3.1 In-mapper Pair Approach

```
class Mapper
    method Initialize
        H = new associative array
    method Map(docid a; doc d)
        for all term w in doc d do
            for all term u in Neighbors(w) do
                H{pair(w,u)} = H{pair(w,u)} + 1

    method close
        for all pair(w,u) in H do
            emit(pair(w,u); count H{pair(w,u)})
```

```
Class Reducer

    method Reduce(pair(u,v); counts [c1, c2, ...])
        s = 0
        for all count c in counts [c1, c2, ...] do
            s = s + c

        Emit(pair(u,v), count s)
```

8.1 In-mapper Pair Approach Illustration

Let's say,

mapper-0 input

24, 31, 7, 81, 31

mapper-1 input

7, 20, 81

24 → 24 % 4 = 0
31 → 31 % 4 = 3
7 → 7 % 4 = 3
81 → 81 % 4 = 1
20 → 20 % 4 = 0

In pair approach, we make pairs of a term with its neighbors. Also, if we use in-mapper aggregation, then it will produce local aggregation of each pair. So, the mappers' output will be -

mapper-0 output

<(24, 31), 2>, <(24, 7), 1>, <(24, 81), 1>,
<(31, 7), 1>, <(31, 81), 1>, <(7, 81), 1>, <(7, 31), 1>
<(81, 31), 1>

mapper-1 output

<(7, 20), 1>, <(7, 81), 1>, <(20, 81), 1>

These outputs will be shuffled by using partition logic and sent to four reducers after performing groupBy operation.

Use: partition = (int) Key.hc() % r

So, we will have -

	R0	R1	R2	R3
Input	<(20,81), [1]> <(24,7), [1]> <(24,31), [1]> <(24,81), [1]>	<(81,31), [1]>		<(7,20), [1]> <(7,31), [1]> <(7,81), [1,1]> <(31,7), [1]> <(31,81), [1]>
Output	<(20,81), 1> <(24,7), 1> <(24,31), 1> <(24,81), 1>	<(81,31), 1>		<(7,20), 1>, <(7,31), 1>, <(7,81), 2>, <(31,7), 1> <(31,81), 1>