COMP9313 2023T2 Final Exam

The deadline for the final exam is:

Tuesday 22nd, August 12:00 pm (AEST)

Please submit your answers through Moodle. Do not wait till the last minute to submit and double check if it is successful. Please send emails to xin.cao@unsw.edu.au and siqing.li@unsw.edu.au if you have any questions.

Question 1. Concepts (4 marks)

(a) (2 marks) Please briefly explain why the order inversion and value-to-key design patterns applied in Project 1 can improve the performance on MapReduce.

(b) (1 marks) In Project 2 we need to filter out the stop words. Please describe the most efficient way and explain the reason.

(c) (1 marks) In Project 3, a student complained that her approach took a lot of time at the last step when writing data to disk, but all the previous operations such as reading the data by textFile(), filtering the data by filter(), and transforming the data by map() and flatmap() are very fast. Could you please explain the reason to her?

Question 2. MapReduce Programming (14 marks)

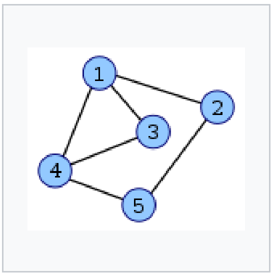
Requirement: *You should explain how the input is mapped into (key, value) pairs by the map stage, i.e., specify what is the key and what is the associated value in each pair, and how the key(s) and value(s) are computed. Then you should explain how the (key, value) pairs produced by the map stage are processed by the reduce stage to get the final answer(s). You only need to provide the pseudo code for the classes including Mapper and Reducer (optionally Combiner etc.if necessary, and the* ***efficiency*** *of your method will be considered).*

(a) (5 marks) Given a large news articles dataset from several years (year as the key and news content as the value), use MapReduce to compute the longest length of terms for each year. You can assume that the terms are separated by a single space character. Note that one term could be split into two lines (the end of the first line and the beginning of the second line).

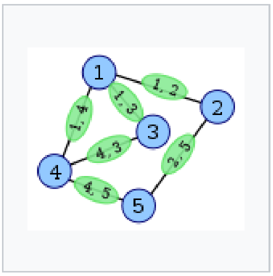
(b) (9 marks) **Problem Background:** Given an undirected graph G, its “line graph” is another graph L(G) that represents the adjacencies between edges of G, such that:

- each vertex of L(G) represents an edge of G; and
- two vertices of L(G) are adjacent if and only if their corresponding edges share a common endpoint ("are incident") in G.

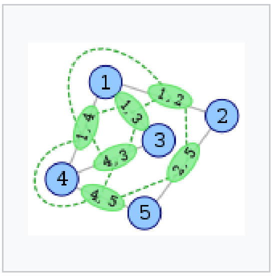
The following figures show a graph (left) and its line graph (right). Each vertex of the line graph is shown labelled with the pair of endpoints of the corresponding edge in the original graph. For instance, the vertex on the right labelled (1,3) corresponds to the edge on the left between the vertices 1 and 3. Vertex (1,3) is adjacent to three other vertices: (1,2) and (1,4) (corresponding to edges sharing the endpoint 1 in G) and (3,4) (corresponding to an edge sharing the endpoint 3 in G). Note that the vertex (4, 3) in the below example should be (3, 4) in the output.



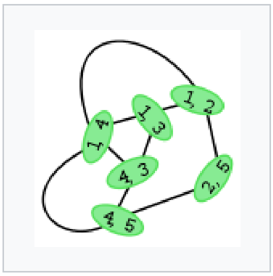
Graph G



Vertices in L(G) constructed from edges in G



Added edges in L(G)



The line graph L(G)

Problem: Given you the adjacency list of an undirected graph G, use MapReduce to generate the adjacency list of its line graph L(G). Note that each edge connecting two nodes i and j is represented by (i, j) in L(G) (if i<j). In the output, the edges in each list should be ranked in ascending order by comparing the first node and then the second node. The adjacency lists should be ranked by the keys according to the same order as well. Take the above figure as an example, sample input and output are as below:

Input:	Output:
1: 2, 3, 4 2: 1, 5 3: 1, 4 4: 1, 3, 5 5: 2, 4	(1, 2): (1, 3), (1, 4), (2, 5) (1, 3): (1, 2), (1, 4), (3, 4) (1, 4): (1, 2), (1, 3), (3, 4), (4, 5) (2, 5): (1, 2), (4, 5) (3, 4): (1, 3), (1, 4), (4, 5) (4, 5): (1, 4), (2, 5), (3, 4)

Question 3. Spark Programming (14 marks)

Provide the PySpark code for the given problems (minor errors are acceptable).

(a) (7 marks) **RDD programming (DataFrame APIs not allowed):** You are given a dataset of sensor readings collected over time. Each record in the dataset consists of a timestamp, a sensor ID, and a numerical sensor reading value. Your task is to implement a Spark RDD program to detect anomalies in the sensor readings based on the following steps:

1. Calculate the average sensor reading value for each sensor.
2. For each sensor reading, calculate the difference between the reading and the average reading for that sensor.
3. Report the reading that is the most different from the average reading for that sensor (could be not unique).

The output should contain three fields: the timestamp, the sensor ID, and the gap between the reading and the average reading. The result should be sorted by the sensor ID alphabetically first and then by the timestamp in ascending order. The sample input and output are as follows:

Input:	Output:
T1 SensorA 10.5 T2 SensorB 25.0 T3 SensorA 12.0 T4 SensorB 25.5 T5 SensorA 100.0 T6 SensorB 26.0	T5 SensorA 59.1667 T2 SensorB 0.5 T6 SensorB 0.5

(b) (7 marks) **DataFrame programming (RDD APIs not allowed):** Given a set of prices of items from different shops (the input format is as shown in the left column), the task is to: find out the shop that has the highest price for each item and sort the result by the item ID in alphabetical order (the format is as shown in the right column). If there exist multiple shops having the same highest price, only report the one with the smallest shop ID.

Input:	Output:
shop1:item1,90;item2,92;item3,80;item4,79;item5,93 shop2:item1,92;item2,77;item5,85 shop3:item3,64;item4,97;item5,93	item1:shop2,92 item2:shop1,92 item3:shop1,80 item4:shop3,97 item5:shop1,93

Question 4. Finding Similar Items (6 marks)

Given three documents A = ("the data is big the value is high") and B = ("the value in the data is high"), using the ***WORDS*** as tokens, answer the following questions.

(i) (2 marks) compute the 2-shingles for A and B, and then compute their Jaccard similarity based on their 2-shingles.

(ii) (4 marks) We want to compute min-hash signature for the two documents A and B given in Question (i), using three pseudo-random permutations of columns defined by the following functions:

h1(n) = (2n + 1) mod M

h2(n) = (7n + 2) mod M

h3(n) = (5n + 3) mod M

Here, n is the row number in original ordering of the 2-shingles (**according to their occurrences in A then B**), and M is the number of all 2-shingles you have computed from A and B. Instead of explicitly reordering the columns for each hash function, we use the implementation discussed in class. Complete the steps of the algorithm and give the resulting signatures for A and B.

Question 5. Mining Data Streams (6 marks)

(a) (3 marks) Counting Bloom Filter

Consider a Counting Bloom filter of size m = 9 and 2 hash functions that both take a string (lowercase) as input:

h1(str) = $\sum_{c\ in\ str}(c - 'a')$ mod 9

h2(str) = (str.length * 2) mod 9

Here, c - ‘a’ is used to compute the position of the letter c in the 26 alphabetical letters, e.g., h1(“bd”) = (1 + 3) mod 9 = 4.

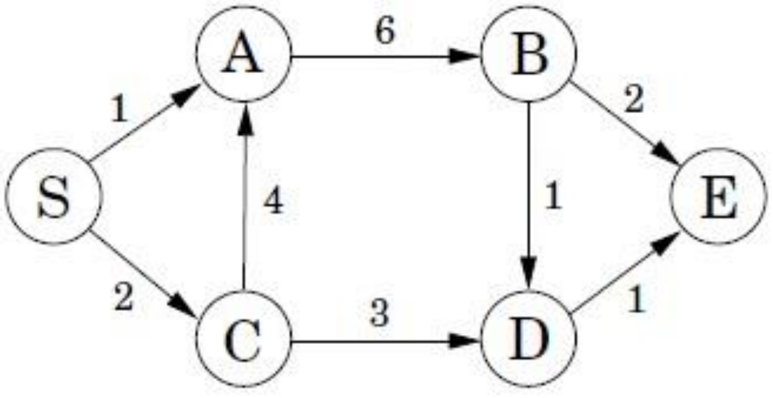
(i) (2 marks) Given a set of string S = {“hi”, “big”, “data”, “spark”}, show the update of the Bloom filter

(ii) (1 mark) Delete “big” from S, and use the bloom filter to check if “nosql” is contained in S.

(b) (3 marks) Finding Frequent Elements

For the heavy hitter problem, given a sequence [1,1,2,3,4,5,1,1,1,5,3,3,1,1,2] and k=3, we want to find element that occurred more than 15/3 = 5 times. Please use the Space-Saving algorithm to find out the approximate results.

Question 6. Graph Data Management (6 marks)



(a) (4 marks) Given the above graph, show the output of mapper (sorted results of all mappers) and reducer (assume only one reducer) in each iteration during the computation of the shortest path from node S to node E (You can follow the example in lecture 8.1 and show only the updates in the output).

(b) (2 marks) Given the above graph (ignore the edge weights), show the matrix for computing the PageRank vector, assuming β = 0.8 (teleport probability = 1- β).

	S	A	B	C	D	E
S						
A						
B						
C						
D						
E						

End Of Paper