CSCI 3412 – Algorithms

Jeremy Diamond

Problem Set 3

Part I – Representing the Social Graph

The social graph is represented as a square 2d array of integers, called a collaboration matrix.A given cell of this array index [n,m] is the number of times character n has been in a comic with m. Any index [n,n] is set to zero. In this representation the books are no longer nodes and are instead a family of mutually connected characters. The exact value of these integers is not of interest to us but, is useful in other applications.

Part II – Six degrees of Spiderman

My method of the six degrees of spider man fills an array with all characters that have collaborated with him directly. All of these characters are then given a Spiderman number of 1. I then fill a second array with every character the characters in the first array collaborate with and give anyone whose number is still the default value of 7 (diameter of the graph) a 2. Repeat the above once more to find all Spiderman numbers (Maximum Spiderman number is a 3 for any connected node).

Part III – Implementation notes

- Read-in

My code was all written in my IDE of choice: Netbeans. This java code leans into the object oriented nature of java and the language's robust libraries. There are classes of "Book" and "Character" defined to hold all needed data for the two types of nodes. Each "Character has not only a name and UID (also a Spiderman number, more on this later) but an array called connections which 12942 bools, one for every comic. If a bool is true then the character appears in that book. Building this raw dataset is a O(n) process.

- Collaboration matrix

A 2d array of ints called colab is initialized with a size of 6486 by 6486. Three nested for loops seed the collaboration matrix.This is the longest runtime process in the program with a O(n^3) however the runtime is still manageable considering that the lowest level loop is only the highest number of collaborators which is significantly smaller than the number of total characters.

Part IV – Conclusions

Many of my processes and algorithms are really non optimal and, with a larger dataset, would probably brake down. However the connectedness of the graph is not lost on me and I will likely use more optimal versions of this idea in the future.

Part V – Code

- Book Class

```
package hw3;

/**
 *
 * @author jdiamond
 */
public class Book {
    public int number;
    public String name;


    public Book (int theNum, String theName)
    {
        number = theNum;
        name = theName;
    }

    public int getNum ()
    {
        return number;
    }

    public String getName ()
    {
        return name;
    }


}
```

-

- Character Class

```
package hw3;
import java.util.*;
/**
 *
 * @author jdiamond
 */
public class Character {

    //.memeber vars
    public int number;
    public String name;
    public int spiderNum = 7;//grater then the diamoiter of the graph
    public ArrayList connections = new ArrayList();//will be made of bools


    public Character (int theNum, String theName)
    {
        number = theNum;
        name = theName;
        for (int i = 0; i <= 12942; i++ )
        {
            connections.add(false);// places bools in arraylist
        }
    }

    public int getNum ()
    {
        return number;
    }

    public String getName ()
    {
        return name;
    }

}
```

- ReadIn Function

```
//based on https://www.caveofprogramming.com/java/java-file-reading-and-writing-files-in-java.html
  public static void ReadIn(ArrayList<Character> theChars, ArrayList<Book> theBooks)
  {
    //locals
    int spaceindex = 0;
    String tempStr = new String();
    int tempInt = 0;
    Character tempChar;
    Book tempBook;
    int charIndex;
    // The name of the file to open.
    String fileName = "porgat.txt";

    // This will reference one line at a time
    String line = null;

    try {
      // FileReader reads text files in the default encoding.
      FileReader fileReader =
        new FileReader(fileName);

      // Always wrap FileReader in BufferedReader.
      BufferedReader bufferedReader =
        new BufferedReader(fileReader);



      //logic for reading in characters
      line = bufferedReader.readLine();
      for (int i = 0; i < 2050; i++ )
      {
        line = bufferedReader.readLine();


        spaceindex = line.indexOf(' ');
        tempStr = line.substring(0, spaceindex);
        tempInt = Integer.parseInt(tempStr);

        tempStr = line.substring(spaceindex+2, (line.length()-1));

        tempChar = new Character(tempInt, tempStr);
        theChars.add(tempChar);


        //System.out.println(line);
      }
      for (int i = 0; i < 6486-2050; i++ )
      {
        line = bufferedReader.readLine();


        spaceindex = line.indexOf(' ');
        tempStr = line.substring(0, spaceindex);
        tempInt = Integer.parseInt(tempStr);

        tempStr = line.substring(spaceindex+2, (line.length()-1));

        tempChar = new Character(tempInt, tempStr);
        theChars.add(tempChar);
        //System.out.println(line);
    }

    //logic for reading in books
    for (int i = 0; i < 12942; i++ )
    {
        line = bufferedReader.readLine();


        spaceindex = line.indexOf(' ');
        tempStr = line.substring(0, spaceindex);
        tempInt = Integer.parseInt(tempStr);

        tempStr = line.substring(spaceindex+2, (line.length()-1));

        tempBook = new Book(tempInt, tempStr);
        theBooks.add(tempBook);
    }

     //logic for reading in connections
    line = bufferedReader.readLine();
     while((line = bufferedReader.readLine()) != null) {

       spaceindex = line.indexOf(' ');
        tempStr = line.substring(0, spaceindex);
        charIndex = Integer.parseInt(tempStr);
        tempChar = theChars.get(charIndex-1);

        line = line.substring(spaceindex+1, (line.length()));
        do{
            if (line.contains(" ")){
                spaceindex = line.indexOf(' ');
                tempStr = line.substring(0, spaceindex);
                tempInt = Integer.parseInt(tempStr);
                tempChar.connections.set(tempInt-6487,true);
                line = line.substring(spaceindex+1, (line.length()));
                if (line.contains(" ") == false ){
                    tempInt = Integer.parseInt(line);
                    tempChar.connections.set(tempInt-6487,true);
                }
            }
            else{

                tempInt = Integer.parseInt(line);
                tempChar.connections.set(tempInt-6487,true);
            }


        } while (line.contains(" "));
    }

    // Always close files.
    bufferedReader.close();
}
catch(FileNotFoundException ex) {
    System.out.println(
        "Unable to open file '" +
        fileName + "'");
}
catch(IOException ex) {
        System.out.println(
            "Error reading file '"
            + fileName + "'");
        // Or we could just do this:
        // ex.printStackTrace();
    }
}
```

- Main Function

```
package hw3;


/**
 *
 * @author jdiamond
 */

import java.io.*;
import java.util.*;
public class HW3 {

   /**
    * @param args the command line arguments
    */
   public static void main(String[] args) {
      // TODO code application logic here
      //Character test = new Character(1,"dfjhsdfkjhg");
      //Book test1 = new Book(1,"dfjhsdfkjhg");
      ArrayList<Character> characters = new ArrayList<Character>();
      ArrayList<Book> books = new ArrayList<Book>();

      ReadIn(characters, books);

      int[][] colab = new int[6486][6486];
      ArrayList tempInts;

      ArrayList nextInts;
      int spiderAssign;

      int choise = 1;
      BufferedReader readInt = new BufferedReader(new InputStreamReader(System.in));
      String choiseString = new String();

      for (int i = 0; i < 12937; i++) //i will be the comic
      {
         tempInts = new ArrayList();

         for (int j = 0; j < 6486; j++) //j will be the character
         {
           if ((boolean)characters.get(j).connections.get(i))
           {
              tempInts.add(j); //tempints now is all the charaters in comic j
           }
         }

         for (int k = 0; k <tempInts.size(); k++)
         {
            for (int l = 0; l <tempInts.size(); l++)
            {
               if (k != l)
                  colab[(int)tempInts.get(k)][(int)tempInts.get(l)]++;
            }
         }

      }


      System.out.println("colab matrix built");
    tempInts = new ArrayList();
    nextInts = new ArrayList();
    tempInts.add(5305);
    spiderAssign = 1;
    characters.get(5305).spiderNum = 0;

    while(spiderAssign <= 3){
    for (int i = 0; i < tempInts.size(); i++)
    {
        for (int j = 0; j < 6486; j++)
        {
            if ((colab[(int)tempInts.get(i)][j] != 0) && (characters.get(j).spiderNum > spiderAssign)){
                characters.get(j).spiderNum = spiderAssign;
                nextInts.add(j);
            }
        }
    }
    tempInts.clear();
    tempInts.addAll(nextInts);
    nextInts = new ArrayList();
    spiderAssign++;
}
    char testChar = characters.get(2051).name.charAt(0);
    while (choise != 0) {
      try{
          System.out.println("please enter a character number from 1 to 6486. enter 0 to exit");

            choiseString = readInt.readLine();
            choise = Integer.parseInt(choiseString);
            if (choise != 0)
            System.out.println(choise + " " + characters.get(choise-1).name + " : " + characters.get(choise-1).spiderNum );
          }
      catch(Exception e)
        {
            e.printStackTrace();
        }
    }

}
```