COLORED GRAPHS AND THEIR PROPERTIES

BEN STEVENS

1. Introduction

This paper is concerned with the upper bound on the chromatic number for graphs of maximum vertex degree Δ under three different sets of coloring rules. For a regularly colored graph, we present a proof of Brooks' Theorem, stating that the chromatic number is at most Δ in all but two cases. For a 2, 1-colored graph, we demonstrate that the 2, 1-chromatic number is at most $\Delta^2 + \Delta$. For a fractionally colored graph, we show that the fractional chromatic number is at most the regular chromatic number and that there always exists an optimal fractional coloring.

2. Preliminary Definitions

This paper deals with a subdiscipline of graph theory known as graph coloring. Before we address graph coloring, however, some definitions of basic concepts in graph theory will be necessary.

While the word "graph" is common in mathematics courses as far back as introductory algebra, usually as a term for a plot of a function or a set of data, in graph theory the term takes on a different meaning. In the context of graph theory, a graph is a collection of vertices and edges, each edge connecting two vertices.

If we take a graph and remove some of its vertices and edges, making sure that each edge is still connected to two vertices (that is, no edge may be "left hanging" with one or both ends not attached to a vertex), we obtain what is called a subgraph.

We formalize these two definitions as follows:

Definition 2.1. *A* ***graph*** *is a set of vertices V coupled with a set of edges E, each edge being incident on a pair of vertices in V. A* ***subgraph*** *of a graph is a subset of V, called V', coupled with a subset of E, called E', such that each edge in E' is incident on two vertices in V'.*

See Figure 1 for some examples of graphs and Figure 2 for an example of a subgraph.

It is important to note in Figure 1 that there are several points on the first two graphs in the figure where the edges cross but there is no bold dot present. These points should not be confused with vertices. There are some graphs where this sort of edge crossing is unavoidable (such as the center graph in Figure 1). Keep in mind, however, that not every point where two edges cross is a vertex: only the points with bold dots represent vertices.

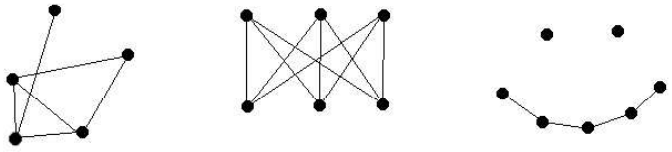

FIGURE 1. Some examples of graphs

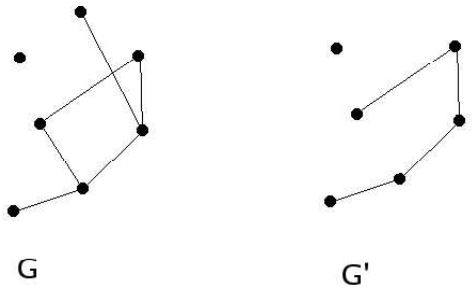

FIGURE 2. G' is a subgraph of G

Definition 2.2. *We call two vertices* ***adjacent*** *if there is a single edge that is incident on both vertices.*

Definition 2.3. *The* ***degree*** *of a vertex v is the number of edges that are incident on v.*

An example of vertices of varying degrees is shown in Figure 3. In the figure, vertex A has degree 0, vertex B has degree 1, vertices C

and D each have degree 2, and vertex E has degree 3. The degree of the vertex in a graph with the most edges incident on it, called the **maximum degree**, will be very important in determining that graph's coloring properties. The maximum degree of the graph in Figure 3 is 3.

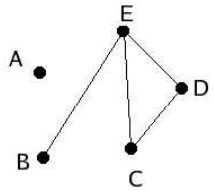

FIGURE 3. Vertices of varying degrees.

The idea of two vertices in a graph being connected will also prove important in the study of graph coloring. In informal terms, two vertices are connected if you can get from one vertex to the other following the graph's edges. We formalize this idea of connected in the following two definitions:

Definition 2.4. *A **path** is a sequence of vertices* $\{v_1, v_2, \cdots, v_{k-1}, v_k\}$ *in a graph such that for each vertex* v_i*, with* $i = 1, 2, \cdots, k-1$*, in the sequence there exists an edge such that the edge is incident on both* v_i *and* v_{i+1}*.*

Note that vertices may be repeated in paths. That is, there may be a path in which $v_i = v_j$ for some $i \neq j$. For example, in Figure 4, $\{A, B, C, B, C, D\}$ is a path.

Definition 2.5. *Two vertices,* u *and* v*, are called **connected** if there exists a path from* u *to* v*. A **connected graph** is a graph in which every distinct pair of vertices is connected.*

An example of a graph that is not connected (also known as disconnected) is shown in Figure 4. In this graph, vertices A and D are connected, as there exists a path between them, such as the path $\{A, B, C, D\}$. However, vertices A and E are not connected, as there is no path between them. From this, we conclude that the graph is disconnected.

Throughout this paper, we will be working exclusively with connected graphs, for reasons that will become clear when we learn about

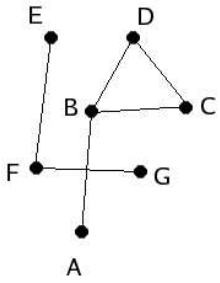

FIGURE 4. A disconnected graph.

graph coloring. However, it is a concept needed in order to define **connectivity**, which we will define in the next section.

In addition to connected graphs, there are many other types of special graphs that are important in the field of graph coloring. Two of these types of graphs are defined by a special type of path known as a circuit, a path that begins and ends at the same point and doesn't cross any given vertex more than once.

Definition 2.6. *A **circuit** is a path in which, given the path's vertex sequence* $\{v_1, v_2, \cdots, v_{k-1}, v_k, v_1\}$, $v_i \neq v_j$ *for* $i \neq j$. *A graph composed of a single circuit is called a **cycle**. A cycle with an even number of vertices is called an **even cycle**, while a cycle with an odd number of vertices is called an **odd cycle**.*

Because the labels are so similar, it is worth reiterating the difference between a circuit and a cycle. Informally, a circuit is a path that forms a closed loop, and is an ordered set of vertices within a graph. A cycle, on the other hand, is itself a graph, one composed entirely of a single circuit.

Referring back to Figure 4, the path $\{B, C, D, B\}$ is a circuit. Two examples of cycles are shown in Figure 5. In the figure, graph G is an even cycle (as it has 6 vertices), while graph G' is an odd cycle (having 5 vertices).

Another important type of connected graph is a graph that does not contain any circuits. This type of graph is called a tree.

Definition 2.7. *A **tree** is a connected graph that contains no circuits. A **tree of a graph** G is a subgraph of G that is a tree. A **spanning tree of a graph** G is a tree of G which contains all of the vertices in G.*

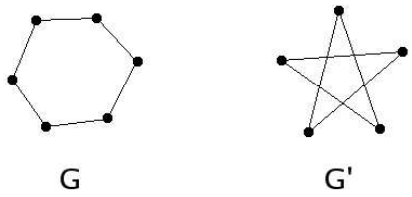

FIGURE 5. Two cycles.

An example of a tree is shown in Figure 6. In Figure 7, graph G' is a tree of graph G, while graph G'' is a spanning tree of graph G.

FIGURE 6. A tree.

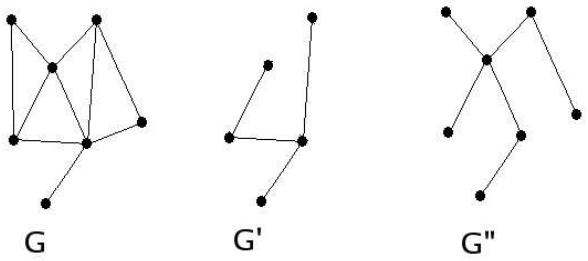

FIGURE 7. G' is a tree of G. G'' is a spanning tree of G.

One last type of graph we should consider is called a complete graph, in which there is an edge between every pair of vertices in the graph.

Definition 2.8. *A **complete graph** is a graph G in which every vertex is adjacent to every other vertex in G. A complete graph containing n vertices is referred to as the complete graph on n vertices and is denoted K_n.*

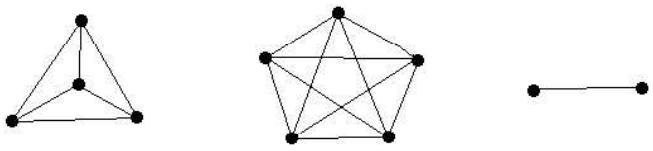

FIGURE 8. From left to right, the complete graphs on four, five and two vertices.

Some examples of complete graphs are shown in Figure 8.

Finally, we address graph coloring. We "color" a graph by assigning various colors to the vertices of that graph. As we will see, this process of coloring is generally governed by a set of coloring rules. For example, the most basic set of coloring rules, referred to as regular coloring, consists of a single rule: no two adjacent vertices may have the same color. We will explore regular coloring in more detail in the next section. For now, we will present some basic definitions regarding graph coloring.

Definition 2.9. *A **colored graph** is a graph in which each vertex is assigned a color. A **properly colored** graph is a colored graph whose color assignments conform to the coloring rules applied to the graph. The **chromatic number** of a graph G, denoted $\chi(G)$, is the least number of distinct colors with which G can be properly colored.*

Figure 9 gives an example of a colored graph. This graph is colored using the colors R, G, B, Y. Moreover, it is properly colored according to regular coloring rules. While it is less readily apparent, the graph's chromatic number is 4.

Note that when we say we have "colored" a vertex, we simply mean that we have assigned a label to it. The concept of colors merely provides a helpful mental image, but keep in mind that they need not actually represent colors as we know them, such as red and blue. As seen in Figure 9, it suffices to label the vertices of a graph, with each label representing a "color".

The concept of the chromatic number of a graph is one of the most interesting in all of graph theory. While there is no general rule defining a graph's chromatic number, we instead place an upper bound on the chromatic number of a graph based on the graph's maximum vertex degree. That is, we say that for a graph G with maximum vertex degree Δ, $\chi(G) \leq f(\Delta)$, where $f(\Delta)$ is some function of the maximum vertex degree. The remainder of this paper deals with the problem of finding

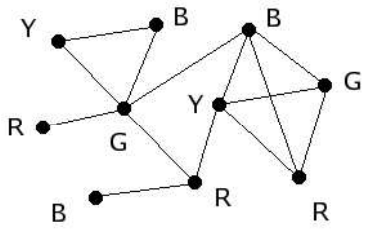

FIGURE 9. A properly colored graph of chromatic number 4.

a suitable upper bound for the chromatic number of any graph in each of three sets of coloring rules. We begin with the simplest set of rules, regular coloring.

3. Regular Coloring

3.1. **Basic Results.** As stated above, regular coloring is a rule for coloring graphs which states that no two adjacent vertices may have the same color. See Figure 10 for an example. In the figure, graph G is properly colored by regular coloring rules, while G' is not, as it contains two adjacent vertices that are both colored with color R.

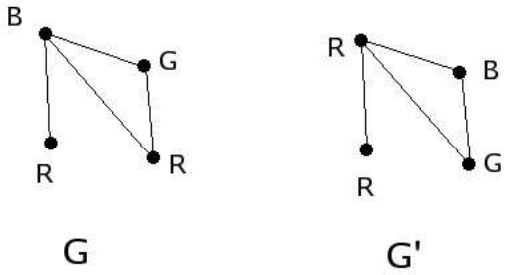

FIGURE 10. Two colored graphs. G is properly colored, G' is not.

Given this coloring rule, it becomes apparent why we may safely ignore disconnected graphs in our exploration of graph coloring. As the coloring rules deal with vertices that are adjacent, the colors on the vertices of each disjoint part of a disconnected graph have no bearing whatsoever on the colors of the vertices on any other disjoint part.

Thus, we may treat the each of the disjoint parts of the graph as if they were individual, connected graphs.

As we are ultimately trying to derive an upper bound for the chromatic number of a regularly colored graph, we will begin by finding the chromatic number of several types of graphs and see if a pattern emerges.

In a complete graph on n vertices, each of the n vertices in the graph is adjacent to every other vertex in the graph. In other words, each vertex in the graph is adjacent to $n-1$ vertices. Thus, every vertex has degree $n-1$, and we conclude that the maximum vertex degree of the complete graph on n vertices is $\Delta = n-1$. So, how does the chromatic number of a complete graph relate to Δ? Let's look at an example, shown in Figure 11.

In this example, the complete graph on 5 vertices, we begin by coloring one vertex with a color, called C_1. We then attempt to color a second vertex. This vertex must be adjacent to the first vertex that we colored, so it cannot be colored with C_1. Thus, we color it with some other color, C_2. The third vertex we choose will be adjacent to both of the first two vertices, so it must be colored with a third color, C_3, and so on, until we have colored all five vertices using a total of 5 colors. As can clearly be seen in the far right side of Figure 11, none of the vertices could be colored any of the other colors without violating our coloring rule. Thus, we conclude that the chromatic number of the complete graph on 5 vertices is 5.

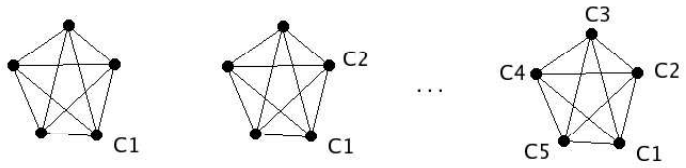

FIGURE 11. Illustrating the process of coloring the complete graph on 5 vertices.

It is not difficult to see how this example is generalized, in the case of the complete graph on n vertices, K_n. As each of the n vertices in K_n is adjacent to every other vertex in the graph, each vertex must be colored with a different color, making the chromatic number n. From our previous relationship of $\Delta = n-1$, we conclude that, for a complete graph G, we have $\chi(G) = \Delta + 1$.

Now let us look at another type of graph, a cycle. A cycle is a graph composed of a single circuit. From the examples of cycles shown in Figure 5, we easily see that every vertex has degree 2. This is intuitively true for any cycle, as a cycle represents a closed path with no repeated vertices, so, if one follows the path in one full circuit, each vertex must have one edge leading into it and one edge leading out of it. Thus, for a cycle, we conclude that $\Delta = 2$.

What, then, is the chromatic number of a cycle? At first glance, it might appear that two colors, C_1 and C_2 should be sufficient to properly color a cycle, as we may simply start at a vertex v_1, color it with C_1, then move around the circuit in one direction, alternating between coloring with C_2 and C_1 as we go. However, we must consider what happens when we arrive at the last vertex in the circuit, v_n which is adjacent to v_1, the first vertex we colored. Examining the other vertex adjacent to v_n, which we'll call v_{n-1}, we find two cases.

In the first case, v_{n-1} is colored with C_1. In this case, we are free to simply color v_n with C_2, and we have properly colored the cycle with 2 colors.

In the second case, v_{n-1} is colored with C_2. In this case, v_n is adjacent to a vertex colored with C_1 and a vertex colored with C_2, so v_n may not be colored with either of these colors. Thus, we must color it with some third color, C_3. Here, we have properly colored the cycle with 3 colors.

This process is illustrated in Figures 12 and 13. In the figures, we begin by coloring vertex v_1, then proceed clockwise around the cycle, coloring each vertex. Figure 13 shows the two cases for the final two vertices we color, v_{n-1} and v_n.

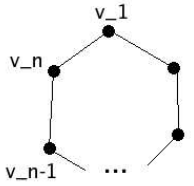

FIGURE 12. In coloring a cycle, we need only concern ourselves with vertices v_1, v_n and v_{n-1}.

With a bit of thought, it is easy to deduce that the first case occurs when the cycle contains an even number of vertices, while the second

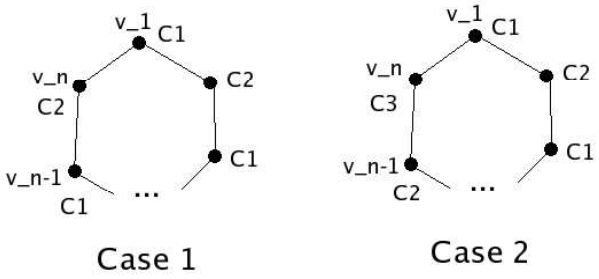

FIGURE 13. The two possible cases for the color of v_n. In case 1, both vertices adjacent to v_n are colored with color C_1, so we are free to color v_n with color C_2. In case 2, v_n is adjacent to a vertex colored with C_1 and a vertex colored with C_2, so it must be colored with a third color, C_3.

case occurs when the number of vertices is odd. See Figure 14 for an illustration of this.

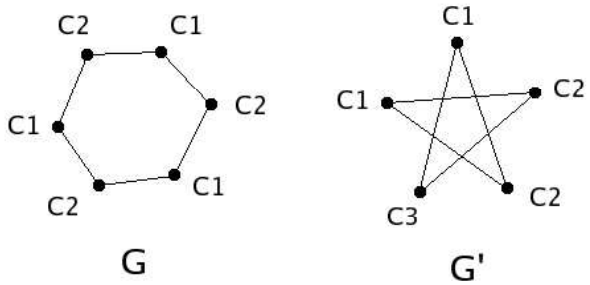

FIGURE 14. Cycles of even and odd length properly colored.

As these are the only two possibilities for cycles, we conclude that, for a cycle G, $\chi(G) \leq 3 = \Delta + 1$.

From these two types of graphs, we begin to see a pattern emerge in the relationship between the chromatic number χ and the maximum vertex degree Δ. For complete graphs, we have $\chi = \Delta+1$ and for cycles we have $\chi \leq \Delta + 1$. Can we generalize this result, that $\chi \leq \Delta + 1$, for all graphs? It turns out that we can, in a very simple proof.

Theorem 3.1. *If the maximum vertex degree of a graph G is Δ, then $\chi(G) \leq \Delta + 1$.*

Proof. We choose any arbitrary vertex in G and color it with one of the $\Delta + 1$ available colors. We then pick any uncolored vertex in G and color it with a color that has not been assigned to any of the vertices adjacent to it. We then repeat this last step until every vertex in G is colored. Because any given vertex v is connected to at most Δ vertices, there can be at most Δ distinct colors already used on the vertices adjacent to v, so there will alway be at least one color available to color v with. □

We have established that, under regular coloring rules, it requires at most $\Delta+1$ colors to properly color any graph of maximum vertex degree Δ. But can we do better? In general, the answer is obviously no, as we have found that complete graphs and odd cycles require $\Delta + 1$ colors to be properly colored. But what about graphs that do not fall into either of these categories? We already know that even cycles require Δ colors to be properly colored, so let's look at a few more examples.

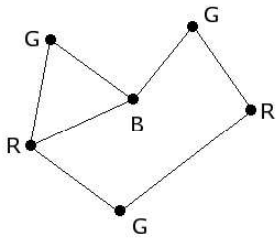

FIGURE 15. A graph of maximum vertex degree 3 properly colored using 3 colors

In Figure 15, we have a graph with a maximum degree $\Delta = 3$. The figure illustrates how the graph can be properly colored using 3 colors. Because the graph contains a triangle, which requires three colors, it will take no less than three colors to properly color the graph. So, in this case, we have $\chi = \Delta$.

In Figure 16, we have a graph with a maximum degree $\Delta = 7$. The graph, however, is properly colored using only 2 colors. This is quite obviously the minimum number of colors, as any connected graph with more than one vertex requires at least 2 colors in order to properly color it. So, here we have a graph in which $\chi < \Delta$.

As it turns out, complete graphs and odd cycles are the **only** graphs that require $\Delta + 1$ colors to properly color. For any graph that is not a complete graph or an odd cycle, we will show that the chromatic

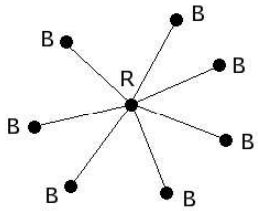

FIGURE 16. A graph of maximum vertex degree 7 properly colored using 2 colors

number of the graph is at most Δ. This result is one of the most important in all of graph theory and is known as Brooks' Theorem.

3.2. **Brooks' Theorem.** Brooks' Theorem states the following:

Theorem 3.2. *For any graph G with maximum vertex degree Δ such that G is not a complete graph or an odd cycle, $\chi(G) \leq \Delta$.*

Before we prove Brooks' Theorem, there are a few more necessary definitions and minor theorems that will be used in the proof.

Definition 3.1. *An **ordered graph** is a graph for which the set of its vertices is an ordered set. We call this ordered set the **vertex order**.*

In general, the set of vertices in a graph is not ordered. In an ordered graph, we assign an order to these vertices, so that they may be used in an algorithm, detailed below. An example of an ordered graph is shown in Figure 17.

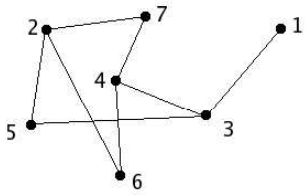

FIGURE 17. An ordered graph. Here, we take the vertex order to be $\{1, 2, 3, 4, 5, 6, 7\}$.

Note that, in Figure 17, the order of the vertices need not have anything to do with the vertices' labels. The fact that they were labeled with the integers 1 through 7 is completely arbitrary. For example,

$\{4, 7, 3, 2, 5, 1, 6\}$, with vertex 4 appearing first in the order, vertex 7 second, and so on, would have also been an acceptable order. However, each vertex in the graph must appear in the vertex order exactly once. Hence, using the graph from Figure 17 again, $\{1, 2, 2, 4, 7, 6\}$ would not be a legitimate vertex order.

As stated above, ordered graphs are useful because one may apply algorithms to them. One such algorithm which we will need in the proof of Brooks' Theorem is called the Greedy Coloring Algorithm.

Definition 3.2. *The **Greedy Coloring Algorithm** is an algorithm used to properly color an ordered graph using k colors (assuming that, given the order of the vertices and the graph's chromatic number, such a coloring is possible). It is implemented as follows:*

1. Assign an order to the set of colors.

2. Considering the first vertex in the vertex order, assign to it the first color.

3. Considering the next vertex, assign to it the lowest-ordered color that has not already been assigned to a vertex adjacent to it.

4. Repeat step 3 until the graph is colored.

For an example of this algorithm, refer back to Figure 17. Suppose we want to properly color the graph in Figure 17 using 3 colors. Suppose the colors are R, G, B. To implement the algorithm, we first assign an order to the colors. As no specific color has any special signifigance, we arbitrarily choose the order $\{G, R, B\}$. Now, we color the first vertex, vertex 1, with the first color, G. Next, we consider the next vertex in the order, vertex 2. As vertex 2 is not adjacent to any vertices that have colors assigned to them, we are free to color this vertex with G as well. Considering the next vertex, vertex 3, we see that it is adjacent to vertex 1, which is already colored with G. Thus, we choose the next available color in the color order, which is R, and color vertex 3 with it. This process continues until the entire graph is colored. See Figure 18 for an illustration of this process.

It is important to note that the Greedy Coloring Algorthm does not always color a graph using its chromatic number of colors. It is possible to select a vertex order that will result in some number of colors greater than the graph's chromatic number being needed to color the graph. See Figure 19 for an example. In the figure, the ordered graph is an even cycle, which we know has a chromatic number of $\chi = 2$. However, given the vertex order, the algorithm colors the graph with 3 colors.

So, if the Greedy Coloring Algorithm sometimes produces less-than-optimal results, why is it useful? What is important about the algorithm is that, for any graph, given the correct vertex order, it will

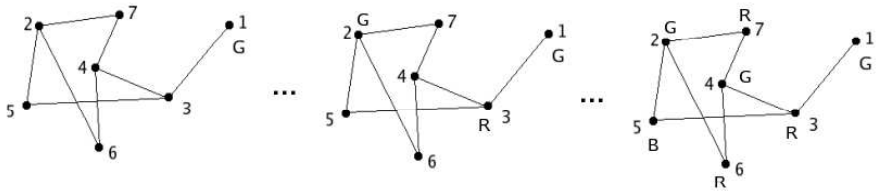

FIGURE 18. An ordered graph with vertex order $\{1, 2, 3, 4, 5, 6, 7\}$ colored using the greedy coloring algorithm with a color order of $\{G, R, B\}$.

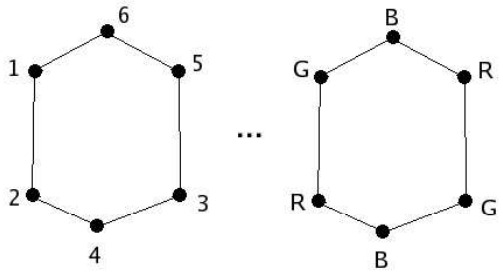

FIGURE 19. An ordered graph with vertex order $\{1, 2, 3, 4, 5, 6\}$ colored using the greedy coloring algorithm with a color order of $\{G, R, B\}$. In this example, the number of colors used is greater than the graph's chromatic number.

produce the optimal result (that is, it will color the graph using a number of colors equal to the graph's chromatic number). We prove this result below.

Theorem 3.3. *For any graph, G, there is an order that can be assigned to the vertices of G for which the greedy coloring algorithm will use the graph's chromatic number of colors to properly color G.*

Proof. By the definition of the chromatic number, we know that at least one such coloring exists. Assign an arbitrary order to the colors used in the graph. Now, we consider the set of all graph colors that use the chromatic number of colors. Of these, we consider the subset of these colorings which use the first color the maximum number of times. Of those colorings, we consider the subset of colorings which use the

second color the maximum number of times, and so on, until you are left with one coloring.

In this coloring, we know that any vertex colored with the second color must be adjacent to at least one vertex colored with the first color (if it were not, it could have been colored with the first color, which would imply that the coloring did not have the maximum number of vertices colored with the first color). Similarly, any vertex colored with the third color will be adjacent to both a vertex of the first color and a vertex of the second color, and so on. In this graph, order the vertices as follows: place any vertex colored with the first color first in the order. Place any unordered vertex colored with the lowest-ordered color available next in the vertex order. Continue this process until all the vertices in the graph are ordered. Now, applying the greedy coloring algorithm to this ordered graph using the same color order as used previously, we will recreate the original graph coloring, which uses the graph's chromatic number of colors. □

Now that we have some basic results concerning the Greedy Coloring Algorithm, we need to devise a method for ordering the vertices of a graph in such a way that we can use the algorithm to prove Brooks' Theorem. We find such a method in the following two theorems.

Theorem 3.4. *Every connected graph has at least one spanning tree.*

Proof. If the graph contains no circuits, it is already a tree and is therefore its own spanning tree. If the graph contains at least one circuit, remove one edge from each circuit in the graph. The graph will still be connected, as a circuit implies that for any two vertices on the circuit, two distinct paths exist between those vertices. By removing one edge from the circuit, we remove one of these paths, which still leaves one path between those two vertices. Further, the removal of an edge from a circuit ensures that the remaining edges will no longer form a circuit. Now, as the new graph is connected and contains no circuits, it is a tree of the original graph. As we did not remove any of the original graph's vertices in this process, the new graph is a spanning tree of the original graph. □

Theorem 3.5. *For any connected graph, G, there is an order in which one can place the vertices of G such that every vertex has a higher-ordered neighbor, with the exception of the last vertex in the order.*

Proof. Consider a spanning tree of G, which we will call G_t. G_t contains all of the vertices of G, and any vertices that are adjacent in G_t are also adjacent in G. Now assume that all of the edges in G_t are given

a weight, which we will call length. Furthermore, let each edge in G_t have a length of 1. Now choose any vertex v_0 in G_t. Given that vertex, we order the vertices of G_t, as well as the corresponding vertices in G, as follows:

1. Find the vertex for which the path from that vertex to v_0 has the greatest length. If more than one such vertex exists, choose any of these vertices. Place this vertex first in the order.
2. Find the unordered vertex for which the path from it to v_0 has the greastest length. If more than one such vertex exists, choose any of these vertices. Place this vertex next in the order.
3. Repeat step 2 until all vertices in G_t except v_0 have been ordered.
4. Place v_0 last in the order.

It can easily be shown that there is only one distinct path between any two given vertices in a tree, as two paths between those vertices would imply that a circuit exists. Given this, by the above ordering algorithm, all the vertices in the ordered graph of G_t, and therefore in the ordered graph of G, will have higher-ordered neighbors, except for v_0, which appears last in the order. □

We will refer to the algorithm introduced in the previous proof as the **Vertex Ordering Algorithm**. An example of the implementation of this algorithm is shown in Figures 20 and 21.

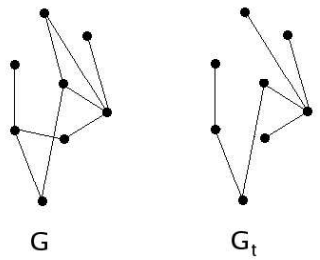

FIGURE 20. Two graphs, G and G_t. G_t is a spanning tree of G.

Figure 20 shows a graph G and a spanning tree of that graph G_t that we will use in the implementation of the Vertex Ordering Algorithm. In Figure 21, we implement the algorithm on G_t from Figure 20, first selecting an arbitrary v_0 from the vertices in G_t. Next, we find the vertex for which path from that vertex to v_0 is of the greatest possible length. In this graph, there are three such vertices, all with paths of length 4 between them and v_0. We choose one of these vertices and place it first in the vertex order (this is the vertex labeled 1). We

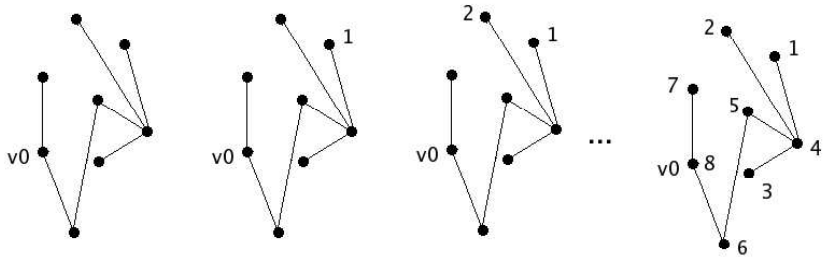

FIGURE 21. An implementation of the Vertex Ordering Algorithm on the graph and spanning tree in Figure 20.

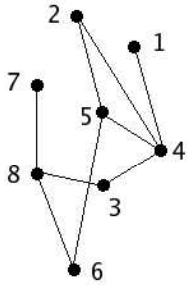

FIGURE 22. The vertex order obtained in Figure 21 mapped onto the vertices of G.

then choose another of these vertices with a path of length 4 to v_0 and place it second in the order (the vertex labeled 2). We repeat this process, ordering the vertices according to their distance from v_0 until we arrive at the vertex ordering shown in the right-most graph in Figure 21. Finally, we map the vertex order from G_t onto the corresponding vertices in G, as shown in Figure 22. As can be seen, every vertex in G except vertex 8, the last in the vertex order, has a higher-ordered neighbor.

Using the Vertex Ordering Algorithm in conjunction with the Greedy Coloring Algorithm, we are now able to prove Brooks' Theorem for all but one type of graph, known as a regular graph.

Definition 3.3. *A **regular graph** is a graph in which every vertex has the same degree. An **n-regular graph** is a regular graph in which all vertices have degree n.*

In Figure 23 we give some examples of regular graphs. Note that a complete graph is a special case of a regular graph, specifically an n-regular graph containing $n + 1$ vertices. Also note that cycles are 2-regular graphs.

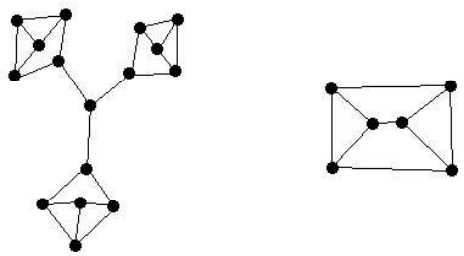

FIGURE 23. Two examples of 3-regular graphs.

Theorem 3.6. *For any non-regular graph G, let Δ be the maximum vertex degree. Then the chromatic number of G is at most Δ.*

Proof. Because G is not regular, at least one of the vertices in G must have a degree of less than Δ. Choose any such vertex and label it v_0. Now apply the Vertex Ordering algorithm to G, using the vertex v_0 assigned here. Using the vertex order generated by the Vertex Ordering Algorithm, we apply the Greedy Coloring Algorithm. As all vertices except v_0 have at least one higher-ordered neighbor, and all vertices in G have at most Δ neighbors, when the Greedy Coloring Algorithm considers a given vertex w, that vertex will have at most $\Delta - 1$ previously colored neighbors, meaning that at least one color will be available to color w. When the algorithm reaches v_0, the final vertex, we know that it has fewer than Δ neighbors, all of which will be colored, meaning at least one color will be available to color v_0. Hence, G can be properly colored using Δ colors. □

From this we conclude that Brooks' Theorem holds for all non-regular graphs. All that remains to be shown, then, is that it also holds true for all regular graphs that are neither complete graphs nor odd cycles. Unless otherwise specified, any further reference to a regular graph will be taken to mean a regular graph that is not a complete graph nor an odd cycle.

Before we attempt to prove Brooks' Theorem for the case of regular graphs, let us first consider some of the implications of regular

graphs with respect to the Vertex Ordering Algorithm. As each vertex in a regular graph will have degree Δ, our choice of which vertex to make v_0 becomes more complicated. The vertex designated as v_0 will have exactly Δ previously colored neighbors when it is considered by the Greedy Coloring Algorithm. Thus, in order to guarantee that the Greedy Coloring Algorithm will color the graph using at most Δ colors, we need to ensure that at least two of the neighbors of v_0 have the same color. If these two vertices do have the same color, at most $\Delta - 1$ colors will be forbidden to be used on v_0, and so there will be at least one color left with which to color it.

These two like-colored neighbors, then, must be non-adjacent. We are guaranteed at least one such pair of non-adjacent neighbors. If all of v_0's neighbors were adjacent to each other, a bit of thought reveals that the graph would be a complete graph, which we are assuming that it is not. This holds true for any v_0, as all vertices in the graph have the same degree.

To ensure that two non-adjacent neighbors of v_0 are colored with the same color, we need to alter the method in which we construct our vertex order. Specifically, we need to be more precise about which spanning tree of the graph must be used. First, note that every tree has at least two vertices of degree 1, as logic dictates that there would need to be a circuit in the graph for this not to be true. From this, we find that there is only one tree that is also a regular graph. This tree has just two vertices connected by an edge, and is shown in Figure 24. This, however, is also the complete graph on two vertices, so we may ignore it. As such, we may safely ignore any 1-regular graphs. Any connected regular graph we will be considering, then, is not a tree and therefore contains at least one circuit.

FIGURE 24. The only regular tree, the tree on 2 vertices. This graph is also the complete graph on 2 vertices.

Consider a 2-regular connected graph, which is a cycle. As we are not considering cycles of odd length, assume that this is an even cycle. As discussed earlier, we can properly color this graph using 2 colors. We therefore may also ignore any 2-regular graphs in our further discussion

of Brooks' Theorem. Thus, we need only prove Brooks' Theorem for Δ-regular graphs where $\Delta \geq 3$. However, before we finish proving Brooks' Theorem, we will need to define a concept known as connectivity.

Definition 3.4. *The* ***connectivity*** *of a connected graph G is the minimum number of vertices that one would need to remove from G in order to disconnect it.*

Some examples of graphs of different connectivities are shown in Figure 25. In the figure, graph G has connectivity 1. Removing vertex v would disconnect G into two disjoint parts. Graph G' has connectivity 2. Removing vertices w_1 and w_2 would disconnect the graph. It can also be seen that there is no single vertex that could be removed from G' that would disconnect the graph. Finally, graph G'' has connectivity 3. By removing vertices z_1, z_2 and z_3 we can disconnect G''. Examing G'' reveals that there is no single vertex or pair of vertices whose removal would disconnect the graph.

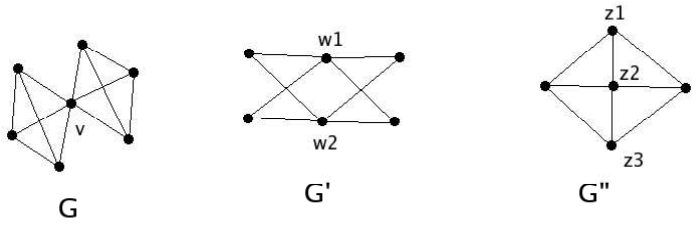

FIGURE 25. Graphs G, G' and G'' have connectivity 1, 2 and 3, respectively.

Now we may resume our proof of Brooks' Theorem. We will prove Brooks' Theorem in the case of regular graphs by addressing three separate cases: regular graphs of connectivity three or greater, regular graphs of connectivity two and regular graphs of connectivity one.

Theorem 3.7. *For any Δ-regular graph G_3 of connectivity three or greater, $\chi(G_3)$ is at most Δ.*

Proof. That G_3 has connectivity of at least three implies that one can remove any two vertices of G_3 without disconnecting it. Given this, consider the set of all spanning trees of the graph. We are trying to find a spanning tree of G_3 such that there is at least one vertex with two neighbors of degree one (in the spanning tree) that are not themselves adjacent in the original graph. Given the nature of spanning trees, we see that these two vertices can be any two nonadjacent vertices

with a common neighbor whose removal will not disconnect G_3. As this graph has connectivity three or greater, we are free to choose any two nonadjacent vertices with a common neighbor in G_3. Hence, there exists at least one spanning tree of G_3 in which those two vertices will have degree one and will both be adjacent to the same vertex. We label these two vertices of v_1 and v_2 and call the vertex to which they are both adjacent v_0.

From this, we implement a modified version of the Vertex Ordering Algorithm. This algorithm should create a vertex order that, when used in conjunction with the Greedy Coloring Algorithm, will properly color the graph using at most Δ colors. The modified algorithm is as follows:

1. Place v_1 first in the vertex order.
2. Place v_2 second in the vertex order.
3. Find the unordered vertex for which the path from that vertex to v_0 is of the greatest length. Place that vertex next in the vertex order. If there is more than one such vertex, arbitrarily choose any of the applicable vertices.
4. Repeat step 3 until all vertices except v_0 are ordered.
5. Place v_0 last in the vertex order.

Now, assigning some arbitrary order to the Δ colors being used, we implement the Greedy Coloring Algorithm on this ordered graph. First it will color both v_1 and v_2 with the first color, as they are not adjacent in the original graph. In this ordering, every vertex except v_0 has a higher-ordered neighbor. Therefore, when a vertex v_k, where $v_k \neq v_0$, is colored by the greedy coloring algorithm, it will have at most $\Delta - 1$ colored neighbors, so at least one color will be available to color v_k. When v_0 is reached, it will have Δ colored neighbors. However, v_1 and v_2 are colored with the same color, leaving at least one color available to color v_0. Hence, Brooks' Theorem applies to regular graphs of connectivity three or greater. □

An example of the process used in the proof is illustrated in Figures 26-28. Figure 26 shows a graph G which is 3-regular and of connectivity 3. Also shown in Figure 26 is a spanning tree of G, called G_t, which contains two vertices of degree one that are both adjacent to the same vertex. Further, their corresponding vertices in G are not adjacent. Hence, G_t meets all of the requirements for our spanning tree described above.

Figure 27 illustrates an implementation of the modified Vertex Ordering Algorithm. In it, vertices v_1 and v_2 are placed first and second in the vertex order, then the vertex farthest away from v_0 is placed

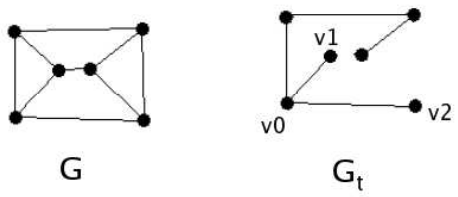

FIGURE 26. Graph G is a 3-regular graph of connectivity 3. G_t is a spanning tree of G.

third, then the next farthest is placed fourth, and so on. Once the other five vertices have been ordered, v_0 is placed sixth in the order.

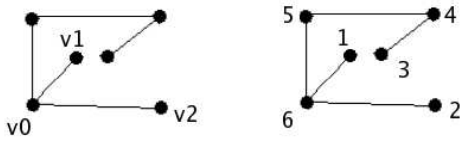

FIGURE 27. Applying the modified Vertex Ordering Algorithm to G_t.

Finally, in Figure 28, the vertex order obtained on G_t is mapped to the corresponding vertices in G and the Greedy Coloring Algorithm is applied. As seen in the figure, there are three colors used, which is the maximum vertex degree of G.

Now that we have established that Brooks' Theorem is valid in the case of a regular graph of connectivity three or greater, we must also prove the theorem in two other cases: regular graphs of connectivity one and of connectivity two.

Theorem 3.8. *Any Δ-regular graph of connectivity one has a chromatic number of Δ.*

Proof. Consider such a graph, called G_1. As G_1 has connectivity one, there is some vertex, v_c, which, if removed, would disconnect G_1 into n disjoint parts, with $n \geq 2$. Now consider the n subgraphs of G_1 formed by removing all vertices (and any edges incident on those vertices)

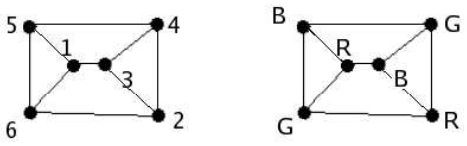

FIGURE 28. Applying the Greedy Coloring Algorithm to G using the vertex order obtained in Figure 27.

contained in all but one of those disjoint parts. An example of such a graph and its subgraphs is shown in Figure 29.

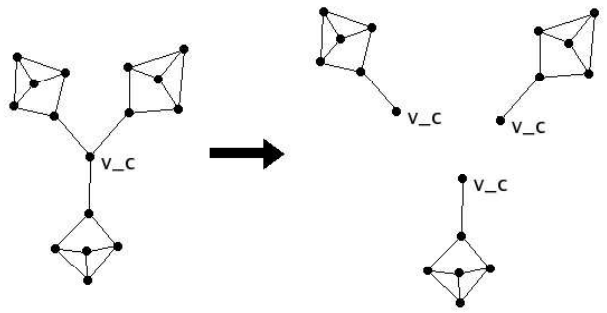

FIGURE 29. G_1, a 3-regular graph with connectivity 1, being split into three subgraphs

Each of these subgraphs is non-regular since in the subgraph, v_c will have a degree at least one less than its degree in G_1. Therefore, each of these subgraphs can be properly colored using at most Δ colors, as Brooks' Theorem applies to all non-regular graphs. Coloring each of these subgraphs using the Vertex Ordering Algorithm and the Greedy Coloring Algorithm, we obtain n colored subgraphs of G_1. If v_c is colored the same color in each of these subgraphs, we can map the n colored subgraphs onto G_1, coloring each vertex in G_1 as it was colored in its respective subgraph.

If v_c is not the same color in each subgraph, we can apply a simple color permutation to each subgraph, exchanging colors until v_c is colored with the same color in each subgraph. We then proceed as above. As each subgraph was properly colored, so, too, will G_1 be, using at

most Δ colors. Brooks' Theorem, then, applies to regular graphs of connectivity one. $\square$

An illustration of the process described in the above proof is shown in Figures 30 and 31.

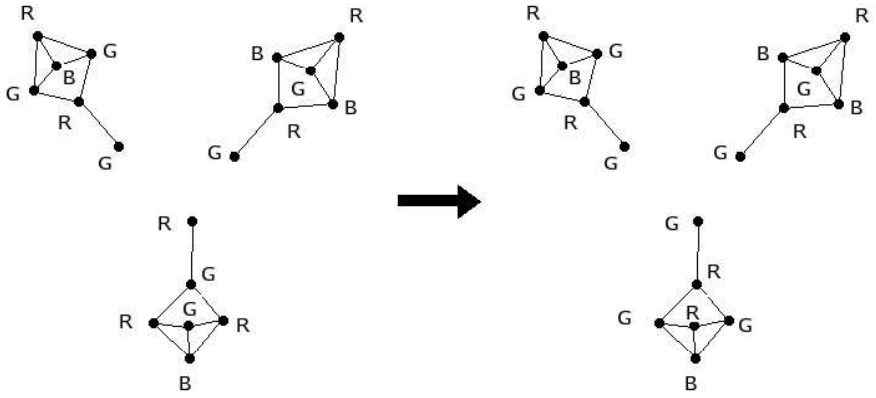

FIGURE 30. Each subgraph of G_1 is properly colored, then colors are permutated so that the colors on v_c match.

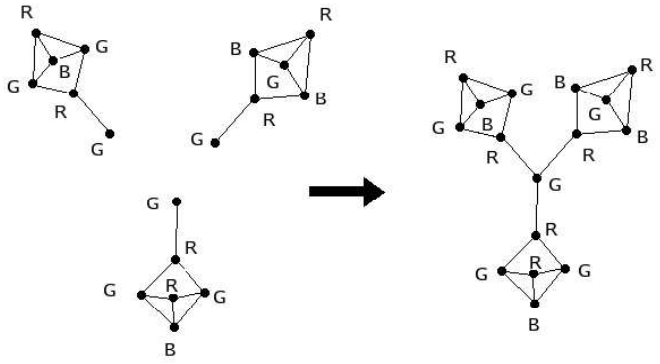

FIGURE 31. The colorings of the subgraphs of G_1 are mapped onto G_1, properly coloring it.

In Figure 30, each of the three subgraphs of G_1 are properly colored using the Vertex Ordering Algorithm and the Greedy Coloring Algorithm. As v_c is not colored with the same color in all three subgraphs, we perform a color permutation, interchanging the colors R and G in

the bottom subgraph, as seen on the right-hand side of Figure 30. Now, v_c is colored with color G in all three subgraphs.

In Figure 31, we color every vertex in G_1 as it was colored in its corresponding subgraph. Now, G_1 is properly colored using $\Delta = 3$ colors.

Theorem 3.9. *Any Δ-regular graph of connectivity two can be properly colored using at most Δ colors.*

Proof. Consider such a graph, called G_2. In G_2 there is at least one set of vertices, $\{v_\alpha, v_\beta\}$ such that removing these vertices will disconnect G_2. Consider all of the sets of three vertices $\{v_0, v_1, v_2\}$ in G_2 such that v_1 and v_2 are both adjacent to v_0, but not to each other. If for at least one of these sets of vertices the removal of v_1 and v_2 will not disconnect G_2, we may proceed to color the graph as we did in the case of a graph with connectivity three or greater.

If all of these sets are such that v_1 and v_2 cannot be removed without disconnecting G_2, we arbitrarily choose one such set. The removal of v_1 and v_2 would then split G_2 into n disjoint parts, with $n \geq 2$. We now form two subgraphs of G_2, G_L and G_R. G_L is formed by removing the vertices in every disjoint part of G_2 except for the part containing v_0. G_R is formed by removing the vertices in the disjoint part containing v_0. This process is illustrated in Figure 32. Note that the figure is a general example, showing only vertices v_0, v_1 and v_2 and none of the other vertices in G_2.

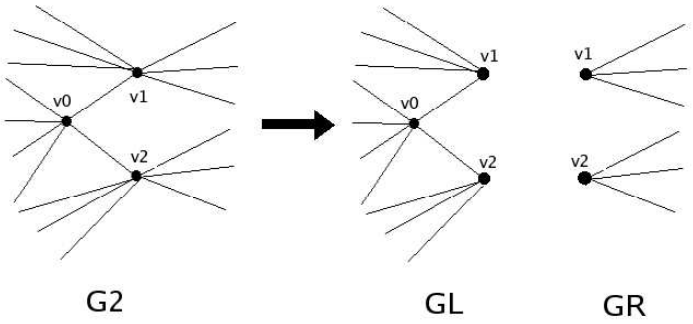

FIGURE 32. A regular graph G_2 with connectivity 2. G_2 is split into two subgraphs, G_L and G_R.

Both of these subgraphs are non-regular, and as such can be properly colored using Δ colors. Coloring both subgraphs using the Vertex

Ordering Algorithm and the Greedy Coloring Algorithm, we consider the colors of v_1 and v_2 in both subgraphs. If v_1 and v_2 are both colored with one color, say color c_L, in G_L and are both colored with one color, c_R, in G_R, we may simply interchange colors c_L and c_R in G_R. Then v_1 and v_2 will both be colored with c_L in both subgraphs. We may then map the colorings of the two graphs onto the corresponding vertices G_2, properly coloring it with Δ colors. We may follow the same reasoning if v_1 and v_2 are colored with two distinct colors in each of G_L and G_R. This process is illustrated in Figure 33.

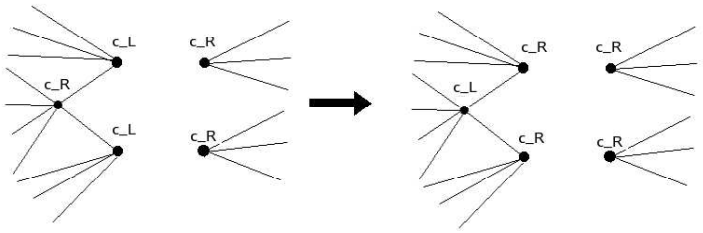

FIGURE 33. After coloring G_L and G_R, if v_1 and v_2 are the same color in both subgraphs, we permute the colors so we may combine the subgraphs to create a properly colored G_2. The same process is used if v_1 and v_2 are different colors in both G_L and G_R.

If v_1 and v_2 are colored with the same color in one subgraph (without loss of generality, we'll assume they're colored with the same color in G_L) and colored with two distinct colors in the other subgraph, we may force the two vertices in G_L to be different colors. We accomplish this by temporarily adding an edge between v_1 and v_2 in G_L while we apply the greedy coloring algorithm to it. Adding this edge does not change the the maximum vertex degree of G_L, as we know that the degrees of v_1 and v_2 in G_L are both strictly less than Δ. Figure 34 provides an example of this strategy.

Adding an edge between v_1 and v_2 will be sufficient to properly color G_2 unless adding the edge in G_L causes G_L to be regular. In this case, we must force v_1 and v_2 to be the same color in G_R. Assume that, in G_R, v_1 is colored with color c_1 and v_2 is colored with color c_2. Given that adding an edge between v_1 and v_2 in G_L causes G_L to be regular, we can conclude that, in G_R, v_1 and v_2 each have degree one. There are then two possible cases for G_R.

In the first case, v_1 and v_2 are both adjacent to one vertex, v_3. In this case, v_3 must be colored with some color c_3, so we may simply

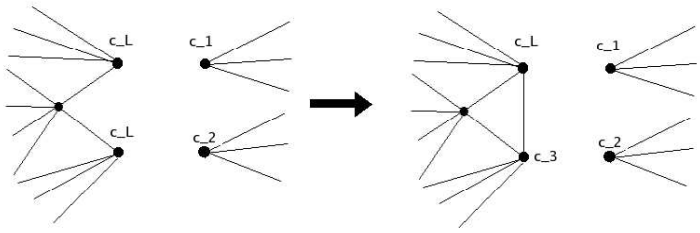

FIGURE 34. After coloring G_L and G_R, if v_1 and v_2 are the same color in G_L but not in G_R, we add a temporary edge between the vertices in G_L.

change the color of v_2 to color c_1, making v_1 and v_2 the same color. This case is shown in Figure 35.

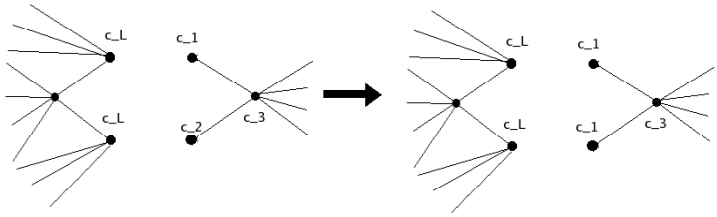

FIGURE 35. If we cannot add a temporary edge between v_1 and v_2 in G_L, we force the vertices in G_R to be the same color. In this case, they are adjacent to the same vertex.

In the second case, v_1 is adjacent to a vertex v_a and v_2 is adjacent to another vertex v_b. If v_a is not colored with color c_2, we may change the color of v_1 to color c_2, making v_1 and v_2 the same color. We follow a similar line of reasoning if v_b is not colored with color c_1. If v_a is colored with c_2 and v_b is colored with c_1, we consider our assumption that $\Delta \geq 3$. As such, there is some third color, c_3, that we may color both v_1 and v_2 with. This situation is shown in Figure 36.

In both cases, we can map the colorings of the two subgraphs onto G_2, which will then be properly colored with Δ colors. Brooks' Theorem, then, applies to regular graphs of connectivity two. The proof of Brooks' Theorem is now complete. □

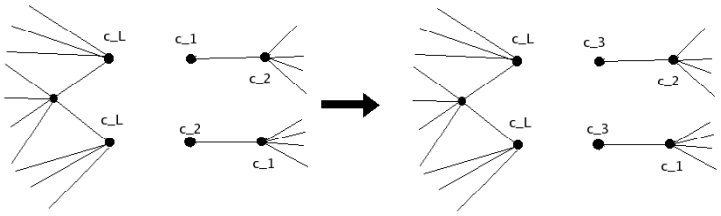

FIGURE 36. As in Figure 35, we force v_1 and v_2 to be the same color in G_R. In this case, the two vertices are adjacent to two distinct vertices.

4. $2,1$-COLORING

As stated above, Regular Coloring is only one of many possible sets of coloring rules. Now we will focus on a different, but related, set of coloring rules known as $2,1$-coloring.

Suppose that each of the $k+1$ colors we wish to use to color a graph are each assigned a distinct integer n_i, where $0 \leq n_i \leq k$. We may then refer to each color by its corresponding integer assignment. That is, we are coloring a graph using the set of colors $\{0, 1, 2, \ldots, k-1, k\}$. Then we define the rules of $2,1$-coloring as follows:

1. Adjacent vertices must have colors that are at least 2 numbers apart.

2. Two distinct vertices separated by exactly two edges cannot be the same color.

First, note that the color order is not cyclical. That is to say, color 0 is not considered to be one number away from color k. Also note that, in this set of rules, if a graph uses colors $0, 1, 3, 4$, we say that it is colored using 5 colors, even though only four were used, as the highest-ordered color used is the fifth color overall. In other words, any colors less than the highest numbered color that were not used in the graph still count toward the total number of colors used.

However, it turns out to be simpler if we define the $2,1$-chromatic number as the number of the highest-ordered color used the the graph. Thus, the $2,1$-chromatic number is one less than the total number of colors used in $2,1$-coloring the graph. We denote the $2,1$-chromatic number as λ.

Just as in the case of regular coloring, we wish to find an upper bound on the $2,1$-chromatic number of a graph. To begin, we will proceed as we did above, by examining the behavior of several common types of graphs under these coloring rules. We begin with cycles.

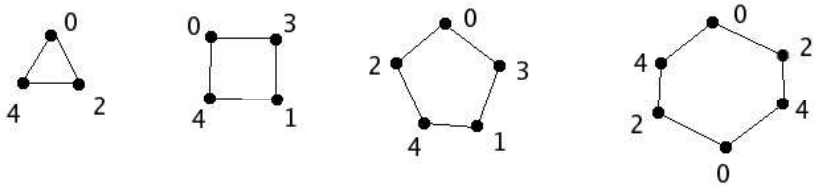

FIGURE 37. Cycles on 3, 4, 5, and 6 vertices that are $2,1$-colored.

Diagrams of the cycles on 3, 4, 5, and 6 vertices properly colored according to $2,1$-coloring rules are shown in Figure 37. All of them can be colored by using 5 colors (colors 0 through 4). This seems to be a pattern, suggesting that cycles can be $2,1$-colored using 5 colors.

We now turn our attention to complete graphs. Because all of the vertices in these graphs are adjacent, we can easily find the $2,1$-chromatic number for complete graphs. To use the minimum number of colors, color one vertex with color 0. Picking any other vertex, we know that it is adjacent to the original vertex, so we color it with color 2. Picking a third, it is adjacent to both the previous vertices, so it must be colored with color 4. Continuing on in this fashion, we see that the kth vertex chosen will be colored with color $2(k-1)$. So, a complete graph on Δ vertices will have a $2,1$-chromatic number of $2(\Delta-1)$, or $2\Delta-2$. See Figure 38 for an example of a $2,1$-colored complete graph.

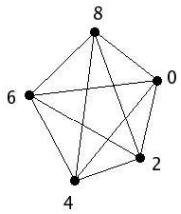

FIGURE 38. A $2,1$-coloring of K_5. In this graph, $\lambda = 8 = 2(5)-2$.

In order to find a suitable upper bound on the $2,1$-chromatic number of a graph of maximum vertex degree Δ, we need to alter the definition of the Greedy Coloring Algorithm used in the proof of Brooks' Theorem.

Definition 4.1. *The Alternative Greedy Coloring Algorithm is an algorithm which colors the vertices of an ordered graph. Its steps are as follows:*

1. Assign an order to the colors being used.

2. Consider the first vertex in the vertex order. If it can legally be colored with the first color, color it with the first color. If it cannot, leave it uncolored.

3. Repeat step 2 for the next vertex in the vertex order until all of the vertices of the graph have been considered.

4. Repeat steps 2 and 3 for the next color in the color order.

5. Repeat step 4 until all of the colors have been considered, or until there are no uncolored vertices, whichever comes first.

In the case of $2,1$-coloring, step 1 is already done, as the colors inherently have an order associated with them. An example of this algorithm being implemented to $2,1$-color a graph is shown in Figure 39. In the figure, the numbers in the left-hand graph represent the vertex order used. In the other two graphs in the figure, the numbers represent the colors assigned to each vertex.

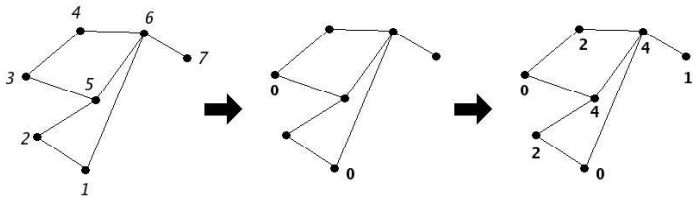

FIGURE 39. An implementation of the Alternative Greedy Coloring Algorithm.

In Figure 39, the algorithm considers vertex 1 and colors it with color 0. It then considers vertex 2, which is adjacent to vertex 1 and as such cannot be colored with color 0. Therefore, it remains uncolored. Vertex 3 is then considered. As it is separated from vertex 1 by more than two edges, it can be colored with color 0. The algorithm then reconsiders every uncolored vertex for each color, resulting in the $2,1$-coloring on the right-hand side of Figure 39.

Using the Alternative Greedy Coloring Algorithm, we are now able to prove a suitable upper-bound on the $2,1$-chromatic number of a graph of maximum vertex degree Δ.

Theorem 4.1. *For any graph G with maximum vertex degree Δ, $\lambda \leq \Delta^2 + \Delta$, where λ is the $2,1$-chromatic number of G.*

Proof. We begin by assigning an order to the vertices of G. We then apply the Alternative Greedy Coloring Algorithm, defined above, to this ordered graph, using $\Delta^2 + \Delta + 1$ colors. Recall that λ is defined such that it is equal to the number of colors used minus one. Assume that G was not properly colored by this algorithm. This implies that at least one vertex was left uncolored at the end of the algorithm.

Let us consider one of these uncolored vertices, which we'll call w. As w was not colored by the algorithm, this means that each time the algorithm arrived at w, it could not color w with the color that was being considered during that iteration of the algorithm. This means that for all i, when the algorithm was considering the ith color, w had either a 2-neighbor that had already been colored with color i or a neighbor colored with either color i or color $i-1$. Note that a neighbor of w colored with color $i+1$ would also prevent w from being colored with color i. However, we do not consider this case, as the Alternative Greedy Coloring Algorithm considers each color in sequence. Thus, no vertices will be colored with color $i+1$ when the algorithm attempts to color vertex w with color i.

As the maximum vertex degree of G is Δ, w has at most Δ neighbors and at most $\Delta(\Delta-1)$ 2-neighbors. As mentioned above, each neighbor precludes the use of at most two colors and each 2-neighbors forbids the use of one color. The worst case scenario, then, is that each of these forbidden colors is distinct. The maximum number of colors that can be forbidden to w, then, is $2\Delta + \Delta(\Delta-1)$, which simplifies to $\Delta^2 + \Delta$. However, we were using $\Delta^2 + \Delta + 1$ colors in our algorithm, so there must have been at least one color that could have been used to color w.

As G can be properly colored using at most $\Delta^2 + \Delta + 1$ colors, its $2,1$-chromatic number λ is at most $\Delta^2 + \Delta$. □

It has been demonstrated that the upper bound on the $2,1$-chromatic number is, in fact, lower than $\Delta^2 + \Delta$. The lowest upper bound currently known is given as $\lambda \leq \Delta^2 + \Delta - 2$. However, the proof of this is quite long and complicated, and will not be covered here.

This concludes our discussion of $2,1$-coloring. The final set of coloring rules we will investigate involves potentially assigning "fractions" of colors to vertices. This set of rules is appropriately titled **fractional coloring**.

5. Fractional Coloring

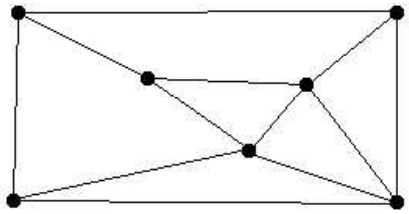

FIGURE 40. A graph G. The chromatic number of G is 4.

5.1. **Properties of Fractional Colorings.** Consider the graph, G, shown in Figure 40. If we were to assign a regular coloring to it, we would find that it cannot be properly colored using 3 colors. However, we could see by trial that it can be properly colored using 4 colors. We would then conclude, correctly, that the graph's chromatic number is 4.

However, what if we were to color each vertex with several different colors, each color weighted as a "fraction of a color"? Consider the coloring shown in Figure 41, in which colors A, C, E, and F are assigned a weight of $\frac{1}{3}$ of a color and colors B, D and G are assigned a weight of $\frac{2}{3}$ of a color.

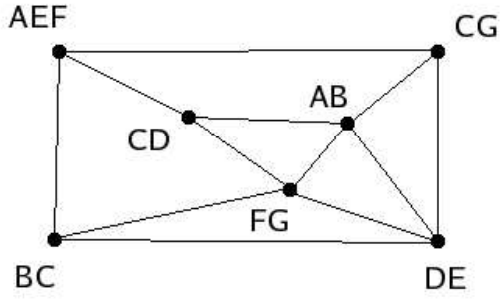

FIGURE 41. A fractional coloring of G. Here, G is colored with fewer than 4 whole colors.

In this coloring, notice that the sum of the weights of the colors on each vertex equals one and that no two adjacent vertices share colors. Furthermore, if we add up the weights of all the colors used, we get $4 \cdot \frac{1}{3} + 3 \cdot \frac{2}{3} = \frac{10}{3} < 4$. Using this fractional coloring scheme, we

have colored the graph using "fewer" colors than the graph's regular chromatic number.

Before we formalize the concept of fractional coloring, we need to define the concept of an independent set.

Definition 5.1. *A set of vertices in a graph is called an* ***independent set*** *if no two vertices in the set are adjacent. Conversely, a set is* ***dependent*** *if at least two of the vertices in the set are adjacent. A* ***maximal independent set*** *is an independent set which would become dependent were any other vertex in the graph added to it. The* ***independence number*** $\beta(G)$ *of a graph G is the size of the largest maximal independent set in G.*[2]

Definition 5.2. *A* ***fractional coloring*** *of a graph G is a set of non-negative weights (colors) assigned to the independent sets of vertices in G such that the sum of the weights on each independent set containing a given vertex v is at least 1.*

Definition 5.3. *The* ***weight*** *of a fractional coloring of a graph G is the sum of the weights of all independent sets in G. The* ***fractional chromatic number*** *of G, denoted $\chi_f(G)$ is the minimum weight of a fractional coloring of G.*

Looking back to the example in Figure 41, the colors A through G each represent an independent set of the graph's vertices. The coloring is distinguished from a regular coloring in that the independent sets corresponding to colors do not need to be disjoint and the weights assigned to these sets are not necessarily one.

We see, then, that a regular coloring of a graph, then, is merely a special case of a fractional coloring. In a regular coloring, each color represents an independent set. Assigning a weight of one to any independent set corresponding to a color and a weight of zero to any other independent sets produces a regular coloring.

The set of regular colorings of a graph is, then, a subset of the set of fractional colorings. As we have just shown in the above example, it is possible to use a smaller total weight in a fractional coloring than is possible in a regular coloring. From these two results we can conclude that the fractional chromatic number is at most equal to the regular chromatic number. That is, $\chi_f(G) \leq \chi(G)$.

It may seem odd that the rules of fractional coloring state that the total weight on any vertex must be **at least** 1. This means that each vertex may be colored with more than the equivalent of one regular color, which would seem to imply that any such coloring would not be

optimal. However, we can show that an optimal coloring always exists such that the weight on each vertex in the coloring is exactly 1.

Theorem 5.1. *Given the fractional chromatic number of a graph G, $\chi_f(G)$, there is a fractional coloring of G with total weight equal to $\chi_f(G)$ in which the total weight of each vertex is equal to 1.*

Proof. Suppose that no such coloring exists. This means that, for every fractional coloring of G with total weight equal to $\chi_f(G)$, there is at least one vertex whose total weight is strictly greater than one.

Suppose one such vertex, w, has a total weight equal to $1+\alpha$, where α is some positive rational number. Consider the independent sets with non-zero weight that include w. If any of them have weight less than or equal to α, we may remove w from that independent set. We continue to remove w from independent sets until either its total weight becomes exactly 1 or there are no independent sets from which we can remove w without reducing its total weight to less than 1.

In the latter case, suppose that removing w from some independent set A with weight $\frac{a}{b}$ would make the total weight of w equal to $1 - \frac{c}{d}$. We now replace independent set A with two independent sets, A_1 and A_2, such that A_1 has weight equal to $\frac{c}{d}$ and includes all vertices that were included in A. A_2, meanwhile, has weight equal to $\frac{a}{b} - \frac{c}{d}$ and includes all vertices that were included in A except for w, which it does not include. Vertex w now has a total weight of exactly 1. Note that, in this process, we did not change the total weight of the graph, as the combined weight of sets A_1 and A_2 is equal to the weight of A, the set they replaced.

We may now repeat this process for all vertices with total weight greater than 1 in G, achieving a fractional coloring of G with total weight equal to $\chi_f(G)$ in which all vertices have total weight equal to 1. □

The problem of finding $\chi_f(G)$ for a specific graph can be expressed by way of a linear programming problem. In the problem, we seek to minimize the sum of the weights of each maximal independent set (we can safely ignore all non-maximal independent sets by assigning each of them a weight of 0) in G subject to the constraints that the sum of the weights of all independent sets that contain a given vertex v must be at least one, for all v in G.

Assume that there are m vertices and n maximal independent sets in G. We then create an $m \times n$ vertex-independence matrix A, in which the i, j position is filled with a 1 if vertex i is in the independent set j

and with a 0 if not. We also create an n-dimensional vector $\mathbf{w}$ whose j-th entry is the weight of the independent set j.

Let $\mathbf{1}$ be a vector in n-space with each entry 1 and let $\mathbf{0}$ be a vector in n-space with each entry 0. We can then succinctly state the above linear programming problem as:

Minimize $\mathbf{1}^T\mathbf{w}$ subject to $A\mathbf{w} \leq \mathbf{1}$ and $\mathbf{w} \geq \mathbf{0}$. Here, the $\leq$ comparison operator is taken to mean that each entry in the left-hand vector is less than or equal to the corresponding entry in the right-hand vector (with a similar definition for the $\geq$ operator).

Linear programming problems like this one can be solved using an algorithm known as the simplex method. While it is outside the scope of this paper, any text on linear programming may be consulted for those interested in the specifics of the simplex method.

But can we always guarantee that a singular solution exists? Given the above problem of finding the fractional chromatic number of a graph, we have m linear constraints (that is, the weights of the maximal independent sets). Geometrically, the linear constraints define a convex polyhedron in m-space. As none of the constraints contradict one another and the polyhedron is not unbounded in the direction of the objective function, linear programming theory guarantees that there is an optimum that is attained at a vertex of the polyhedron. That is to say, the solution is a single point.

As the problem is a collection of linear equations and the coefficients in the constraints are all rational numbers, we may conclude that the solutions are all rational numbers as well. Therefore, the minimal solution of the objective function is rational. From this we conclude that there is always a solution to the problem of finding the fractional chromatic number of a graph.[1]

6. Conclusion

While this paper provides a good introduction to the study of graph coloring, there are many other questions in the field that bear addressing. Regular coloring, $2,1$-coloring and fractional coloring are only three examples out of countless potential sets of coloring rules. Each set of coloring rules has its own upper bound on the corresponding chromatic number. Even within these three examples, there are questions outside the scope of this paper. For example, as previously mentioned, it has been shown that the upper bound on the $2,1$-chromatic number is less than the upper bound proved in this paper. Regardless, the strategies and reasoning used in this paper would serve as an excellent starting point for exploring further into the field of graph coloring.

References

[1] Wikipedia. Linear Programming. http://en.wikipedia.org/wiki/Linear_programming.
[2] C.L. Liu. *Introduction to Combinatorial Mathematics*. McGraw-Hill Book Company, 1968.