CSC 323 Spring 2015: GraphColoring (C++)

Ravi Patel

April 28, 2015

**Algorithm steps:**

step 0: prepare and initialize everything

- load adjacentMatrix from the input pairs

- print the matrix

- set nodeColor array to zero

- newColor <-- 0

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

step 1: newColor ++

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

Step 2: newNode <-- get an uncolor node

if newNode does not have any adjacent node that are already colored with the newColor

nodeColor[newNode} <-- newColor

Step 3: repeat step 2 until all uncolored nodes are checked.

// for debugging purpose, ...

Step 4: print (to save paper)

(a) print the newColor, one text line

(b) print the index of nodeColor array in one text line;

(c) print the nodeColor lineup with the index above, in one text line.

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

Step 6: copy nodeColor[i] to adjacnetMatrix[i][i]

Step 7: print newColor

print adjecantMatrix

Source Code:

#include <iostream>

#include <fstream>

using namespace std;

bool checkColor(int\* nodeColor, int NumberOfNodes){

for (int row=1; row < NumberOfNodes ; row++) {

if(nodeColor[row]==0){

return true;

}

}

return false;

}

void printNodeColor(int\* nodeColor, int NumberOfNodes, ostream& outputFile){

for (int row=1; row < NumberOfNodes ; row++) {

outputFile<<nodeColor[row];

}

outputFile<<endl;

}

int main( int argc , char\* argv[]){

ifstream inputFile;

inputFile.open(argv[1]);

ofstream outputFile;

outputFile.open(argv[2]);

int NumberOfNodes, from, to;

inputFile>>NumberOfNodes;

cout<<NumberOfNodes<<endl;

int\*\* matrix;

int\* nodeColor;

NumberOfNodes++;

matrix = new int\* [NumberOfNodes];

nodeColor = new int [NumberOfNodes];

int newColor = 0;

int newNode=1;

for(int row=0; row<NumberOfNodes; row++){

matrix[row] = new int[NumberOfNodes];

for(int col=0; col<NumberOfNodes; col++){

matrix[row][col]=0;

}

}

for(int row=0; row < NumberOfNodes; row++){

nodeColor[row]=0;

}

while(inputFile>>from && inputFile>>to){

for (int row=0; row < NumberOfNodes; row++) {

for(int col=0; col < NumberOfNodes; col++){

if(row==from && col==to){

matrix[from][to]=1;

}

}

}

}

while(checkColor(nodeColor, NumberOfNodes)){

newColor++;

for(int row=1; row < NumberOfNodes; row++){

if(nodeColor[row]==0){

newNode = row;

}

for (int col=1; col < NumberOfNodes; col++) {

if (matrix[row][col] == 0 || matrix[row][col] == newColor ) {

break;

}

nodeColor[newNode] = newColor;

}

}

outputFile<<"New Color: "<<newColor<<endl;

outputFile<<"Node Color Array: ";

printNodeColor(nodeColor, NumberOfNodes, outputFile);

}

for(int row=0; row<NumberOfNodes; row++){

for(int col=0; col<NumberOfNodes; col++){

if(row == col){

matrix[row][col]=nodeColor[row];

}

outputFile<<matrix[row][col]<<" ";

}

outputFile<<endl;

}

}

Input1:

10

2 7 7 2 4 7 7 4 2 4 4 2 1 7 7 1 4 1 1 4 4 10 10 4 2 5 5 2 2 6 6 2 2 10 10 2 5 6 6 5 10 6 6 10 10 9 9 10 10 1 1 10 1 9 9 1 1 8 8 1 1 3 3 1 3 8 8 3 8 9 9 8 3 9 9 3

Output1:

New Color: 1

Node Color Array: 0000000000

New Color: 2

Node Color Array: 0022002222

New Color: 3

Node Color Array: 0322032222

New Color: 4

Node Color Array: 4322432222

0 0 0 0 0 0 0 0 0 0 0

0 4 0 1 1 0 0 1 1 1 1

0 0 3 0 1 1 1 1 0 0 1

0 1 0 2 0 0 0 0 1 1 0

0 1 1 0 2 0 0 1 0 0 1

0 0 1 0 0 4 1 0 0 0 0

0 0 1 0 0 1 3 0 0 0 1

0 1 1 0 1 0 0 2 0 0 0

0 1 0 1 0 0 0 0 2 1 0

0 1 0 1 0 0 0 0 1 2 1

0 1 1 0 1 0 1 0 0 1 2

Input2:

20

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

Output2:

New Color: 1

Node Color Array: 00000000000000000000

New Color: 2

Node Color Array: 00020000020000022000

New Color: 3

Node Color Array: 00320000320000322000

New Color: 4

Node Color Array: 04320004320004322000

New Color: 5

Node Color Array: 54320054320054322000

New Color: 6

Node Color Array: 54320654320654322006

New Color: 7

Node Color Array: 54327654327654322076

New Color: 8

Node Color Array: 54327654327654322876

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

1 5 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 0

1 0 4 0 1 1 1 0 0 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 0

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

1 0 1 0 0 7 1 1 0 0 0 0 0 1 1 0 0 0 1 0 0

0 0 1 1 0 1 6 0 0 0 0 0 1 0 1 0 0 0 0 1 0

0 0 0 0 0 1 0 5 0 0 0 0 0 1 0 0 0 0 1 0 0

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

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

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

0 0 0 0 1 0 0 0 1 0 0 7 0 0 0 0 0 0 0 0 0

0 0 0 1 0 0 1 0 0 1 0 0 6 0 0 0 0 0 0 0 0

0 0 0 0 0 1 0 1 0 0 0 0 0 5 1 0 0 0 0 0 0

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

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

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

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

1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 8 0 0

0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 7 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6

Input3:

10

1 8 8 1 1 9 9 1 1 10 10 1 1 4 4 1 9 8 8 9 3 8 8 3 9 3 3 9 9 10 10 9 10 4 4 10 4 7 7 4 4 2 2 4 10 2 2 10 10 6 6 10 2 6 6 2 5 6 6 5

Output3:

New Color: 1

Node Color Array: 0000000000

New Color: 2

Node Color Array: 2000000200

New Color: 3

Node Color Array: 2000003203

New Color: 4

Node Color Array: 2000043243

New Color: 5

Node Color Array: 2000553243

New Color: 6

Node Color Array: 2006653243

New Color: 7

Node Color Array: 2077653243

New Color: 8

Node Color Array: 2887653243

0 0 0 0 0 0 0 0 0 0 0

0 2 0 0 0 0 0 0 1 1 0

0 0 8 0 0 1 0 0 0 0 0

0 0 0 8 0 0 0 0 0 1 0

0 0 0 0 7 0 0 0 0 0 1

0 0 0 0 0 6 0 0 0 0 0

0 0 0 0 0 0 5 0 0 0 0

0 0 0 0 0 0 0 3 0 0 0

0 1 0 0 0 0 0 0 2 0 0

0 0 0 1 0 0 0 0 0 4 1

0 0 1 0 1 0 0 0 0 0 3