Brélaz's Graph Coloring Algorithm

Ming Yuan, Aaron Devlin

University of Washington

TCSS 543

IMPLEMENATION ANALYSIS

1. (10 points) In the worst case, Brélaz's Dsatur algorithm has $O(|V|^2 \cdot |E|)$ running time using a naïve implementation. Describe how to improve this by using a careful choice of data structures and supporting algorithms.

A TreeSet can be utilized to store the vertices in the graph which will be colored.   All the colors which can be used stored as a HashSet. With TreeSets, the complexity for READ, INSERT, & DELETE are O(log |V|). So, when finding a node with the maximum saturation degree, we are finding the last element in the TreeSet, which is the farthest to the right in the tree. And every time we color a new node, we do not need to compute the saturation degree of every node (whose time complexity is O(|V|*|E|)) again. We only need to update the saturation degree and the colors all the neighbors have used of the adjacent nodes of the colored one, whose time complexity is O(|E|*log|V|).

We have store the graph by store its vertices in a HashMap, and store its edges by store all the adjacent nodes of each vertex in another HashMap. For every vertex, we store the information about its saturation degree, adjacent degree, the color used to color it, the colors of all its neighbors and its label, so that we can easy to update related information every time we color a new node. And the rule it is sorted in TreeSet is that: all the nodes are sorted increasing via the order of saturation degree, and the nodes with the same saturation degree will be sorted increasing via the order of their adjacent degree.

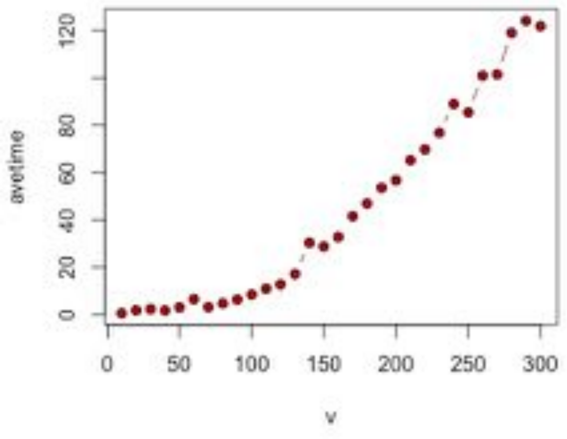
2. (10 points) Implement Brélaz's Dsatur algorithm. What is the running time complexity using your choice of data structures?

When updating the color of a node, the adjacent vertices must be updated. In this implementation, this update takes O(|E|log|V|). This is because we need to check all its neighbors whose time complexity is O(|E|), and for each neighbor, we need to update its adjacent colors and saturation degree whose time complexity is O(1), and this implementation will make the TreeSet adjust the order of its nodes, whose time complexity is O(log|V|). And to extract the node with maximal saturation degree and the maximal adjacent degree, we only need to spend O(log|V|) because of the structure of a TreeSet. This is already being carried out |V| times. This leads to a running-time complexity of O(|V|*(|E|*log|V|+log|V|))=O(|V|*|E|*log(|V|)).

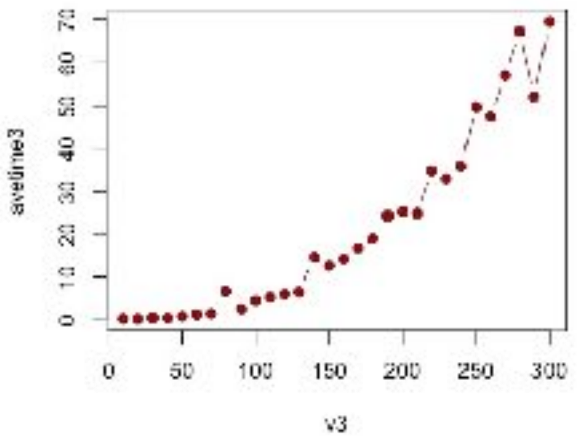
3. (10 points) Generate random graphs using the strategy described in Section 3.1 of Brélaz's paper, with $|V|$ = 10, 20, 30, ... vertices (try to go as far as possible). Apply Brélaz's Dsatur algorithm to color them and measure the running time (for each value of

|*V*|, measure the time needed to color 100 random graphs of that size at once, then divide the time by 100). Plot your results and compare them against your asymptotic worst-case analysis above. What can you say about the observed (i.e. average) running time?

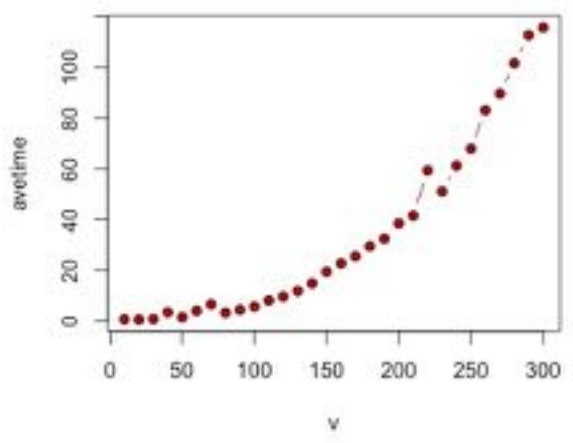
A plot is created based on first two groups of data. These groups have similar density of edges so the influence to running time and number of colors used is correlated to the number of vertices. Since the results of the four groups are similar, the images plotted for the third group were not included. The images generated by forth group are included since the data points have a rougher upward trend. This is most likely due to the density of edges of this group being very low, which will lead to a result that may be unstable.



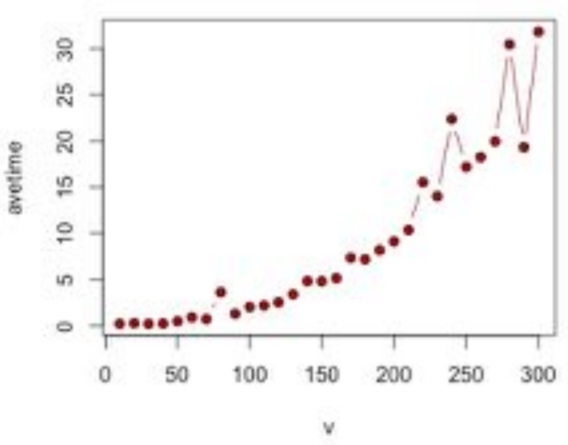
Group 1: |V| vs. Average Coloring Time for 100 Random Graphs



Group 2: |V| vs. Average Coloring Time for 100 Random Graphs

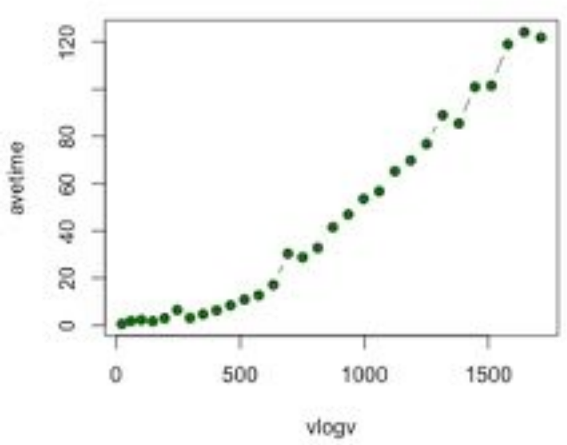


Group 3: |V| vs. Average Coloring Time for 100 Random Graphs

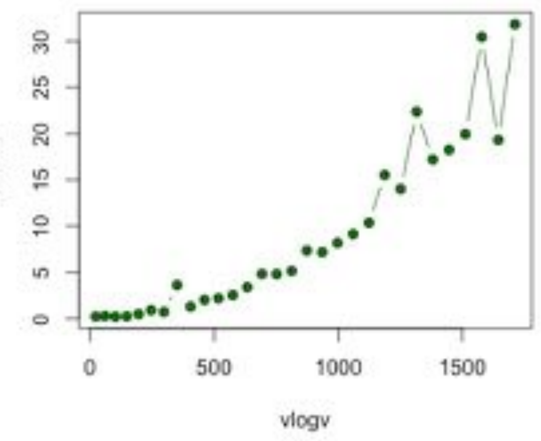


Group 4: |V| vs. Average Coloring Time for 100 Random Graphs

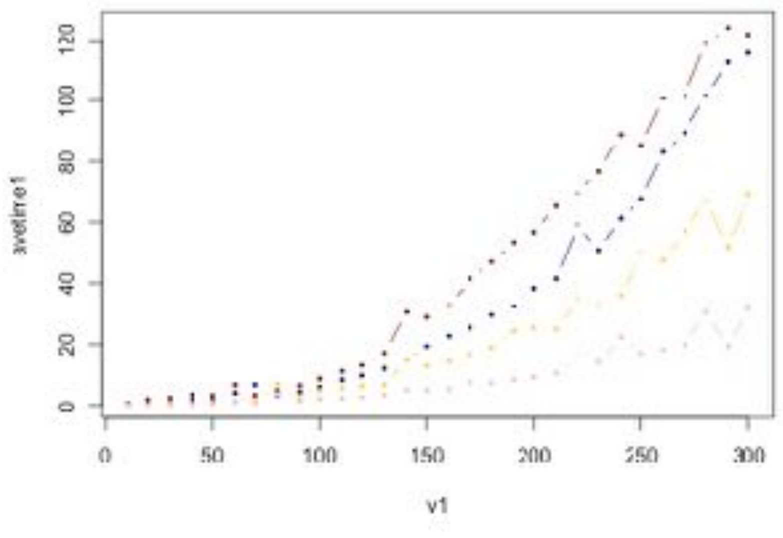
The relationship between |V|log|V| and running time has been plotted. But although the density of edges in one group doesn't change, the number of edges will also increase as the number of nodes increase. So the form of the curve doesn't change a lot.



Group 1: |V|log|V| vs. Average Coloring Time for 100 Random Graphs



Group 4: |V|log|V| vs. Average Coloring Time for 100 Random Graphs

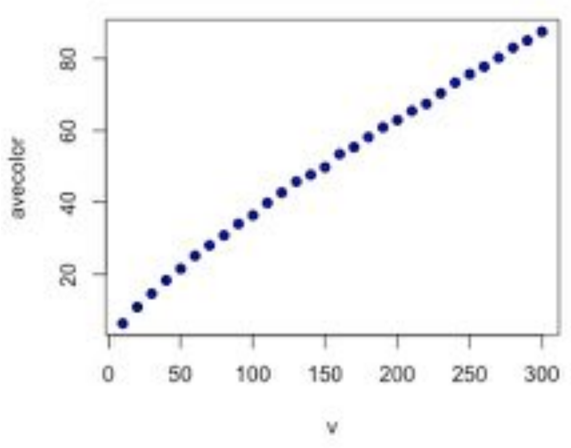


|V| vs. Average Coloring Time for 100 Random Graphs for 4 groups.(The red one represents group1, the blue one represents group2, the yellow one represents group3, and the pink one represents group4)

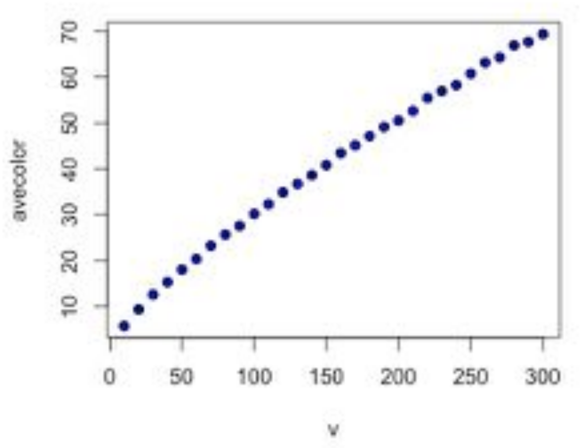
The relationship between edges and running time images were not plotted since the data is not big enough (only four groups). But by observing the results directly, we can come to the conclusion that they are proportional, or at least the running time will decrease while the density of edges decreases.

4. (5 bonus points) Plot the estimated (i.e. observed on average) minimum number of colors needed to color the vertices of a random graph as a function of |*V*|. What can you say about the dependence between |*V*| and that number?

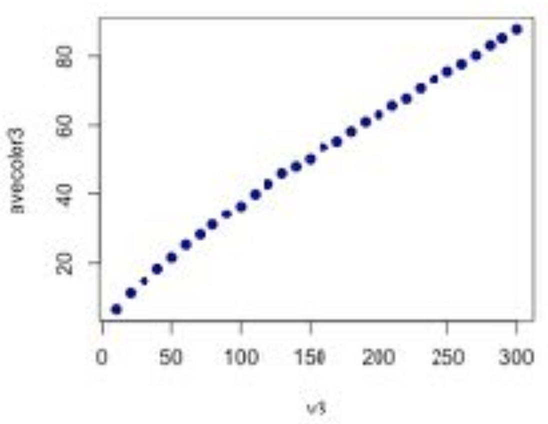
The results plotted below depict a positive linear relationship between the number of vertices and the average minimum number of colors necessary for graph coloring.



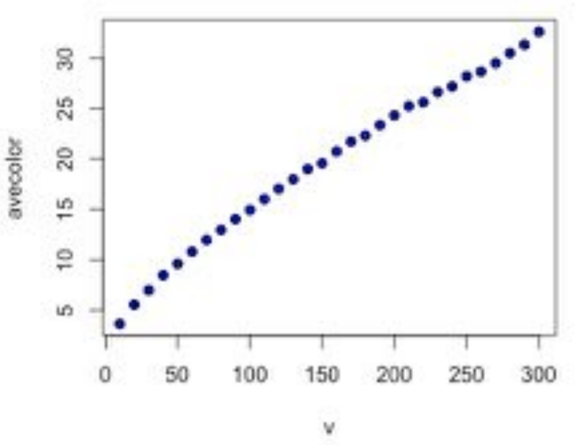
Group 1: |V| vs. Average Coloring Number for 100 Random Graphs



Group 2: |V| vs. Average Coloring Number for 100 Random Graphs



Group 3: |V| vs. Average Coloring Number for 100 Random Graphs



Group 4: |V| vs. Average Coloring Number for 100 Random Graphs

The