**Problem 1 (3 points)**

**Assume that a large table is distributed across multiple files, each containing partial rows of the table.**

**Each row is composed of the following data:**

**(student name, department, salary).**

**For example, (Bob, Computing, 30,000) means Bob graduated from the department of Computing and**

**the salary of his first job is 30,000.**

**The objective is to determine, in each department, the total number of graduated students whose salary**

**is more than 25,000 in their first job.**

**(a) What are the relationships between MapReduce and Apache Hadoop?**

MapReduce is a programming model designed for processing large datasets in parallel across a distributed cluster. It consists of two main phases: the Map phase and the Reduce phase. The Map phase processes input data and produces intermediate key-value pairs, while the Reduce phase aggregates the intermediate results to produce the final output.

Apache Hadoop is an open-source framework that implements the MapReduce model. It provides a distributed file system (HDFS) for storing large datasets and a MapReduce engine for processing them. Hadoop handles the distribution of data, scheduling of tasks, and fault tolerance, allowing developers to focus on writing Map and Reduce functions.

**(b) Provide a pseudo-code for the Map workers, specifying the input and output (key, value) pairs.**

The Map function processes each row of the table and outputs intermediate key-value pairs where:

* The key is the department.
* The value is a tuple containing the student name and salary.

Input: Each row of the table in the format (student name, department, salary).

Output: Intermediate key-value pairs where:

* Key: Department (string).
* Value: Tuple (student name, salary).

function Map(key, value):

// key: file name

// value: line of text (student name, department, salary)

for each line in value:

student\_name, department, salary = parse(line)

if salary > 25000:

Emit(department, 1)

**(c) Provide a pseudo-code for the Reduce workers, specifying the input and output (key, value) pairs.**

The Reduce function aggregates the intermediate results from the Map phase. It counts the number of students in each department whose salary is greater than 25,000.

Input: Intermediate key-value pairs where:

* Key: Department (string).
* Value: List of tuples (student name, salary).

**Output**: Final key-value pairs where:

* Key: Department (string).
* Value: Total count of students with salary > 25,000 (integer).

Problem 2 (2 points)

We have four text files as follows, storing the student grades of four subjects

|  |  |  |  |
| --- | --- | --- | --- |
| math.txt | physics.txt | chemistry.txt | art.txt |
| James, 81 | James, 57 | James, 78 | James, 67 |
| John, 83 | John, 78 | John, 92 | John, 89 |
| Robert, 75 | Robert, 68 | Robert, 68 | Robert, 88 |
| Michael, 71 | Michael, 71 | Michael, 91 | Michael, 87 |
| David, 79 | David, 79 | David, 77 | David, 87 |
| Mary, 73 | Mary, 69 | Mary, 74 | Mary, 79 |
| Linda, 83 | Linda, 79 | Linda, 89 | Linda, 94 |
| Susan, 67 | Susan, 76 | Susan, 87 | Susan, 78 |
| Lisa, 76 | Lisa, 74 | Lisa, 92 | Lisa, 91 |

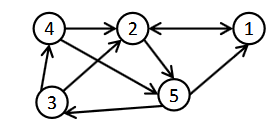
Our goal is to calculate the total scores of students in all four subjects.

(a) Write a pseudo-code for the Map workers, specifying the input and output (key, value) pairs.

(b) Write a pseudo-code for the Reduce workers, specifying the input and output (key, value) pairs.

Problem 3 (3 points) Assume that the connections among 5 webpages are represented as a graph as follows. We use the PageRank equation (with random teleports) to update the rank value of each webpage.

Assume that the initial (iteration 0) rank value of each webpage is 1/5, and the damping factor β is 0.8.



Graph = { ‘1’: [‘2’],

‘2’: [‘1’,’5’],

‘3’:[‘2’,’4’]

‘4’:[‘2’]

‘5’:[‘1’,’3’]}

(a) Formulate the PageRank equation for each webpage, ensuring the inclusion of specific weights.

The formula of PageRank Equation is:

* *β* is the damping factor (0.8 in this case).
* *N* is the total number of pages (5 in this case).
* *M*(*i*) is the set of pages that link to page *i*.
* *L*(*j*) is the number of outbound links from page *j*.

So for each webpage, the formula is:

Webpage 1:

In-neighbours: 2, 5

Out-degrees: out-degree of 2 = 2,  out-degree of 5 = 2

Webpage 2:

In-neighbours: 1,3,4

Out-degrees: out-degree of 1 = 1 out-degree of 3 = 2, out-degree of 4 = 1

Webpage 3:

In-neighbours: 5

* Out-degrees: out-degree of 5 = 2

Webpage 4:

In-neighbours: 3

* Out-degrees: out-degree of 3 = 2

Webpage 5:

In-neighbours: 2

* Out-degrees: out-degree of 2 = 2

(b) Compute the PageRank values for all five webpages during iterations 1 and 2. Subsequently,

arrange the webpages in descending order based on their PageRank values.

Iteration 1:

Iteration 2:

The webpages in descending order based on their PageRank values after iteration 2 are:

1. Page 2
2. Page 1
3. Page 5
4. Page 3
5. Page 4