**CS323-22: Project 9.1 (JAVA)**

**Yida Tao**

**Due date: Dec. 12, 2017**

Algorithm Steps:

step 0: numNode <-- get from input file

- dynamically allocate adjacencyMatrix // initiallize to 0

- load adjacencyMatrix from the input pairs

// Note: for one edge from the input, you need to fill two cells in the matrix

// 1 5 means an edge <1, 5> and an edge <5, 1>

- print adjacencyMatrix

- constructNodeList

- newColor <-- 0

// get new new color (using 1, 2, 3, ... as the color scheme)

step 1: newColor ++

currentNode <-- listHead's next // the node after the dummy node

//use the newColor to color as many un-colored nodes as possible

Step 2: if currentNode.Color == 0 // if currentNode is an uncolor node

\*and\* checkAdjacent (currentNode.ID, newColor) == 0

// currentNode does not have any adjacent node

// that are already colored with the newColor

// check adjancyMatrix

then

- adjacentMatrix[currentNode][currentNode] <-- newColor

- currentNode.Color <-- newColor

- currentNode <-- currentNode's next

Step 3: repeat step 2 until the end of the node list.

Step 4: print adjacencyMatrix

Step 5: repeat steps 1 to 3 until all nodes are colored.

Step 6: print newcolor // with indication, i.e., number of colors used is ...

print adjacencyMatrix

step 7: close all files

**Source Code**

**public** **class** node {

**int** nodeId;

**int** color = 0;

**int** numEdge = 0;

node next = **null**;

**public** node(){

nodeId = 0;

}

**public** node(**int** i, **int** num){

nodeId = i;

numEdge = num;

}

**public** **void** printNode(){

System.***out***.print(nodeId + ":" + numEdge + ":" + color);

}

}

**public** **class** graphColoring {

**int** adjMatrix[][];

node listHead;

**int** uncolorNode;

**int** newColor;

**int** numNode;

**public** graphColoring(**int** size){

numNode = size;

uncolorNode = size;

newColor = 0;

adjMatrix = **new** **int**[size + 1][size + 1];

listHead = **new** node();

**for**(**int** i = 0; i < size + 1; i++){

**for**(**int** j = 0; j < size + 1; j++){

adjMatrix[i][j] = 0;

}

}

}

**public** **void** loadMatrix(**int** n1, **int** n2){

adjMatrix[n1][n2] = 1;

adjMatrix[n2][n1] = 1;

}

**public** **void** printMatrix(){

**for**(**int** i = 1; i < numNode + 1; i++){

**for**(**int** j = 1; j < numNode + 1; j++){

System.***out***.print(adjMatrix[i][j] + " ");

}

System.***out***.println();

}

}

**public** **void** insertOneNode(node listh , node n){

node curr = listh;

**while**(curr.next != **null** && n.numEdge > curr.next.numEdge){

curr = curr.next;

}

n.next = curr.next;

curr.next = n;

}

**public** **void** constructNodeList(){

**int** edgecount = 0;

**for**(**int** i = 1; i < numNode + 1; i++){

**for**(**int** j = 1; j < numNode + 1; j++){

**if**(i != j){

edgecount += adjMatrix[i][j];

}

}

insertOneNode(listHead, **new** node(i,edgecount));

edgecount = 0;

}

}

**public** **void** printList(){

node curr = listHead;

**while**(curr.next != **null**){

curr.printNode();

curr = curr.next;

System.***out***.print("-->");

}

System.***out***.println("endlist");

}

**public** **int** checkAdjacent(**int** nodeId, **int** color){

**int** num = 0;

**for**(**int** i = 1; i < numNode + 1; i++){

//there is a edge between nodeid and i, and the color of i equal color

**if**(adjMatrix[nodeId][i] == 1 && adjMatrix[i][i] == color){

num++;

}

}

**return** num;

}

}

**import** java.io.\*;

**import** java.util.Scanner;

**public** **class** project91 {

**public** **static** **void** main(String[] args) {

Scanner scan = **null**;

graphColoring gc = **null**;

node curr = **null**;

FileOutputStream fos1 =**null**;

**try** {

fos1 = **new** FileOutputStream(args[1]);

System.*setOut*(**new** PrintStream(fos1));

} **catch** (FileNotFoundException e1) {

e1.printStackTrace();

}

**int** n1 = 0;

**int** n2 = 0;

//step 0 -- start

**try** {

scan = **new** Scanner(**new** File(args[0]));

n1 = scan.nextInt();

gc = **new** graphColoring(n1);

**while**(scan.hasNextInt()){

n1 = scan.nextInt();

n2 = scan.nextInt();

System.***out***.println(n1 + " " + n2);

gc.loadMatrix(n1, n2);

}

} **catch** (FileNotFoundException e) {

e.printStackTrace();

} **finally**{

**if**(scan != **null**) scan.close();

}

System.***out***.println("initial adjacencyMatrix and nodelist:");

gc.printMatrix();

gc.constructNodeList();

gc.printList();

//step 0 -- end

//step 5

**while**(gc.uncolorNode != 0){

//step 1

System.***out***.println();

System.***out***.println("trial " + gc.newColor);

curr = gc.listHead.next;

gc.newColor++;

//step 2-3

**while**(**true**){

**if**(curr.color == 0 && gc.checkAdjacent(curr.nodeId, gc.newColor) == 0){

System.***out***.print(curr.nodeId + " ");

curr.color = gc.newColor;

gc.adjMatrix[curr.nodeId][curr.nodeId] = gc.newColor;

gc.uncolorNode--;

}

**if**(curr.next == **null**) **break**;

curr = curr.next;

}

System.***out***.println("are colored with " + gc.newColor);

//step 4

gc.printMatrix();

}

//step 6

System.***out***.println();

System.***out***.println("number of colors used:" + gc.newColor);

System.***out***.println("Final adjacentcyMatrix with the assign colors in the diagonal of the matrix");

gc.printMatrix();

gc.printList();

//step 7

**try** {

fos1.close();

System.*setOut*(**new** PrintStream(**new** FileOutputStream(FileDescriptor.***out***)));

} **catch** (IOException e) {

e.printStackTrace();

}

System.***out***.println("Done");

}

}

**Input**

19

1 10

1 17

1 16

1 4

5 2

4 2

6 2

6 3

9 3

12 3

3 4

16 4

15 4

11 4

8 4

9 4

10 4

18 5

5 7

6 5

13 5

14 5

14 6

19 6

12 6

18 7

13 7

12 9

10 16

8 11

14 13

17 16

**Output 1**

1 10

1 17

1 16

1 4

5 2

4 2

6 2

6 3

9 3

12 3

3 4

16 4

15 4

11 4

8 4

9 4

10 4

18 5

5 7

6 5

13 5

14 5

14 6

19 6

12 6

18 7

13 7

12 9

10 16

8 11

14 13

17 16

initial adjacencyMatrix and nodelist:

0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0

0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 1 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0

1 1 1 0 0 0 0 1 1 1 1 0 0 0 1 1 0 0 0

0 1 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 1 0

0 1 1 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1

0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0

0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0

0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0

1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0

0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0

0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0

0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0

0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0

0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0

0:0:0-->19:1:0-->15:1:0-->18:2:0-->17:2:0-->11:2:0-->8:2:0-->14:3:0-->13:3:0-->12:3:0-->10:3:0-->9:3:0-->7:3:0-->2:3:0-->16:4:0-->3:4:0-->1:4:0-->6:6:0-->5:6:0-->endlist

trial 0

19 15 18 17 11 14 12 10 2 are colored with 1

0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0

0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 1 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0

1 1 1 0 0 0 0 1 1 1 1 0 0 0 1 1 0 0 0

0 1 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 1 0

0 1 1 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1

0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0

0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0

0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0

1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0

0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0

0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0

0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 0

0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0

0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0

1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0

0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0

0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1

trial 1

8 13 9 16 6 are colored with 2

0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0

0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 1 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0

1 1 1 0 0 0 0 1 1 1 1 0 0 0 1 1 0 0 0

0 1 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 1 0

0 1 1 0 1 2 0 0 0 0 0 1 0 1 0 0 0 0 1

0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0

0 0 0 1 0 0 0 2 0 0 1 0 0 0 0 0 0 0 0

0 0 1 1 0 0 0 0 2 0 0 1 0 0 0 0 0 0 0

1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0

0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0

0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0

0 0 0 0 1 0 1 0 0 0 0 0 2 1 0 0 0 0 0

0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0

0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0

1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 2 1 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0

0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0

0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1

trial 2

7 3 1 are colored with 3

3 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0

0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 3 1 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0

1 1 1 0 0 0 0 1 1 1 1 0 0 0 1 1 0 0 0

0 1 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 1 0

0 1 1 0 1 2 0 0 0 0 0 1 0 1 0 0 0 0 1

0 0 0 0 1 0 3 0 0 0 0 0 1 0 0 0 0 1 0

0 0 0 1 0 0 0 2 0 0 1 0 0 0 0 0 0 0 0

0 0 1 1 0 0 0 0 2 0 0 1 0 0 0 0 0 0 0

1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0

0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0

0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0

0 0 0 0 1 0 1 0 0 0 0 0 2 1 0 0 0 0 0

0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0

0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0

1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 2 1 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0

0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0

0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1

trial 3

5 4 are colored with 4

3 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0

0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 3 1 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0

1 1 1 4 0 0 0 1 1 1 1 0 0 0 1 1 0 0 0

0 1 0 0 4 1 1 0 0 0 0 0 1 1 0 0 0 1 0

0 1 1 0 1 2 0 0 0 0 0 1 0 1 0 0 0 0 1

0 0 0 0 1 0 3 0 0 0 0 0 1 0 0 0 0 1 0

0 0 0 1 0 0 0 2 0 0 1 0 0 0 0 0 0 0 0

0 0 1 1 0 0 0 0 2 0 0 1 0 0 0 0 0 0 0

1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0

0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0

0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0

0 0 0 0 1 0 1 0 0 0 0 0 2 1 0 0 0 0 0

0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0

0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0

1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 2 1 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0

0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0

0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1

number of colors used:4

Final adjacentcyMatrix with the assign colors in the diagonal of the matrix

3 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0

0 1 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 3 1 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0

1 1 1 4 0 0 0 1 1 1 1 0 0 0 1 1 0 0 0

0 1 0 0 4 1 1 0 0 0 0 0 1 1 0 0 0 1 0

0 1 1 0 1 2 0 0 0 0 0 1 0 1 0 0 0 0 1

0 0 0 0 1 0 3 0 0 0 0 0 1 0 0 0 0 1 0

0 0 0 1 0 0 0 2 0 0 1 0 0 0 0 0 0 0 0

0 0 1 1 0 0 0 0 2 0 0 1 0 0 0 0 0 0 0

1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0

0 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0

0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0

0 0 0 0 1 0 1 0 0 0 0 0 2 1 0 0 0 0 0

0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0

0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0

1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 2 1 0 0

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0

0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0

0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1

0:0:0-->19:1:1-->15:1:1-->18:2:1-->17:2:1-->11:2:1-->8:2:2-->14:3:1-->13:3:2-->12:3:1-->10:3:1-->9:3:2-->7:3:3-->2:3:1-->16:4:2-->3:4:3-->1:4:3-->6:6:2-->5:6:4-->endlist