RDD的shuffle过程，非常昂贵：包含——磁盘IO,序列化、反序列化，网络IO

就是MapReduce的如下过程：

Map阶段输出：序列化，并写到本地， 即序列化和磁盘IO

传输阶段：reduce fetch数据时，要从其他节点拉取，则有网络IO

reduce阶段： 拉取数据，反序列化，按键排序，从本地读取（反序列化和网络IO）

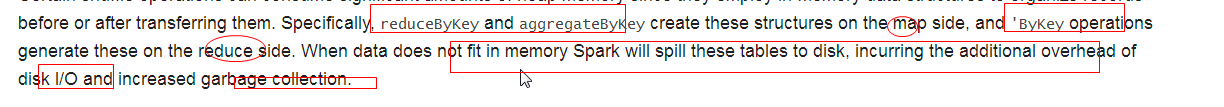


reduceByKey和AggregateByKey的好处：在map端执行，不用发生shuffle

而其他ByKey操作在reduce端执行

那在map端执行，就不用发生shuffle，也就不用reduce就计算出了。

由于shuffle会消耗大量内存，当内存不足时，就会：写磁盘-磁盘IO 、gc回收——应该是full gc，特别费时间。



shuffle会产生大量中间结果，存放到本地磁盘，目录有spark.local.dir指定



RDD的持续时间，采用LRU策略，保留最近的RDD， 但也可以手动去除，no waiting:

rdd.unpersist()

