package com.twitter.follow\_recommendations.common.utils

object MergeUtil {

/\*\*

\* Takes a seq of items which have weights. Returns an infinite stream of each item

\* by their weights. All weights need to be greater than or equal to zero. In addition,

\* the sum of weights should be greater than zero.

\*

\* Example usage of this function:

\* Input weighted Item {{CS1, 3}, {CS2, 2}, {CS3, 5}}

\* Output stream: (CS1, CS1, CS1, CS2, CS2, CS3, CS3, CS3, CS3, CS3, CS1, CS1, CS1, CS2,...}

\*

\* @param items items

\* @param weighted provides weights for items

\* @tparam T type of item

\*

\* @return Stream of Ts

\*/

def weightedRoundRobin[T](

items: Seq[T]

)(

implicit weighted: Weighted[T]

): Stream[T] = {

if (items.isEmpty) {

Stream.empty

} else {

val weights = items.map { i => weighted(i) }

assert(

weights.forall {

\_ >= 0

},

"Negative weight exists for sampling")

val cumulativeWeight = weights.scanLeft(0.0)(\_ + \_).tail

assert(cumulativeWeight.last > 0, "Sum of the sampling weights is not positive")

var weightIdx = 0

var weight = 0

def next(): Stream[T] = {

val tmpIdx = weightIdx

weight = weight + 1

weight = if (weight >= weights(weightIdx)) 0 else weight

weightIdx = if (weight == 0) weightIdx + 1 else weightIdx

weightIdx = if (weightIdx == weights.length) 0 else weightIdx

items(tmpIdx) #:: next()

}

next()

}

}

}