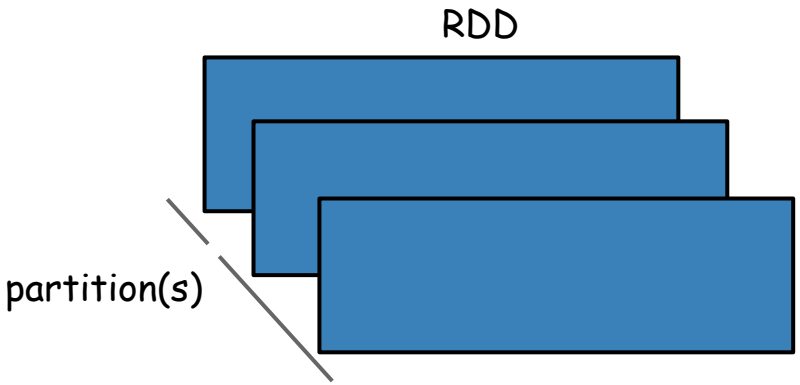
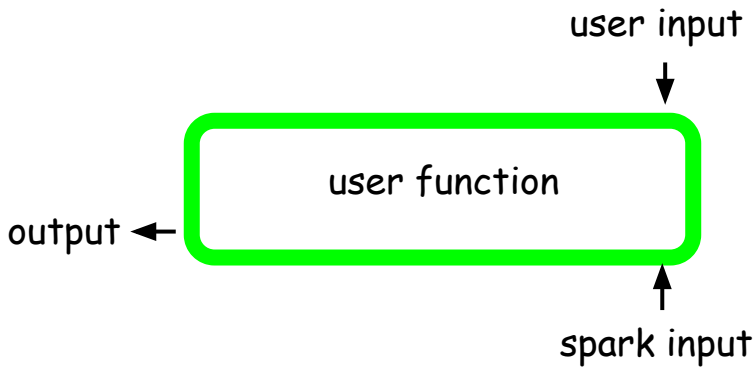
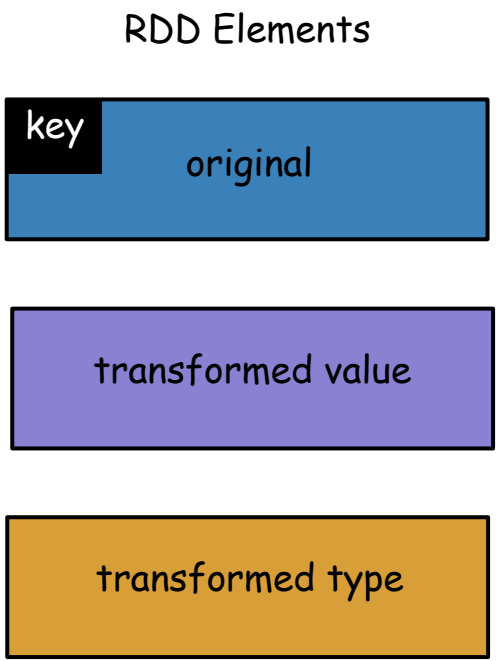
pySpark pictures

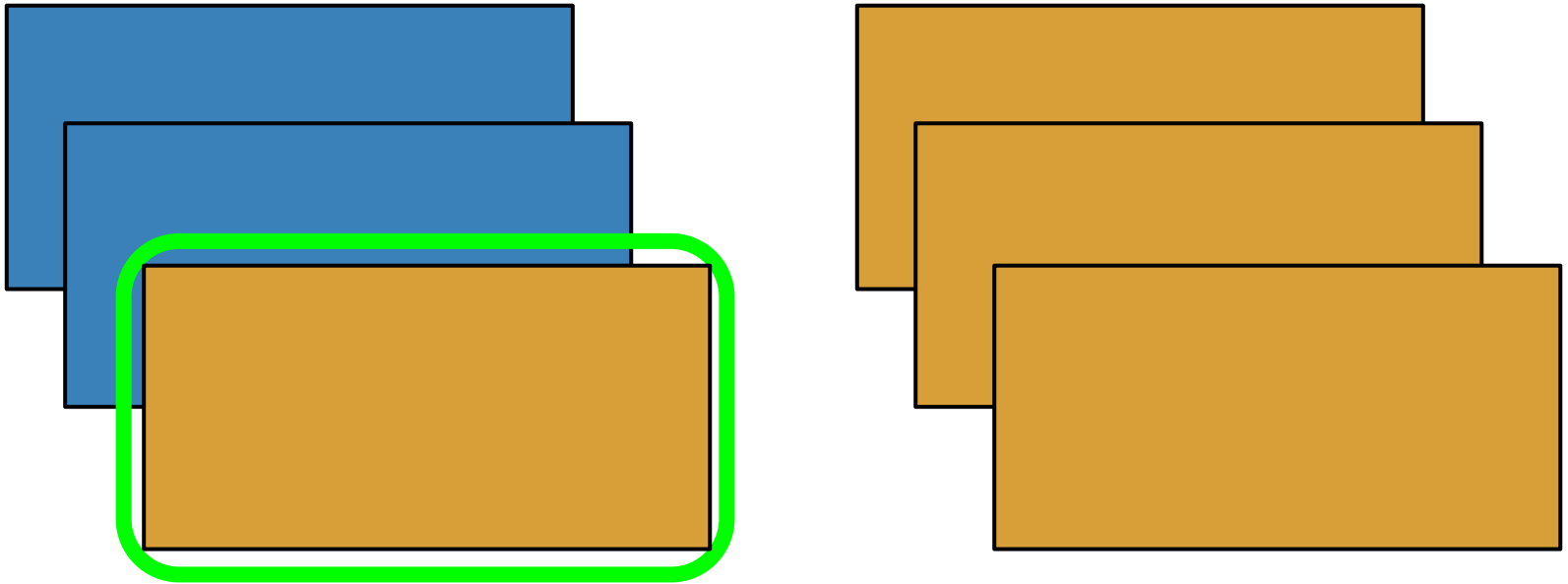
Learn the pySpark API through pictures


RDD
partition(s)

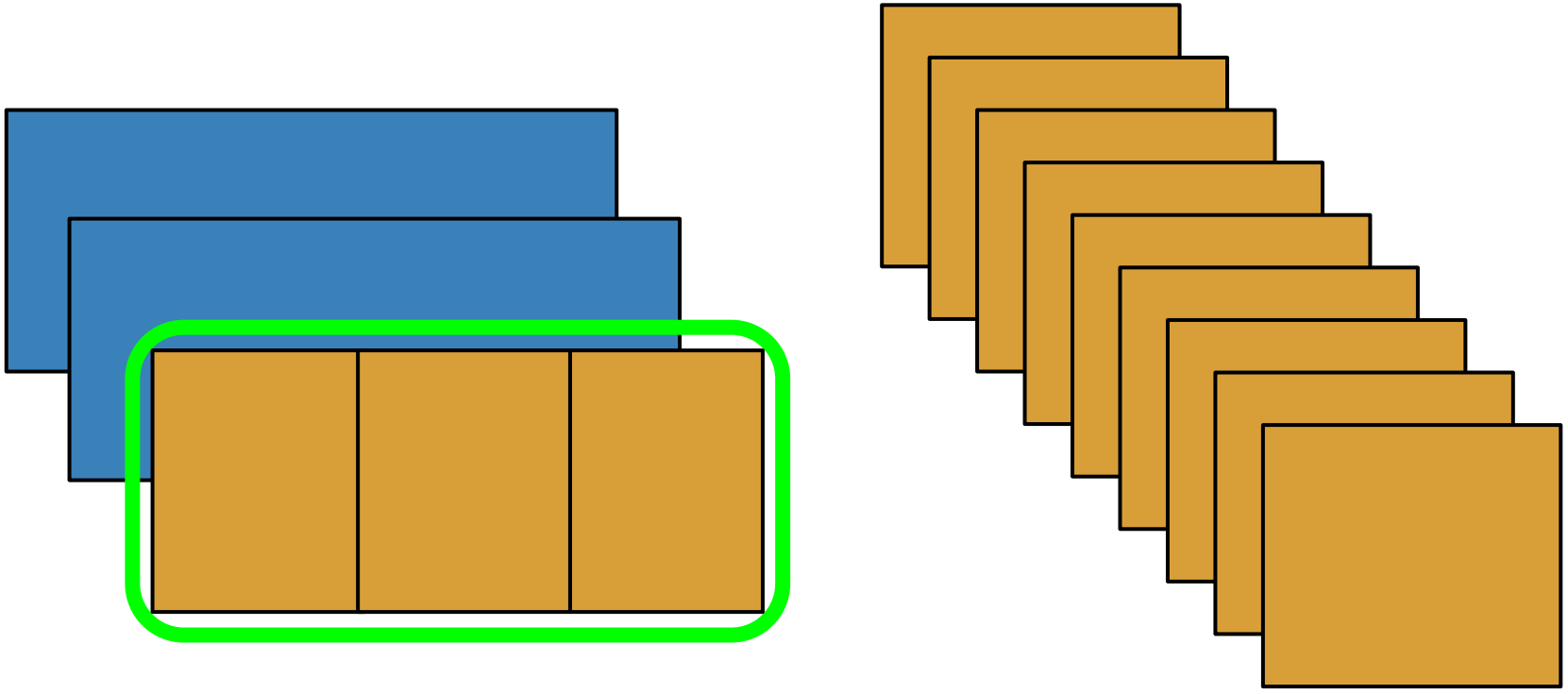

user input
user function
output
spark input


RDD Elements
key
original
transformed value
transformed type

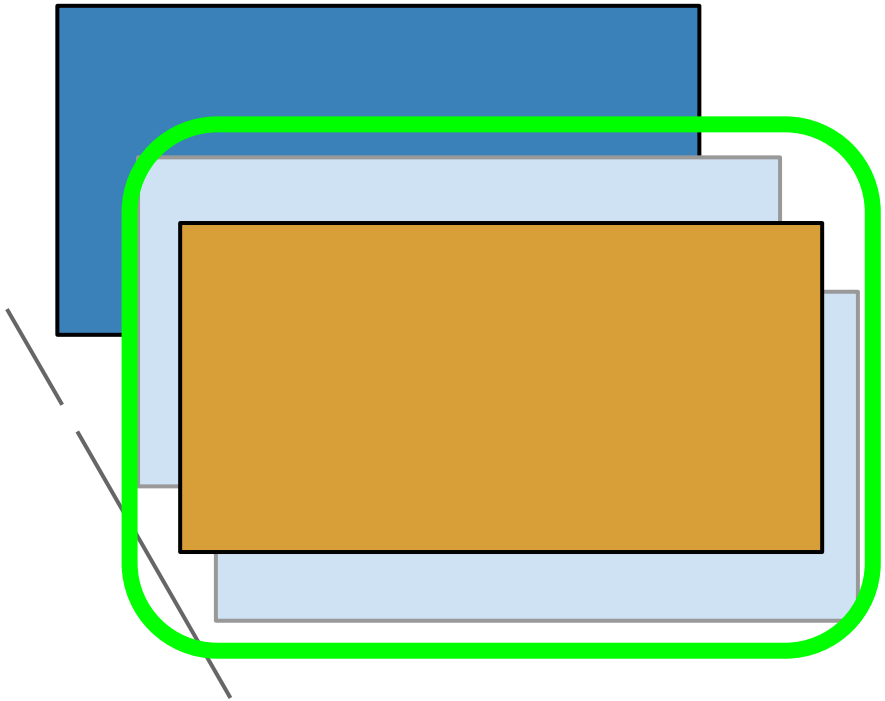
map

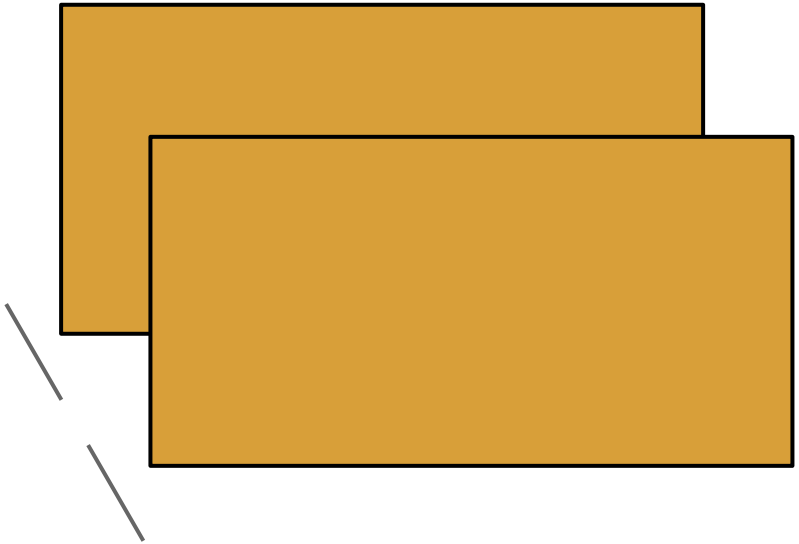


flatMap

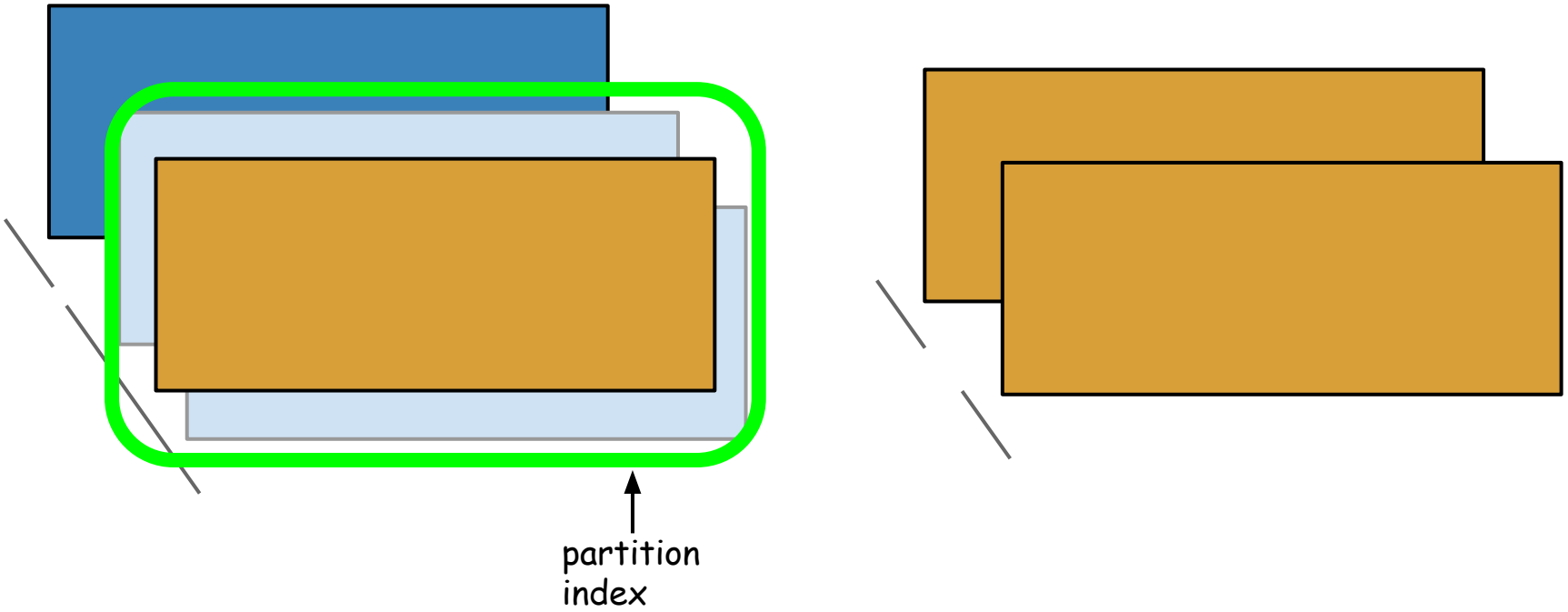


mapPartitions

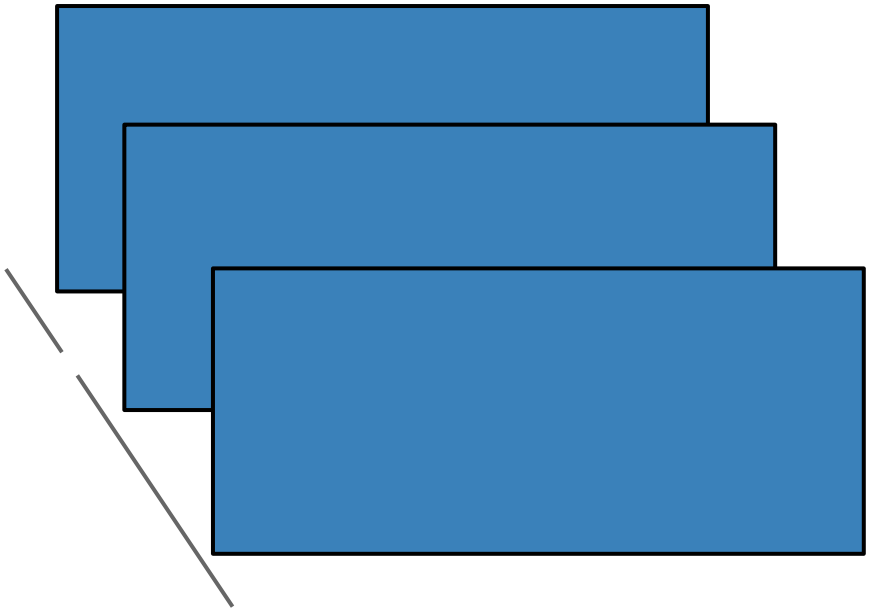




mapPartitionsWithIndex

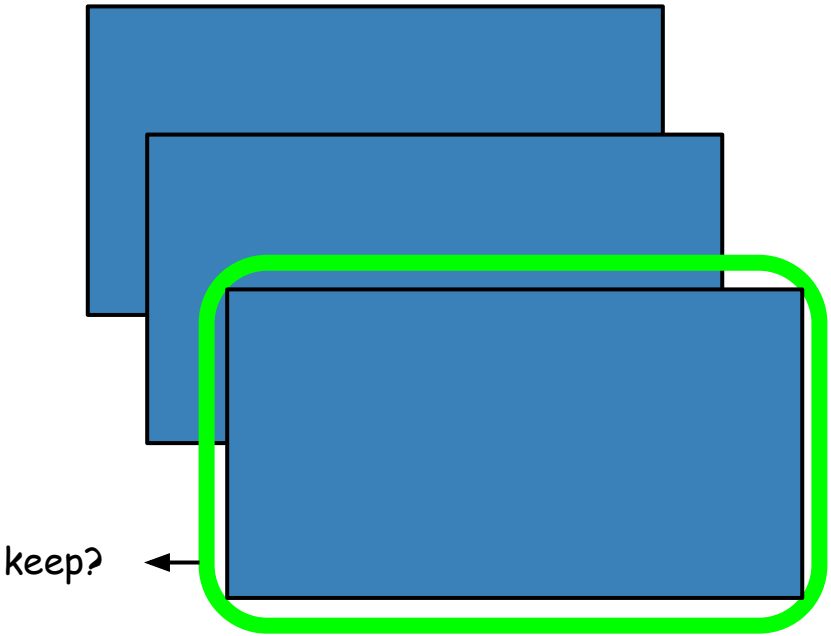


getNumPartitions

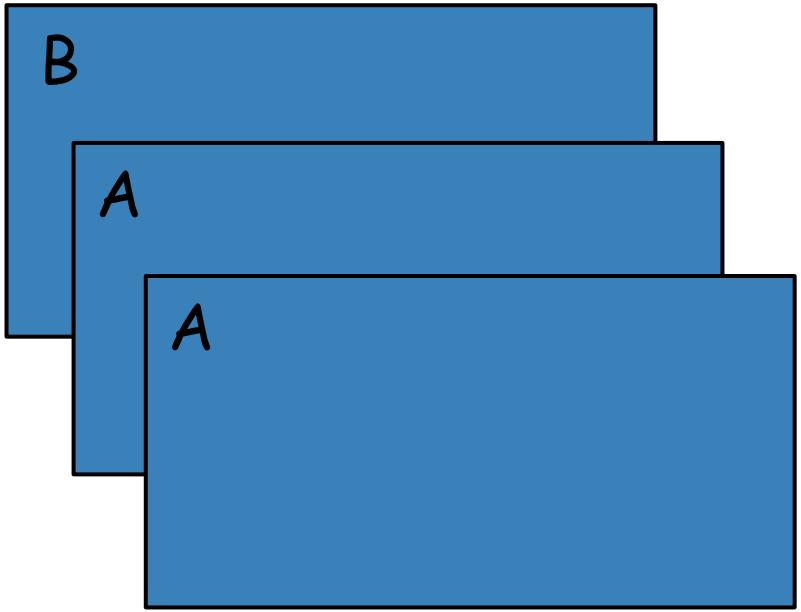


2

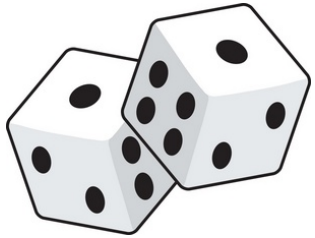
filter

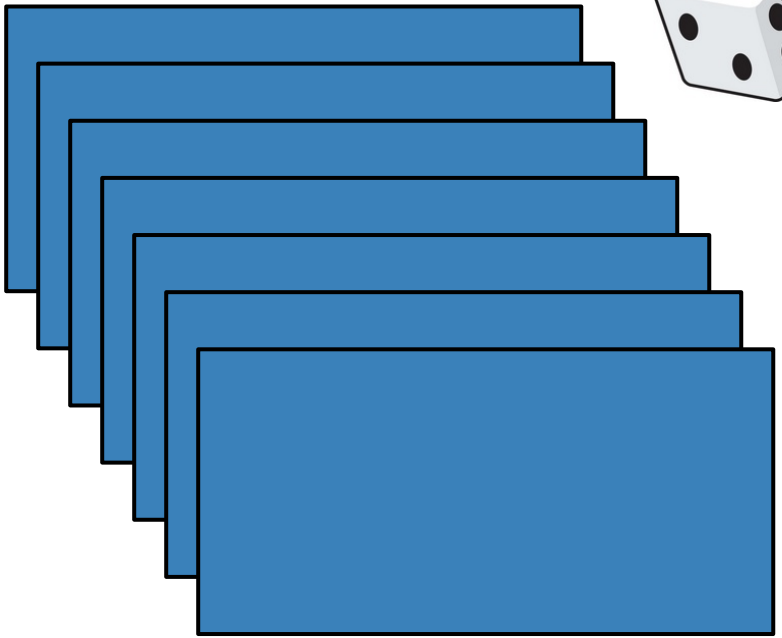


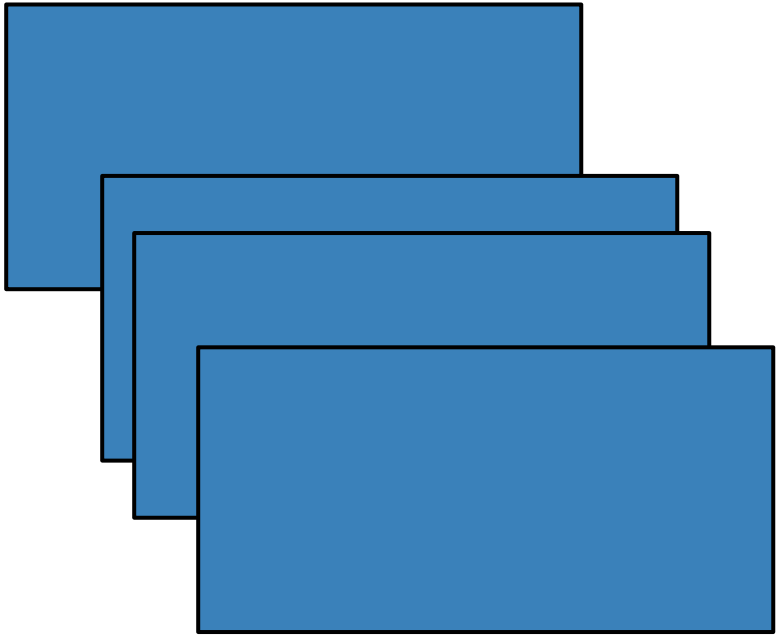
distinct



sample

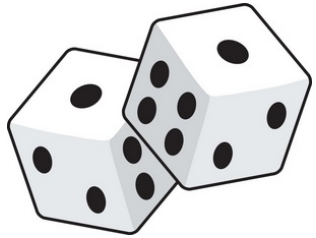


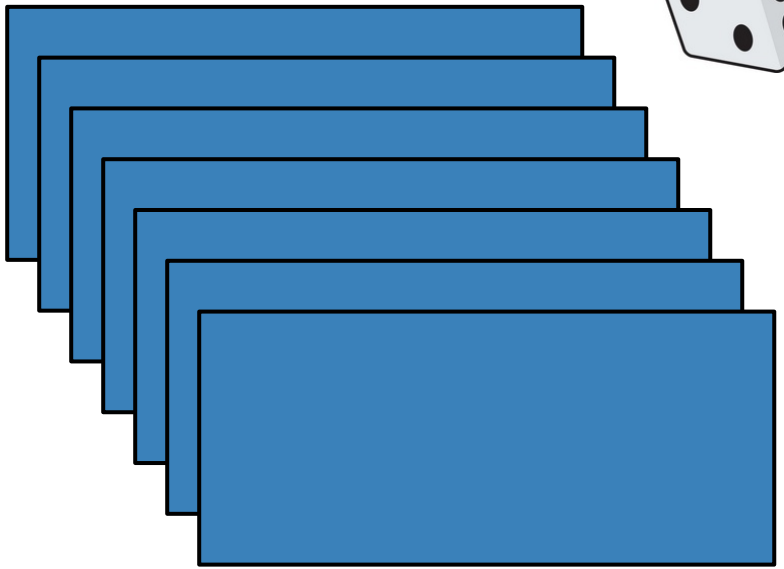


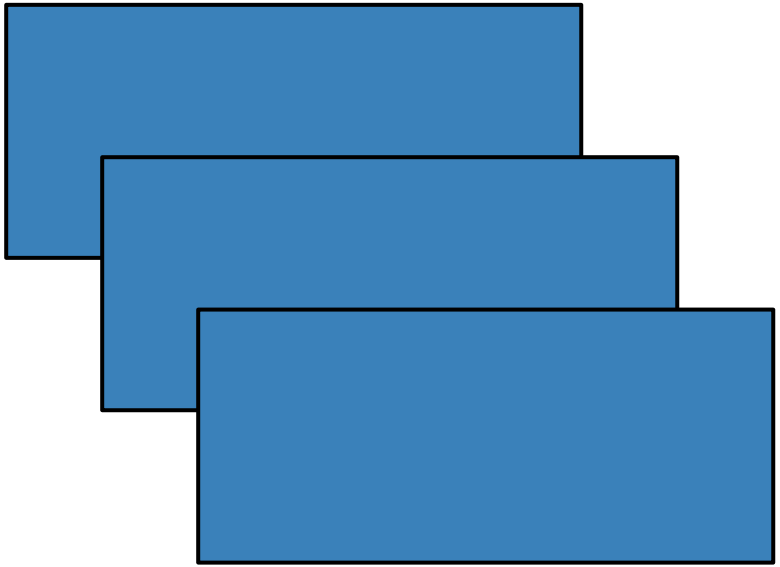


takeSample

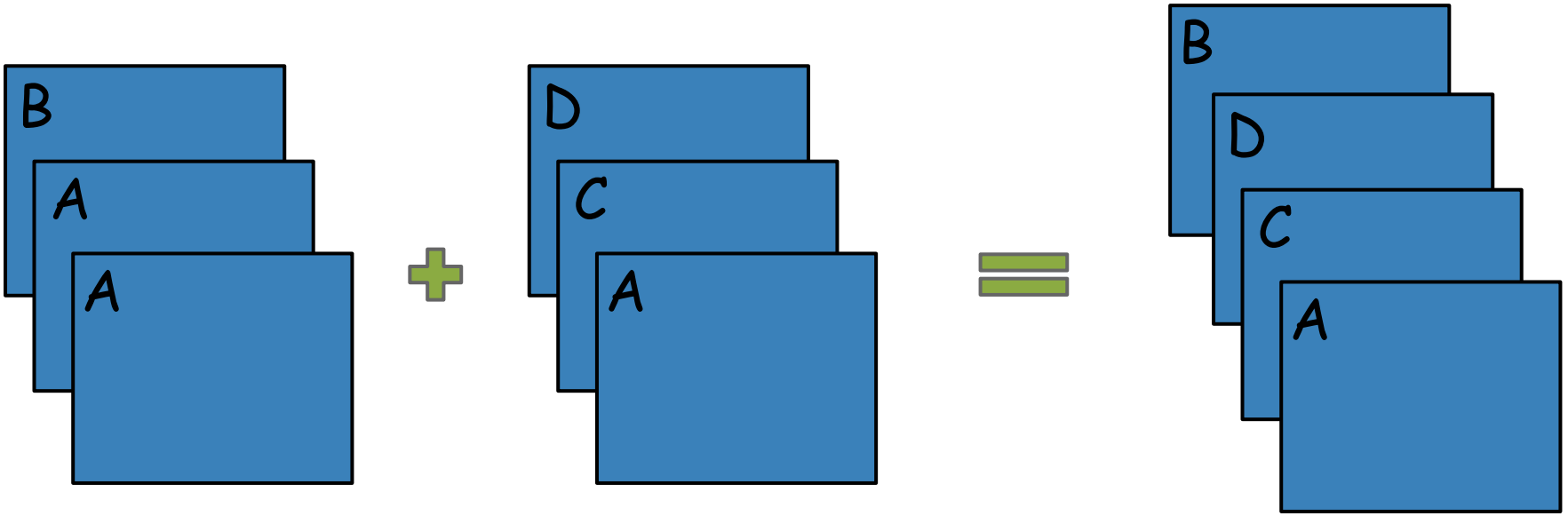
num = 3



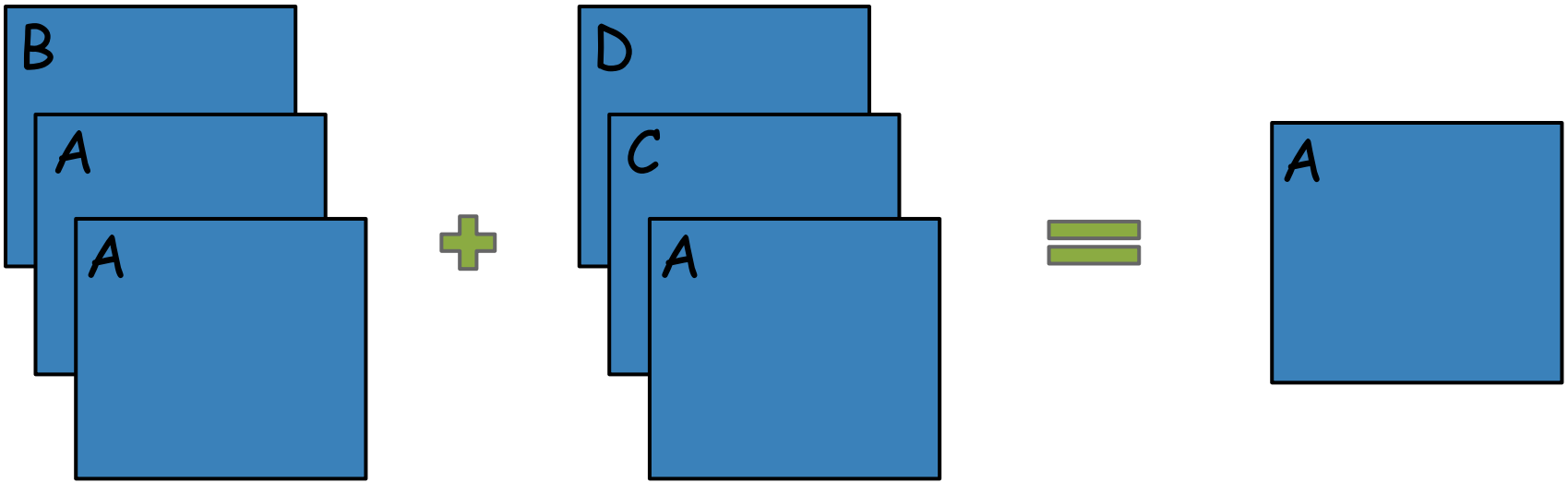




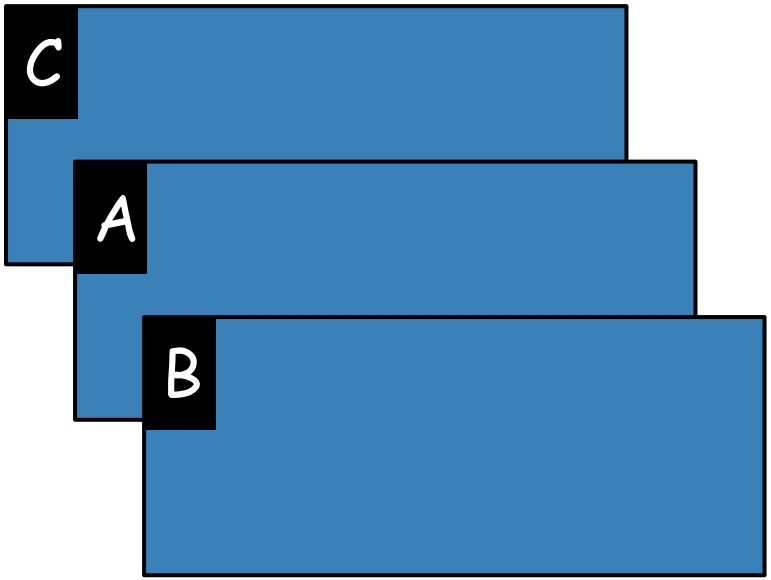
union (+)

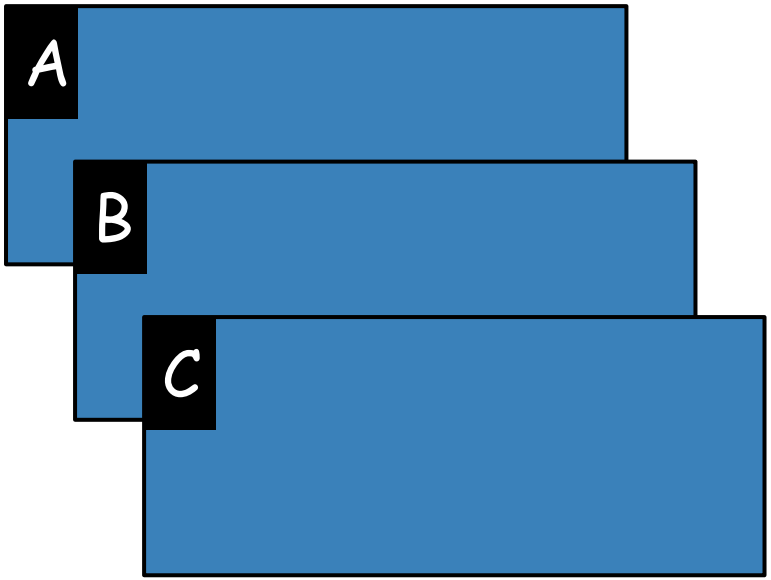


intersection

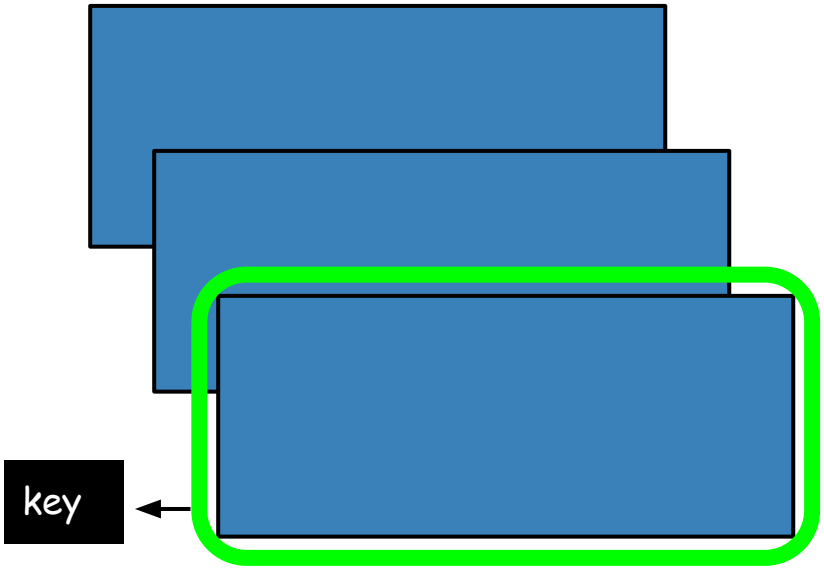


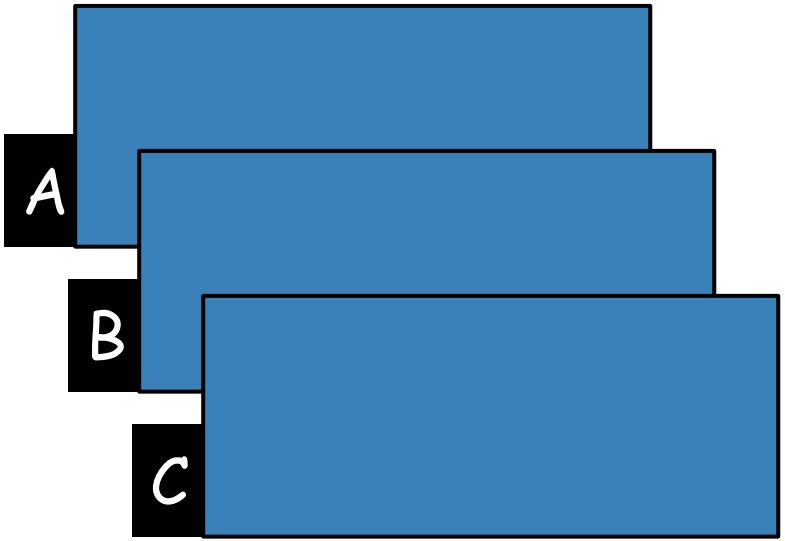
sortByKey



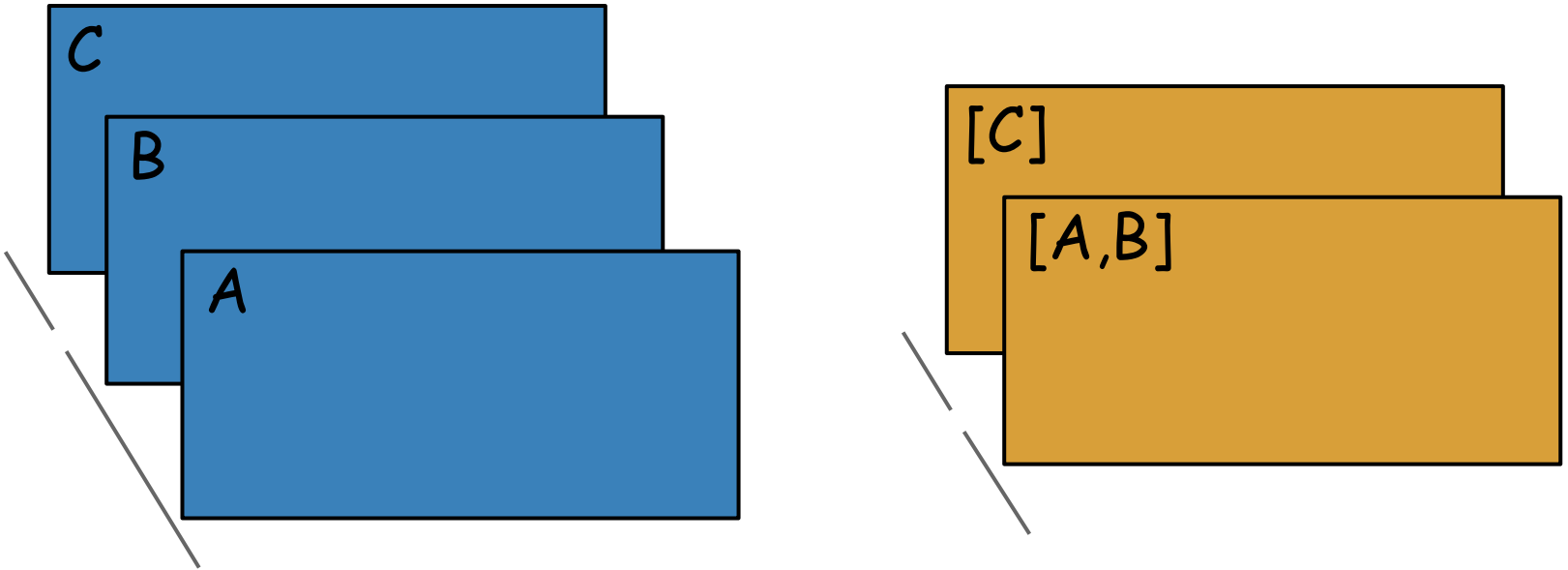


sortBy

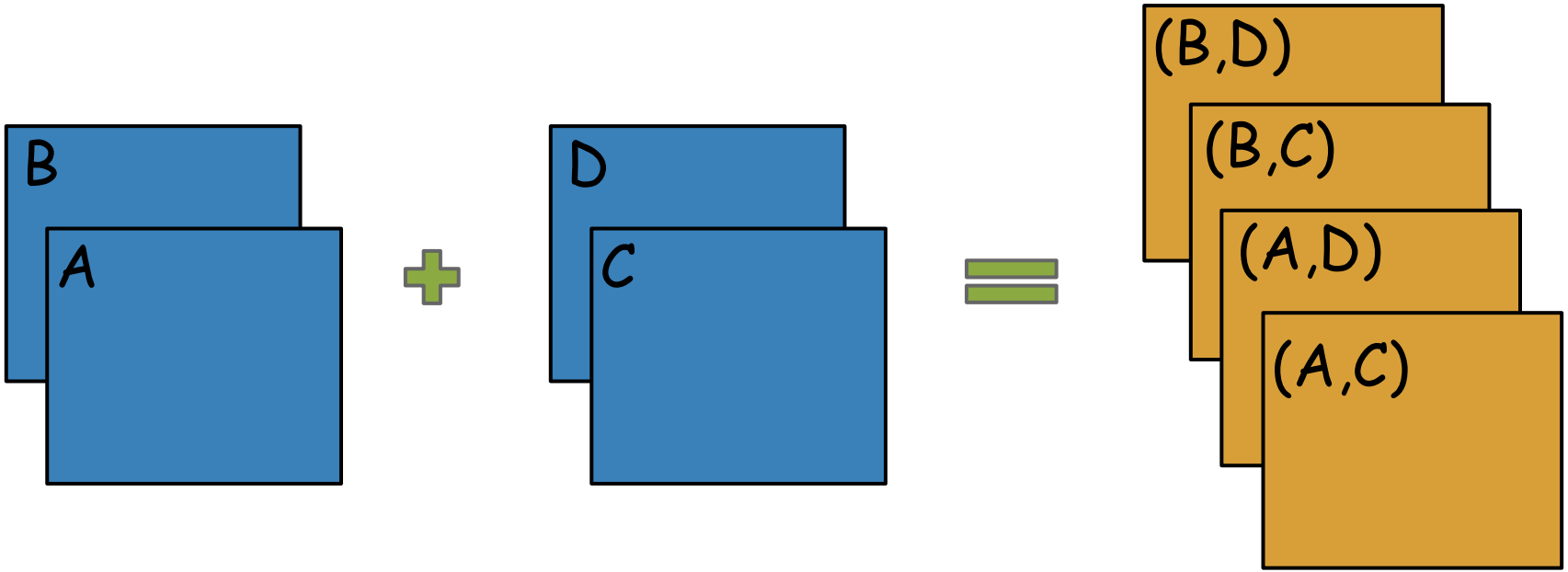




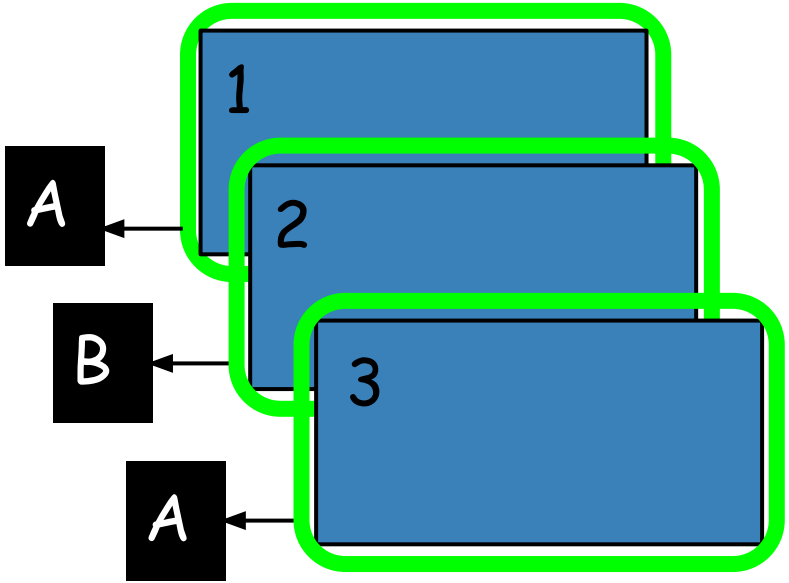
glom



cartesian



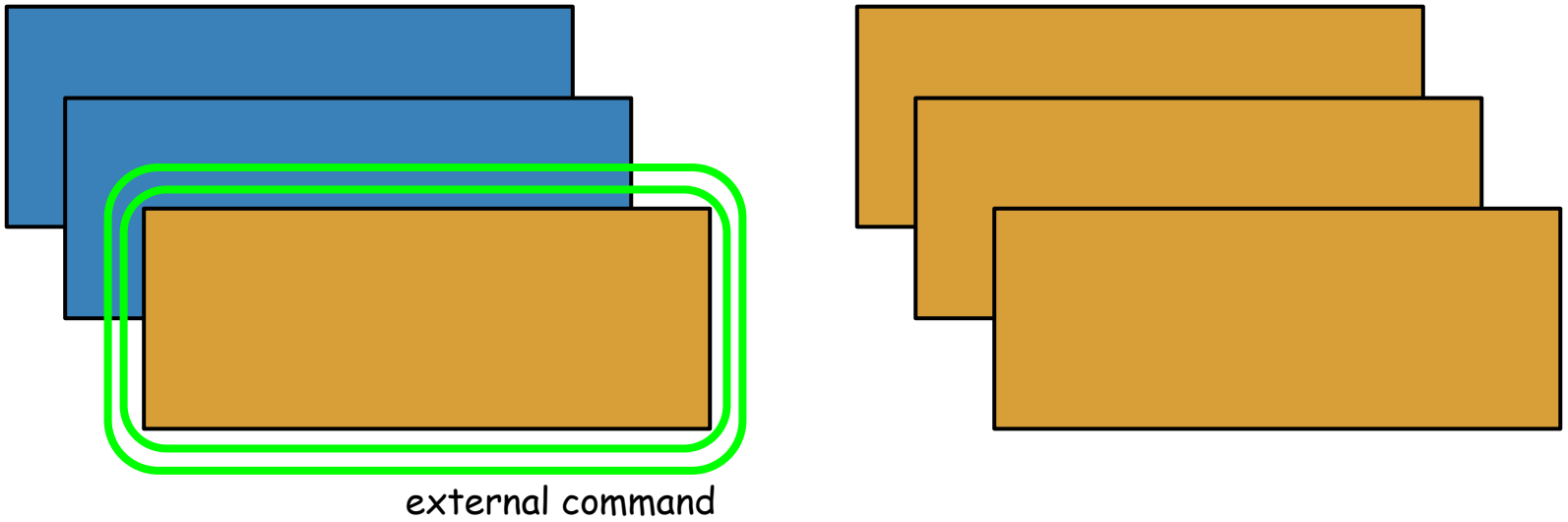
groupBy



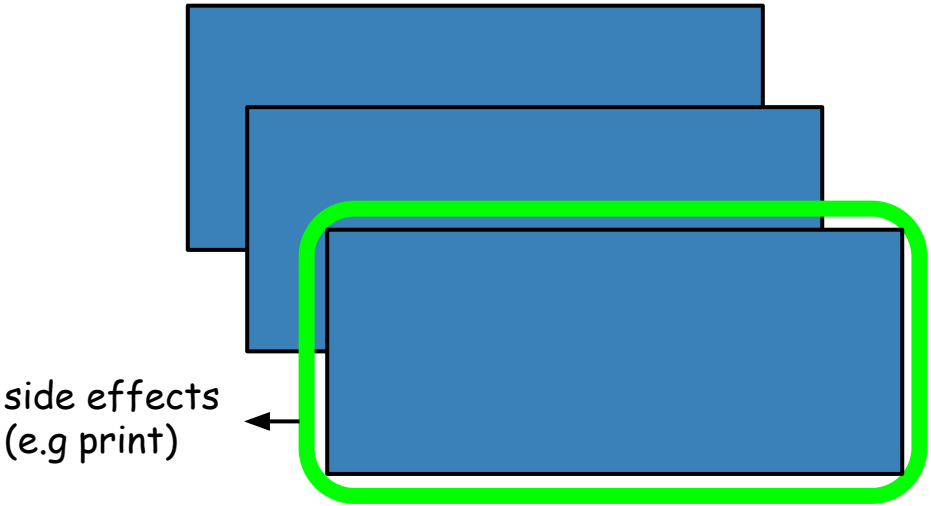
(B,[2])

(A,[1,3])

pipe

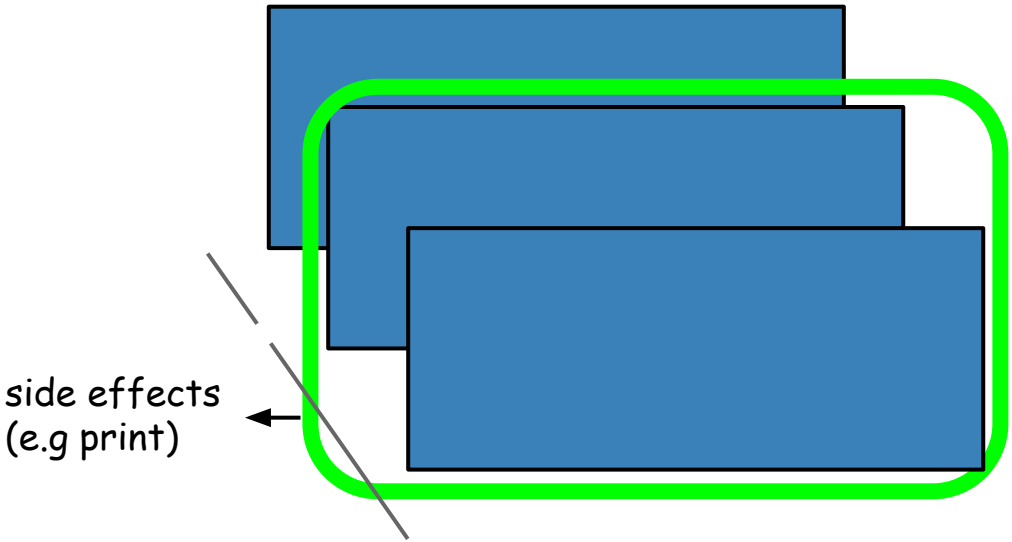


foreach



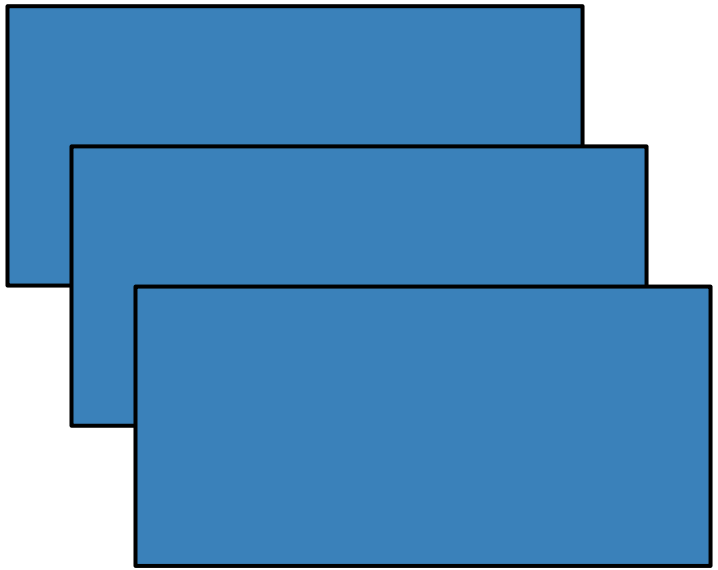
*no return value,
original RDD unchanged

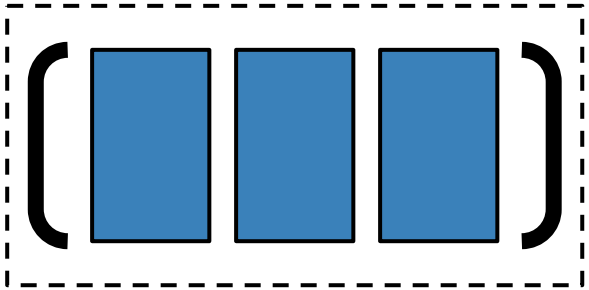
foreachPartition



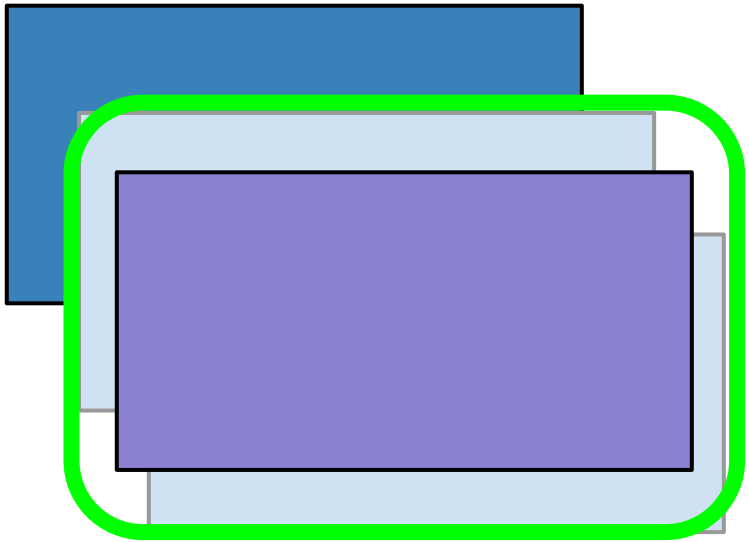
*no return value,
original RDD unchanged

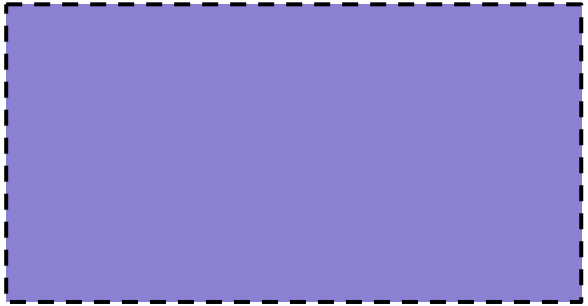
collect



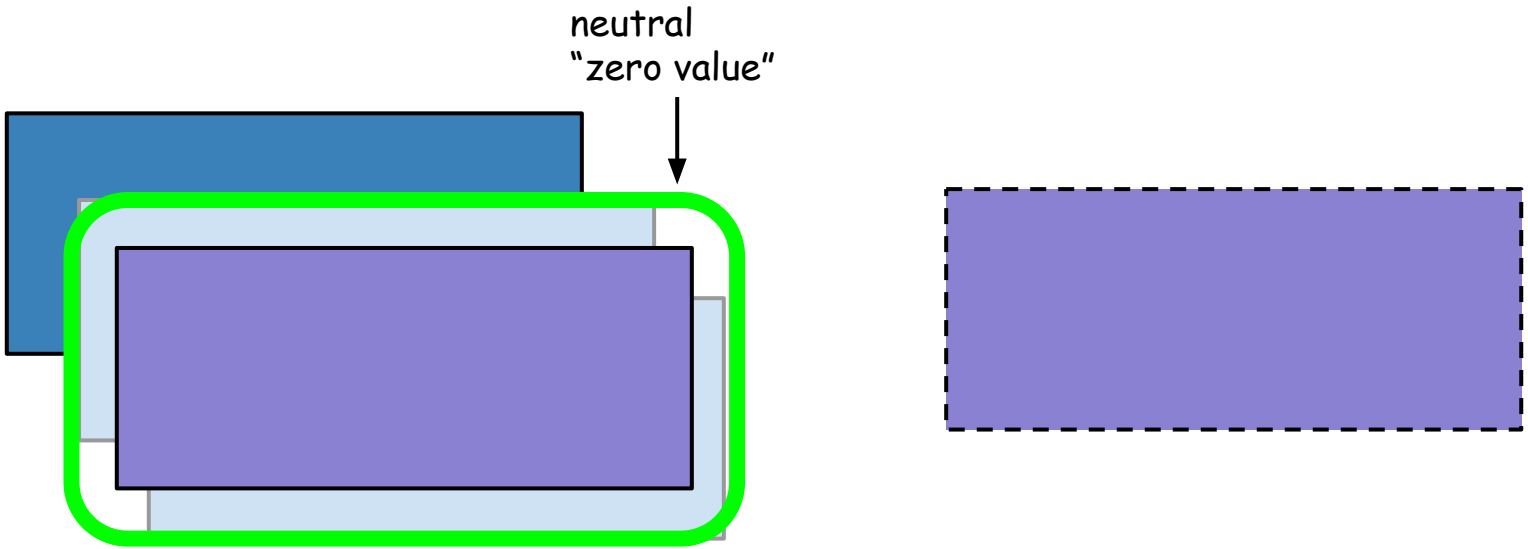


reduce

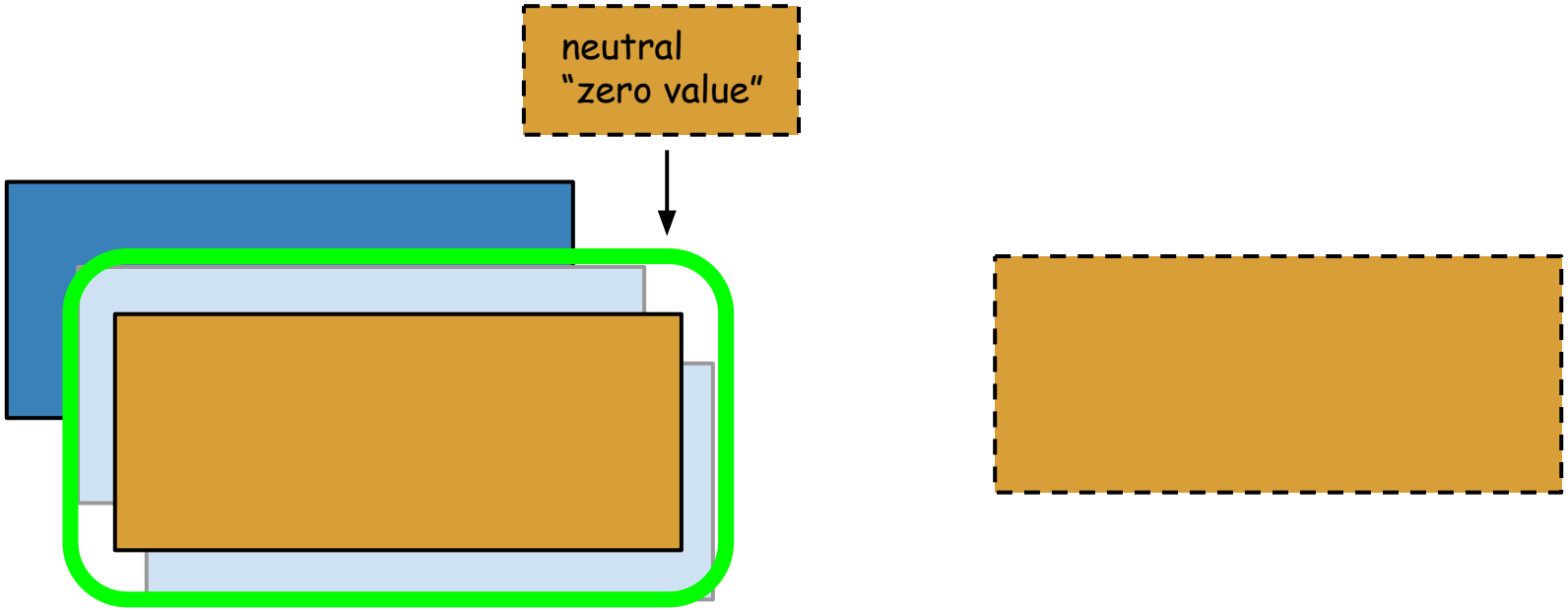




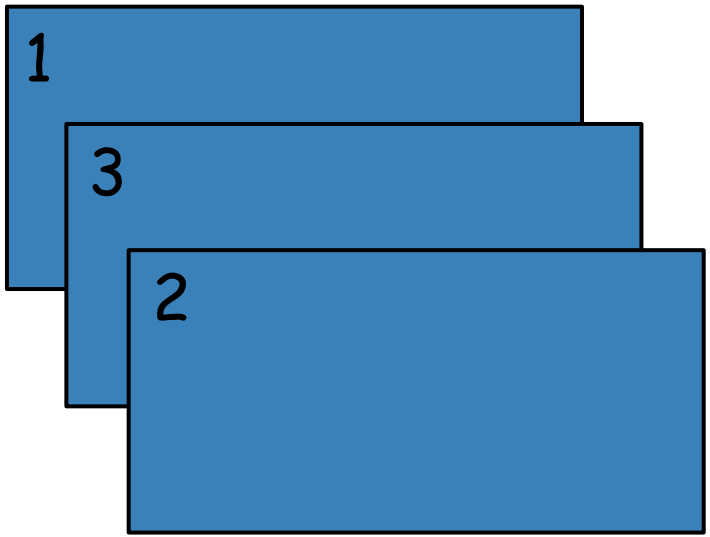
fold



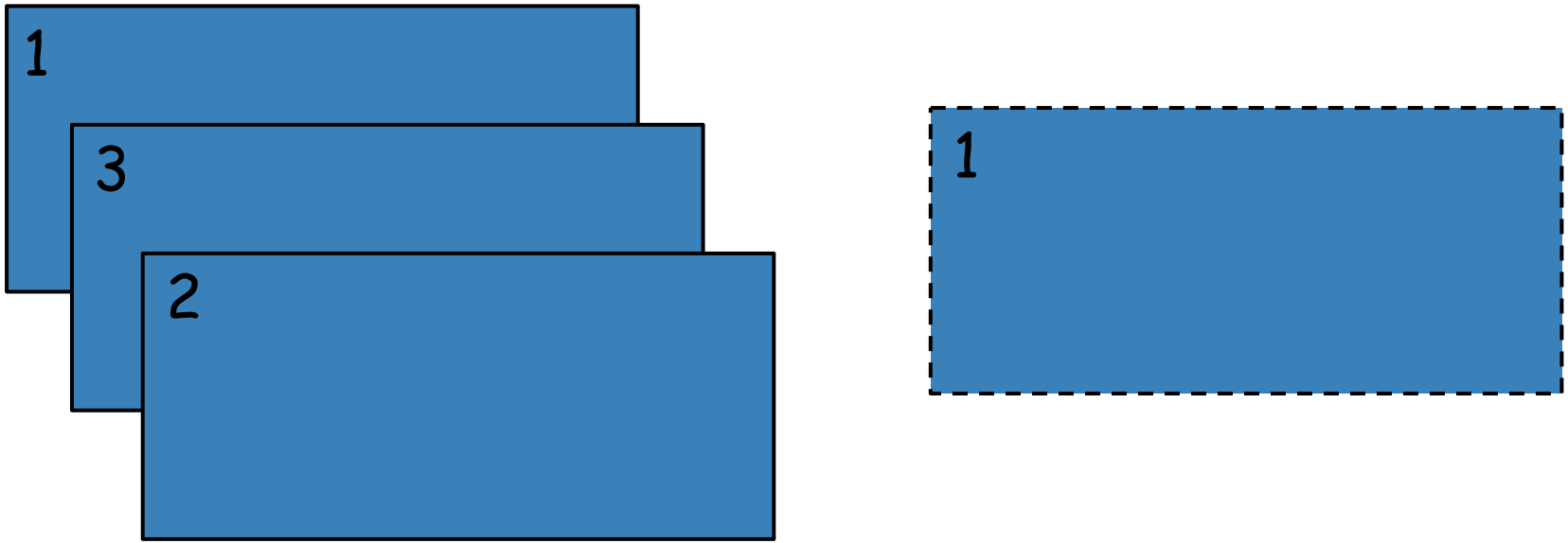
aggregate



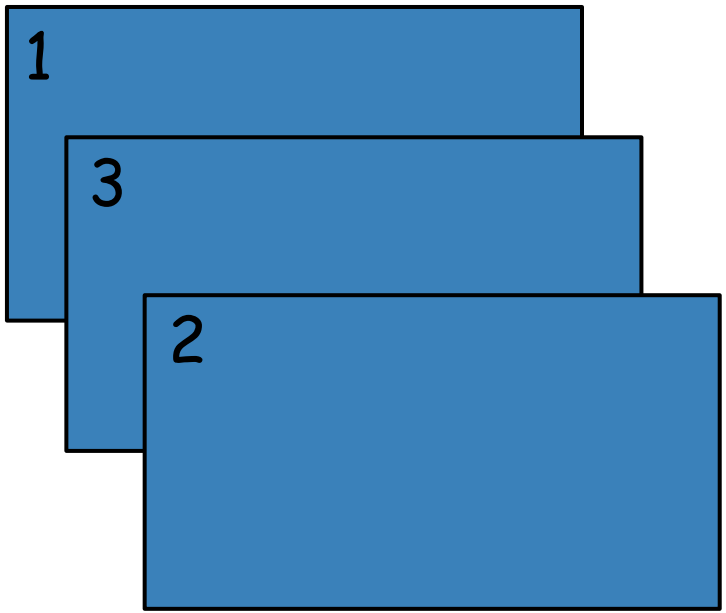
max



min

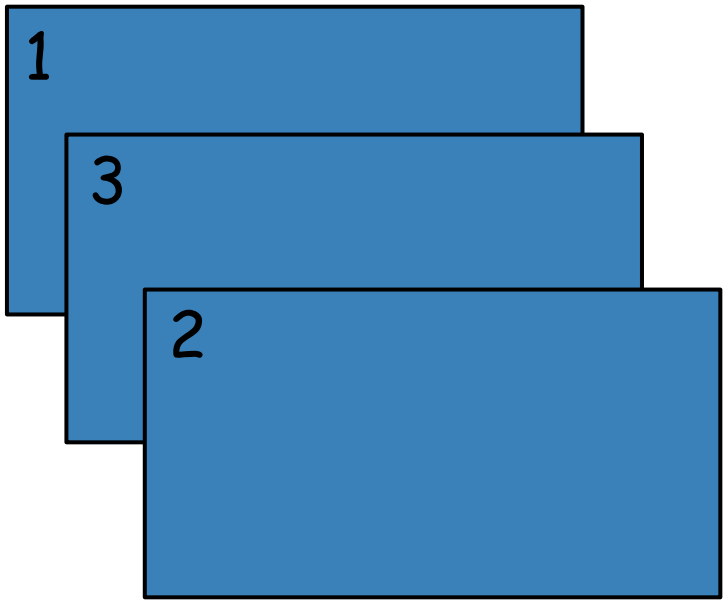


sum



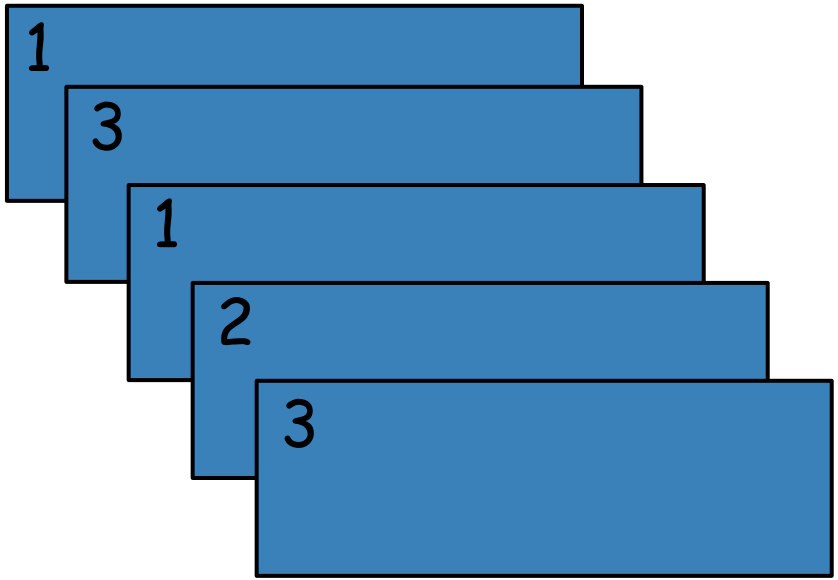
6

count



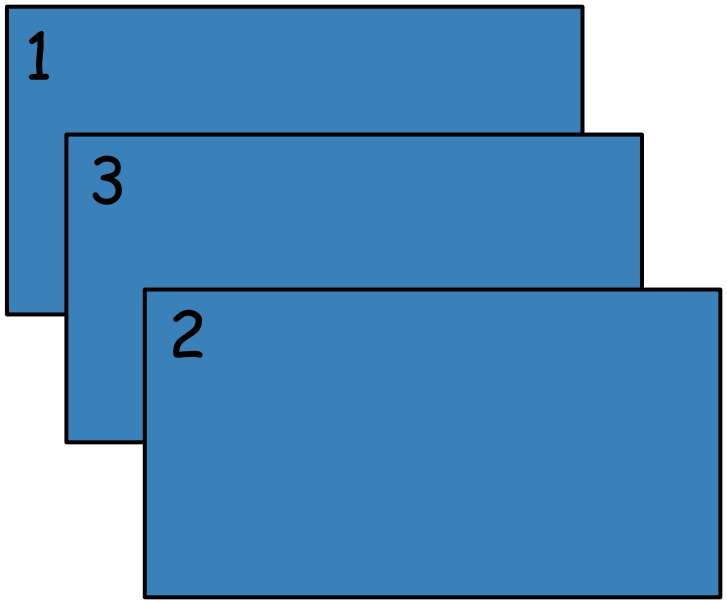
histogram

buckets = 2



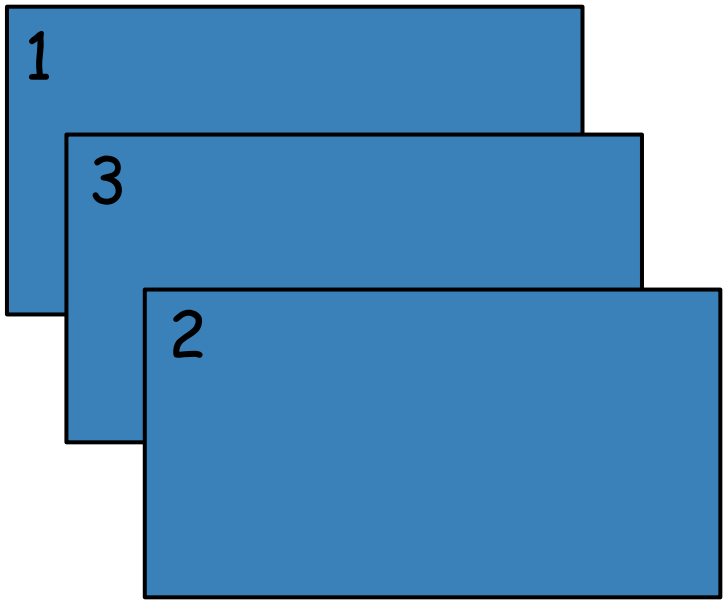
((1,2,3),[2,3])

mean



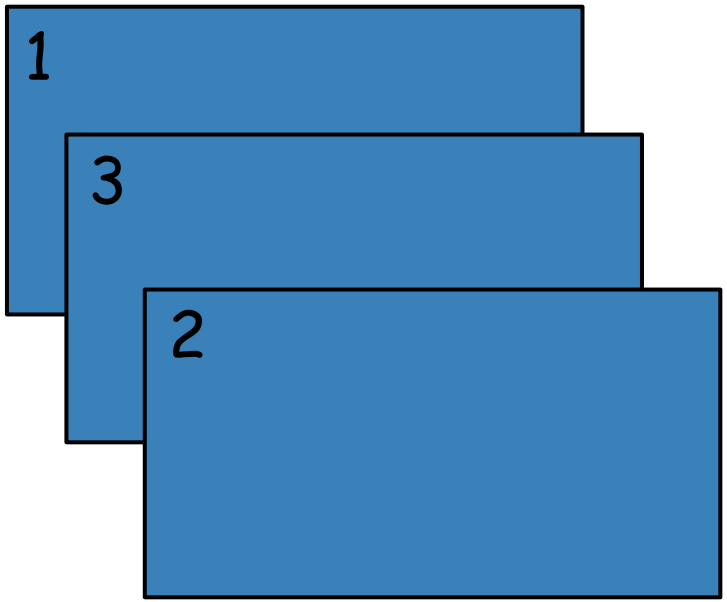
2.0

variance



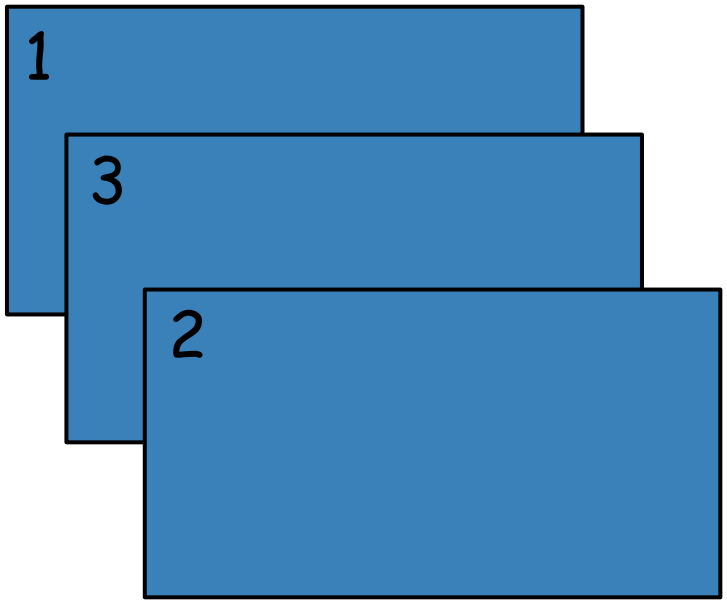
0.666...

stdev



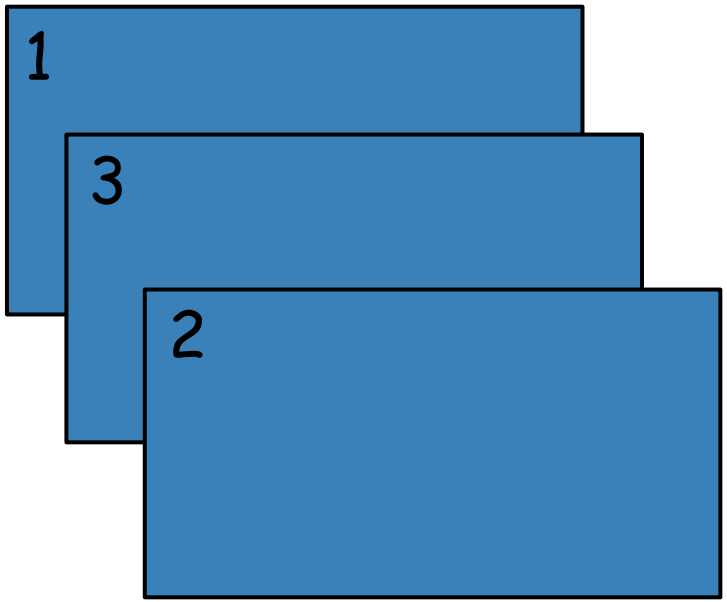
0.816...

sampleStdev



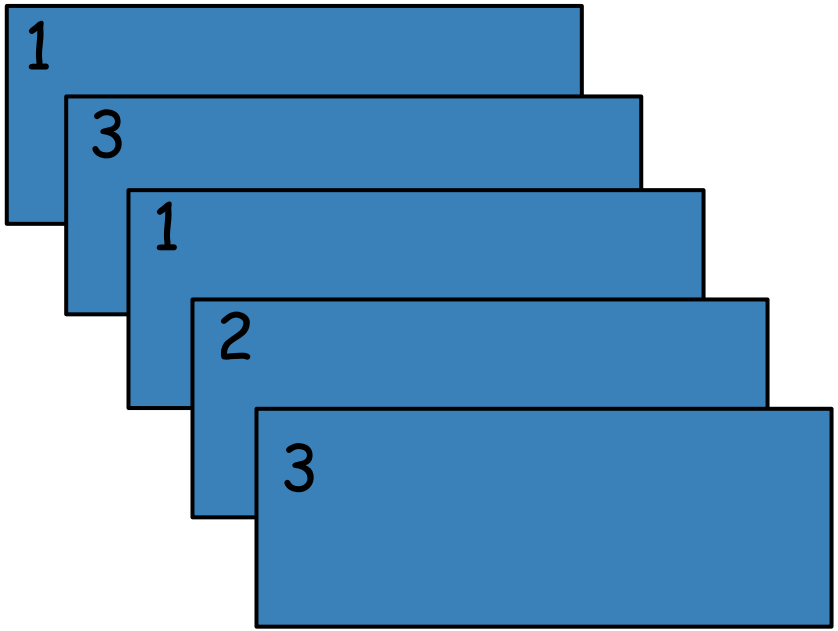
1.0

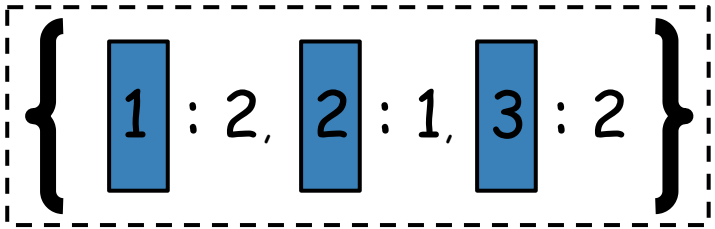
sampleVariance



1.0

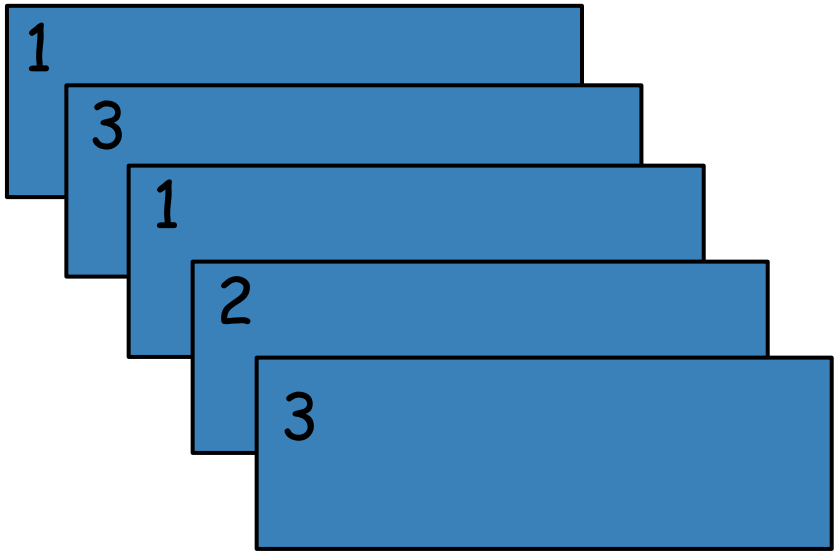
countByValue

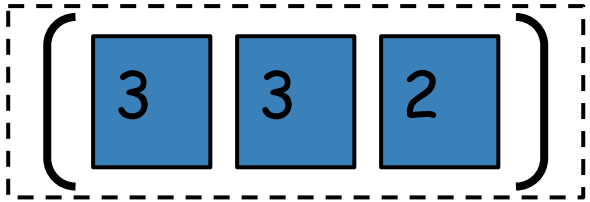




top

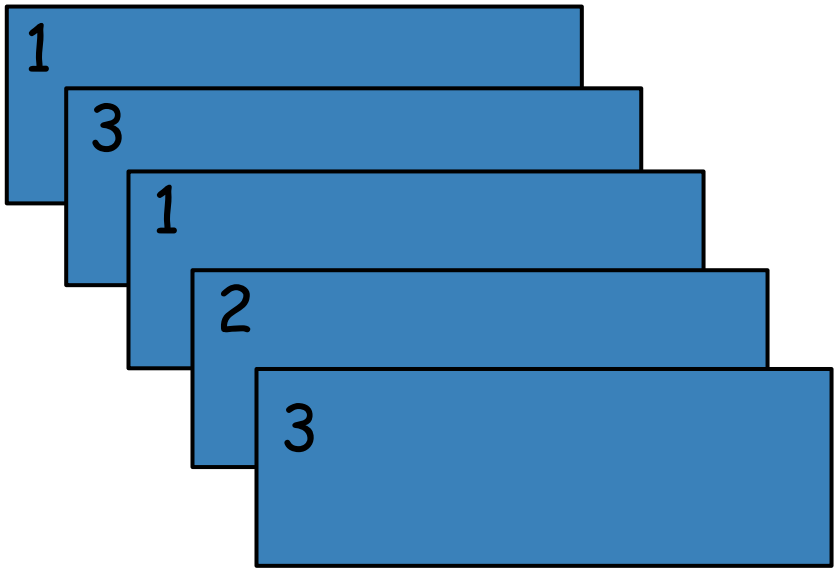
num = 3

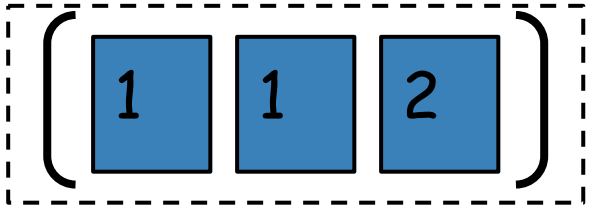




takeOrdered

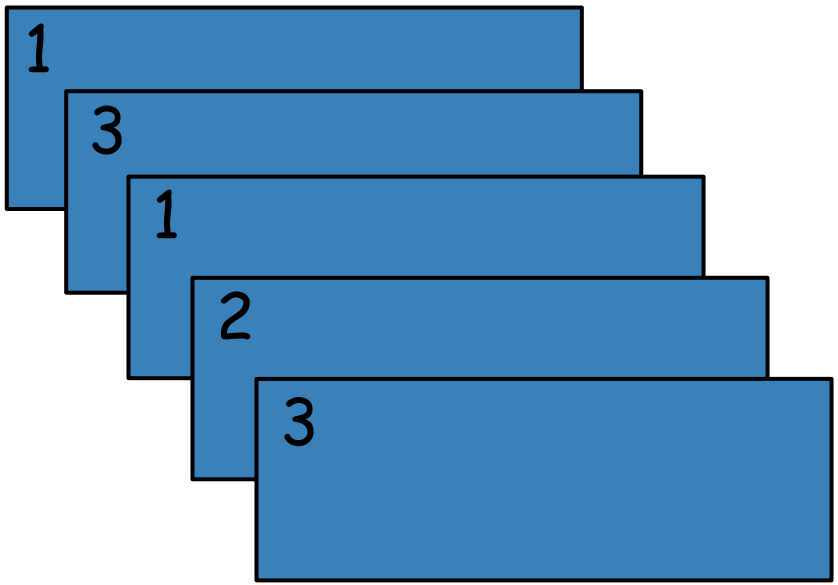
num = 3

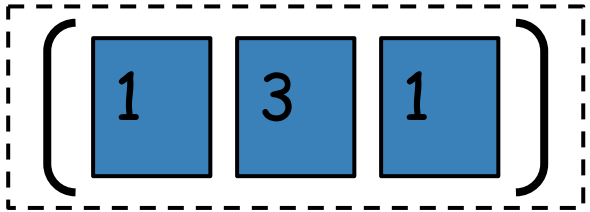




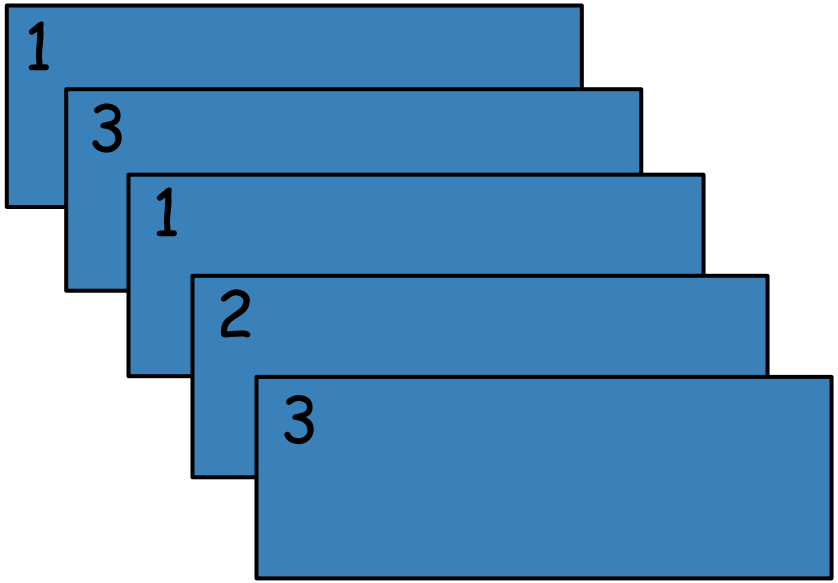
take

num = 3

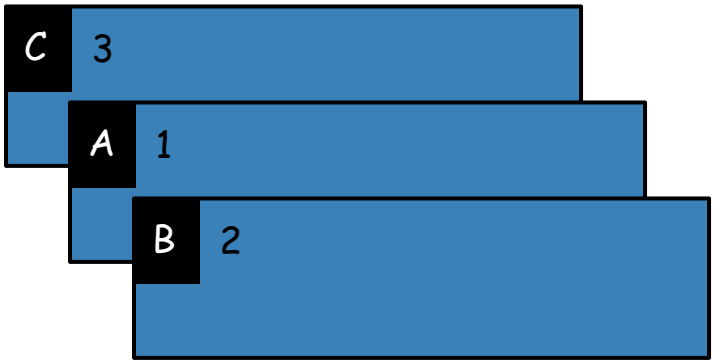


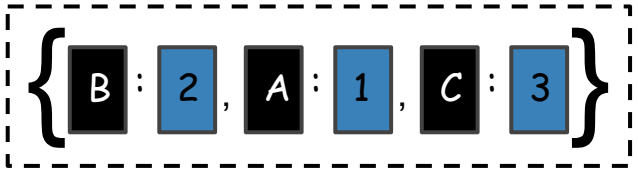


first

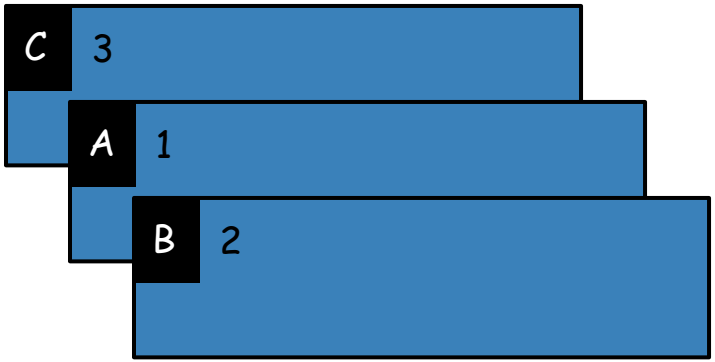


collectAsMap

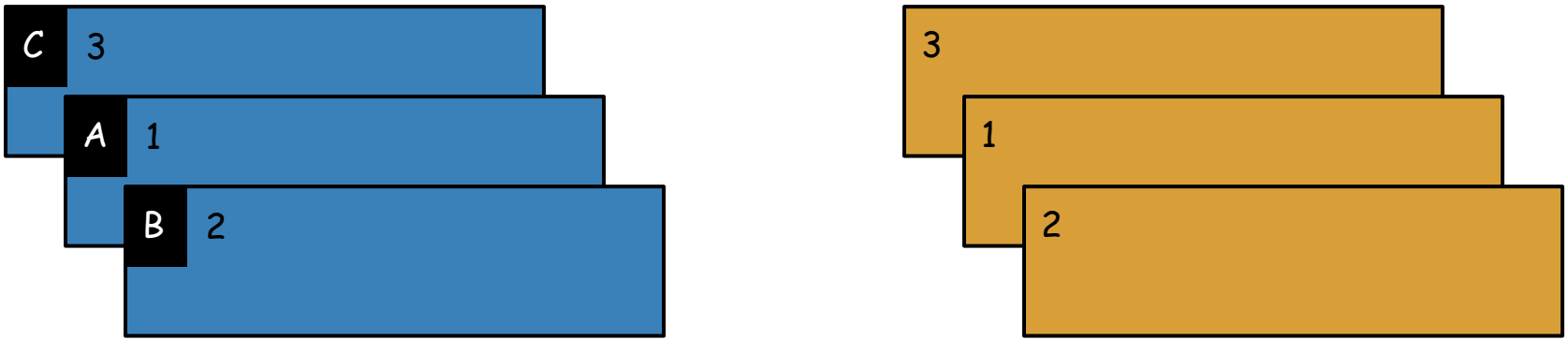




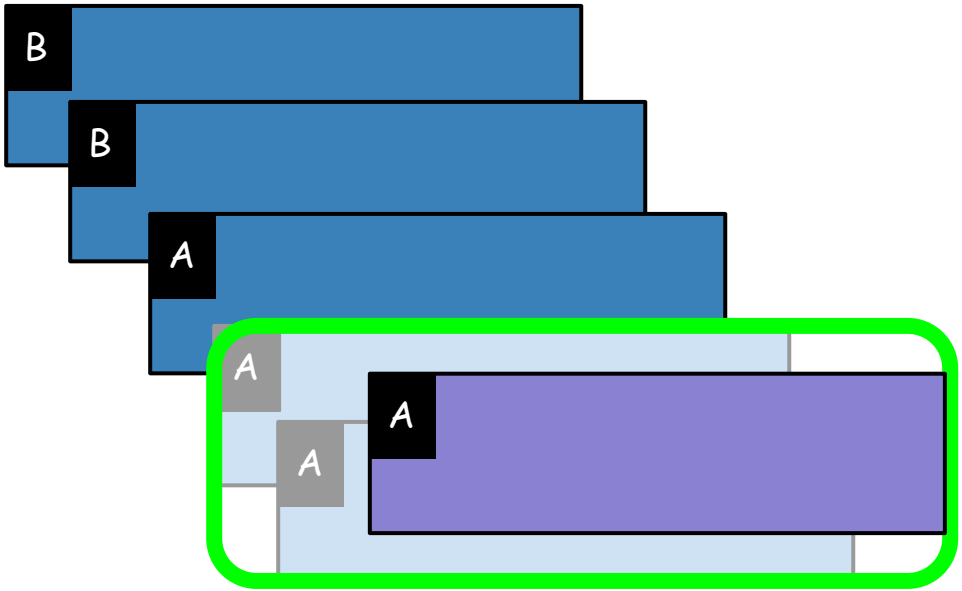
keys



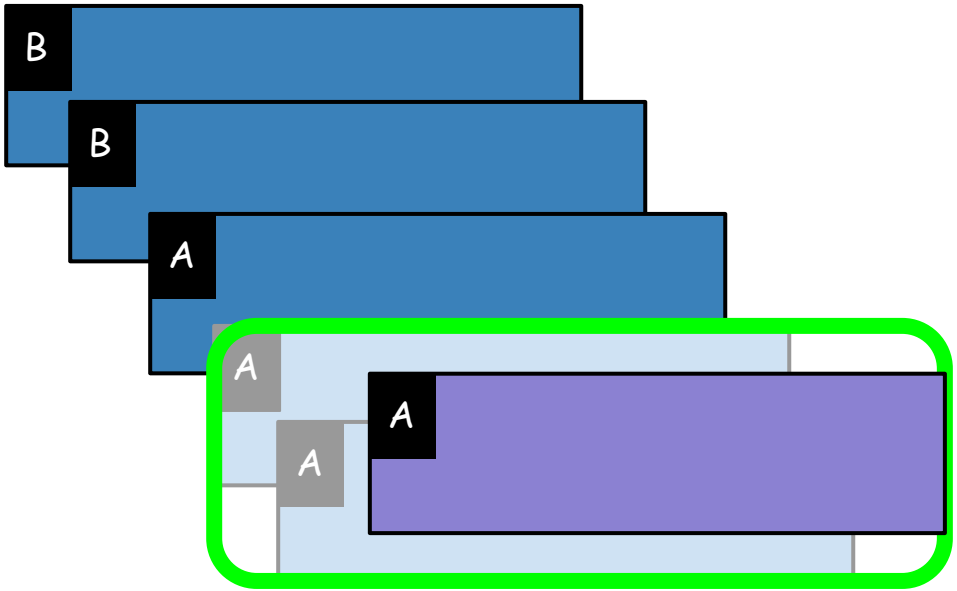
values

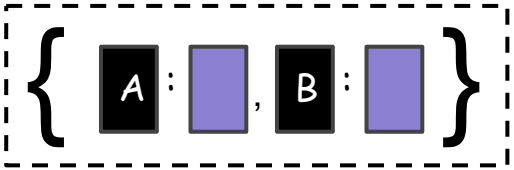


reduceByKey

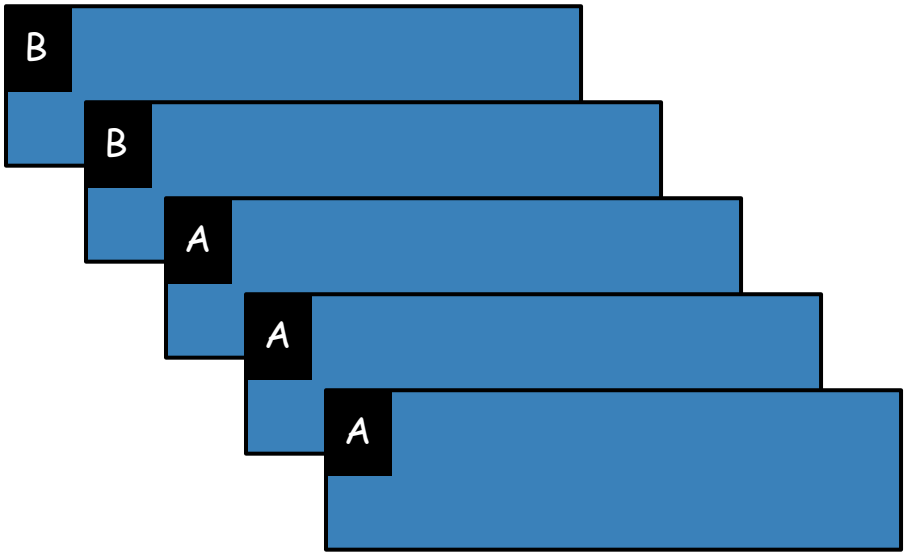


reduceByKeyLocally



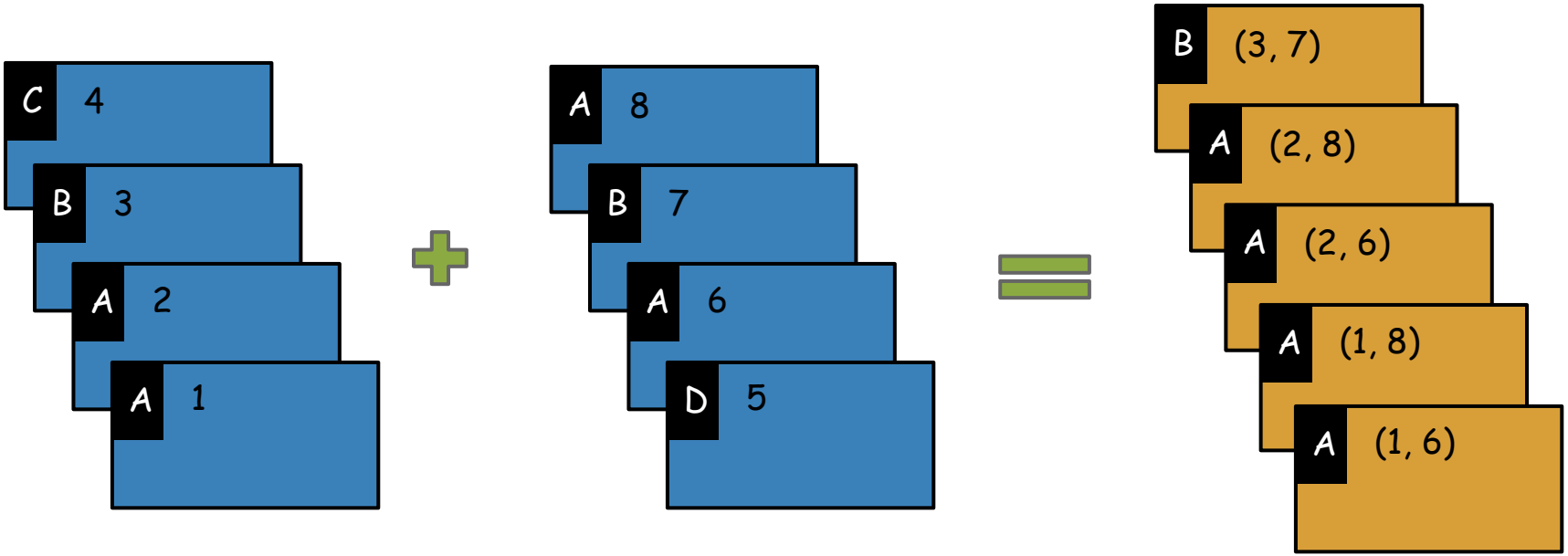


countByKey

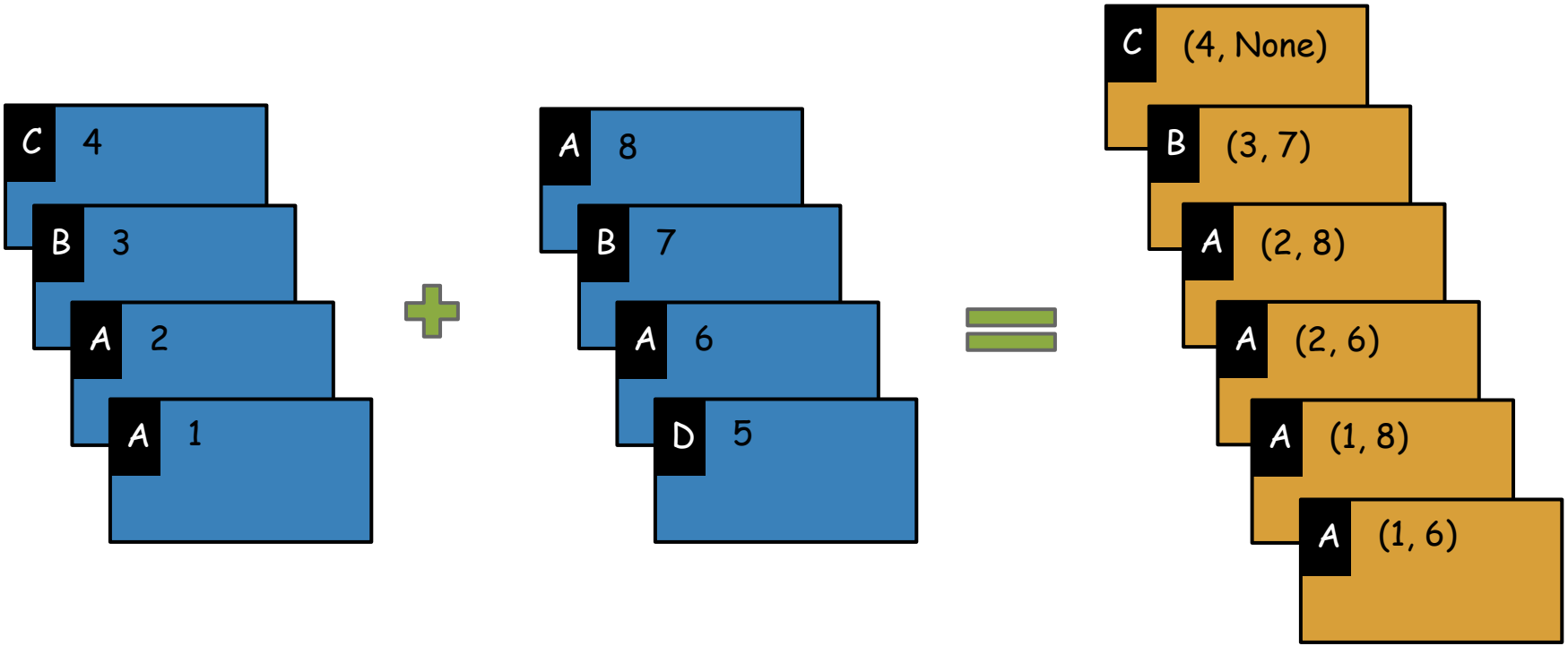


{ A : 3 , B : 2 }

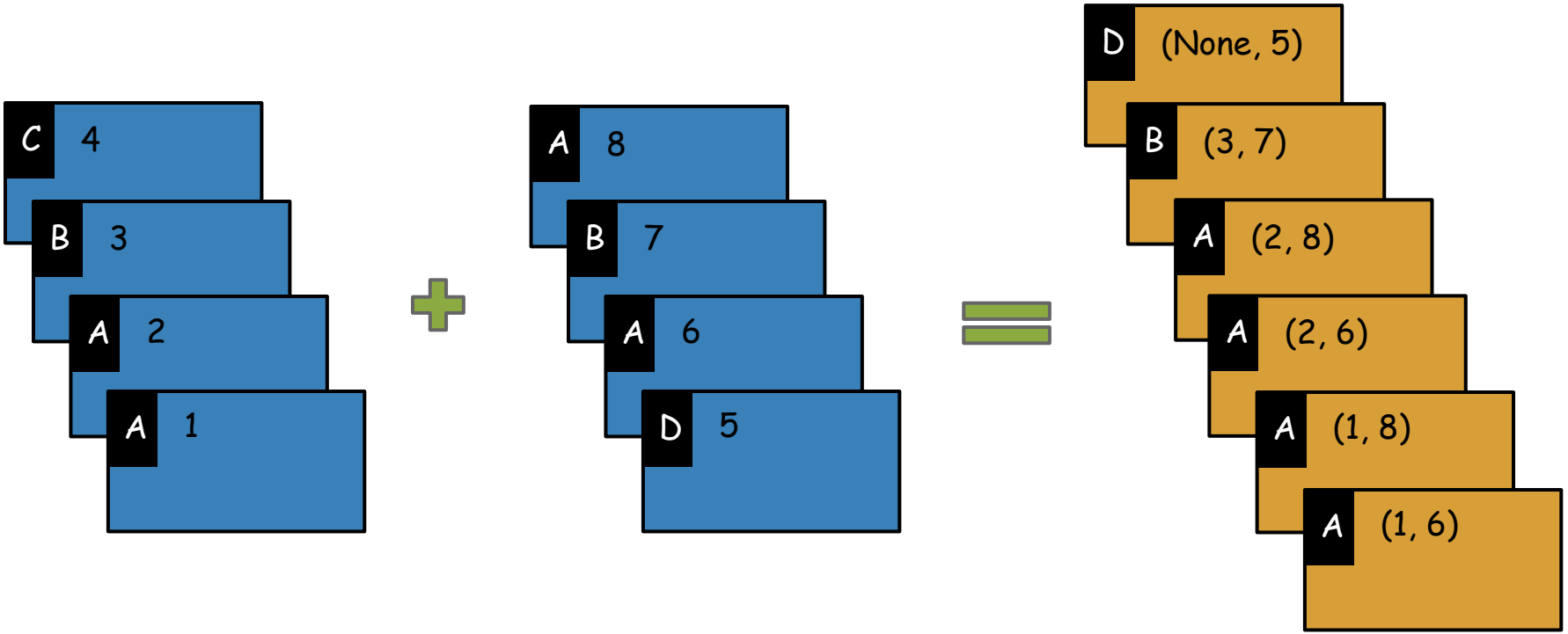
join



leftOuterJoin



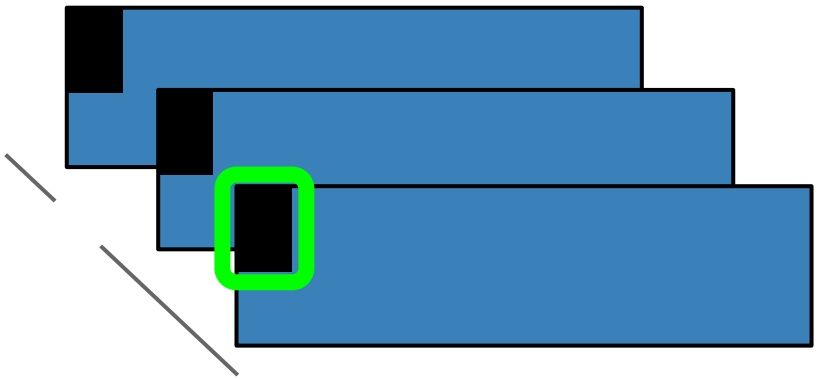
rightOuterJoin

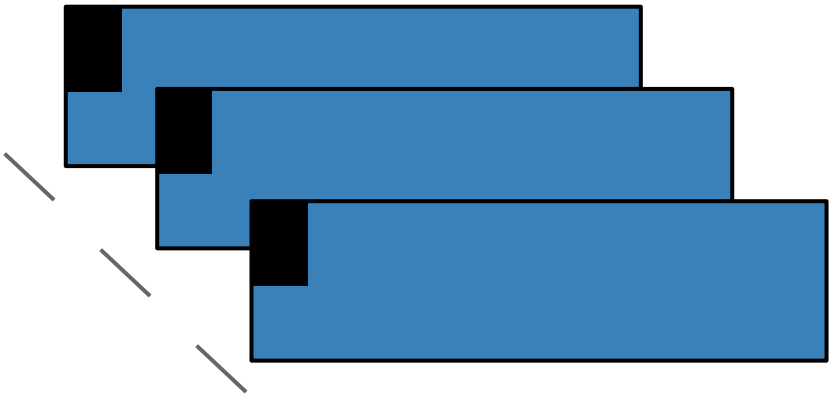


partitionBy

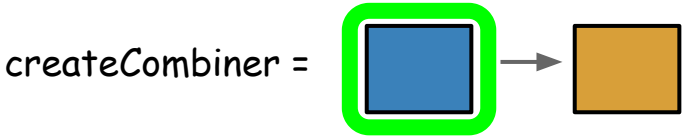
new partition index = key % numPartitions

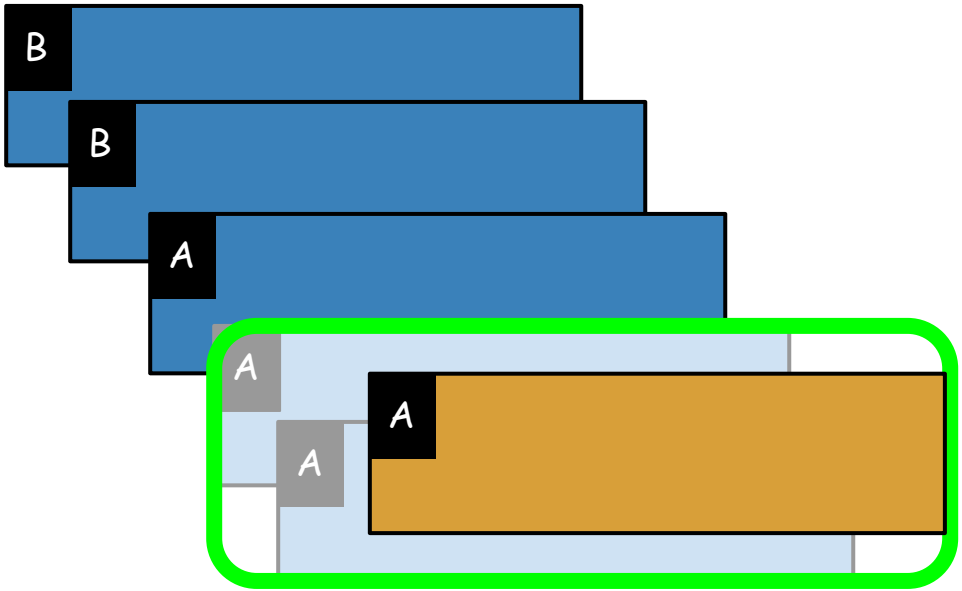
numPartitions = 3





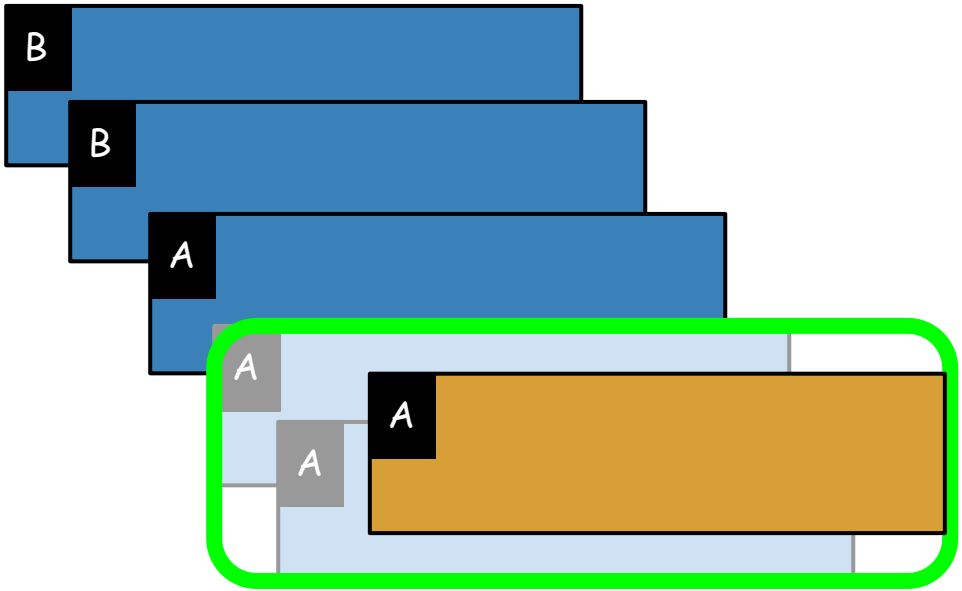
combineByKey



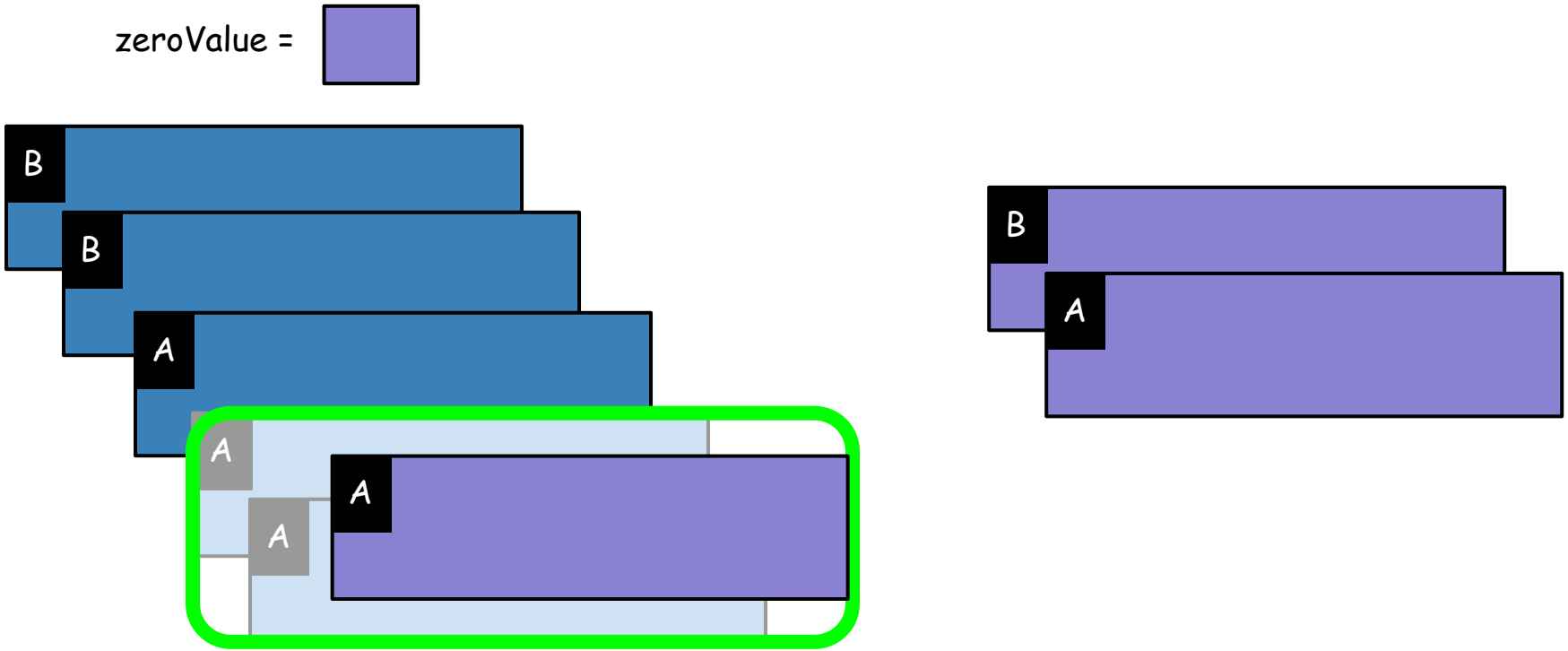


aggregateByKey

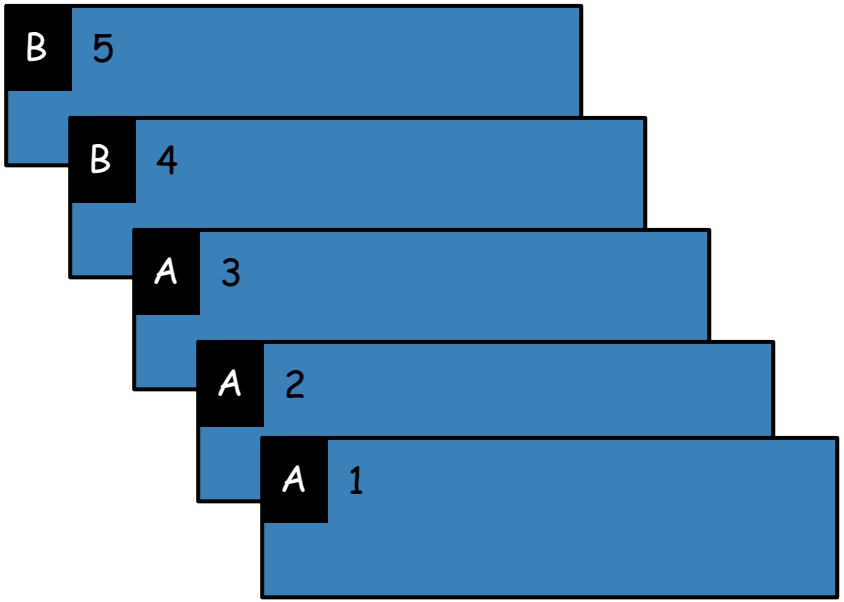
zeroValue =

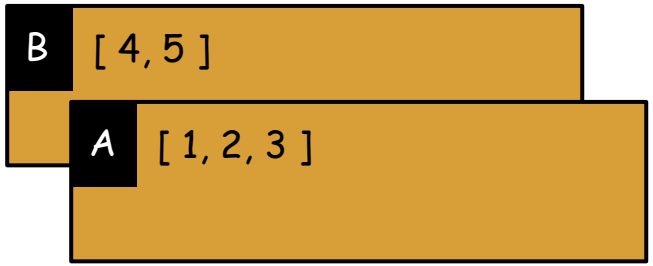


foldByKey

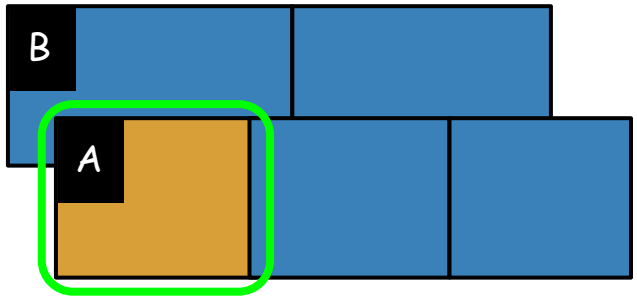


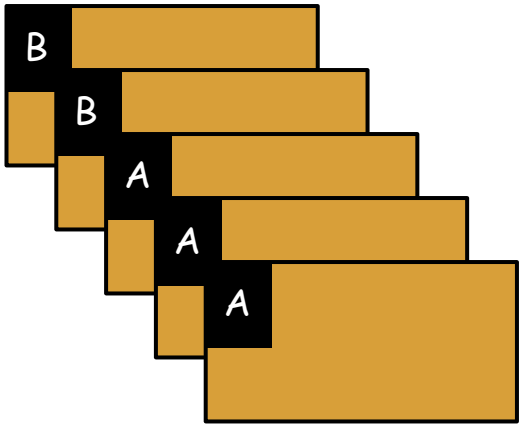
groupByKey



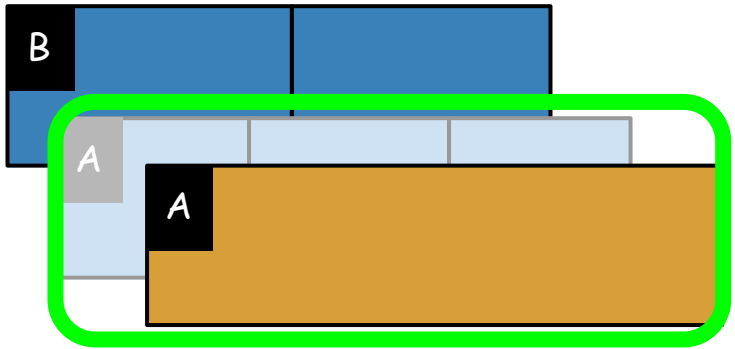


flatMapValues

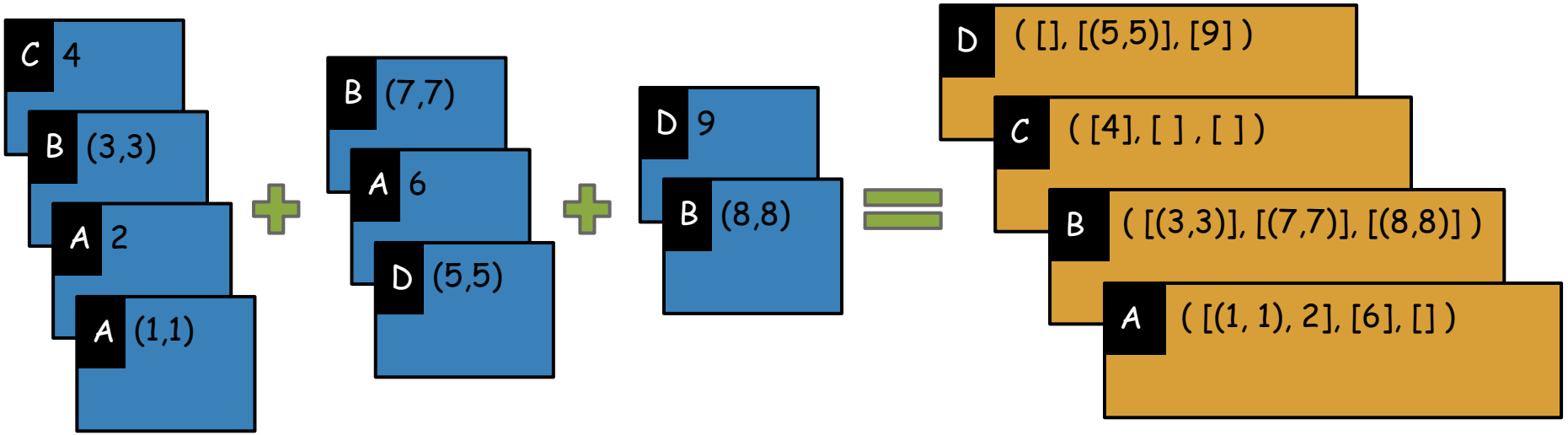




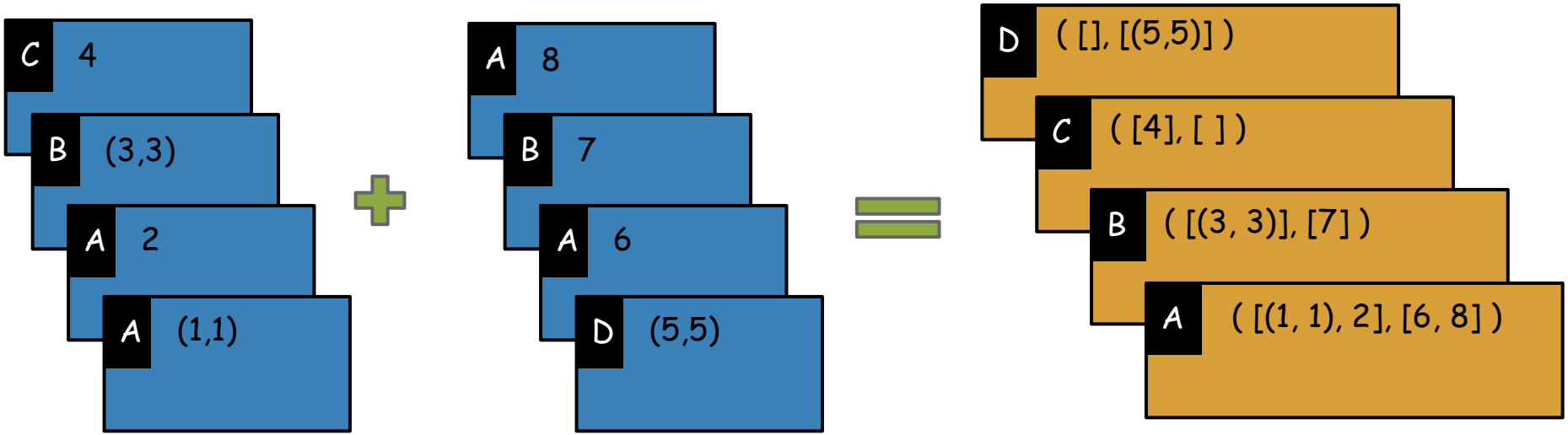
mapValues



groupWith

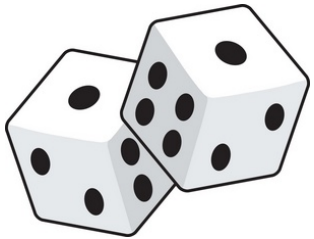


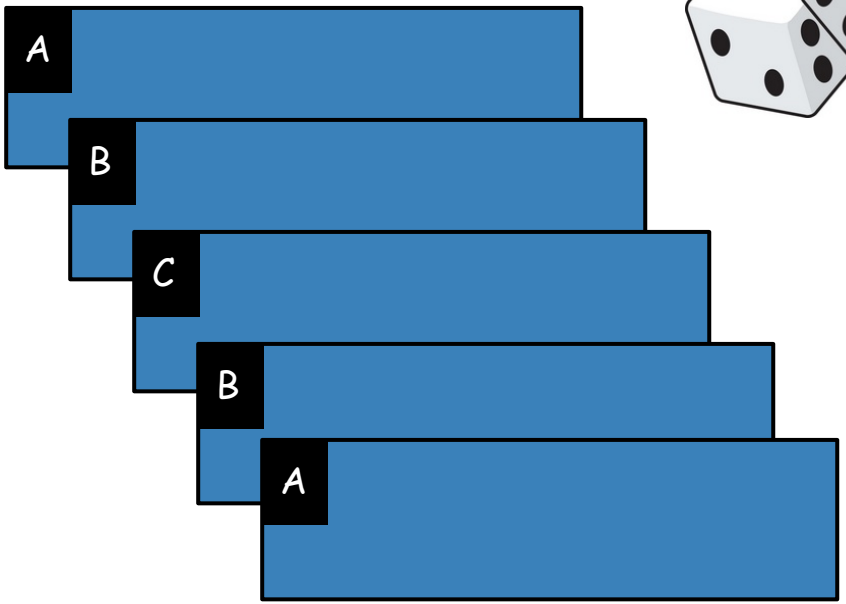
cogroup

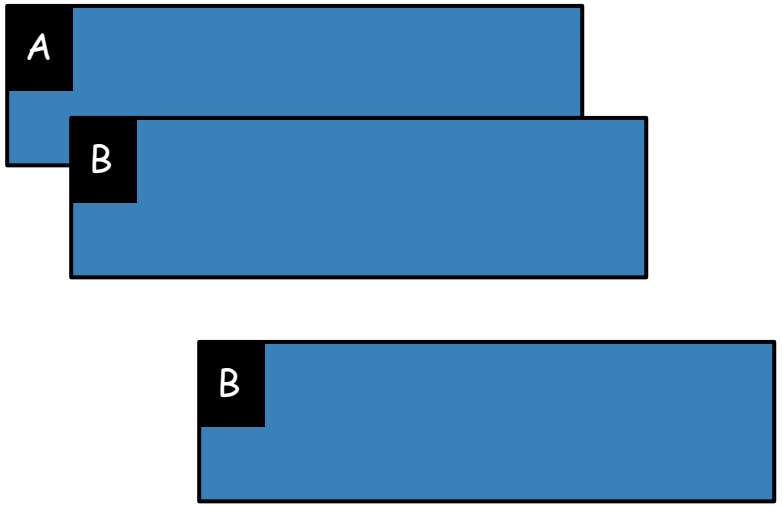


sampleByKey

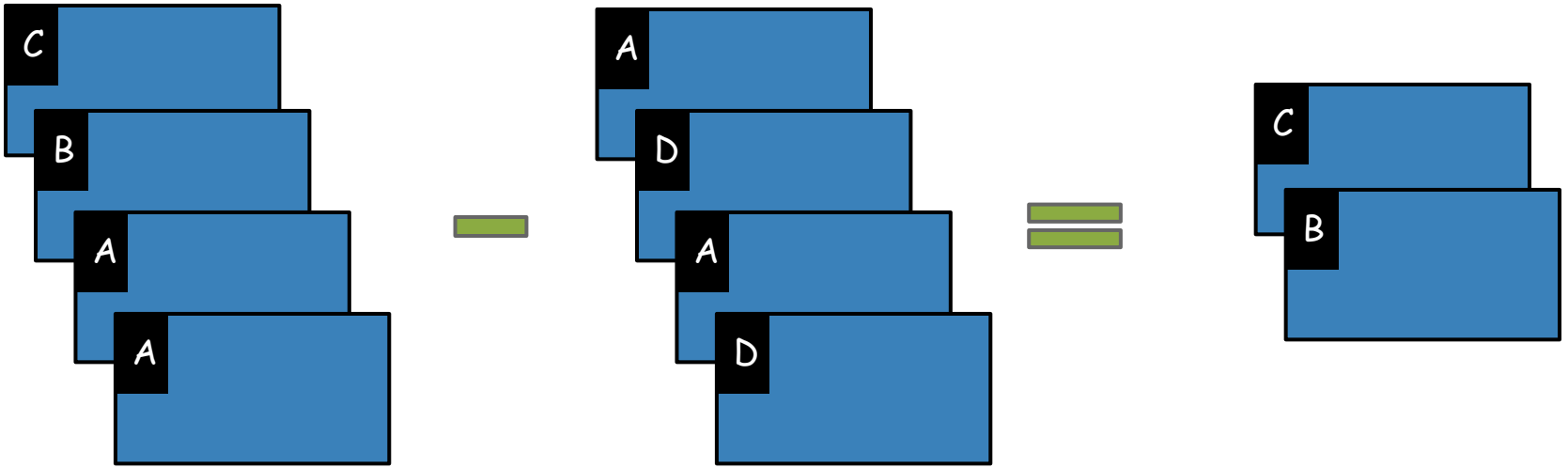
fractions = {'A': 0.5, 'B':1, 'C': 0.2}



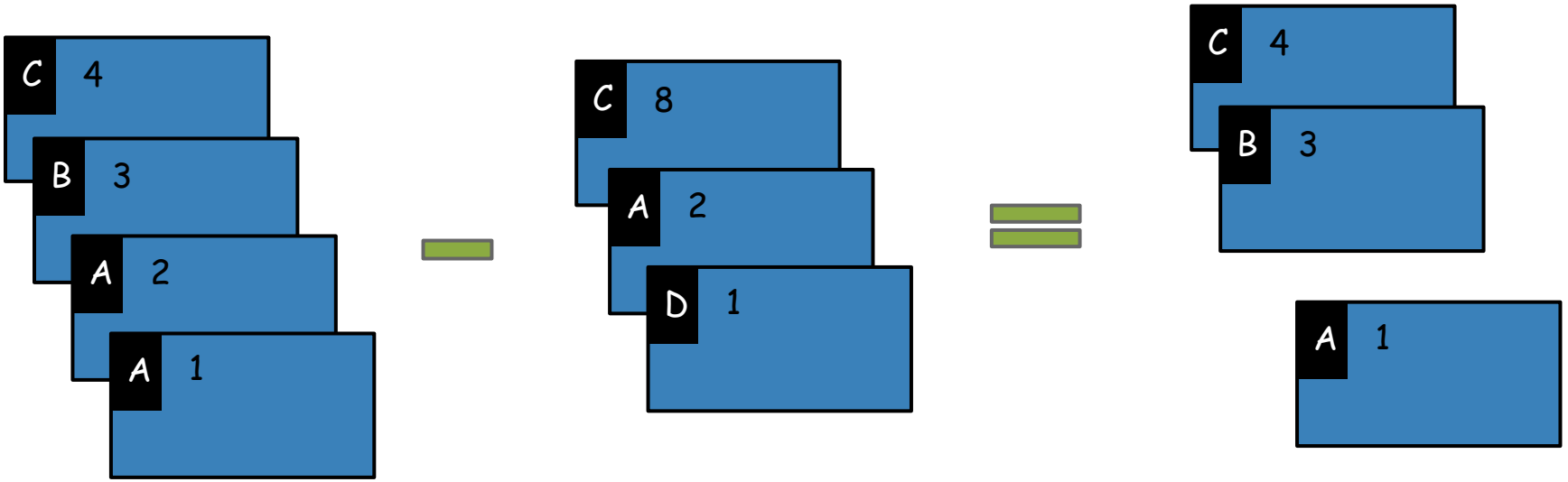




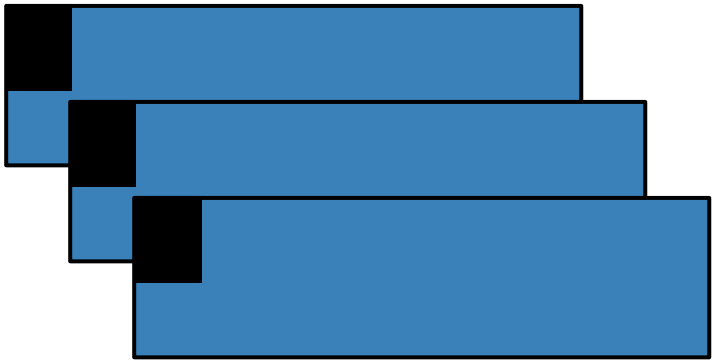
subtractByKey



subtract

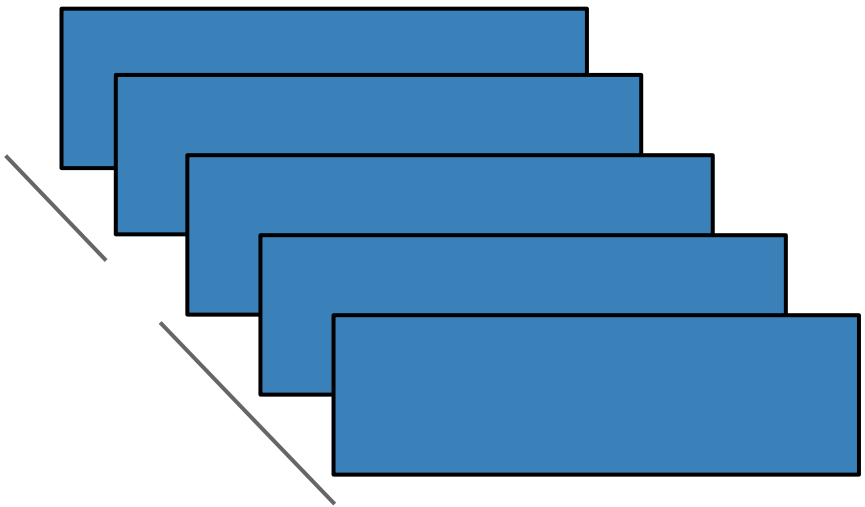


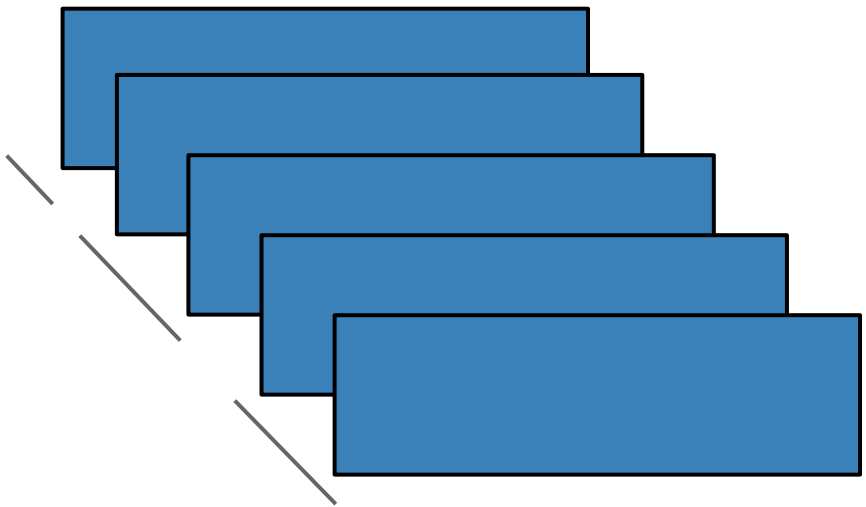
keyBy



repartition

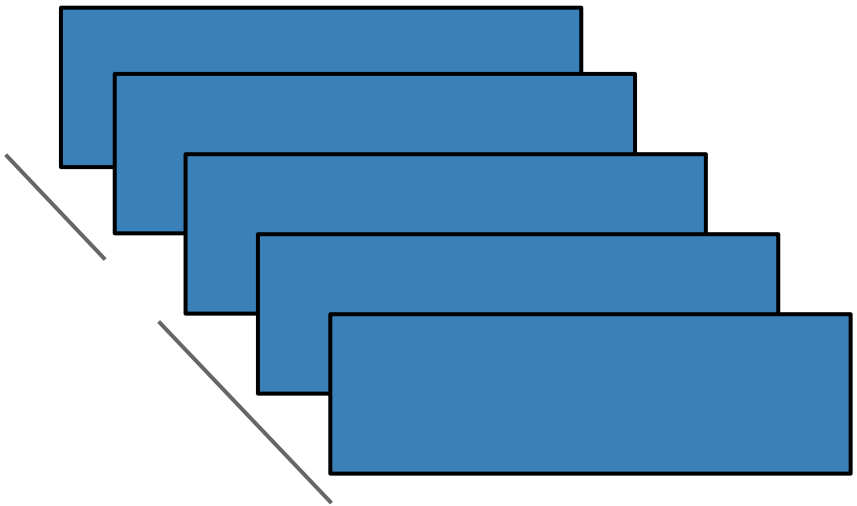
numPartitions = 3

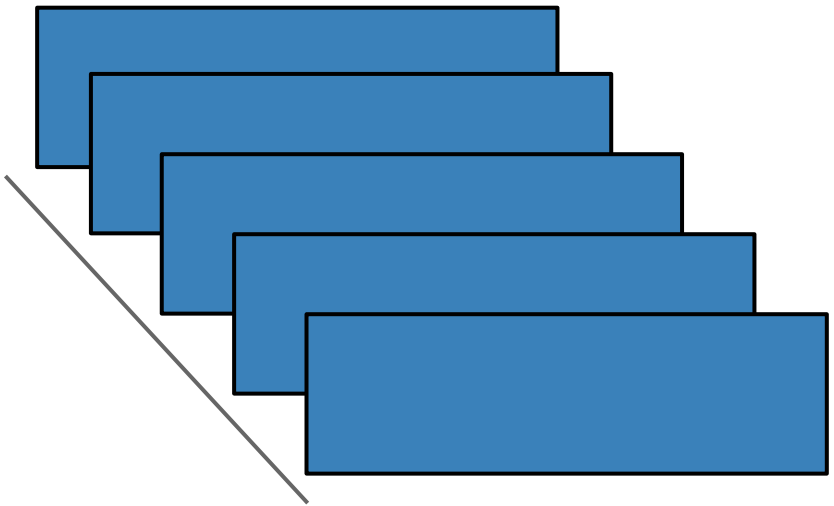




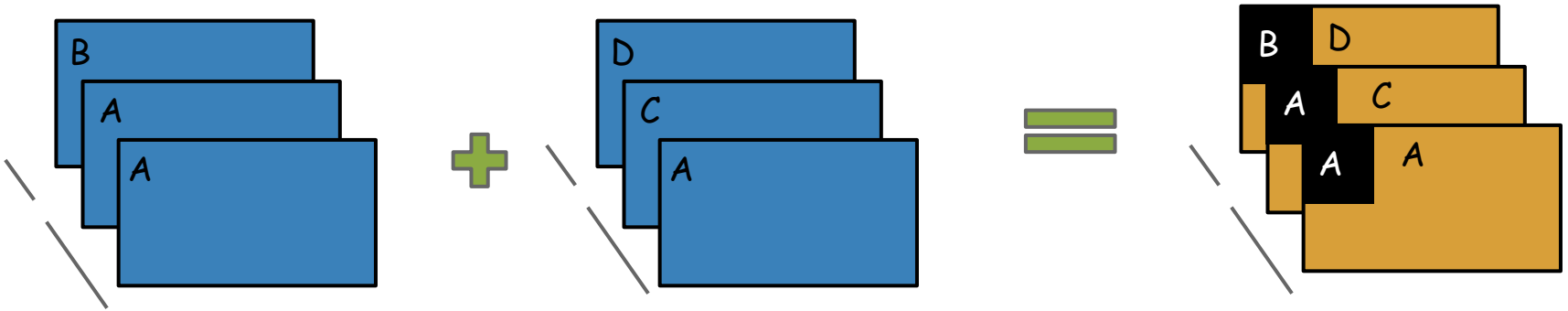
coalesce

numPartitions = 1

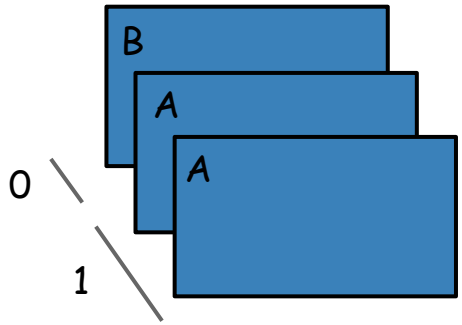


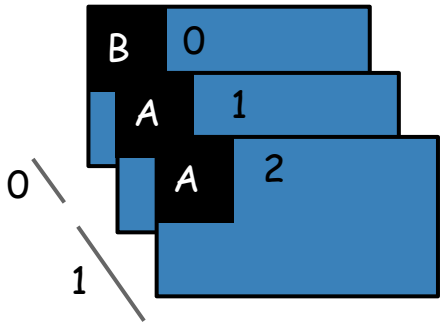


zip



zipWithIndex





zipWithUniqueId

uniqueId = element index * #partitions + partition index