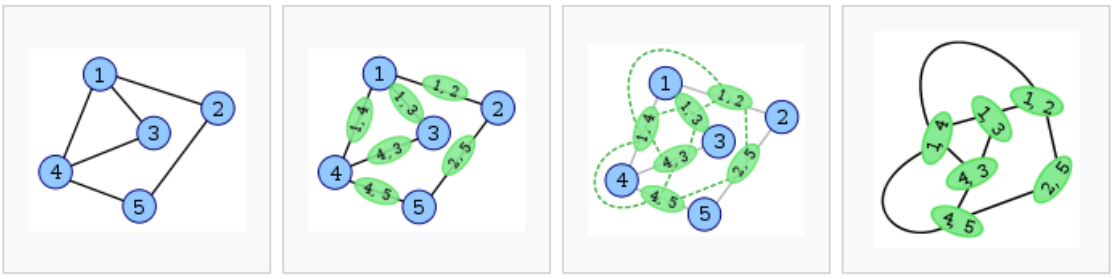
COMP9313 2018s2 Assignment

Question 1. MapReduce (5 pts)

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).



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. Write the pseudocode for this problem, and consider the efficiency of your solution.

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)

Answer: Use secondary sort

```
class VertexPair
        vertex1, vertex2
        VertexPair (v1, v2)
                if(v1 < v2)
                        vertex1 = v1, vertex2 = v2
                else
                        vertex1 = v2, vertex2 = v1

        compareTo(Pair p)
                ret = this.vertex1 < p.getVertex1
                if(ret == 0) ret = this.vertex2 < p.getVertex2
                return ret

class EdgePair
        edge1, edge2
        compareTo(Pair p)
                ret = this.edge1.compareTo(p.getEdge1)
                if(ret == 0) ret = this.edge2.compareTo(p.getEdge2)
                return ret

class Mapper
        method Map(vertexID, list of neighbours L)
                initialize a list edgeList
                foreach vertex v in L
                        L.add( new VertexPair(vertexID, v) )
                foreach pair p1 in edgeList
                        foreach pair p2 in edgeList.remove(p1)
                                Emit( new EdgePair(p1, p2), p2 )

class Partitioner
        method int getPartition(key, value, numPartitions)
                return key.getEdge1.hashCode() & Integer.MAX_VALUE % numPartitions

Class PairGroupingComparator extends WritableComparator
        method int compare(WritableComparable wc1, WritableComparable wc2)
                return ((EdgePair) wc1).getEdge1().compareTo(((EdgePair)
wc1).getEdge1())

class Reducer
        method Reduce(key, pairList [])
                Emit(key.getEdge1(), pairList)
```

A combiner could be used to further improve the performance.

Question 2. LSH (5 pts)

(i) Given two documents A ("the sky is blue the sun is bright") and B ("the sun in the sky is bright"), using the ***words*** as tokens, compute the 2-shingles for A and B, and then compute their Jaccard similarity based on their 2-shingles.

Answer:

9 2-shingles: the sky, sky is, is blue, blue the, the sun, sun is, is bright, sun in, in the

Sim(A, B) = 4/9

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

$h1(n) = 5n - 1 \bmod M$

$h2(n) = 2n + 1 \bmod M$

Here, n is the row number in original ordering of the 2-shingles (according to their occurrence 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, in which we read each data in a column once in a sequential order and update the min hash signatures as we pass through them.

Complete the steps of the algorithm and give the resulting signatures for A and B.

Answer: M = 9

Row index	2-shingles	A	B
0	the sky	1	1
1	sky is	1	1
2	is blue	1	0
3	blue the	1	0
4	the sun	1	1
5	sun is	1	0
6	is bright	1	1
7	sun in	0	1
8	in the	0	1

Intially:

	h1	h2
A	∞	∞
B	∞	∞

0 – the sky:

	h1	h2
A	8	1

B	**8**	**1**

1 – sky is:

	h1	**h2**
A	**4**	**1**
B	**4**	**1**

2 – is blue:

	h1	**h2**
A	**0**	**1**
B	**4**	**1**

3 – blue the:

	h1	**h2**
A	**0**	**1**
B	**4**	**1**

4 – the sun:

	h1	**h2**
A	**0**	**0**
B	**1**	**0**

The following rows will not change the result.