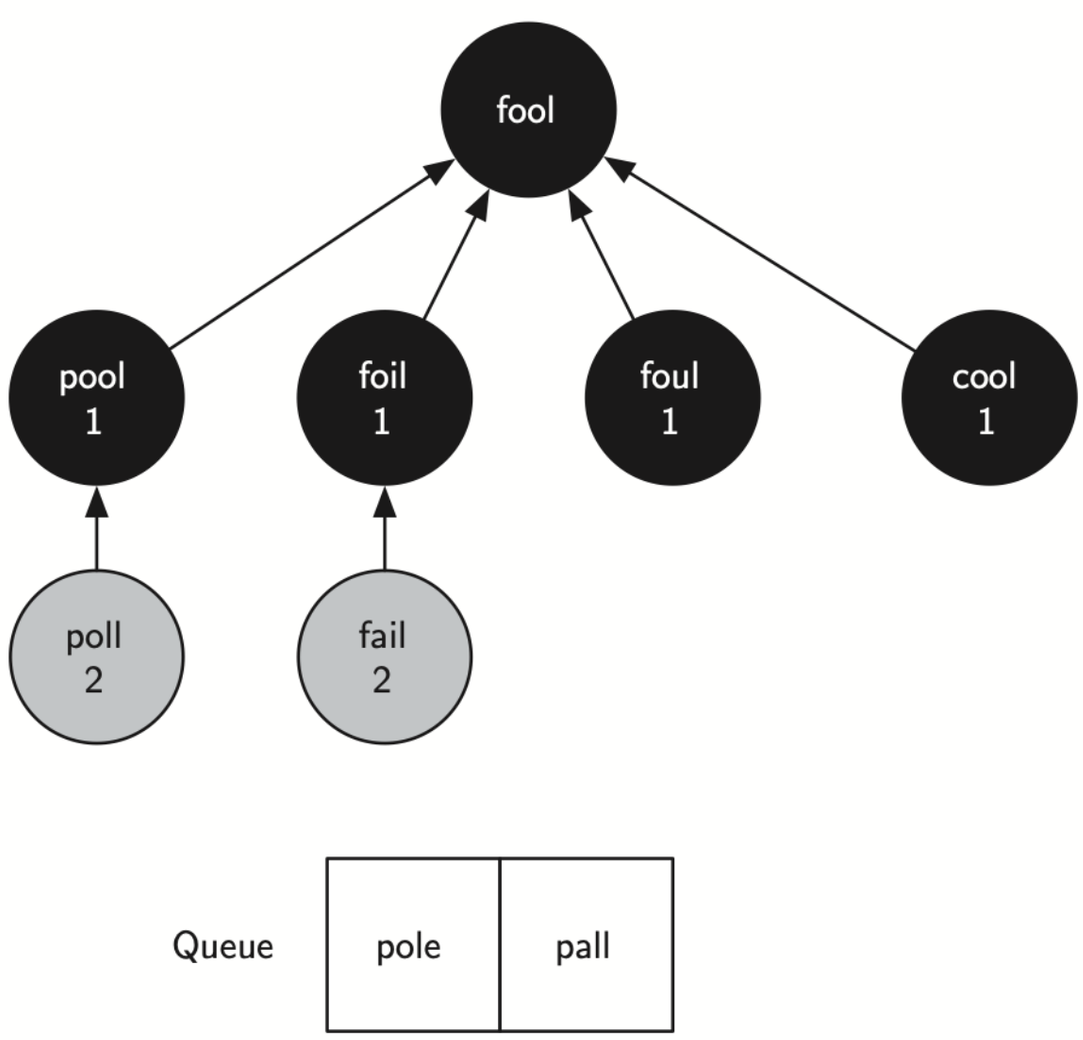
Breadth First Search

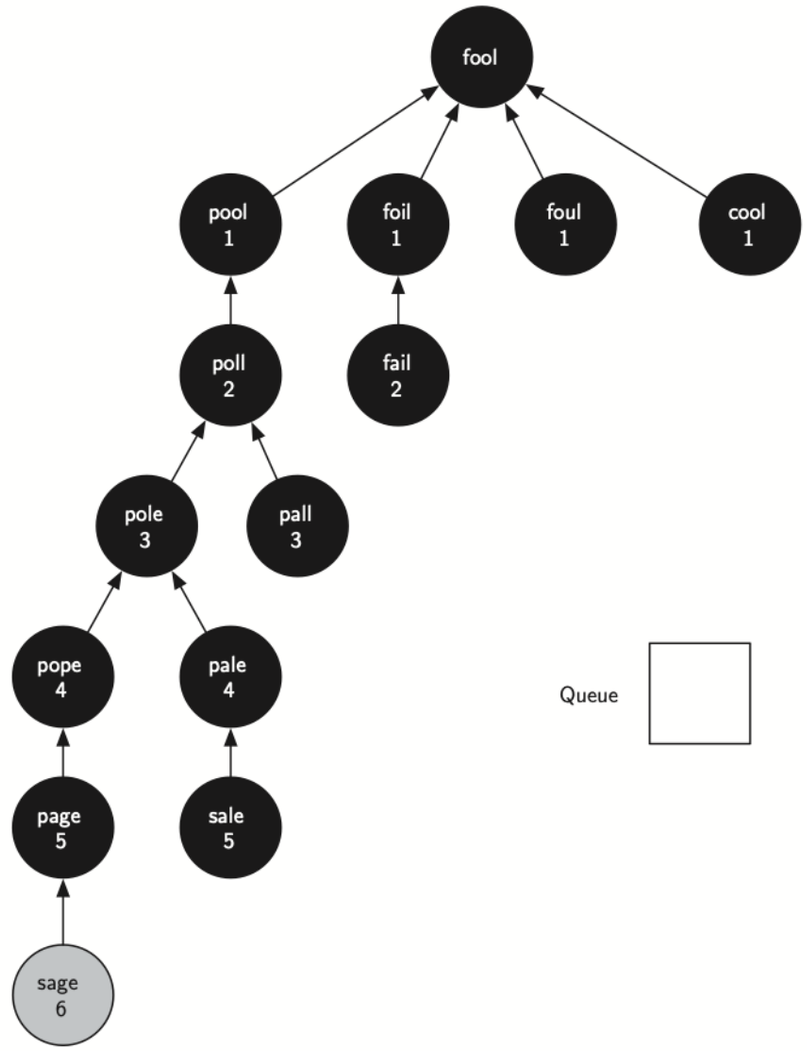
Second Step in the Breadth First Search



Breadth First Search

Constructing the Breadth First Search Tree





(a) Breadth First Search Tree After Completing One Level

(b) Final Breadth First Search Tree

Breadth First Search

Constructing the Breadth First Search Tree

- You can represent your problem in terms of an unweighted graph
- The solution to your problem is to find the shortest path between two nodes in the graph

Questions?