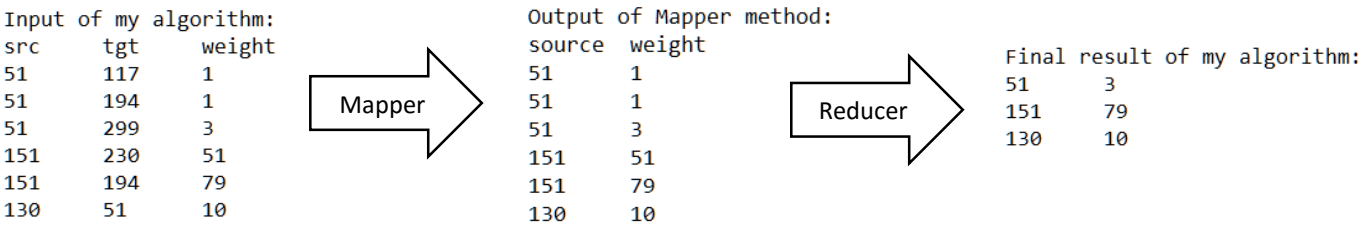
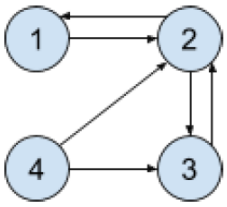
Description:

Part a:

1. In mapper procedure, the code splits the line into tokens separated by a tab. Then it will determine if the edge is null before it emits a key pair of <source, weight> using the second and third tokens in each line.
2. In reducer procedure, it just compares the list of values of each key to get the maximum result among the list of values.
3. Finally, it collects and outputs the key pair of <key, result>.



Part b:




```
Input of my algorithm:
src     tgt
4       3
1       2
2       3
4       2
2       1
3       2
```

Pseudo code

1. Read file

2. Extract the next value from the line

3. Set next source node as key

4. Set next target node as value

```
key Set
(4, 1, 2, 4, 2, 3)
value set
(3, 2, 3, 2, 1, 2)
```

5: class Mapper

6: *// for each i in the value set, determine if there is a key j+1 that equals to it.*

7: *// If the answer is yes, then collect the (j, i + 1) to context*

8: method Map (key, value, context)

9: for all i ∈ value do:

10: for all (j + 1) ∈ key do:

11: if (i == (j + 1)) then

12: context.collect (j, i + 1)

```
Output of Mapper method:
4       2
1       3
1       1
2       2
4       3
4       1
2       2
3       3
3       1
```

13: class Reducer

14: *// for each value in the value list, determine if it equals to its key.*

15: *// If the answer is no, then collect the (key, values) to context as final result.*

16: method Reduce (key, <list of values>, context)

17: for each value in <list of values>:

18: if (value != key) then

19: context.collect (key, value)

20: Output the final result

```
Output of Reducer method:
4       2
1       3
4       3
4       1
2       2
3       1
```