1a]

Input to mapper – as provided in the inline example (src, tgt, weight)

Mapper splits each line of input using 'tab' as separator to create a string array, then picks up only the second and third elements (tgt, weight) of each array. 'weight' is parsed as an integer.

Output of mapper:

tgt	weight
51	1
51	1
51	3
151	51
151	79
130	10

Reducer aggregates by the key 'tgt' and returns tgt, max(weight)

Output of reducer:

tgt	weight
51	3
151	79
130	10

1b] Approach used: Breadth First Search algorithm

1. Initial MapReduce:
   Mapper reads input data (Source – Target tuples)
   Reducer: Finds list of unique nodes (1,2,3,4)
2. MapReduce
   Mapper: Reads list of unique nodes and input data (Source – Target tuples)
   'Emits messages' from a node to its in-neighbours
   message = (source node, number of hops)
   Reducer: Writes unique nodes and messages received, to an external resource:

Node	Messages
1	(2,1)
2	(3,1), (1,1), (4,1)
3	(2,1), (4,1)
4	

3. MapReduce

Mapper: Reads list of unique nodes and input data (Source – Target tuples)
'Propagates messages' from a node to its in-neighbours
message = (origin, number of hops++)
Reducer: Appends messages received, to the external resource:

Node	Messages
1	(2,1), (3,2), (4,2)
2	(3,1), (1,1), (4,1)
3	(2,1), (4,1), (1,2), (4,2)
4	

4. Repeat step 3, as needed: Since we want only up to 2-hop neighbours, step 3 is not repeated
5. Final MapReduce:
   Mapper: Reads pairs of nodes (Recipient, Origin) generated by Reducer in step 2
   Reducer: Eliminates cyclical node pairs (single-step two-way message exchange between the same two nodes – marked in red)

Message Recipient	Message Origin	Hops
1	2	1
1	3	2
1	4	2
2	3	1
2	1	1
2	4	1
3	2	1
3	4	1
3	1	2
3	4	1
3	1	2
3	4	2

Message Recipient	Message Origin
1	3
1	4
2	4
3	4
3	1
3	4
3	1
3	4