HW3

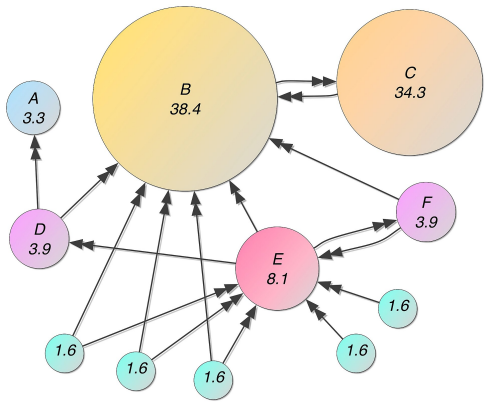
PageRank

What is the most important website on the internet? Who is the "key player" on a sports team? Which countries are the most central players in the world economy? There is no one correct answer to any of these questions, but there is a most profitable one. PageRank (https://en.wikipedia.org/wiki/PageRank) is an algorithm for ranking individual elements of complex systems, invited by Sergey Brin and Larry Page. It was the first and most famous algorithm used by the Google Search engine, and it is fair to say that the internet as we know it today would not exist without PageRank.

In this assignment, we will implement PageRank. There are many good ways to implement this algorithm, but in this assignment we will use our newfound skills with object-oriented programming and iterators.

How it works

For the purposes of this example, let's assume that we are talking about webpages. PageRank works by allowing a "random surfer" to move around webpages by following links. Each time the surfer lands on a page, it then looks for all the links on that page. It then picks one at random and follows it, thereby arriving at the next page, where the process repeats. Eventually, the surfer will visit all the pages one or more times. Pages that the surfer visits more frequently have higher *PageRank scores*. Because the surfer moves between linked pages, PageRank expresses an intuitive idea: **important pages are linked to other important pages.** This diagram (https://en.wikipedia.org/wiki/PageRank#/media/File:PageRanks-Example.jpg) from Wikipedia gives a nice illustration. Note that more important webpages (higher PageRank) tend to be connected to other important webpages.



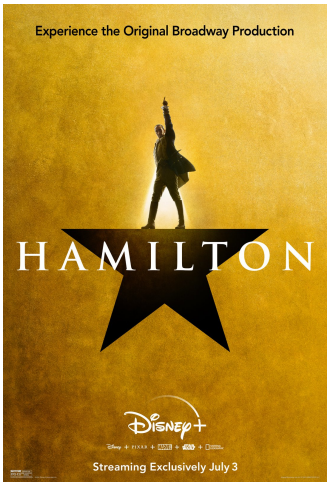
A schematic for PageRank.

Data

You can complete this assignment using data from one of two sources.

Option 1: Hamilton

This data set comes from the hit Broadway musical "Hamilton."



The Hamilton data set

The good folks at The Hamilton Project (http://hamilton.newtfire.org/) analyzed the script for us, obtaining data on **who talks about whom** in each of the show's songs. When character A mentions character B, we'll think of this as a *link* from A to B, suggesting that B might be important.

If you use this data set, listening to the soundtrack while working is strongly recommended.

Option 2: Global Airline Network

The global airline network

Back in the Before Times, lots of people flew on airplanes. This data set includes a "link" from Airport A to Airport B whenever there is a flight from B to A. This data set was collected by the OpenFlights Project (https://openflights.org/data.html).

(A). Define Functions

In this part, all you have to do is hit `shift + enter` on the code block supplied below. This block defines two functions. The first one retrieves the data from the internet and saves it to your local computer, while the second reads in the data, producing a list of tuples. It's not important for you to be familiar with the code in these functions right now -- we'll discuss them soon.

In [178]:
```
import urllib
import csv

def retrieve_data(url):
    """
    Retrieve a file from the specified url and save it in a local file
    called data.csv. The intended values of url are:

    1. https://philchodrow.github.io/PIC16A/homework/HW3-hamilton-data.csv
    2. https://philchodrow.github.io/PIC16A/homework/HW3-flights-data.csv
    """

    # grab the data and parse it
    filedata = urllib.request.urlopen(url)
    to_write = filedata.read()

    # write to file
    with open("data.csv", "wb") as f:
        f.write(to_write)

def read_data(path):
    """
    read downloaded data from a .csv file, and return a list of tuples.
    each tuple represents a link between states.
    """
    with open(path, "r") as f:
        reader = csv.reader(f)
        return [(row[0], row[1]) for row in list(reader)]
```

(B). Grab the Data

The data live at the following URLs:

- **Hamilton**: `https://philchodrow.github.io/PIC16A/homework/HW3-hamilton-data.csv`
- **Airline**: `https://philchodrow.github.io/PIC16A/homework/HW3-flights-data.csv`

In each data set, each row corresponds to a "link" between objects. In Hamilton, the pairs have format `mentioner, mentioned` while in the airline network the rows have format `destination, origin.`

Pick one of these data sets, and set the variable `url` appropriately by uncommenting one of the two lines below. Then, call `retrieve_data()` and `read_data()`. The `path` argument for `read_data()` should be set to `"data.csv"`. Create a variable called `data` to hold the return value of `read_data()`.

Your solution

In [179]:
```
# uncomment the second line if you'd prefer to
# work with the flights data.
url = "https://philchodrow.github.io/PIC16A/homework/HW3-hamilton-data.csv"
#url = "https://philchodrow.github.io/PIC16A/homework/HW3-flights-data.csv"

# Call your functions below
retrieve_data(url)
data = read_data('data.csv')
print(data)
```

```
[('burr', 'hamilton'), ('burr', 'weeks'), ('burr', 'madison'), ('burr', 'jay'), ('burr', 'theod
osiaDaughter'), ('burr', 'betsy'), ('burr', 'theodosiaMother'), ('burr', 'hamilton'), ('burr',
'hamilton'), ('burr', 'hamilton'), ('burr', 'washington'), ('burr', 'hamilton'), ('burr', 'mart
haWashington'), ('burr', 'schuylerSis'), ('burr', 'washington'), ('burr', 'burr'), ('burr', 'ge
neralMontgomery'), ('burr', 'hamilton'), ('burr', 'philipS'), ('burr', 'peggy'), ('burr', 'ange
lica'), ('burr', 'eliza'), ('burr', 'hamilton'), ('burr', 'reynolds'), ('burr', 'hamilton'),
('burr', 'washington'), ('burr', 'hamilton'), ('burr', 'philipS'), ('burr', 'generalMercer'),
('burr', 'madison'), ('burr', 'jefferson'), ('burr', 'washington'), ('burr', 'hamilton'), ('bur
r', 'washington'), ('burr', 'jefferson'), ('burr', 'jefferson'), ('burr', 'madison'), ('burr',
'burr'), ('burr', 'hamilton'), ('burr', 'hamilton'), ('burr', 'jAdams'), ('burr', 'jefferson'),
('burr', 'hamilton'), ('burr', 'jefferson'), ('burr', 'burr'), ('burr', 'ness'), ('burr', 'hami
lton'), ('burr', 'pendleton'), ('burr', 'angelica'), ('burr', 'eliza'), ('hamilton', 'burr'),
('hamilton', 'angelica'), ('hamilton', 'philipH'), ('hamilton', 'lafayette'), ('hamilton', 'eli
za'), ('hamilton', 'laurens'), ('hamilton', 'mulligan'), ('hamilton', 'washington'), ('hamilto
n', 'eliza'), ('hamilton', 'lee'), ('hamilton', 'laurens'), ('hamilton', 'conway'), ('hamilto
n', 'hamilton'), ('hamilton', 'washington'), ('hamilton', 'lee'), ('hamilton', 'laurens'), ('ha
milton', 'burr'), ('hamilton', 'washington'), ('hamilton', 'hamilton'), ('hamilton', 'burr'),
('hamilton', 'lee'), ('hamilton', 'burr'), ('hamilton', 'eliza'), ('hamilton', 'peggy'), ('hami
lton', 'angelica'), ('hamilton', 'hamilton'), ('hamilton', 'laurens'), ('hamilton', 'mulliga
n'), ('hamilton', 'lafayette'), ('hamilton', 'burr'), ('hamilton', 'kingGeorge'), ('hamilton',
'burr'), ('hamilton', 'lafayette'), ('hamilton', 'laurens'), ('hamilton', 'burr'), ('hamilton',
'hamilton'), ('hamilton', 'reynolds'), ('hamilton', 'eliza'), ('hamilton', 'angelica'), ('hamil
ton', 'philipH'), ('hamilton', 'eliza'), ('hamilton', 'eacker'), ('hamilton', 'philipH'), ('ham
ilton', 'eliza'), ('hamilton', 'reynolds'), ('hamilton', 'jefferson'), ('hamilton', 'madison'),
('hamilton', 'burr'), ('hamilton', 'reynolds'), ('hamilton', 'washington'), ('hamilton', 'jeffe
rson'), ('hamilton', 'washington'), ('hamilton', 'kingLouis'), ('hamilton', 'lafayette'), ('ham
ilton', 'burr'), ('hamilton', 'burr'), ('hamilton', 'angelica'), ('hamilton', 'maria'), ('hamil
ton', 'reynolds'), ('hamilton', 'angelica'), ('hamilton', 'madison'), ('hamilton', 'jefferso
n'), ('hamilton', 'eliza'), ('hamilton', 'schuylerSis'), ('hamilton', 'jAdams'), ('hamilton',
'jefferson'), ('hamilton', 'washington'), ('hamilton', 'madison'), ('hamilton', 'jefferson'),
('hamilton', 'hamilton'), ('hamilton', 'philipH'), ('hamilton', 'eliza'), ('hamilton', 'burr'),
('hamilton', 'jefferson'), ('hamilton', 'jAdams'), ('hamilton', 'burr'), ('hamilton', 'hamilto
n'), ('hamilton', 'burr'), ('hamilton', 'laurens'), ('hamilton', 'washington'), ('hamilton', 'e
liza'), ('ensemble', 'washington'), ('ensemble', 'kingGeorge'), ('ensemble', 'jefferson'), ('en
semble', 'burr'), ('ensemble', 'hamilton'), ('ensemble', 'jAdams'), ('ensemble', 'jefferson'),
('company', 'hamilton'), ('company', 'mulligan'), ('company', 'lafayette'), ('company', 'hamilt
on'), ('company', 'washington'), ('company', 'hamilton'), ('company', 'admiralHowe'), ('compan
y', 'washington'), ('company', 'kingGeorge'), ('company', 'schuylerSis'), ('company', 'angelic
a'), ('company', 'reynolds'), ('company', 'washington'), ('company', 'jefferson'), ('company',
'hamilton'), ('company', 'burr'), ('company', 'jefferson'), ('company', 'eliza'), ('company',
'jAdams'), ('company', 'burr'), ('men', 'hamilton'), ('men', 'angelica'), ('men', 'jAdams'),
('men', 'jefferson'), ('men', 'burr'), ('women', 'hamilton'), ('women', 'angelica'), ('women',
'washington'), ('women', 'eliza'), ('women', 'burr'), ('women', 'jefferson'), ('angelica', 'ham
ilton'), ('angelica', 'hamilton'), ('angelica', 'angelica'), ('angelica', 'franklin'), ('angeli
ca', 'schuylerSis'), ('angelica', 'eliza'), ('angelica', 'angelica'), ('angelica', 'eliza'),
('angelica', 'burr'), ('angelica', 'paine'), ('angelica', 'jefferson'), ('angelica', 'schuylerS
is'), ('angelica', 'hamilton'), ('angelica', 'jefferson'), ('angelica', 'angelica'), ('angelic
a', 'eliza'), ('angelica', 'angelica'), ('angelica', 'hamilton'), ('angelica', 'eliza'), ('ange
lica', 'angelica'), ('angelica', 'eliza'), ('eliza', 'hamilton'), ('eliza', 'washington'), ('el
iza', 'hamilton'), ('eliza', 'eliza'), ('eliza', 'eliza'), ('eliza', 'eliza'), ('eliza', 'angel
ica'), ('eliza', 'schuylerSis'), ('eliza', 'angelica'), ('eliza', 'eliza'), ('eliza', 'hamilto
n'), ('eliza', 'hamilton'), ('eliza', 'philipH'), ('eliza', 'angelica'), ('eliza', 'jAdams'),
('eliza', 'angelica'), ('eliza', 'washington'), ('eliza', 'hamilton'), ('eliza', 'hamilton'),
('washington', 'rochambeau'), ('washington', 'hamilton'), ('washington', 'burr'), ('washingto
n', 'lee'), ('washington', 'hamilton'), ('washington', 'hamilton'), ('washington', 'lee'), ('wa
shington', 'lafayette'), ('washington', 'hamilton'), ('washington', 'burr'), ('washington', 'gr
een'), ('washington', 'knox'), ('washington', 'jefferson'), ('washington', 'jefferson'), ('wash
ington', 'hamilton'), ('washington', 'burr'), ('washington', 'hamilton'), ('washington', 'jeffe
rson'), ('washington', 'madison'), ('washington', 'jefferson'), ('mulligan', 'mulligan'), ('mul
ligan', 'hamilton'), ('mulligan', 'burr'), ('mulligan', 'mulligan'), ('mulligan', 'burr'), ('la
fayette', 'hamilton'), ('lafayette', 'hamilton'), ('lafayette', 'burr'), ('lafayette', 'lafayet
```

```
te'), ('laurens', 'hamilton'), ('laurens', 'lee'), ('laurens', 'burr'), ('laurens', 'angelic
a'), ('laurens', 'laurens'), ('laurens', 'sAdams'), ('laurens', 'burr'), ('kingGeorge', 'washin
gton'), ('kingGeorge', 'jAdams'), ('jefferson', 'hamilton'), ('jefferson', 'reynolds'), ('jeffe
rson', 'eliza'), ('jefferson', 'hamilton'), ('jefferson', 'washington'), ('jefferson', 'hamilto
n'), ('jefferson', 'washington'), ('jefferson', 'lafayette'), ('jefferson', 'hamilton'), ('jeff
erson', 'washington'), ('jefferson', 'madison'), ('jefferson', 'burr'), ('jefferson', 'hamilto
n'), ('jefferson', 'lafayette'), ('jefferson', 'washington'), ('jefferson', 'sally'), ('jeffers
on', 'madison'), ('jefferson', 'jAdams'), ('jefferson', 'hamilton'), ('jefferson', 'burr'), ('j
efferson', 'washington'), ('jefferson', 'hamilton'), ('madison', 'hamilton'), ('madison', 'wash
ington'), ('madison', 'hamilton'), ('madison', 'hamilton'), ('madison', 'burr'), ('madison', 'j
efferson'), ('madison', 'hamilton'), ('madison', 'burr'), ('madison', 'jefferson'), ('madison',
'hamilton'), ('madison', 'jAdams'), ('philipH', 'eacker'), ('philipH', 'philipH'), ('philipH',
'philipS'), ('philipH', 'burr'), ('philipH', 'philipH'), ('lee', 'lee'), ('lee', 'washington'),
('peggy', 'peggy'), ('peggy', 'schuylerSis'), ('seabury', 'seabury'), ('seabury', 'kingGeorg
e'), ('reynolds', 'reynolds'), ('doctor', 'hamilton')]
```

(C). Examine the structure of the data

This would also be a good time to inspect the data to make sure you understand how it is structured. Write a function `describe(n)` that describes the meaning of the `n` th row of the data set you chose. In the Hamilton data set, your function should do the following:

```
describe(5)

# output
"Element 5 of the Hamilton data set is ('burr', 'betsy'). This means that Burr mentions B
etsy in a song."
```

In context of the airline flights data, your function should instead do this:

```
describe(5)

# output
"Element 5 of the flights data set is ('SIN', 'BKK'). This means that there is a flight f
rom BKK to SIN."
```

Please attend to capitalization and formatting. While the standard string concatenation operator `+` is completely fine for this task, the fancy `str.format()` function may make your code somewhat simpler. [This page (https://realpython.com/python-formatted-output/)](https://realpython.com/python-formatted-output/) has some useful examples in case you'd like to try this.

Your Solution

In [232]:
```
'''describes the meaning of the nth row of the data set '''
def describe(n):
    myString = 'Element {0} of the Hamilton data set is {1}. ' \
    'This means that {2} mentions {3} in a song.'.format(n, data[n], data[n][0].capitalize(), dat
    return myString

describe(5)
```

Out[232]:
```
"Element 5 of the Hamilton data set is ('burr', 'betsy'). This means that Burr mentions Betsy i
n a song."
```

In [230]:
```
describe(0)
```

Out[230]:
```
"Element 0 of the Hamilton data set is ('burr', 'hamilton'). This means that Burr mentions Hami
lton in a song."
```

(D). Data to Dictionary

Write a function called `data_to_dictionary` that converts the data into a dictionary such that:

1. There is a single key for each character (in Hamilton) or airport (in flights).
2. The value corresponding to each key is a list of the characters/airports to which that key links. The list should contain repeats if there are multiple links.

Here's an example of the desired behavior on a fake data set.

```
data = [("a", "b"),
        ("a", "b"),
        ("a", "c"),
        ("b", "c"),
        ("b", "a")]

data_to_dictionary(data)

# output
{"a" : ["b", "b", "c"], "b" : ["a", "c"]}
```

Your Solution

In [233]:
```
'''converts the data set into a dictionary'''
def data_to_dictionary(myData):
    myDict = dict() #initalize empty dict

    #go through data to make dict
    for n in range(len(myData)):
        if myData[n][0] not in myDict.keys(): #for keys not in new dict
            myDict[myData[n][0]] = [myData[n][1]]
        elif myData[n][0] in myDict.keys():  #for keys already in new dict
            myDict[myData[n][0]].append(myData[n][1])
    return myDict

print(data_to_dictionary(data))
```

```
{'burr': ['hamilton', 'weeks', 'madison', 'jay', 'theodosiaDaughter', 'betsy', 'theodosiaMothe
r', 'hamilton', 'hamilton', 'hamilton', 'washington', 'hamilton', 'marthaWashington', 'schuyler
Sis', 'washington', 'burr', 'generalMontgomery', 'hamilton', 'philipS', 'peggy', 'angelica', 'e
liza', 'hamilton', 'reynolds', 'hamilton', 'washington', 'hamilton', 'philipS', 'generalMerce
r', 'madison', 'jefferson', 'washington', 'hamilton', 'washington', 'jefferson', 'jefferson',
'madison', 'burr', 'hamilton', 'hamilton', 'jAdams', 'jefferson', 'hamilton', 'jefferson', 'bur
r', 'ness', 'hamilton', 'pendleton', 'angelica', 'eliza'], 'hamilton': ['burr', 'angelica', 'ph
ilipH', 'lafayette', 'eliza', 'laurens', 'mulligan', 'washington', 'eliza', 'lee', 'laurens',
'conway', 'hamilton', 'washington', 'lee', 'laurens', 'burr', 'washington', 'hamilton', 'burr',
'lee', 'burr', 'eliza', 'peggy', 'angelica', 'hamilton', 'laurens', 'mulligan', 'lafayette', 'b
urr', 'kingGeorge', 'burr', 'lafayette', 'laurens', 'burr', 'hamilton', 'reynolds', 'eliza', 'a
ngelica', 'philipH', 'eliza', 'eacker', 'philipH', 'eliza', 'reynolds', 'jefferson', 'madison',
'burr', 'reynolds', 'washington', 'jefferson', 'washington', 'kingLouis', 'lafayette', 'burr',
'burr', 'angelica', 'maria', 'reynolds', 'angelica', 'madison', 'jefferson', 'eliza', 'schuyler
Sis', 'jAdams', 'jefferson', 'washington', 'madison', 'jefferson', 'hamilton', 'philipH', 'eliz
a', 'burr', 'jefferson', 'jAdams', 'burr', 'hamilton', 'burr', 'laurens', 'washington', 'eliz
a'], 'ensemble': ['washington', 'kingGeorge', 'jefferson', 'burr', 'hamilton', 'jAdams', 'jeffe
rson'], 'company': ['hamilton', 'mulligan', 'lafayette', 'hamilton', 'washington', 'hamilton',
'admiralHowe', 'washington', 'kingGeorge', 'schuylerSis', 'angelica', 'reynolds', 'washington',
'jefferson', 'hamilton', 'burr', 'jefferson', 'eliza', 'jAdams', 'burr'], 'men': ['hamilton',
'angelica', 'jAdams', 'jefferson', 'burr'], 'women': ['hamilton', 'angelica', 'washington', 'el
iza', 'burr', 'jefferson'], 'angelica': ['hamilton', 'hamilton', 'angelica', 'franklin', 'schuy
lerSis', 'eliza', 'angelica', 'eliza', 'burr', 'paine', 'jefferson', 'schuylerSis', 'hamilton',
'jefferson', 'angelica', 'eliza', 'angelica', 'hamilton', 'eliza', 'angelica', 'eliza'], 'eliz
a': ['hamilton', 'washington', 'hamilton', 'eliza', 'eliza', 'eliza', 'angelica', 'schuylerSi
s', 'angelica', 'eliza', 'hamilton', 'hamilton', 'philipH', 'angelica', 'jAdams', 'angelica',
'washington', 'hamilton', 'hamilton'], 'washington': ['rochambeau', 'hamilton', 'burr', 'lee',
'hamilton', 'hamilton', 'lee', 'lafayette', 'hamilton', 'burr', 'green', 'knox', 'jefferson',
'jefferson', 'hamilton', 'burr', 'hamilton', 'jefferson', 'madison', 'jefferson'], 'mulligan':
['mulligan', 'hamilton', 'burr', 'mulligan', 'burr'], 'lafayette': ['hamilton', 'hamilton', 'bu
rr', 'lafayette'], 'laurens': ['hamilton', 'lee', 'burr', 'angelica', 'laurens', 'sAdams', 'bur
r'], 'kingGeorge': ['washington', 'jAdams'], 'jefferson': ['hamilton', 'reynolds', 'eliza', 'ha
milton', 'washington', 'hamilton', 'washington', 'lafayette', 'hamilton', 'washington', 'madiso
n', 'burr', 'hamilton', 'lafayette', 'washington', 'sally', 'madison', 'jAdams', 'hamilton', 'b
urr', 'washington', 'hamilton'], 'madison': ['hamilton', 'washington', 'hamilton', 'hamilton',
'burr', 'jefferson', 'hamilton', 'burr', 'jefferson', 'hamilton', 'jAdams'], 'philipH': ['eacke
r', 'philipH', 'philipS', 'burr', 'philipH'], 'lee': ['lee', 'washington'], 'peggy': ['peggy',
'schuylerSis'], 'seabury': ['seabury', 'kingGeorge'], 'reynolds': ['reynolds'], 'doctor': ['ham
ilton']}
```

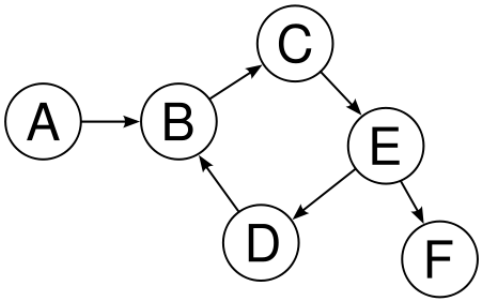
In [182]:
```
#check to see if function works correctly with data1 as well
data1 = [("a", "b"),
         ("a", "b"),
         ("a", "c"),
         ("b", "c"),
         ("b", "a")]

data_to_dictionary(data1)
```

Out[182]:
```
{'a': ['b', 'b', 'c'], 'b': ['c', 'a']}
```

(E). Define a PR_DiGraph class

A **directed graph**, or DiGraph, is just a set of arrows ("edges") between objects ("nodes"). It is a natural way to represent data that represents one-way relationships, such as links from one webpage to another or mentions of one character by another. We already saw a directed graph above when we introduced the idea of PageRank. Here's a paired-down example.



Example of a directed graph.

Implement a `PR_DiGraph` class with a custom `__init__()` method and a `linked_by()` method. The `__init__()` method should accept two arguments: `data` and `iteration_limit`. The `__init__()` method should then construct an instance variable `self.link_dict` which is simply the output of `data_to_dictionary` applied to the argument `data`. `__init__()` should also construct an instance variable `self.iteration_limit`, which simply takes the value of `iteration_limit` supplied to `__init__()`. Don't worry about that one for now.

Then, define a method `self.linked_by(x)` which, when called, returns the value `self.link_dict[x]`.

Finally, add an `__iter__` method, which returns an object of class `PR_Iterator`. We will define this class in the next part.

Example session (using Hamilton):

```
D = PR_DiGraph(data, iteration_limit = 10000)
D.linked_by('peggy')

# output
['peggy', 'schuylerSis']
```

Your Solution

In [183]:
```
'''
class in that uses creates a dictionary with links and has functions in order to iterate through
thorugh the dicitonary
'''
class PR_DiGraph:
    '''create instance variables that 1) creates a dictionary 2) set iteration limit '''
    def __init__(self, data, iteration_limit):
        self.link_dict = data_to_dictionary(data)
        self.iteration_limit = iteration_limit

    '''return a list of things linked to x'''
    def linked_by(self, x):
        return self.link_dict[x]

    '''returns a page rank iterator'''
    def __iter__(self):
        return PR_Iterator(self)
```

In [97]:
```
D = PR_DiGraph(data, iteration_limit = 10000)
D.linked_by('philipH')
```

```
philipH
```

Out[97]:
```
['eacker', 'philipH', 'philipS', 'burr', 'philipH']
```

In [184]:
```
D = PR_DiGraph(data, iteration_limit = 10000)
D.linked_by('peggy')
```

Out[184]:
```
['peggy', 'schuylerSis']
```

(F). Implement PR_Iterator

Define a `PR_Iterator` class with a custom `__next__()` method.

The `__init__` method of this class should create instance variables to store the `PR_DiGraph` object from which it was constructed; a counter `i`, starting at 0, to log the number of steps taken, and a `current_state` variable whose value is one of the keys of the `link_dict` of the `Pr_DiGraph`. You can choose its initial value arbitrarily; in my solution code I chose `self.current_state = "hamilton"`.

We are going to use iteration to implement the PageRank algorithm. This means we are going to imagine a surfer who is following the links in our data set. **Implement the following two methods:**

1. `follow_link()`.
   A. Pick a random new character mentioned by the current character, or new airport served by the current airport. Let's call this `next_state`.
   B. If `next_state != current_state`, set `current_state` to `next_state`.
   C. Otherwise (if `next_state == current_state`), teleport (see below).
   D. You might run into `KeyError`s, in which case you should again teleport (use a `try-except` block).
2. `teleport()`.
   A. Set the current state to a new state (key of the link dict) completely at random.

Hint: use `random.choice` from the `random` module to choose elements of lists.

Finally, **implement** `__next__()`. `__next__()` should do `follow_link()` with 85% probability, and do `teleport()` with 15% probability. You should also define a custom `StopIteration` condition to ensure that only as many steps are taken as the `iteration_limit` supplied to the `PR_DiGraph` initializer.

1. To do something with 85% probability, use the following:

```
if random.random() < 0.85:
    # do the thing
else:
    # do the other thing
```

Example Usage

After you define your class, run the following code and show that it works. Note: your precise sequence may be different from mine.

```
D = PR_DiGraph(data, iteration_limit = 5)
for char in D:
    print(char)
```

```
following link : current state = burr
following link : current state = washington
following link : current state = burr
following link : current state = hamilton
teleporting    : current state = washington
```

I have added printed messages here for you to more clearly see what should be happening, but it is not necessary for you to do this. It is sufficient for your output to look like:

```
D = PR_DiGraph(data, iteration_limit = 5)
for char in D:
    print(char)
```

```
burr
washington
burr
hamilton
washington
```

Your Solution

In [185]:
```
import random
'''class for iterating thorugh the page rank directed graph '''
class PR_Iterator:

    '''creates instance variables, sets first current state'''
    def __init__(self, prItem):
        self.prItem = prItem
        self.current_state = 'hamilton' #current state
        self.i = 0 #my counter

    '''
    depending on current_state value, we will either
    1) follow the link to a random choice from the link
    2) teleport and get a completely new link
    following the link means to pick a random new character mentioned by the current character
    '''
    def follow_link(self):
        #try except to catch any key errors
        try:
            #get all list of possible choices
            temp1 = self.prItem.linked_by(self.current_state)
            #get 1 entry from list from temp 1
            next_state = random.choice(temp1)
            #if statement to check if self link
            if self.current_state == next_state:
                self.teleport()
            else:
                self.current_state = next_state

        except KeyError:
            self.teleport()

    ''' Set the current state to a new state (key of the link dict) completely at random '''
    def teleport(self):
        prItemKeys = self.prItem.link_dict.keys() #get keys
        #note: keys must be in a list format to use random.choice
        self.current_state = random.choice(list(prItemKeys))

    '''
    iterates until we hit limit and decides whether we will teleport or follow the link
    teleport() has 15% chance and follow_link() has a 85% probability
    '''
    def __next__(self):
        #check if we need stop iteration
        self.i += 1 #increment
        if self.i > self.prItem.iteration_limit:
            raise StopIteration

        flip = random.random() #rand num between 0 and 1

        #either teleport or follow link based on prob of flip
        if flip < 0.85 :
            self.follow_link()
        elif flip >= 0.85:
            self.teleport()

        return self.current_state

D = PR_DiGraph(data, iteration_limit = 5)
```

In [186]:
```
# run the below
D = PR_DiGraph(data, iteration_limit = 5)
for char in D:
    print(char)
```

```
burr
ness
burr
generalMontgomery
madison
```

(G). Compute PageRank

Finally, we are ready to compute the PageRank in our data set. Initialize a `PR_DiGraph` with a large iteration limit (say, 1,000,000). Use a `for` -loop to allow your surfer to randomly move through the data set. The number of times that the surfer visits state `x` is the PageRank score of `x` .

Create a `dict` which logs how many times a given state appears when iterating through the `PR_Digraph` . So, this dictionary holds the PageRank score of each state.

Your Solution

In [192]:
```
D = PR_DiGraph(data, iteration_limit = 1000000)
pageRank = dict()

#going through D to find values and keys for pageRank dict
for key in D:
    if key not in pageRank.keys():
        pageRank[key] = 1
    elif key in pageRank.keys():
        pageRank[key] += 1

print(pageRank) #print the whole dict
```

```
{'reynolds': 29311, 'seabury': 17159, 'laurens': 27350, 'lee': 33423, 'doctor': 17411, 'hamilto
n': 166320, 'jefferson': 71903, 'lafayette': 34221, 'angelica': 47829, 'eliza': 52026, 'jAdam
s': 31022, 'women': 16873, 'burr': 99801, 'kingGeorge': 28803, 'sAdams': 3326, 'washington': 92
098, 'company': 17044, 'men': 17112, 'kingLouis': 1795, 'madison': 36816, 'ensemble': 17098, 'p
eggy': 20600, 'schuylerSis': 18932, 'philipH': 26483, 'philipS': 8018, 'mulligan': 21414, 'eack
er': 6176, 'generalMercer': 1647, 'rochambeau': 3942, 'knox': 3904, 'jay': 1660, 'theodosiaDaug
hter': 1724, 'maria': 1826, 'weeks': 1748, 'paine': 1947, 'admiralHowe': 738, 'ness': 1663, 'sa
lly': 2788, 'pendleton': 1688, 'green': 3875, 'marthaWashington': 1730, 'theodosiaMother': 165
2, 'generalMontgomery': 1765, 'betsy': 1645, 'conway': 1723, 'franklin': 1971}
```

(H). Display Your Result

Use your favorite approach to show the results in sorted format, descending by PageRank score. The entries at the top should be the entries with highest PageRank. What are the most important elements in the data set?

You may show either the complete list or just the top 10.

Check your code by comparing your top 10 to mine. Because we are using a randomized algorithm, your results will not agree exactly with mine, but they should be relatively close. If your top 10 list is very different, then you might want to revisit your previous solutions.

For Hamilton, my top 10 were:

```
[('hamilton', 166062),
 ('burr', 99180),
 ('washington', 92246),
 ('jefferson', 72450),
 ('eliza', 51485),
 ('angelica', 48042),
 ('madison', 37421),
 ('lafayette', 34297),
 ('lee', 33678),
 ('jAdams', 31121)]
```

For the flights data, my top 10 were:

```
[('LHR', 18043), # London Heathrow
 ('ATL', 16370), # Atlanta
 ('JFK', 14795), # New York JFK
 ('FRA', 14156), # Frankfurt
 ('CDG', 14073), # Charles de Gaulle (Paris)
 ('LAX', 13199), # Los Angeles
 ('ORD', 12915), # Chicago O'Hare
 ('PEK', 12525), # Beijing
 ('AMS', 12410), # Amsterdam Schiphol
 ('PVG', 11517)] # Shanghai
```

Your solution

In [217]:
```
#sort by value using sorted, note: keys with highest values with be at the bottom
endList = sorted(pageRank.items(), key=lambda x:x[1])
#getting the 10 keys wiht the highest values
endList[-1:-11:-1]
```

Out[217]:
```
[('hamilton', 166320),
 ('burr', 99801),
 ('washington', 92098),
 ('jefferson', 71903),
 ('eliza', 52026),
 ('angelica', 47829),
 ('madison', 36816),
 ('lafayette', 34221),
 ('lee', 33423),
 ('jAdams', 31022)]
```

(I). Submit!

Check that your code is appropriately documented (comments and docstrings), and turn it in.