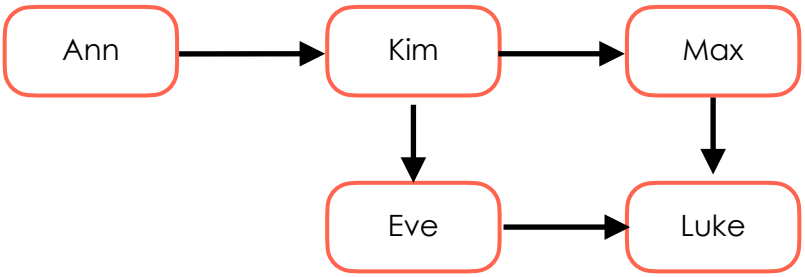
Advanced: PSET 3

Betweenness Centrality

In this problem, you will be implementing a network analysis algorithm: Betweenness Centrality, which measures how much control a node has over a graph. In this problem set, we are analysing a social network.

What is betweenness centrality?

Imagine the following social network with **people** as vertices, and **follows** as **directed edges**. An edge from vertex A to vertex B means that vertex A **follows** vertex B. Consider the following graph:



Using a BFS algorithm, for each pair of people, we can calculate the shortest paths like so:

- **Ann to Eve:** [Kim]
- **Ann to Max**: [Kim]
- **Ann to Luke**: [Kim, Eve]
- **Kim to Luke**: [Eve]

Based on the above, we can see that **Kim** appears in 3 shortest paths, while **Eve** appears in 2 shortest paths. As such, **Kim** is the node that has highest betweenness centrality. In this problem, you will be creating a function to calculate betweenness centrality of each vertex in a network.

Note!:

- For shortest paths in this problem, we don't include the start and end of the shortest path. This is also why we don't consider single edges when calculating shortest paths (e.g. Ann to Kim)
- From node A to node B, if there are two shortest paths, we always take the path where the name is less in alphabetical order. For instance, from Ann to Luke, we have:
  - [Kim, Max] and [Kim, Eve]
  - We choose [Kim, Eve] because Eve is less then Max if compared (E < M)

A. Starting Code

network5.csv

```
Ann,Kim
Kim,Max
Kim,Eve
Max,Luke
Eve,Luke
```

- We have provided several csv files (inside the networks folder) which contain the edges (follows) in a network
- These will be used to test your code
- Each row represents a directed edge from column 0 to column 1
- For instance, the first row means that Ann follows Kim and there is an edge from Ann to Kim
- Remember that edges are directional!

`main`

```
def main():
    if len(argv) < 2:
        print("Require network file to load edges")
        return

    adjList = {}

    with open(argv[1], "r") as csv_file:
        myReader = csv.reader(csv_file)
        for row in myReader:
            u, v = row[0], row[1]
            if u not in adjList:
                adjList[u] = []
            if v not in adjList:
                adjList[v] = []
            adjList[u].append(v)

    print(greatestBetweennessCentrality(adjList))
```

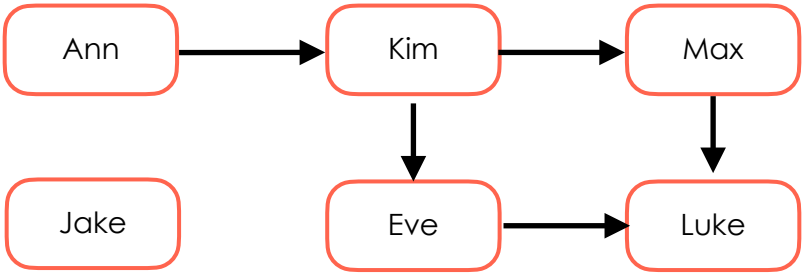
In main, we help you to read from the csv file, through a command line argument, and create the adjacency list for the edges. This is stored in a dictionary where each key is a vertex in the network and value is a list of adjacent edges to that vertex. To test, for instance, **network5.csv**, you may run **`python centrality.py network5.csv`**

B. Your Task

Implement the following functions:

1. `shortestPaths`

- This function takes in a user (vertex) and the adjacency list, and returns a dictionary for each shortest path that exists from that vertex to any other in the network. For instance, consider the following network:



`shortestPath("Ann", adjList)` should return:

```
{
      "Eve": [ "Kim" ],
      "Max": [ "Kim" ],
      "Luke": [ "Kim", "Eve" ]
}
```

- The shortest path from "Ann" (start) to "Eve" is ["Kim"], meaning to get to "Eve" in the shortest way possible, "Ann" must go through "Eve"
- This dictionary **should not** include keys who the start has a direct edge to (e.g. "Kim") and keys who the start has no path to (e.g. "Jake").
- When there are multiple shortest paths, always choose the one that has lower alphabetical order

2. `betweennessCentrality`

- This function takes in the adjacency list and returns a dictionary where each vertex is a key and it's betweenness centrality is the value
- It should make use of the `shortestPaths` function
- For instance:

`betweennessCentrality(adjList)` for the above network should return:
`{'Ann': 0, 'Kim': 3, 'Eve': 2, 'Max': 0, 'Luke': 0, 'Jake': 0}`

C. Sample Output

```
python centrality.py network5.csv
{'Ann': 0, 'Kim': 3, 'Eve': 2, 'Max': 0, 'Luke': 0}

python centrality.py network6.csv
{'Elle': 0, 'Draco': 1, 'Ron': 3, 'Jacob': 5, 'Tom': 0, 'Harry': 0}

python centrality.py network7.csv
{'Kanye': 5, 'Khalid': 2, 'SZA': 4, 'Justin': 0, 'Shawn': 0, 'Beyonce': 0}

python centrality.py network8.csv
{'Jake': 0, 'Alex': 0, 'Charlotte': 0, 'Emma': 4, 'Ava': 5, 'Ben': 0,
'James': 9, 'Joshua': 7}
```

D. Submission

To test your code, run the following command:

```
python utils/test.py
```

To submit your code, run the following command:

```
python utils/submit.py
```

A report.txt should be generated for you to view your results

---