package com.twitter.recos.graph\_common

import com.twitter.graphjet.bipartite.api.EdgeTypeMask

import com.twitter.graphjet.bipartite.NodeMetadataLeftIndexedPowerLawMultiSegmentBipartiteGraph

import com.twitter.graphjet.stats.StatsReceiver

/\*\*

\* The GraphBuilder builds a NodeMetadataLeftIndexedPowerLawMultiSegmentBipartiteGraphBuilder given a set of

\* parameters.

\*/

object NodeMetadataLeftIndexedPowerLawMultiSegmentBipartiteGraphBuilder {

/\*\*

\* This encapsulates all the state needed to initialize the in-memory graph.

\*

\* @param maxNumSegments is the maximum number of segments we'll add to the graph.

\* At that point, the oldest segments will start getting dropped

\* @param maxNumEdgesPerSegment determines when the implementation decides to fork off a

\* new segment

\* @param expectedNumLeftNodes is the expected number of left nodes that would be inserted in

\* the segment

\* @param expectedMaxLeftDegree is the maximum degree expected for any left node

\* @param leftPowerLawExponent is the exponent of the LHS power-law graph. see

\* [[com.twitter.graphjet.bipartite.edgepool.PowerLawDegreeEdgePool]]

\* for details

\* @param expectedNumRightNodes is the expected number of right nodes that would be inserted in

\* the segment

\* @param numRightNodeMetadataTypes is the max number of node metadata types associated with the

\* right nodes

\*/

case class GraphBuilderConfig(

maxNumSegments: Int,

maxNumEdgesPerSegment: Int,

expectedNumLeftNodes: Int,

expectedMaxLeftDegree: Int,

leftPowerLawExponent: Double,

expectedNumRightNodes: Int,

numRightNodeMetadataTypes: Int,

edgeTypeMask: EdgeTypeMask)

/\*\*

\* This apply function returns a mutuable bipartiteGraph

\*

\* @param graphBuilderConfig is the graph builder config

\*

\*/

def apply(

graphBuilderConfig: GraphBuilderConfig,

statsReceiverWrapper: StatsReceiver

): NodeMetadataLeftIndexedPowerLawMultiSegmentBipartiteGraph = {

new NodeMetadataLeftIndexedPowerLawMultiSegmentBipartiteGraph(

graphBuilderConfig.maxNumSegments,

graphBuilderConfig.maxNumEdgesPerSegment,

graphBuilderConfig.expectedNumLeftNodes,

graphBuilderConfig.expectedMaxLeftDegree,

graphBuilderConfig.leftPowerLawExponent,

graphBuilderConfig.expectedNumRightNodes,

graphBuilderConfig.numRightNodeMetadataTypes,

graphBuilderConfig.edgeTypeMask,

statsReceiverWrapper

)

}

}