DATE:

Name: Prerna Sunil Jadhav
Sap ID: 60004220127
Batch: C2-2
Course: Big Data Infrastructure laboratory.
Course Code: DJ19CEEL6011

EXPERIMENT 06

AIM: Implement Matrix Multiplication and word frequency count using MapReduce

THEORY: MapReduce is a technique in which a huge program is sub-divided into small tasks and run parallely to make computation faster, save time, and mostly used in distributed system.
It has 2 important parts:

- MAPPER: It takes raw data i/p and organizes into key, value pairs.
  for Eg: In a dictionary, you search for the word 'Data' and its ~~accos~~ associated meaning is "the facts and statistic~~ally~~ collected together for reference or analysis". Here the key is "Data" and the value associated with" its facts and statistics collected together for reference or analysis"
- Reducer: It is responsible for processing data in parallel and produce final output.

DATE:

⇒ MATRIX MULTIPLICATION Using MAP REDUCE

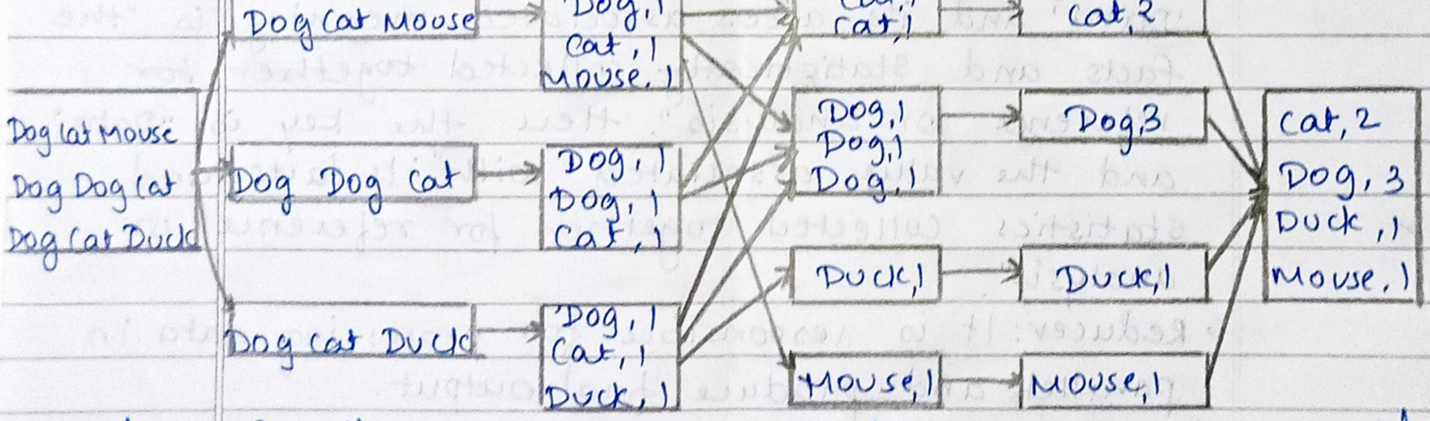
- The Map function:

```
for each element mij of M do
    produce (key,value) pairs as ((i,k),(M,j,mij)),
    for k=1,2,3,...up to the no.of columns of N.
for each eleement njk of N do
    produce (key,value) pairs as ((i,k),(N,j,njk)),
    for i=1,2,3,... up to the no.of rows of M.
return set of (key,value) pair that each key (i,k) has a
list with values (M,j,mij) & (N,j,njk) for all values of j.
```

- The Reduce function:

```
for each key (i,k) do
    sort values begin with M by j in listM.
    sort value begin with N by j in listN.
    multiply mij and njk for jth value of each list.
    sum up mij * njk
return (i,k), $\sum_{j=1} m_{ij} * n_{jk}$
```

⇒ ~~MATRIX~~ COUNT FREQUENCY OF WORDS Using MAP REDUCE.



CONCLUSION: Thus, we studied and implement MapReduce

Name:	Prerna Sunil Jadhav
Sap Id:	60004220127
Class:	T. Y. B. Tech (Computer Engineering)
Course:	Big Data Infrastructure Laboratory
Course Code:	DJ19CEEL6011
Experiment No.:	06

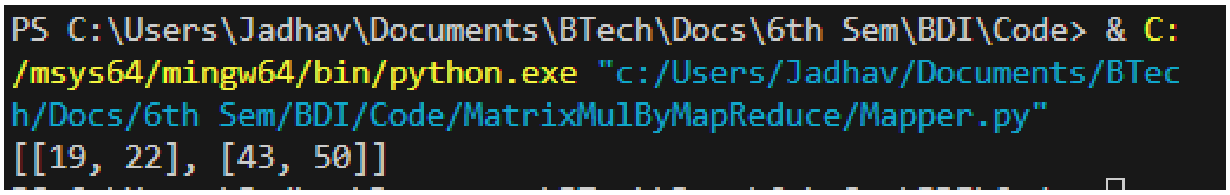
AIM: Implement Matrix Multiplication and Word Frequency Count using Map Reduce

MATRIX MULTIPLICATION USING MAP REDUCE.

CODE:

```
def matrix_multiply_mapper(matrix_a, matrix_b):
    result = []
    for i in range(len(matrix_a)):
        for j in range(len(matrix_b[0])):
            for k in range(len(matrix_b)):
                result.append(((i, j), (matrix_a[i][k] * matrix_b[k][j])))
    return result

def matrix_multiply_reducer(mapped_result):
    intermediate_result = {}
    for key, value in mapped_result:
        if key in intermediate_result:
            intermediate_result[key].append(value)
        else:
            intermediate_result[key] = [value]

    final_result = []
    for key, values in intermediate_result.items():
        total = sum(values)
        final_result.append((key, total))
    return final_result

def matrix_multiply(matrix_a, matrix_b):
    mapped_result = matrix_multiply_mapper(matrix_a, matrix_b)
    reduced_result = matrix_multiply_reducer(mapped_result)
    final_result = [[0 for _ in range(len(matrix_b[0]))] for _ in
range(len(matrix_a))]

    for key, value in reduced_result:
        i, j = key
        final_result[i][j] = value
    return final_result

# Example usage:
matrix_a = [[1, 2],
            [3, 4]]

matrix_b = [[5, 6],
            [7, 8]]

result = matrix_multiply(matrix_a, matrix_b)
print(result)
```

OUTPUT:

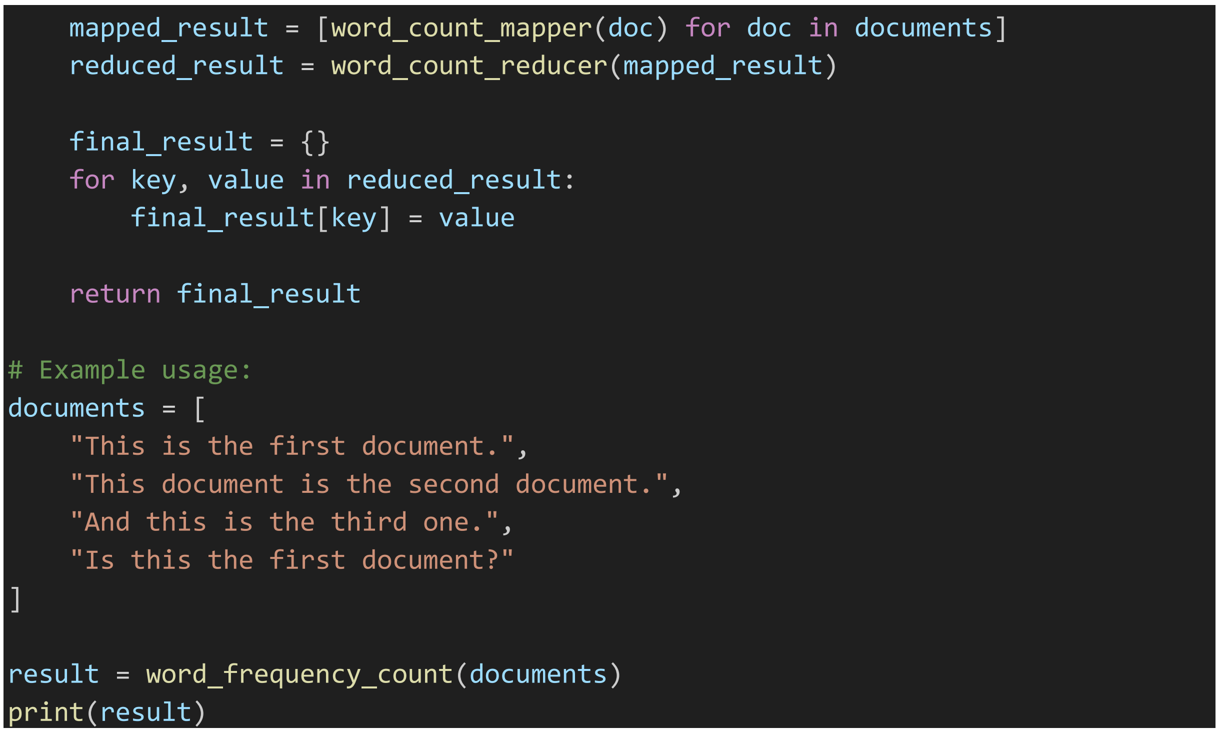


```
PS C:\Users\Jadhav\Documents\BTech\Docs\6th Sem\BDI\Code> & C:/msys64/mingw64/bin/python.exe "c:/Users/Jadhav/Documents/BTech/Docs/6th Sem/BDI/Code/MatrixMulByMapReduce/Mapper.py"
[[19, 22], [43, 50]]
```

<u>WORD FREQUENCY COUNT USING MAP REDUCE</u>.

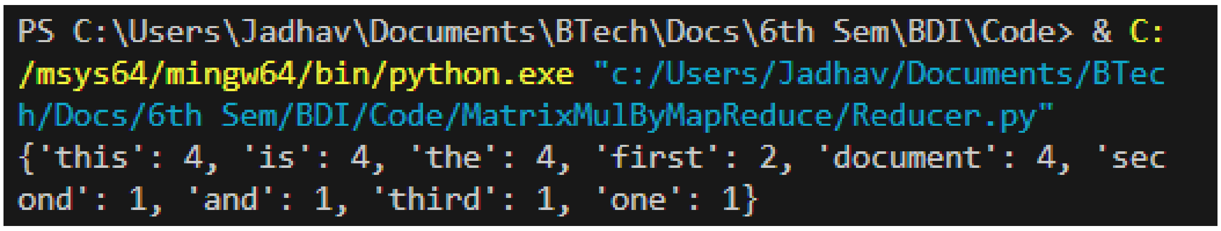
CODE:

```
import re

def word_count_mapper(document):
    words = re.findall(r'\w+', document.lower())
    word_count = {}
    for word in words:
        word_count[word] = word_count.get(word, 0) + 1
    return list(word_count.items())

def word_count_reducer(mapped_result):
    intermediate_result = {}
    for item in mapped_result:
        for key, value in item:
            if key in intermediate_result:
                intermediate_result[key] += value
            else:
                intermediate_result[key] = value
    return list(intermediate_result.items())

def word_frequency_count(documents):
```



```
    mapped_result = [word_count_mapper(doc) for doc in documents]
    reduced_result = word_count_reducer(mapped_result)

    final_result = {}
    for key, value in reduced_result:
        final_result[key] = value

    return final_result

# Example usage:
documents = [
    "This is the first document.",
    "This document is the second document.",
    "And this is the third one.",
    "Is this the first document?"
]

result = word_frequency_count(documents)
print(result)
```

OUTPUT:



```
PS C:\Users\Jadhav\Documents\BTech\Docs\6th Sem\BDI\Code> & C:
/msys64/mingw64/bin/python.exe "c:/Users/Jadhav/Documents/BTec
h/Docs/6th Sem/BDI/Code/MatrixMulByMapReduce/Reducer.py"
{'this': 4, 'is': 4, 'the': 4, 'first': 2, 'document': 4, 'sec
ond': 1, 'and': 1, 'third': 1, 'one': 1}
```