University of Washington
Department of Computer Science and Engineering
CSE 417, Winter 2020
Yiliang Wang

February 5, 2020

Homework 4 Problem 6

Programming Problem 6 Extra Credit (10 points):

Implement a graph coloring algorithm that performs better than the simple greedy coloring algorithm. Compare your results with the maximum degree of the graph, along with the simple greedy algorithm. You should report results for values of p in the range 0.002 and 0.02. How many colors are needed on the average.

Answer:

```
import java.util.HashMap;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Random;

public class Graph{
    public int n;
    public int m;
    public double p;
    public HashMap<Integer, ArrayList<Integer>> vertexMap = new HashMap<>();
    public ArrayList<int[]> edgeList = new ArrayList<>();
    public int[] degrees;

    public Graph(int n, double p) {
        this.n = n;
        this.p = p;
        m = (int) (n * (n - 1) * p / 2.0);
        degrees = new int[this.n];
        getRandomGraph();
    }

    void getRandomGraph(){
        Random rand = new Random();
        for (int i = 0; i < n - 1; i++)
            for (int j = i + 1; j < n; j++)
                if (rand.nextDouble() < p){
                    AddEdge(i, j);
                }
```

```
    }

void AddEdge(int i, int j) {
    if(!vertexMap.containsKey(i)){
        vertexMap.put(i, new ArrayList<>());
    }
    vertexMap.get(i).add(j);
    degrees[i]++;

    if(!vertexMap.containsKey(j)){
        vertexMap.put(j, new ArrayList<>());
    }
    vertexMap.get(j).add(i);
    degrees[j]++;
}

public Integer WelshPowellColoring() {

    //sort the vertices in order of descending degrees
    Pair[] indices = new Pair[n];
    for (int i = 0; i < n; i++) {
        indices[i] = new Pair(i, degrees[i]);
    }
    Arrays.sort(indices);
    ArrayList<?> sortedVertex[]= new ArrayList[n];

    // color the noncolored vertices based on the degree order
    // color all the vertices that are not connected to the
    // coloured vertex with the same color.
    for(int i = 0; i<n; i++){
        if (indices[i].value != 0){
            sortedVertex[i] = vertexMap.get(indices[i].index);
        }
        else{
            sortedVertex[i] = new ArrayList<Integer>();
        }
    }

    int[] colors = new int[n];
    Arrays.fill(colors, -1);
    int maxColor = 0;
    for (int i = 0; i < n; i++){
        if (colors[i] != -1){
            continue;
        }
```

```
            else{
                maxColor ++;
                colors[i] = maxColor;
                for (int j = i+1; j < n; j++){
                    if (!(sortedVertex[i].contains(j)) && (colors[j] == -1)){
                        // make sure j's neighbor are not colored this color
                        boolean flag = true;
                        for(int k = 0; k < sortedVertex[j].size(); k++) {
                            flag &=
                            (colors[(int) sortedVertex[j].get(k)]
                            != maxColor);
                        }
                        if(flag)
                            colors[j] = maxColor;
                    }
                    else{
                        continue;
                    }
                }
            }
        }
        //System.out.println(Arrays.toString(colors));
        return maxColor;
    }

    public Integer greedyColoring(){
        int[] color = new int[n];
        int maxColor = 1;
        Arrays.fill(color, -1);
        for (int i = 0; i < n; i++){
            if (color[i] == -1){
                int[] colorToChoose = new int[maxColor];
                if(vertexMap.containsKey(i)){
                    for (int j:vertexMap.get(i)){
                        if(color[j]!= -1){
                            colorToChoose[color[j]] = 1;
                        }
                    }
                }
                int k = 0;
                while(k < maxColor){
                    if(colorToChoose[k] == 0){
                        color[i] = k;
                        break;
                    }
```

```
                    k++;
                }
                if(k == maxColor){
                    maxColor ++;
                    color[i] = maxColor-1;
                }
            }
        }
        return maxColor;
    }

    public static void main(String[] args){
        //Graph g = new Graph(5, 0.8);
        //System.out.println(g.vertexMap);
        //System.out.print(g.WelshPowellColoring());

        for(double i = 0.002; i <= 0.021; i += 0.001){
            int wpcolor = 0;
            int gdcolor = 0;
            for(int j = 0; j <= 101; j++){
                Graph g = new Graph(1000, i);
                wpcolor += g.WelshPowellColoring();
                gdcolor += g.greedyColoring();
            }
            System.out.printf("␣n:%d,␣p:%.3f\n" +
                              "avgWelshPowellColorNumber:%.2f␣" +
                              "avgGreedyColorNumber%.2f\n",
                              1000, i, wpcolor/100.0, gdcolor/100.0);
        }


    }
}

class Pair implements Comparable<Pair> {
    public final int index;
    public final int value;

    public Pair(int index, int value) {
        this.index = index;
        this.value = value;
    }

    @Override
    public int compareTo(Pair other) {
```

```
        return other.value - this.value;
    }

    /*results:
     n:1000, p:0.002
    avgWelshPowellColorNumber:3.28 avgGreedyColorNumber4.28
     n:1000, p:0.003
    avgWelshPowellColorNumber:4.07 avgGreedyColorNumber5.09
     n:1000, p:0.004
    avgWelshPowellColorNumber:4.39 avgGreedyColorNumber5.42
     n:1000, p:0.005
    avgWelshPowellColorNumber:5.03 avgGreedyColorNumber6.14
     n:1000, p:0.006
    avgWelshPowellColorNumber:5.26 avgGreedyColorNumber6.42
     n:1000, p:0.007
    avgWelshPowellColorNumber:5.77 avgGreedyColorNumber7.04
     n:1000, p:0.008
    avgWelshPowellColorNumber:6.14 avgGreedyColorNumber7.25
     n:1000, p:0.009
    avgWelshPowellColorNumber:6.49 avgGreedyColorNumber7.68
     n:1000, p:0.010
    avgWelshPowellColorNumber:6.96 avgGreedyColorNumber8.24
     n:1000, p:0.011
    avgWelshPowellColorNumber:7.27 avgGreedyColorNumber8.50
     n:1000, p:0.012
    avgWelshPowellColorNumber:7.50 avgGreedyColorNumber8.94
     n:1000, p:0.013
    avgWelshPowellColorNumber:8.09 avgGreedyColorNumber9.19
     n:1000, p:0.014
    avgWelshPowellColorNumber:8.31 avgGreedyColorNumber9.57
     n:1000, p:0.015
    avgWelshPowellColorNumber:8.60 avgGreedyColorNumber9.94
     n:1000, p:0.016
    avgWelshPowellColorNumber:9.06 avgGreedyColorNumber10.27
     n:1000, p:0.017
    avgWelshPowellColorNumber:9.35 avgGreedyColorNumber10.57
     n:1000, p:0.018
    avgWelshPowellColorNumber:9.63 avgGreedyColorNumber11.01
     n:1000, p:0.019
    avgWelshPowellColorNumber:10.01 avgGreedyColorNumber11.28
     n:1000, p:0.020
    avgWelshPowellColorNumber:10.24 avgGreedyColorNumber11.55

    */
}
```