Homework 1 Solution

Question 1: Market Basket Analysis (Frequent Item Pairs using MapReduce)

We aim to identify frequently purchased pairs of items from transactions.

Mapper:

1. Input: Each transaction (a list of items separated by commas).
2. Key: A pair of items appearing together in a transaction.
3. Value: 1 (indicating occurrence).

Example Input (Transaction Line)

```
milk, bread, cereal
milk, sugar, bread, eggs
```

Example Mapper Output

```
(milk, bread)   1
(milk, cereal)  1
(bread, cereal) 1
(milk, sugar)   1
(milk, eggs)    1
```

Reducer:

1. Input: Key (item pair), List of Values (count occurrences).
2. Operation: Sum up all counts for each item pair.
3. Output: (Item Pair, Total Occurrences)

Example Reducer Output:

```
(milk, bread)   3
(milk, cereal)  2
(bread, cereal) 1
```

Question 2: Plagiarism Check using MapReduce

Problem is about finding matching sentences between different articles.

MAPPER 1 : EMITTING SENTENCES WITH THEIR ARTICLE IDS

Each sentence is treated as a key, and the article it belongs to is the value.

1. Input: Each sentence from an article.
2. Key: Sentence text.
3. Value: The article ID.

Input

```
Article 1: sentence 1, sentence 2, sentence 3, sentence 4
Article 2: sentence 4, sentence 5, sentence 6, sentence 7
Article 3: sentence 8, sentence 9, sentence 2, sentence 4
Article 4: sentence 1, sentence 4, sentence 3, sentence 10
```

Output: (Key: Sentence, Value: Article ID)

```
(sentence 1, Article 1)
(sentence 2, Article 1)
(sentence 3, Article 1)
(sentence 4, Article 1)
(sentence 4, Article 2)
(sentence 5, Article 2)
(sentence 6, Article 2)
(sentence 7, Article 2)
(sentence 8, Article 3)
(sentence 9, Article 3)
(sentence 2, Article 3)
(sentence 4, Article 3)
(sentence 1, Article 4)
(sentence 4, Article 4)
(sentence 3, Article 4)
(sentence 10, Article 4)
```

REDUCER 1 - GROUP ARTICLES BY SENTENCE

The reducer groups sentences as keys and lists all articles where they appear.

1. Input: Sentence as key, articles Id that contain the sentence(same as output of mapper 1).
2. Output: Sentence as key, list of articles that contain the sentence.

Output

```
(sentence 1, [Article 1, Article 4])
(sentence 2, [Article 1, Article 3])
(sentence 3, [Article 1, Article 4])
(sentence 4, [Article 1, Article 2, Article 3, Article 4])
(sentence 5, [Article 2])
(sentence 6, [Article 2])
(sentence 7, [Article 2])
(sentence 8, [Article 3])
```

```
(sentence 9, [Article 3])
(sentence 10, [Article 4])
```

MAPPER 2 - EMITTING ARTICLE PAIRS WITH EACH COMMON SENTENCE

1. We generate all pairs of articles sharing the same sentence.
2. We emit pairs of articles as keys and 1 as the value (indicating one common sentence).
    a. Input: (Sentence, [List of Articles])
    b. Key: Pairs of articles.
    c. Value: 1 (indicating one common sentence).

Output

```
(Article 1, Article 4)   1  # sentence 1
(Article 1, Article 3)   1  # sentence 2
(Article 1, Article 4)   1  # sentence 3
(Article 1, Article 2)   1  # sentence 4
(Article 1, Article 3)   1  # sentence 4
(Article 1, Article 4)   1  # sentence 4
(Article 2, Article 3)   1  # sentence 4
(Article 2, Article 4)   1  # sentence 4
(Article 3, Article 4)   1  # sentence 4
```

REDUCER 2 - COUNTING SHARED SENTENCES FOR EACH ARTICLE PAIR

1. The reducer sums up values for each article pair.
    a. Input: Pairs of articles, Values (Same as output of mapper 2)
    b. Operation: Sum up occurrences for each pair.
    c. Output: (Article Pair, Number of Matching Sentences)

Output (Final Output)

```
(Article 1, Article 4)   3  # sentence 1, sentence 3, sentence 4
(Article 1, Article 3)   2  # sentence 2, sentence 4
(Article 1, Article 2)   1  # sentence 4
(Article 2, Article 3)   1  # sentence 4
(Article 2, Article 4)   1  # sentence 4
(Article 3, Article 4)   1  # sentence 4
```

Final Insights

1. Article 1 and Article 4 have 3 sentences in common → highest similarity.
2. Article 1 and Article 3 have 2 sentences in common.
3. Other article pairs have only 1 sentence in common.