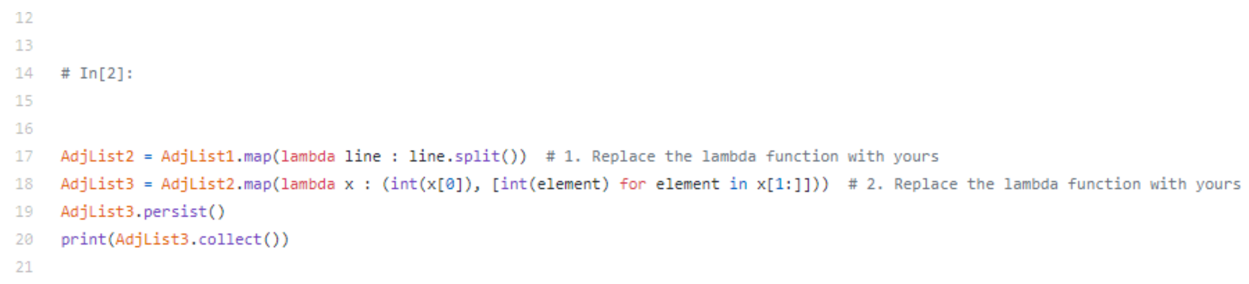
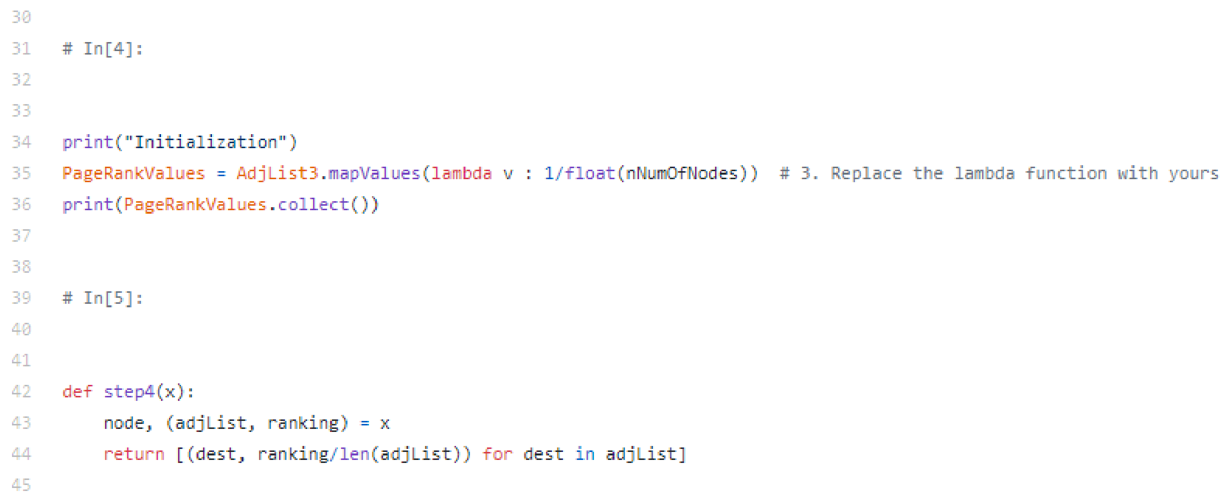
1. We basically import an adjacency list and use the first 2 lambda functions to turn it into tuples. We then check how many nodes in the adjacency list, and use the 3rd lambda function to initialize each node's initial value. Then we create a function just for the 4th lambda function, and its purpose is basically to calculate the destination for each node and calculate their ranking. Now we use a for loop to calculate 30 times of our pagerank, the 5th lambda is for adding up the previous pagerank value, the 6th lambda is for adding up the decay that happens in pagerank algorithm.

2. The result I have not quite matching the terminal txt file professor provides, but overall is an expected result. Node 4 is the most important node among all 5 nodes.

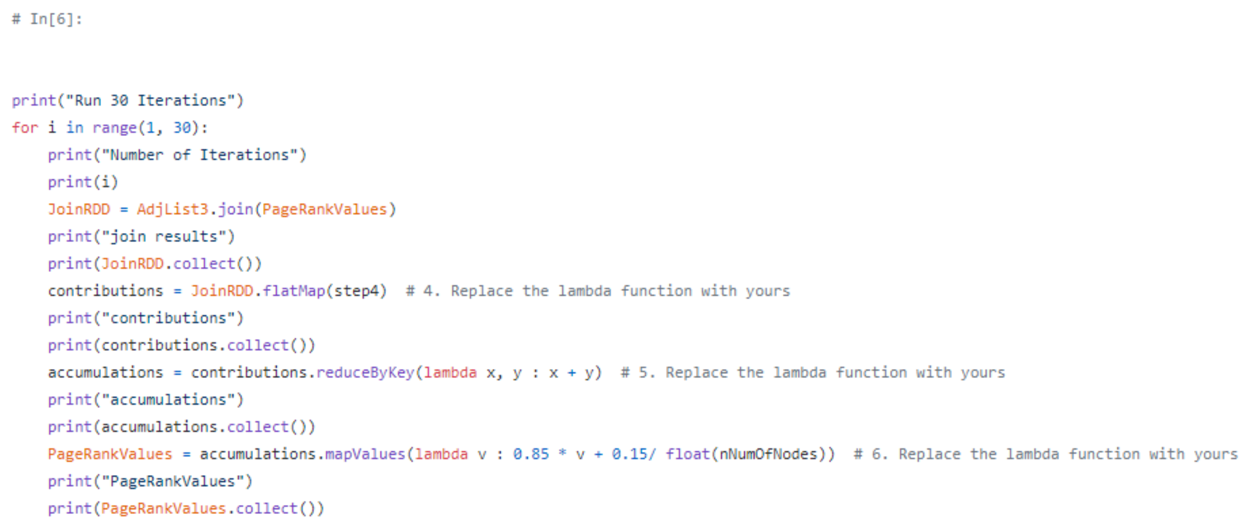
2.1



```


# In[2]:


AdjList2 = AdjList1.map(lambda line : line.split())  # 1. Replace the lambda function with yours
AdjList3 = AdjList2.map(lambda x : (int(x[0]), [int(element) for element in x[1:]]))  # 2. Replace the lambda function with yours
AdjList3.persist()
print(AdjList3.collect())

```



```

# In[4]:


print("Initialization")
PageRankValues = AdjList3.mapValues(lambda v : 1/float(nNumOfNodes))  # 3. Replace the lambda function with yours
print(PageRankValues.collect())


# In[5]:


def step4(x):
    node, (adjList, ranking) = x
    return [(dest, ranking/len(adjList)) for dest in adjList]

```



```
# In[6]:


print("Run 30 Iterations")
for i in range(1, 30):
    print("Number of Iterations")
    print(i)
    JoinRDD = AdjList3.join(PageRankValues)
    print("join results")
    print(JoinRDD.collect())
    contributions = JoinRDD.flatMap(step4)  # 4. Replace the lambda function with yours
    print("contributions")
    print(contributions.collect())
    accumulations = contributions.reduceByKey(lambda x, y : x + y)  # 5. Replace the lambda function with yours
    print("accumulations")
    print(accumulations.collect())
    PageRankValues = accumulations.mapValues(lambda v : 0.85 * v + 0.15/ float(nNumOfNodes))  # 6. Replace the lambda function with yours
    print("PageRankValues")
    print(PageRankValues.collect())
```

The output is too long to post here, I provide separate txt files in github.

2.2)The output is not quite matching, but i think the reason is the decay value is not ideal.