APPENDIX

```
1
2  package roudolf;
4  * @author roudolfndumbe
6
7  import java.io.BufferedWriter;
31
32
33 public class SubGraphs {
34
35     /**
36      * @param args
37      */
38     public static void main(String[] args) {
39         // TODO Auto-generated method stub
40
41          String filePath = args[0];    //input file
42             File xmlFile = new File(filePath);
43             DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
44             DocumentBuilder dBuilder;
45
46         try {
47                 dBuilder = dbFactory.newDocumentBuilder();
48                  Document doc = dBuilder.parse(xmlFile);
49                  doc.getDocumentElement().normalize();
50
51                  int NS = NumberOfSubgraphs(doc) ;
52
53                 ArrayList<Element> edgeL = new ArrayList<>();
54                 ArrayList<Element> nodeL = new ArrayList<>();
55                 Map<String, ArrayList<Element>> edgeM = new HashMap<>();
56                 Map<String, ArrayList<Element>> nodeM = new HashMap<>();
57
58                 System.out.println("Enter n for Network or m for Molecule database: ");
59                 String option;
60                 Scanner scanIn = new Scanner(System.in);
61                 option = scanIn.nextLine();
62                 scanIn.close();
63
64                  switch (option) {
65                   case "m":  MoleculeSubgraph(doc, args[1], edgeM, nodeM, edgeL, nodeL);
66                              break;
67
68                   case "n":  NetworkSubgraph(doc, args[1], edgeM, nodeM, edgeL, nodeL);
69                              break;
70
71                   default: System.out.println("Invalid letter entered (n or m): ");
72                              break;
73                  }
74
75                   System.out.println("Subgraphs created successfully");
76
77           } catch (SAXException | ParserConfigurationException | IOException e1)
78             {
79                 e1.printStackTrace();
80             }
81     }
82
83
84
85      private static int NumberOfSubgraphs(Document doc) {
86          ArrayList<String> input = new ArrayList<>();
87          ArrayList<String> uniquewords = new ArrayList<>();
88          NodeList edges = doc.getElementsByTagName("edge");
89          Element ed = null;
90          //loop for each node
91          for(int i=0; i<edges.getLength(); i++)
92          {
93              ed = (Element) edges.item(i);
94              Node dataEdge = ed.getElementsByTagName("data").item(0);
95             if(dataEdge.getTextContent().equals("subgraph"))
96             {
97               input.add(dataEdge.getNextSibling().getTextContent());
98             }
99          }
00          for(String word : input)
01          {
```

```
          if(!uniquewords.contains(word))
           {
               uniquewords.add(word);
           }
       }
       return uniquewords.size();
   }



private static void MoleculeSubgraph(Document doc, String filepath, Map<String, ArrayList<Element>> edgeM,  Map<String, ArrayList<E
                                     ArrayList<Element> edgeL,  ArrayList<Element> nodeL) throws IOException
{

       NodeList nodes = doc.getElementsByTagName("node");
       Element nd = null;
       Element nd1 = null;
      NodeList edges = doc.getElementsByTagName("edge");
       Element ed = null;
         String kE = null;
         String kN = null;
       ArrayList<String> subs = new ArrayList<>();
       ArrayList<String> unique = new ArrayList<>();


         //Filling the map made up nodes for the subgraphs
         for(int i=0; i<nodes.getLength();i++)
         {
             nd = (Element) nodes.item(i);
             subs.add(nd.getElementsByTagName("data").item(0).getNextSibling().getNextSibling().getTextContent());
         }
         for(String s: subs) {
             if(!unique.contains(s)) {
            unique.add(s);
             }
             else { continue; }
         }
         for(String u: unique) {
               kN = u;
               kE = u;
               nodeL = new ArrayList<>();
               edgeL = new ArrayList<>();
            for(int j=0; j<nodes.getLength();j++)
             {
                  nd1 = (Element) nodes.item(j);
              if( nd1.getElementsByTagName("data").item(0).getNextSibling().getNextSibling().getTextContent().equals(u) )
                {
                 if(!nodeL.contains(nd1)) {
                     nodeL.add(nd1);
                     }

                    for(int k=0; k<edges.getLength();k++)
                      {
                          ed = (Element) edges.item(k);
                        if( (ed.getAttributes().getNamedItem("source").getTextContent().equals(nd1.getAttributes().getNamedItem("ic
                             ||(ed.getAttributes().getNamedItem("target").getTextContent().equals(nd1.getAttributes().getNamedItem(
                           {
                             if(!edgeL.contains(ed)) {
                                     edgeL.add(ed);
                              }
                           }
                       }
                  }
             }

            if(!nodeM.containsValue(nodeL) || !nodeM.containsKey(kN)){
                   nodeM.put(kN, nodeL);
             }
            if(!edgeM.containsValue(edgeL) || !edgeM.containsKey(kE)){
                edgeM.put(kE, edgeL);
             }
        }

          String iv = "\"";
          int g=0;
```

```
178          BufferedWriter writer = null;
179      try {
180              writer = new BufferedWriter(new FileWriter(filepath));
181             writer.write("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n" +
182                          "<graphml xmlns=\"http://graphml.graphdrawing.org/xmlns\" "
183                        + "xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" "
184                        + "xsi:schemaLocation=\"http://graphml.graphdrawing.org/xmlns http://graphml.graphdrawing.org/xmlns/1.0/graph
185                        + " ");
186              writer.newLine();
187
188             for(String k: nodeM.keySet())
189             {
190                   writer.write("<graph id=\"G"+g+"\" edgedefault=\"directed\"> ");
191                   writer.newLine();
192                 for(Element n: nodeM.get(k))
193                 {
194                   writer.write( "<node id="+iv+n.getAttributes().getNamedItem("id").getTextContent()+iv+" labels="+iv+n.getAttribute
195                           +"<data key="+iv+n.getFirstChild().getAttributes().getNamedItem("key").getTextContent()+iv+">"+n.getElement
196                           +"<data key="+iv+n.getElementsByTagName("data").item(0).getNextSibling().getAttributes().getNamedItem("key'
197                           +"<data key="+iv+n.getElementsByTagName("data").item(0).getNextSibling().getNextSibling().getAttributes().g
198                           +"<data key="+iv+n.getElementsByTagName("data").item(0).getNextSibling().getNextSibling().getNextSibling().
199                           +"<data key="+iv+n.getElementsByTagName("data").item(0).getNextSibling().getNextSibling().getNextSibling().
200                           +"<data key="+iv+n.getElementsByTagName("data").item(0).getNextSibling().getNextSibling().getNextSibling().
201                           +"</node>");
202                    writer.newLine();
203                 }
204                for(Element e: edgeM.get(k))
205                {
206                   writer.write("<edge source="+iv+e.getAttributes().getNamedItem("source").getTextContent()+iv+" target="+iv+e.getAtt
207                           +"<data key="+iv+e.getFirstChild().getAttributes().getNamedItem("key").getTextContent()+iv+">"+e.getElement
208                           +"</edge>");
209                   writer.newLine();
210                }
211                   writer.write("</graph> ");
212                   writer.newLine();
213                   writer.newLine();
214               g++;
215             }
216            writer.write("</graphm> ");
217           writer.flush();
218
219      } catch(Exception e) {
220          // if any I/O error occurs
221          e.printStackTrace();
222      }finally {
223          // releases system resources from the streams
224          if(writer!=null)
225            writer.close();
226      }
227
228
229
230
231
232  private static void NetworkSubgraph(Document doc, String filepath, Map<String, ArrayList<Element>> edgeM,  Map<String, ArrayList<Elem
233                                      ArrayList<Element> edgeL,  ArrayList<Element> nodeL) throws IOException
234    {
235
236         NodeList nodes = doc.getElementsByTagName("node");
237         Element nd = null;
238        NodeList edges = doc.getElementsByTagName("edge");
239         Element ed = null;
240         Element ed1 = null;
241           String kE = null;
242           String kN = null;
243           String source = null;
244           String target = null;
245
246           //Filling the map made up edges for the subgraphs
247           for(int i=0; i<edges.getLength();i++)
248             {
249                 ed = (Element) edges.item(i);
250                 nodeL = new ArrayList<>();
251                 edgeL = new ArrayList<>();
252              for(int j=0; j<edges.getLength();j++)
```

```
253             {
254                  ed1 = (Element) edges.item(j);
255             if(!(ed1.getAttributes().getNamedItem("id").getTextContent().equals(ed.getAttributes().getNamedItem("id").getTextContent
256             {
257              if((ed1.getAttributes().getNamedItem("source").getTextContent().equals(ed.getAttributes().getNamedItem("source").getTe
258                  ||(ed1.getAttributes().getNamedItem("target").getTextContent().equals(ed.getAttributes().getNamedItem("source").get
259                {
260                 if(!edgeL.contains(ed1)) {
261                     edgeL.add(ed1);
262                     }
263
264                 if(ed1.getAttributes().getNamedItem("source").getTextContent().equals(ed.getAttributes().getNamedItem("source").getT
265                 {
266                   source = ed1.getAttributes().getNamedItem("target").getTextContent();
267                 }
268                 if(ed1.getAttributes().getNamedItem("target").getTextContent().equals(ed.getAttributes().getNamedItem("source").getT
269                 {
270                   target = ed1.getAttributes().getNamedItem("source").getTextContent();
271                 }
272                 if(((ed1.getAttributes().getNamedItem("source").getTextContent().equals(source)) || (ed1.getAttributes().getNamedIte
273                    && ((ed1.getAttributes().getNamedItem("target").getTextContent().equals(source)) || (ed1.getAttributes().getNamed
274                 {
275                     if(!edgeL.contains(ed1)) {
276                         edgeL.add(ed1);
277                         }
278                 }
279                  kE = ed.getAttributes().getNamedItem("source").getTextContent();
280
281                //Filling the map made up nodes for the subgraphs
282                  for(int k=0; k<nodes.getLength();k++)
283                  {
284                   nd = (Element) nodes.item(k);
285                   if((nd.getAttributes().getNamedItem("id").getTextContent().equals(ed1.getAttributes().getNamedItem("source").getTe
286                     ||(nd.getAttributes().getNamedItem("id").getTextContent().equals(ed1.getAttributes().getNamedItem("target").getT
287                           {
288                             if(!nodeL.contains(nd)) {
289                                 nodeL.add(nd);
290                                 }
291                            kN = ed.getAttributes().getNamedItem("source").getTextContent();
292                           }
293                  }
294                }
295              }
296           }
297             if(!nodeM.containsValue(nodeL) || !nodeM.containsKey(kN)){
298                   nodeM.put(kN, nodeL);
299             }
300            if(!edgeM.containsValue(edgeL) || !edgeM.containsKey(kE)){
301                   edgeM.put(kE, edgeL);
302             }
303        }
304
305             String iv = "\"";
306             int i=0;
307             BufferedWriter writer = null;
308        try {
309                writer = new BufferedWriter(new FileWriter(filepath));
310               writer.write("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n" +
311                            "<graphml xmlns=\"http://graphml.graphdrawing.org/xmlns\" "
312                          + "xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" "
313                          + "xsi:schemaLocation=\"http://graphml.graphdrawing.org/xmlns http://graphml.graphdrawing.org/xmlns/1.0/graph
314                          + " ");
315                writer.newLine();
316
317                for(String k: edgeM.keySet())
318                {
319                      writer.write("<graph id=\"G"+i+"\" edgedefault=\"directed\"> ");
320                      writer.newLine();
321                     for(Element n: nodeM.get(k))
322                     {
323                       writer.write( "<node id="+iv+n.getAttributes().getNamedItem("id").getTextContent()+iv+" labels="+iv+n.getAttribute
324                              +"<data key="+iv+n.getFirstChild().getAttributes().getNamedItem("key").getTextContent()+iv+">"+n.getElement
325                              +"<data key="+iv+n.getElementsByTagName("data").item(0).getNextSibling().getAttributes().getNamedItem("key'
326                              +"<data key="+iv+n.getElementsByTagName("data").item(0).getNextSibling().getNextSibling().getAttributes().g
327                              +"<data key="+iv+n.getElementsByTagName("data").item(0).getNextSibling().getNextSibling().getNextSibling().
328                              +"<data key="+iv+n.getElementsByTagName("data").item(0).getNextSibling().getNextSibling().getNextSibling().
```

```
                        +"</node>");
                    writer.newLine();
                }
                for(Element e: edgeM.get(k))
                {
                   writer.write("<edge source="+iv+e.getAttributes().getNamedItem("source").getTextContent()+iv+" target="+iv+e.getAtt
                           +"<data key="+iv+e.getFirstChild().getAttributes().getNamedItem("key").getTextContent()+iv+">"+e.getElements
                           +"<data key="+iv+e.getElementsByTagName("data").item(0).getNextSibling().getAttributes().getNamedItem("key")
                           +"</edge>");
                   writer.newLine();
                }
                   writer.write("</graph> ");
                   writer.newLine();
                   writer.newLine();
                i++;
            }
           writer.write("</graphm> ");
          writer.flush();

       } catch(Exception e) {
           // if any I/O error occurs
           e.printStackTrace();
       }finally {
           // releases system resources from the streams
           if(writer!=null)
             writer.close();
       }
   }
```

Appendix 1: ***Java program to create set of graphs from one graph in a graphml file***

```java
package Yves;

import java.io.File;
/*
 This program sets up a graphml file exported from the neo4j database by removing attributes from the eqdes so
 as to have only the Source and Target as attributes, as required by and thus can be ran on the gSpan algorithm
 */
public class SetUpFile {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

         String filePath = args[0];    //input file
            File xmlFile = new File(filePath);
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder;
            try {
                dBuilder = dbFactory.newDocumentBuilder();
                Document doc = dBuilder.parse(xmlFile);
                doc.getDocumentElement().normalize();

                //update attribute value
                removeAttributeValue(doc);

              //delete element
                deleteElement(doc);

                //write the updated document to file or console
                doc.getDocumentElement().normalize();
                TransformerFactory transformerFactory = TransformerFactory.newInstance();
                Transformer transformer = transformerFactory.newTransformer();
                DOMSource source = new DOMSource(doc);
                StreamResult result = new StreamResult(new File(args[1]));  //output file
                transformer.setOutputProperty(OutputKeys.INDENT, "yes");
                transformer.transform(source, result);
                System.out.println("Graphml file set up successfully");

            } catch (SAXException | ParserConfigurationException | IOException | TransformerException e1) {
                e1.printStackTrace();
            }
        }


    private static void deleteElement(Document doc) {
        NodeList nodes = doc.getElementsByTagName("node");
        Element nd = null;
        //loop for each node
        for(int i=0; i<nodes.getLength();i++){
            nd = (Element) nodes.item(i);
            Node dataNode = nd.getElementsByTagName("data").item(0);
           if(dataNode.getAttributes().getNamedItem("key").getTextContent().equals("dn")
                     || dataNode.getAttributes().getNamedItem("key").getTextContent().equals("color"))
            nd.removeChild(dataNode);
        }
    }

    private static void removeAttributeValue(Document doc) {
        NodeList edges = doc.getElementsByTagName("edge");
        Element ed = null;

        for(int i=0; i<edges.getLength();i++){
            ed = (Element) edges.item(i);
            ed.removeAttribute("id");
           ed.removeAttribute("label");
        }
    }

}
```

Appendix 2: ***Java program to setup input file for gSpan algorithm***

```java
package Roudolf;

import java.io.File;

/*
 This program modifies a given graphml file the output from the gSpan algorithm passed as a command line argument, by adding labels and other a
 recognised by Neo4j and thus cypher queries can be applied to the nodes
 */

public class ModifyDOM {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

         String filePath = args[0];    //input file
            File xmlFile = new File(filePath);
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder dBuilder;
            try {
                dBuilder = dbFactory.newDocumentBuilder();
                Document doc = dBuilder.parse(xmlFile);
                doc.getDocumentElement().normalize();

                //update attribute value
                updateAttributeValue(doc);

              //delete element
                deleteElement(doc);

                //write the updated document to file or console
                doc.getDocumentElement().normalize();
                TransformerFactory transformerFactory = TransformerFactory.newInstance();
                Transformer transformer = transformerFactory.newTransformer();
                DOMSource source = new DOMSource(doc);
                StreamResult result = new StreamResult(new File(args[1]));  //output file
                transformer.setOutputProperty(OutputKeys.INDENT, "yes");
                transformer.transform(source, result);
               System.out.println("Graphml file updated successfully");

          } catch (SAXException | ParserConfigurationException | IOException | TransformerException e1) {
              e1.printStackTrace();
          }
      }

  private static void deleteElement(Document doc) {
      NodeList nodes = doc.getElementsByTagName("node");
      Element nd = null;
      //loop for each node
      for(int i=0; i<nodes.getLength();i++){
          nd = (Element) nodes.item(i);
          Node dataNode = nd.getElementsByTagName("data").item(0);
         if(dataNode.getAttributes().getNamedItem("key").getTextContent().equals("dn")
                   || dataNode.getAttributes().getNamedItem("key").getTextContent().equals("color"))
          nd.removeChild(dataNode);
      }

  }

  private static void updateAttributeValue(Document doc) {
      NodeList nodes = doc.getElementsByTagName("node");
      Element nd = null;
      NodeList edges = doc.getElementsByTagName("edge");
      Element ed = null;

      for(int i=0; i<edges.getLength();i++){
          ed = (Element) edges.item(i);
          ed.getElementsByTagName("data").item(0).getAttributes().getNamedItem("key").setTextContent("label");
      }

          //loop for each node
          for(int i=0; i<nodes.getLength();i++){
              nd = (Element) nodes.item(i);
              String data = nd.getElementsByTagName("data").item(0).getFirstChild().getNodeValue();
                  nd.setAttribute("labels", data);
```

```
92
93                  Element datalabel= doc.createElement("data");
94                  datalabel.setAttribute("key", "labels");
95                  datalabel.appendChild(doc.createTextNode(data));
96                  nd.appendChild(datalabel);
97
98                  Element dataname= doc.createElement("data");
99                  dataname.setAttribute("key", "name");
100                 dataname.appendChild(doc.createTextNode(data));
101                 nd.appendChild(dataname);
102
103                 Element datasymbol= doc.createElement("data");
104                 datasymbol.setAttribute("key", "symbol");
105                 datasymbol.appendChild(doc.createTextNode(data));
106                 nd.appendChild(datasymbol);
107
108                 Element dataAN= doc.createElement("data");
109                 dataAN.setAttribute("key", "AN");
110                 dataAN.appendChild(doc.createTextNode(data));
111                 nd.appendChild(dataAN);
112
113                 Element dataID= doc.createElement("data");
114                 dataID.setAttribute("key", "ID");
115                 dataID.appendChild(doc.createTextNode(nd.getAttribute("id")));
116                 nd.appendChild(dataID);
117
118     }
119 }
120
121
122
```

Appendix 3: ***Java code to Modify (adding labels) output file from gSpan algorithm***