Part 3

Data Preparation:

1. Extract the venue and corresponding authors from the references and save it in a dictionary with ‘venue’ as the key and ‘list of authors’ as the value.
2. Upon extracting the venues we found that there are more than 1,38,000 unique venues Hence, using the LSH club the venues which are 60% similar and ignore the rest. For this, we have used the LSH implementation of Part 2
3. Upon finding the venues which are similar, take out the keys from the dictionary. Rename all the similar venues of a particular bucket with the same name. Each bucket should have a different name, but all the venues matched inside the bucket will have the same name.
4. This List will be used as input data for Estimating the number of times a publication venue is cited in the last 500 citations using the DGIM algorithm.
5. Since we know which venues are similar we can now generate a new dictionary with the key as the venue name (it will be one for the entire bucket) and value is a list of all the authors of all the matched venue
6. This Dictionary will be used as input data for Finding the Number of unique authors for each publication venue.
7. Refer to the data prep ipynb file for details

Question 1:

Import necessary libraries

```
from kafka.consumer import KafkaConsumer
from kafka.producer.kafka import KafkaProducer
from kafka.errors import KafkaError
from struct import pack, unpack
import mmh3
import math
import json
```

Load data from pickle file

```
import pickle
with open('assignment3_Q1_pickle', 'rb') as f:
    res = pickle.load(f)              # load method is used to load the contents of a pickle file

print(res)
```

Calculate trailing xeros

```
def trailing_zeros(n):
    if n == 0:
        return 32
    p = 0
    while (n >> p) & 1 == 0:
        p += 1

    return p
```

All the hashes are provided to Flajolet martin to find max trailing zeros

```
def flajolet_martin(hashes):
    maxZero = 0
    for i in hashes:
        maxZero = max(maxZero, trailing_zeros(i))
    return 2 ** maxZero
```

Initialize Kafka producer

```
producer = KafkaProducer(bootstrap_servers=['127.0.0.1:9092'])
```

Initialize Kafka Consumer:

```
consumer = KafkaConsumer('myTopic05', bootstrap_servers=['127.0.0.1:9092'], auto_offset_reset='earliest',
                         enable_auto_commit=True,
                         group_id='flajolet-martin')
```

For every venue and author key-value pair, send the author data via a producer to a topic. Then read the data via a consumer who is listening on that topic. Once we have the data we convert it and then split it as per comma. The split data is then hashed and stored in a hash array. The hashed array is then passed to the flajolet_martin function for computation. The outcome is then again pushed back on the target topic via the producer.

```
for k,v in res.items():

    producer.send('myTopic05', str(v).encode('utf-8'))

    message = next(consumer)
    value = message.value
    words = value.decode('utf-8').split(',')
    hashes = [mmh3.hash(word) for word in words]
    estimate = flajolet_martin(hashes)
    producer.send('output-topic', (str(k) +"   ===>   " + str(estimate)).encode('utf-8'))
```

Output:

```
Proc. IEEE Int. Conf. Image Processing (ICIP'08)   ===>   32
25th USENIX Security Symposium (USENIX Security 16)   ===>   2
Segan: Speech enhancement generative adversarial network   ===>   8
IEEE Signal Processing in Medicine and Biology Symposium   ===>   4
Proceedings of the Workshop on Human Language Technology -HLT '93   ===>   4
49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)   ===>   8
Sixth International Conference on Wireless Communications and Signal Processing   ===>   8
American Journal of Epidemiology   ===>   8
Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining   ===>   32
Continuous recurrent neural networks with adversarial training   ===>   1
Proceedings of the 27th international conference on machine learning   ===>   1
Unpaired image-to-image translation using cycle-consistent adversarial networks   ===>   64
```

Like this for unique venues we can see the estimates

But the number venues obtained are more than 16000. And creating and accessing so many venues has heavy toll on the system. Hence we created 20 venues for this purpose. We are going with the assumption that there are 20 different venues.

Import required libraries

```
from kafka.admin import KafkaAdminClient, NewTopic
import re
import string
```

Register Kafka admin client

```
admin_client = KafkaAdminClient(bootstrap_servers='127.0.0.1:9092')
```

Generate 20 topics and save its value in a list called venueTopic_list

```
venueTopic_list =[]
for i in range(0,20):
    s = 'venueTopic' + str(i)
    venueTopic_list.append(s)
    new_topics = NewTopic(s, num_partitions=1, replication_factor=1)
    admin_client.create_topics([new_topics])
```

We divided all the venues into 20 buckets

```
key = len(list(res.keys()))
bucket = key/20
```

For data in every bucket we initialize the consumer to that particular topic and send data via producer. For evry new bucket change the topic and place producer and consumer on that topic. We apply the same steps and output is directed to output topic.

```
i=0
j=0
topic = venueTopic_list[j]
for k,v in res.items():
    i = i+1
    if i%bucket == 0:
        j = j+1
        if j ==20:
            break
        topic = venueTopic_list[j]
    producer.send(topic, str(v).encode('utf-8'))
    consumer = KafkaConsumer(topic, bootstrap_servers=['127.0.0.1:9092'], auto_offset_reset='earliest',
                         enable_auto_commit=True,
                         group_id='flajolet-martin')
    message = next(consumer)
    value = message.value
    words = value.decode('utf-8').split(',')
    hashes = [mmh3.hash(word) for word in words]
    estimate = flajolet_martin(hashes)
    producer.send('output-topic', (str(k) +"   ===>   " + str(estimate)).encode('utf-8'))
```

And we get the same output here as well

Estimate the number of times a publication venue is cited in the last 500 citations using the DGIM algorithm.

Read the dataset created for this question in the data preparation step

```
l1 = []
with open(r'C:/Users/hp/Big Data Analytics (BDA)/Assignment3_question2_dataset.txt', 'r') as fp:
    for line in fp:
        x = line[:-1]
        l1.append(x)

print(l1)
```

Take out last 500 elements in the list to new list

```
l2 = l1[-500:]
```

Find unique number of elements in last 500 items extracted above

```
s = set(l2)
len(s)
```

```
372
```

Check the count of each item in the list. This is solely used only for testing purpose so that comparison can be done.

```
count = pd.Series(l2).value_counts()
count
```

```
Combinatorica
64
Proc. IEEE Conference on Computer Vision and Pattern Recognition CVPR '07
4
Proceedings of the 9th International Conference on Architectural Support for Pr
4
International Joint Conference on Artificial Intelligence (IJCAI)
4
Symposium on Usable Privacy and Security (SOUPS)
3

..
```

Use the Dgim algorithm to find occurrences of an item. For this, we need a bit vector(with True/False values). Hence, converting the original list 'l1' into a bit vector. For the item in which we want to find the occurrences, if it is present in 'l1' then update the bit vector with 'True'. For anything other than the searched item, update the vector as 'False'. We have to check for the last 500 elements hence, set N=500. In the below example, the item for which we have to fund the occurrences is "Combinatorica"

```
from dgim import Dgim
dgim = Dgim(N=500, error_rate=0.1)
for i in l1:
    if i=="Combinatorica":
        dgim.update(True)
    else:
        dgim.update(False)
dgim_result = dgim.get_count()
```

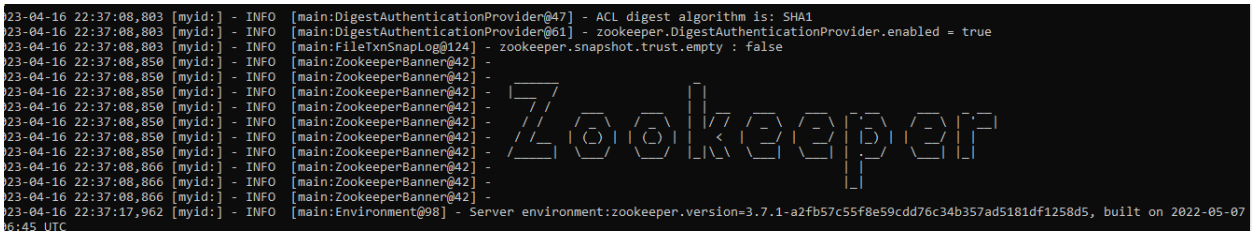
```
dgim_result
```

```
63
```

Additionally, some commands to handle kafka and zookeeper

Run zookeeper:

1. Create zoo.cfg file (config file)
2. Move to e ⇒ e:
3. cd E:\apache-zookeeper-3.7.1-bin\apache-zookeeper-3.7.1-bin\bin
4. Run zkServer.cmd



Run Kafka:

1. Modify server.properties file
2. cd E:\kafka_2.13-3.4.0\kafka_2.13-3.4.0
3. bin\windows\kafka-server-start config\server.properties
4. Create a topic: '**myTopic**' is the topic name
   https://stackoverflow.com/questions/46158296/kafka-broker-not-available-at-starting

   bin\windows\kafka-topics --bootstrap-server 127.0.0.1:9092 --create --topic **myTopic** --partitions 1 --replication-factor 1

5. List topics in the Kafka cluster:
   bin\windows\kafka-topics --bootstrap-server 127.0.0.1:9092 --list

6. Check the properties of a topic:
   bin\windows\kafka-topics --bootstrap-server 127.0.0.1:9092 --describe --topic **myTopic**

7. Create producer:
   bin\windows\kafka-console-producer --bootstrap-server 127.0.0.1:9092 --topic myTopic
   ===> enter your mesage

8. Create Consumer:
   bin\windows\kafka-console-consumer --bootstrap-server 127.0.0.1:9092 --topic myTopic

   ===> to print from beginning
   bin\windows\kafka-console-consumer --bootstrap-server 127.0.0.1:9092 --topic myTopic --from-beginning

9. List consumer groups:

bin\windows\kafka-consumer-groups --bootstrap-server 127.0.0.1:9092 --list

10. Describe a consumer group: ‘console-consumer-18647’ is the group name
    bin\windows\kafka-consumer-groups --bootstrap-server 127.0.0.1:9092 --describe --group **console-consumer-18647**

11. The Kafka cluster stores information about consumers using consumer offset and stores all info related to consumers like from where the messages are consumed by the consumer and from which topic. So if we list all topics from the Kafka cluster we can see the topic named **__consumer_offsets.**

12. We can create multiple producers and multiple consumers for a topic. Anything produced by a producer will be consumed by all consumers of that topic. When we create multiple consumers, Kafka creates multiple consumer groups, each for one consumer. When we close the consumer the consumer group will be removed.

13. Define consumer group while creating a consumer: “myConsumerGroup” is the consumer group name
    bin\windows\kafka-console-consumer --bootstrap-server 127.0.0.1:9092 --topic myTopic --group **myConsumerGroup**