Extra Assignment: Yue, Shenghua

perform join in spark without built-in spark join method

```
In [1]: import pyspark
```

read input data

```
In [2]: input_path = "input.txt"
        records = spark.sparkContext.textFile(input_path)
        print(records.count())
        records.collect()

        8

Out[2]: ['A,x,v1',
         'A,x,v2',
         'A,y,v3',
         'A,y,v4',
         'A,z,v5',
         'B,x,v6',
         'B,y,v7',
         'B,p,v8']
```

split the record

```
In [3]: records_map = records.map(lambda x: x.split(","))
        print(records_map.count())
        records_map.collect()

        8

Out[3]: [['A', 'x', 'v1'],
         ['A', 'x', 'v2'],
         ['A', 'y', 'v3'],
         ['A', 'y', 'v4'],
         ['A', 'z', 'v5'],
         ['B', 'x', 'v6'],
         ['B', 'y', 'v7'],
         ['B', 'p', 'v8']]
```

construct (key, value) pairs

```
In  [4]: records_pairs = records_map.map(lambda x: (x[1], (x[0], x[2])))
         print(records_pairs.count())
         records_pairs.collect()

         8

 Out[4]: [('x', ('A', 'v1')),
          ('x', ('A', 'v2')),
          ('y', ('A', 'v3')),
          ('y', ('A', 'v4')),
          ('z', ('A', 'v5')),
          ('x', ('B', 'v6')),
          ('y', ('B', 'v7')),
          ('p', ('B', 'v8'))]
```

groupby key to get (key, iterable tuples)

```
In  [6]: grouped = records_pairs.groupByKey()
         grouped_map = grouped.map(lambda x: (x[0], list(x[1])))
         grouped_map.collect()

 Out[6]: [('y', [('A', 'v3'), ('A', 'v4'), ('B', 'v7')]),
          ('p', [('B', 'v8')]),
          ('x', [('A', 'v1'), ('A', 'v2'), ('B', 'v6')]),
          ('z', [('A', 'v5')])]
```

self-defined join function

```
In  [7]: def join(x):
             A, B = [], []
             for ele in x[1]:
                 if ele[0] == 'A':
                     A.append(ele[1])
                 if ele[0] == 'B':
                     B.append(ele[1])
             if not A or not B: return
             return [(x[0], (a, b)) for a in A for b in B]
```

use join function to map the RDD, followed by filter none value result and finally using flatMap to convert to unnest the value

```
In [11]: result = grouped_map.map(lambda x: join(x)).filter(lambda x: x).flatMap(lambda x: x)
         result.collect()

Out[11]: [('y', ('v3', 'v7')),
          ('y', ('v4', 'v7')),
          ('x', ('v1', 'v6')),
          ('x', ('v2', 'v6'))]

In  [ ]:
```