

Azat Dovgeldiyev

CSC 555

Homework 1

## Part 1.

### a) Compute

```
print(2**12)
print((2**3)**4)
print(4**6)
print(8**4)
```

```
4096
4096
4096
4096
```

```
print("837 MOD 100 is =",837%100)
print("842 MOD 20 is =",842%20)
print("21 MOD 111 is =",21%111)
print("111 MOD 21 is =",111%21)
```

```
837 MOD 100 is = 37
842 MOD 20 is = 2
21 MOD 111 is = 21
111 MOD 21 is = 6
```

b) Given vectors  $V1 = (1, 2, 3)$  and  $V2 = (1, 2, 3)$  and a 3x3 matrix  $M = [(2, 1, 3), (1, 2, 2), (1, 0, 1)]$ , compute:

```
import numpy as np
V1 = (1, 2, 3)
V2 = (1, 2, 3)
#V2 - V1
print(np.subtract(V2,V1))
```

```
[0 0 0]
```

```
#V1 + V1
print(np.add(V1,V1))
```

```
[2 4 6]
```

```
#|V1| (Euclidean vector length, not the number of dimensions)
#converting tuple to array-like
#formula -> sum of power of vector in square root

a=np.asarray(V1)**2
print(np.sqrt(a.sum()))
#using norm
print(np.linalg.norm(V1))
```

```
3.7416573867739413
3.7416573867739413
```

```
#|V2|
#converting tuple to array-like
a=np.asarray(V2)**2
print(np.sqrt(a.sum()))
#using norm
print(np.linalg.norm(V2))
```

```
3.7416573867739413
3.7416573867739413
```

```

▶ #|V2|
#converting tuple to array-like
a=np.asarray(V2)**2
print(np.sqrt(a.sum()))
#using norm
print(np.linalg.norm(V2))

```

```

3.7416573867739413
3.7416573867739413

```

```

▶ M = [(2, 1, 3), (1, 2, 2), (1, 0, 1)]
#M * V2 (matrix times vector, transpose it as necessary)
np.dot(M,V2)

```

```

]: array([13, 11, 4])

```

```

▶ np.linalg.matrix_power(M,2)

```

```

]: array([[ 8,  4, 11],
          [ 6,  5,  9],
          [ 3,  1,  4]])

```

```

▶ np.linalg.matrix_power(M,3)

```

```

]: array([[31, 16, 43],
          [26, 16, 37],
          [11,  5, 15]])

```

c) Suppose we are flipping a coin with Head (H) and Tail (T) sides. The coin is not balanced with 0.6 probability of H coming up (and 0.4 of T). Compute the probabilities of getting:

HTHT

THTT

Exactly 1 Head out of a sequence of 3 coin flips.

Exactly 1 Tail out of sequence of 3 coin flips.

```

▶ #HTHT
prob_H = 0.6
prob_T = 0.4
print("Probability of getting HTHT is: ",prob_H*prob_T*prob_H*prob_T)
print("Probability of getting THTT is: ",round((prob_T*prob_H*prob_T*prob_T),4))

```

```

Probability of getting HTHT is: 0.0576
Probability of getting THTT is: 0.0384

```

```

▶ #Exactly 1 Head out of a sequence of 3 coin flips.
print("Probability of getting exactly 1 Head in 3 flips: ",3*prob_H*prob_T*prob_T)
print("Probability of getting exactly 1 Tail in 3 flips: ",round((3*prob_H*prob_H*prob_T),3))

```

```

Probability of getting exactly 1 Head in 3 flips: 0.288
Probability of getting exactly 1 Tail in 3 flips: 0.432

```

- d) Consider a database schema consisting of two tables, Employee (ID, Name, Address), Project (PID, Name, Deadline), Assign(EID, PID, Date). Assign.EID is a foreign key referencing employee's ID and Assign.PID is a foreign key referencing the project.

Write SQL queries for:

- i. Find projects that are not assigned to any employees (PID and Deadline of the project).

```
SELECT p.id, p.deadline from PROJECT p
WHERE p.id NOT IN (SELECT p_id FROM ASSIGN a
INNER JOIN EMPLOYEE e WHERE a.eid = e.id)
```

- ii. For each date, find how many assignments were made that day.

```
SELECT Date, COUNT() as Assignments FROM Assign
GROUP BY Date
ORDER BY Assignments DESC;
```

- iii. Find all projects that have either 0, 1, or 2 employees assigned to them (note that the answer should include 0 employees to be correct).

```
SELECT p.PID , p.NAME, p.Deadline, COUNT(a.PID) as empCount
FROM Project p
LEFT JOIN Assign a
On a.PID = p.PID
GROUP BY p.PID, p.Name, p.Deadline
HAVING COUNT(a.PID) <=2;
```

#### e) Mining of Massive Datasets, Exercise 1.3.3

"First, a hash function  $h$  takes a hash-key value as an argument and produces a bucket number as a result. The bucket number is an integer, normally in the range 0 to  $B - 1$ , where  $B$  is the number of buckets."

$$h(x) = x \bmod 15.$$

0, 3, 5 and their multiples such as 0, 3, 5, 6, 9, 10, 12 and 15 do not work for  $x$ . Any other non-negative integers are acceptable values. 1/15 of the integers will be assigned to each of the buckets. Using values for 4 and 7 which are acceptable we can distribute into buckets.

1	15	1
5	15	5
9	15	9
13	15	13
17	15	2
21	15	6
25	15	10
29	15	14

1	15	1
8	15	8
15	15	0
22	15	7
29	15	14
36	15	6
43	15	13
50	15	5

d) Hadoop Distributed Filesystem.

i) Describe the purpose of replication in HDFS

It increases the availability of Data at any time. When file comes in and it stored in HDFS, user should replicate it so in case of failure files don't lost. In sum, if any node containing block of data which is used for processing crashes, user can get the same block of data from another node.

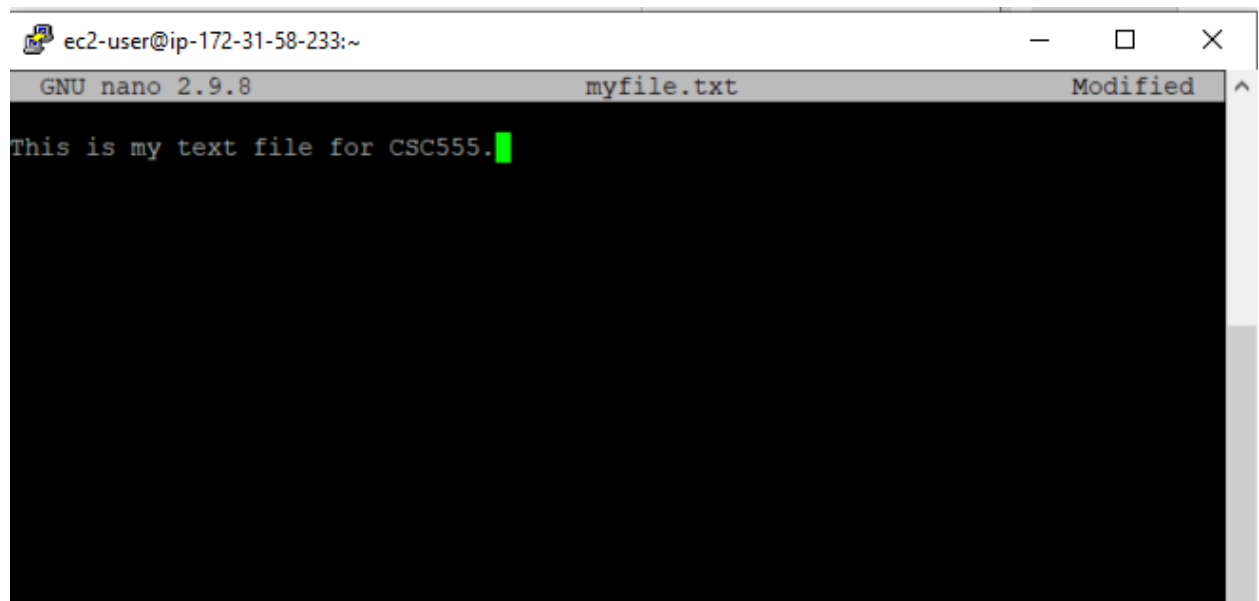
ii) What are the guarantees offered by a replication factor of 3 (3 copies of each block).

When we make 3 copies, we might lose 2 machines but not lose any data. So, we can tolerate 2 nodes or 1 full rack going out without losing any data. Hence, ideally HDFS stores one copy of block of data in one node of one Rack and other two copies in different nodes of different Rack. This ensures Fault tolerance and High availability.

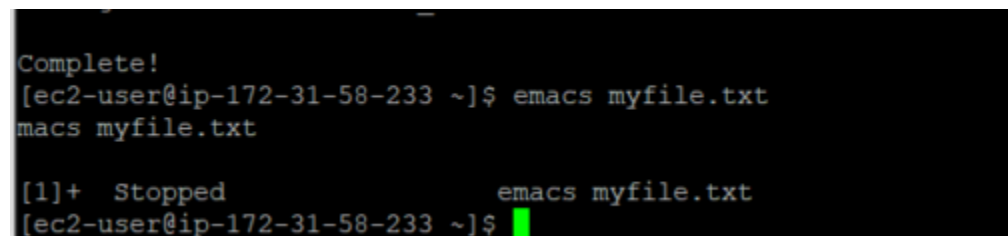
iii) What is the storage cost for a file of size N MBs, if the HDFS replication factor is set to 3?

The default setting to make 3 copies 2 on the same rack and 1 remote, 2 node or 1 full rack going out without losing data.  $N/128\text{MB}$  (N MB divided by 128 MB) will result x number of blocks. Since the default replication factor is 3, each block will be replicated thrice.

## Part 2: Linux Intro



```
ec2-user@ip-172-31-58-233:~  
GNU nano 2.9.8      myfile.txt      Modified  
This is my text file for CSC555.█
```




```
Complete!  
[ec2-user@ip-172-31-58-233 ~]$ emacs myfile.txt  
macs myfile.txt  
  
[1]+  Stopped                  emacs myfile.txt  
[ec2-user@ip-172-31-58-233 ~]$ █
```

```
Complete!
[ec2-user@ip-172-31-58-233 ~]$ emacs myfile.txt
emacs myfile.txt
```

```
[1]+  Stopped                  emacs myfile.txt
[ec2-user@ip-172-31-58-233 ~]$ vim myfile.txt
[ec2-user@ip-172-31-58-233 ~]$
```

```
[1]+  Stopped                  emacs myfile.txt
[ec2-user@ip-172-31-58-233 ~]$ vim myfile.txt
[ec2-user@ip-172-31-58-233 ~]$ ls
myfile.txt  myfile.txt~
[ec2-user@ip-172-31-58-233 ~]$ cat myfile.txt
This is my text file for CSC555 vim.
[ec2-user@ip-172-31-58-233 ~]$
```

 ec2-user@ip-172-31-58-233:~

GNU nano 2.9.8

mycopy.txt

```
This is the copy of the original file.
```

```
mycopy.txt  myfile.txt  myfile.txt~
[ec2-user@ip-172-31-58-233 ~]$ nano mycopy.txt
[ec2-user@ip-172-31-58-233 ~]$
```

```
[ec2-user@ip-172-31-58-233 ~]$ cd CSC555/
[ec2-user@ip-172-31-58-233 CSC555]$ ls
mycopy.txt  myfile.txt
[ec2-user@ip-172-31-58-233 CSC555]$
```

```
CSC555  myfile.txt~  myzipfile.zip
[ec2-user@ip-172-31-58-233 ~]$ unzip myzipfile.zip
Archive:  myzipfile.zip
  inflating: mycopy.txt
  extracting: myfile.txt
[ec2-user@ip-172-31-58-233 ~]$
```

Size of grail file.

```
-rw-rw-r-- 1 ec2-user ec2-user 73K Aug  9 2000 grail
```

```

[2]+ Stopped                  emacs lucky.py
[ec2-user@ip-172-31-58-233 ~]$ python lucky.py
*****
    My Lucky Numbers
*****
My lucky number is 2!
My lucky number is 4!
My lucky number is 6!
My lucky number is 8!
My lucky number is 10!
My lucky number is 12!
My lucky number is 14!
My lucky number is 16!
My lucky number is 18!
My lucky number is 20!
[ec2-user@ip-172-31-58-233 ~]$

```

Write python code to read a text file (you can use myfile.txt) and output a word count total for each word with the number of times that word occurs in the entire file. That is, if the file has the word “Hadoop” occurs in the file 5 times, your code should print “Hadoop 5”. It is similar to Hadoop word count you will run in the next part.

```

ec2-user@ip-172-31-58-233:~
File Edit Options Buffers Tools Python Help
file = open("myfile.txt")
count = {}
for word in file.read().split():
    if word not in count:
        count[word] = 1
    else:
        count[word] += 1
for k, v in count.items():
    print(k, v)
file.close();

[ec2-user@ip-172-31-58-233 ~]$ emacs wordCount.py

[4]+ Stopped                  emacs wordCount.py
[ec2-user@ip-172-31-58-233 ~]$ python wordCount.py
('file.', 1)
('for', 1)
('This', 2)
('text', 1)
('is', 2)
('vim.', 1)
('original', 1)
('file', 1)
('of', 1)
('CSC555', 1)
('copy', 1)
('my', 1)
('the', 2)

```

## Part 3: Lab

For this part of the assignment, you will run wordcount on a single-node Hadoop instance. I am going to provide detailed instructions to help you get Hadoop running. The instructions are following Hadoop: The Definitive Guide instructions presented in Appendix A: Installing Apache Hadoop.

```
[ec2-user@ip-172-31-58-233 ~]$ hadoop fs -put bioproject.xml /data1/
[ec2-user@ip-172-31-58-233 ~]$ hadoop fs -ls /data1
Found 1 items
-rw-r--r--  1 ec2-user supergroup  231149003 2020-09-21 15:16 /data1/bioproject.xml
[ec2-user@ip-172-31-58-233 ~]$
```

Report the time that the job took to execute as screenshot

```
ec2-user@ip-172-31-58-233:~
Merged Map outputs=2
GC time elapsed (ms)=1080
CPU time spent (ms)=43240
Physical memory (bytes) snapshot=563118080
Virtual memory (bytes) snapshot=6318002176
Total committed heap usage (bytes)=333586432
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
  Bytes Read=231153099
File Output Format Counters
  Bytes Written=20056175

real    1m14.149s
user    0m4.012s
sys     0m0.225s
[ec2-user@ip-172-31-58-233 ~]$ hadoop fs -du /data1/wordcount1/
0          /data1/wordcount1/_SUCCESS
20056175   /data1/wordcount1/part-r-00000
[ec2-user@ip-172-31-58-233 ~]$
```

ec2-user@ip-172-31-58-233:~

```
antarcticus</OrganismName>      5
arctic 21
arctica 27
arctica<I> 2
arctica</i> 3
arctica</i>, 1
arctica.</Description> 2
arctica</Name> 5
arctica</OrganismName> 5
arcticus 31
arcticus</i> 2
arcticus</Name> 4
arcticus</OrganismName> 4
holarctica 77
humans.Antarctic 1
palearctica 66
palearctica</Name> 1
sub-Antarctic 4
sub-arctic 4
subantarctic 1
subantarcticus 7
subantarcticus</Name> 1
subantarcticus</OrganismName> 1
subarctic 21
[ec2-user@ip-172-31-58-233 ~]$
```