Module 11 Assignment

Result:

Incorrectly partitioned nodes: 2

Code:

package assignments;  
  
  
import graph.GraphUtils;  
import graph.MGraph;  
import org.apache.commons.math3.linear.EigenDecomposition;  
import org.apache.commons.math3.linear.OpenMapRealMatrix;  
import org.apache.commons.math3.linear.RealVector;  
import org.apache.tinkerpop.gremlin.structure.Vertex;  
import org.apache.tinkerpop.gremlin.tinkergraph.structure.TinkerGraph;  
  
import java.util.Arrays;  
import java.util.HashMap;  
import java.util.Iterator;  
import java.util.Map;  
  
public class Module11 {  
 private String GRAPH\_FILE = "GraphDatabases\\students.graphml";  
  
 public Module11(){  
 TinkerGraph graph = GraphUtils.*readGraphML*(GRAPH\_FILE);  
  
 // Delete isolated students  
 Iterator<Vertex> nodes = graph.vertices();  
 while(nodes.hasNext()){  
 Vertex node = nodes.next();  
 if((long)node.property("degree").value() == 0){  
 node.remove();  
 }  
 }  
  
 MGraph matrixGraph = new MGraph(graph);  
 OpenMapRealMatrix laplacianMatrix = matrixGraph.getLaplacian();  
 RealVector eigenVector = getSecondSmallestEigenVector(new EigenDecomposition(laplacianMatrix));  
  
 nodes = graph.vertices();  
 while(nodes.hasNext()){  
 Vertex node = nodes.next();  
 int vertexIndex = matrixGraph.getVertexIndexFromID(node.id().toString());  
  
 //assign partition  
 node.property("partition", Math.*sin*(eigenVector.getEntry(vertexIndex)));  
 }  
  
 GraphUtils.*saveGraphML*(graph, "GraphDatabases\\students\_partitioned.graphml");  
 System.*out*.println("Incorrectly partitioned nodes: " + countIncorrectNodes(graph));  
 }  
  
 public RealVector getSecondSmallestEigenVector(EigenDecomposition eigenDecomposition){  
 double[] eigenvalues = eigenDecomposition.getRealEigenvalues();  
 Map<Double, Integer> indexMap = new HashMap<>();  
  
 for(int i = 0; i < eigenvalues.length; i++){  
 indexMap.put(eigenvalues[i], i);  
 }  
  
 Arrays.*sort*(eigenvalues);  
 return eigenDecomposition.getEigenvector(indexMap.get(eigenvalues[1]));  
 }  
  
 public int countIncorrectNodes(TinkerGraph graph){  
 int count = 0;  
 Iterator<Vertex> nodes = graph.vertices();  
 while(nodes.hasNext()){  
 Vertex node = nodes.next();  
 int school = Integer.*parseInt*(node.property("school").value().toString());  
 double partition = Double.*parseDouble*(node.property("partition").value().toString());  
  
 // partition < 0 corresponds to school = 0  
 // partition >= 0 corresponds to school = 1  
 if( (school == 1 && partition < 0) ||(school == 0 && partition >= 0) ){  
 count++;  
 }  
 }  
 return count;  
 }  
  
 public static void main(String[] args){  
 new Module11();  
 }  
}