Lab 7 Graph Notes

In AdjacencyMatrix.py, if the class name says AdjacencyList, change it to AdjacencyMatrix. Also, self.adj is suppose to be self.a which is initialized to a new_boolean_matrix. The code for new_boolean_matrix and new_boolean_array are below. new_boolean_array is called to initialize the *seen* array in BFS and DFS.

```
def new_boolean_matrix(self,n):
    return numpy.zeros([n, n], numpy.bool_)

def new_boolean_array(self,n):
    return numpy.zeros(n, numpy.bool_)
```

BFS+DFS

After a vertex is either removed from the queue or popped from the stack, print it out.

Part 2:

(a)

1. In Bookstore.py, create an instance of ChainedHashTable called *indexKeys*
2. In the loadCatalog function, create a variable 'i' to keep track of the line number as you loop through f. The line number is the book's index. Remember to increase 'i' when necessary.
3. After appending a book to bookCatalog, add the key and 'i' to indexKeys: indexKeys(key,i)
4. After loading all of the books, on the next line outside of the *with open(…)* block, create an instance of an AdjacencyList called *similarGraph* that takes the size of the bookCatalog as the parameter.
5. Next, open the file again using *with open(…) as f:*
6. In a for loop populate the similarGraph as follows:

```
i = 0
for line in f:
    (key, title, group, rank, similar) = line.split("^")
    l = similar.split()
    for k in range(1, len(l)):
        j = indexKeys.find(l[k])
        if j is not None:
            similarGraph.add_edge(i,j)
    i += 1
```

7. The reason we have to use two separate *with open(…)* blocks is because only after adding all the books in the file to bookCatalog will we know the number of vertices to initialize the similarGraph with.

(b) In AdajencyList.py, create another bfs method called bfs2(r,k). The code for bfs2 is the same as bfs, except you need to add a parameter 'd' to keep track of the distance at each step in the while loop. Each step of the while loop increases the distance ‘d’ by 1. The while loop stops when d > k. Create a menu option that will call this method on the Bookstore’s similarGraph.

(c) In AdjacencyList.py, create another dfs method called dfs2(r1,r2). r1 and r2 are the source vertex and destination vertex, respectively. The code for dfs2 is the same as dfs, except you need to add a parameter ‘d’ to keep track of the distance of each subsequent vertex encountered.

- Initialize d to equal a numpy.zeros array of size n and initialize d[r1] = 0.
- In the for loop, when ‘j’ a neighbor of ‘i’ is discovered for the first time, set d[j] equal to one more than the distance of its predecessor, ‘i’.
- Then check if j equals r2, the destination. If equal, return d[j].
- After the while loop exits, return -1, which means r1 and r2 are not connected.
- Create a menu option that will call this method on the Bookstore’s similarGraph.