Edge Coloring Algorithms

Research Report by Timothy Ho

1 – Introduction: What is edge coloring?

Edge coloring is a problem in graph theory where all the edges in a given graph must be assigned a color. Furthermore, all edges that are adjacent to each other must be given different colors. In other words, all the edges incident to any specific vertex must contain no repeated colors.

The goal of this problem is to minimize the number of colors needed to color all the edges in the graph legally. It would be trivial to assign a different color to every edge in the graph. In the example below, two identical graphs are shown: one uses four colors, the other uses three colors. The three-colored graph on the right is considered "better" and is optimal for the given graph.

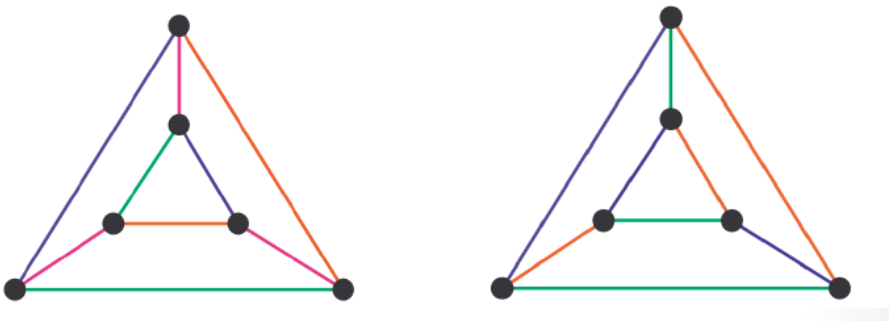

Figure 1: Two identical graphs with different edge colorings.

The applications of edge coloring mainly include scheduling and minimizing the number of rounds or time slots needed to complete all tasks. For example, an input graph could be a schedule of teams that play each other in season with the vertices representing different teams and edges that represent a game between two teams. An optimal edge coloring would represent a minimal number of weeks to complete the schedule of games. Another application of edge coloring is wireless communications with vertices representing nodes that need to talk to each other and edges indicating which nodes are within range of each other. An optimal edge coloring can show the minimum number of different frequencies needed so that all nodes can communicate without overlapping each other's lines.

Table 1. Games to be scheduled.

Team	1	2	3	4	5	6	7	8	9	Total
1	–	3		3				3	3	12
2	3	–	3				3	3		12
3		3	–	2	3	3				11
4	3		2	–	3	3				11
5			3	3	–	2			3	11
6			3	3	2	–	3			11
7		3				3	–	3	3	12
8	3	3					3	–	3	12
9	3				3		3	3	–	12
	12	12	11	11	11	11	12	12	12	104

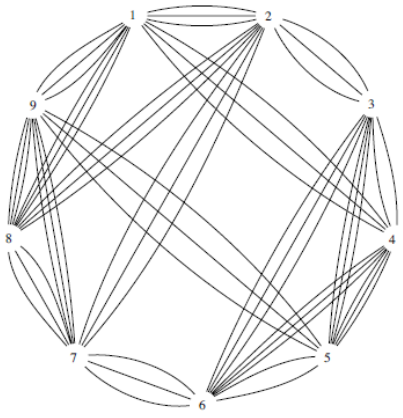

Figure 2: A sports schedule (left) translated into a multigraph (right) for edge coloring.

2 – Vizing's Theorem

Before going into specifics of edge coloring algorithms, it is necessary to define a couple of key terms and introduce Vizing's Theorem, which is essential to the study of edge coloring.

χ'(G): This is referred to as the **edge chromatic number** or **chromatic index**, which is the minimum number of colors needed to **edge color** graph G.

Note that this term is different from χ(G), which is simply called the **chromatic number** and defines the minimum number of colors to **vertex color** graph G. Vertex coloring is a different type of graph problem but the relationship between edge coloring and vertex coloring will be shown later.

Δ(G): This number, delta, is the **maximum degree** of graph G. That is, within the set of vertices in G, Δ represents the vertex (or vertices) with the highest number of incident edges within the graph.

There is a close relationship between the maximum degree of a graph and its edge chromatic number. Vizing's Theorem states that for any simple graph, the edge chromatic number is either Δ or Δ + 1. For multigraphs, a loose upper bound was shown by Shannon to be (3Δ/2). This bound will be explained in more detail in the multigraph edge coloring section to fit Vizing's Theorem. In mathematical terms:

$$\Delta(G) \leq \chi'(G) \leq \Delta(G) + 1, \text{ for any simple graph } G$$

A Class 1 graph is defined as a graph G where χ'(G) = Δ(G). Bipartite graphs and high degree planar graphs are all of Class 1. Any graph not in Class 1 is considered a Class 2 graph. While most random graphs are Class 1, it is an NP-hard problem to determine whether a graph is of Class 1 or Class 2.

To restate Vizing's Theorem, all simple graphs can be edge colored with either Δ or Δ + 1 colors. The goal of a good edge coloring algorithm is one that guarantees using at most Δ + 1 colors.

3 – The Greedy Algorithm

A simple but non-optimal greedy algorithm can edge color a graph using 2Δ – 1 different colors. This approach may seem trivial but algorithms that guarantee Vizing's Δ + 1 colors need to have knowledge of the actual graph ahead of time. An interesting topic beyond the scope of this paper considers how greedy edge coloring is optimal for online graphs – graphs that are not known ahead of time or graphs that need to be changed or adjusted on the fly during processing. So while greedy edge coloring is simple, it has relevant uses for modern applications. The algorithm is as follows:

1) Label the edges in some arbitrary order.
2) Have 2Δ – 1 colors available and indexed in some arbitrary order.
3) Cycle once through each edge and assign it the lowest index color available (i.e. the first legal color).

The proof for why this algorithm uses at most 2Δ – 1 colors is also fairly simple. Consider the worst-case scenario, coloring an edge between two vertices of degree Δ with all other edges already assigned unique colors (shown below in Figure 3). The assigned colors at most sum to the value 2Δ – 2 if each vertex has been given Δ – 1 different colors already. However the edge to be colored, the black edge, is

shared by both vertices; thus only one more color is needed. 1 + (2Δ – 2) is equal to the value 2Δ – 1, which is the maximum number of colors that a greedy edge coloring algorithm will use.

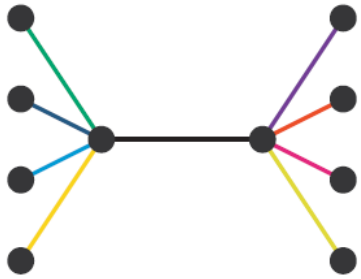

Figure 3: Worst-case scenario for greedy edge coloring where Δ(subgraph) = 5, 9 colors (2Δ – 1) are used.

4 – Algorithms and Proofs for Different Types of Graphs

One important strategy for edge coloring algorithms is to determine the type of graph being colored. If a graph or its subgraphs can be classified into a type that is solved or easily colored, the algorithm can correctly choose the optimal coloring. This report will examine a few basic types of these graphs.

4.1 Cycles

Cycles are one the easiest types of graph to recognize and color. All cycles have **Δ = 2**. If a cycle has an **even number** of edges, it can be 2-colored (Class 1) by alternating colors. If a cycle has an **odd number** of edges, it can be 3-colored (Class 2), alternating colors with the third color assigned to the odd edge. This is visually demonstrated in the figure below. All cycles can therefore be colored with Δ or Δ + 1 colors.

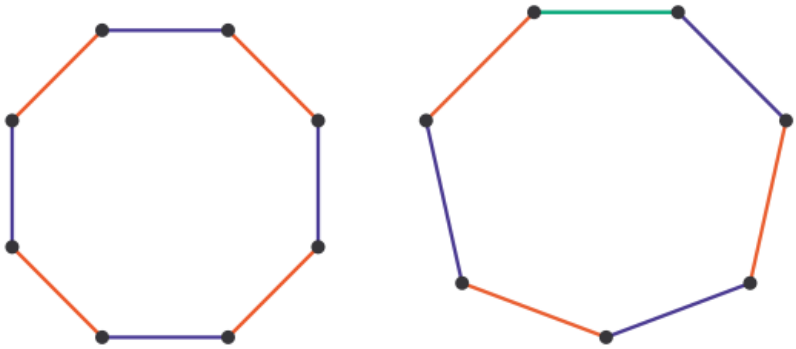

Figure 4: Edge coloring an even cycle (left) and an odd cycle (right).

4.2 Complete Graphs

A complete graph, $\mathbf{K_N}$, is defined as a collection of N vertices and an edge from each vertex to every other vertex. In other words, each distinct pair of vertices in the graph has an edge connecting them. For any complete graph K_N, **Δ = N – 1**. Similar to cycles, if **N is even**, the graph is Class 1 and can be edge-colored with N – 1 colors. If **N is odd**, the graph is Class 2 and needs N colors.

This can be proven with induction using the base cases of K_2 and K_3. A K_2 trivially only needs one color since the graph has one edge and a K_3 needs three colors since each edge is adjacent to the other two. Every complete graph K_N can be visualized as a K_{N-1} with an extra vertex on the outside and edges connecting the outer vertex to every vertex in the original K_{N-1}. This will be demonstrated visually in the algorithm below. If K_{N-1} is Class 2, then each vertex will be missing a different color and that color can be used to color each edge to the new vertex for K_N. A simple algorithm to color a complete graph, K_N, is listed below (Bar-Noy [2]).

<u>If N is odd</u>:
1) Arrange the vertices as an n-polygon and color each edge with a different color.

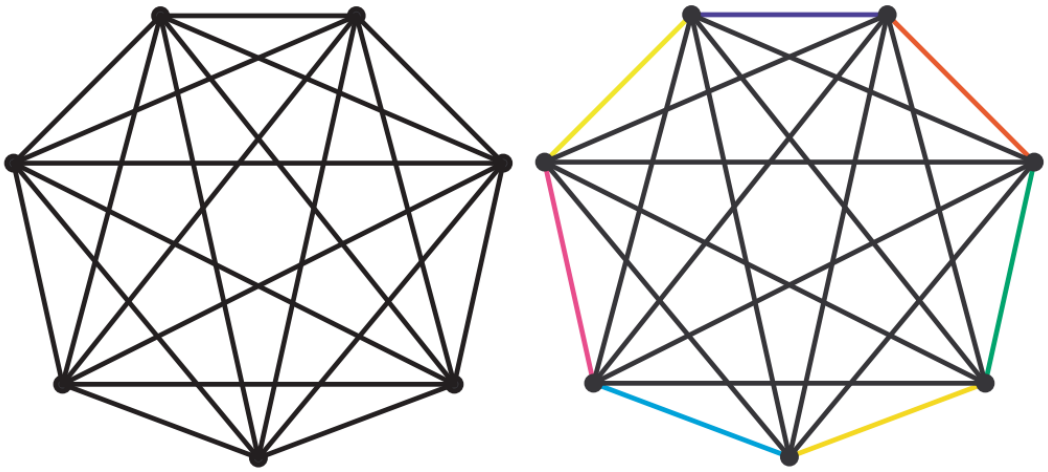

Figure 5: Coloring the outer edges of a K_7 graph.

2) For each outer edge, color its "inner" parallel edges the same color. Repeat for each outer edge.

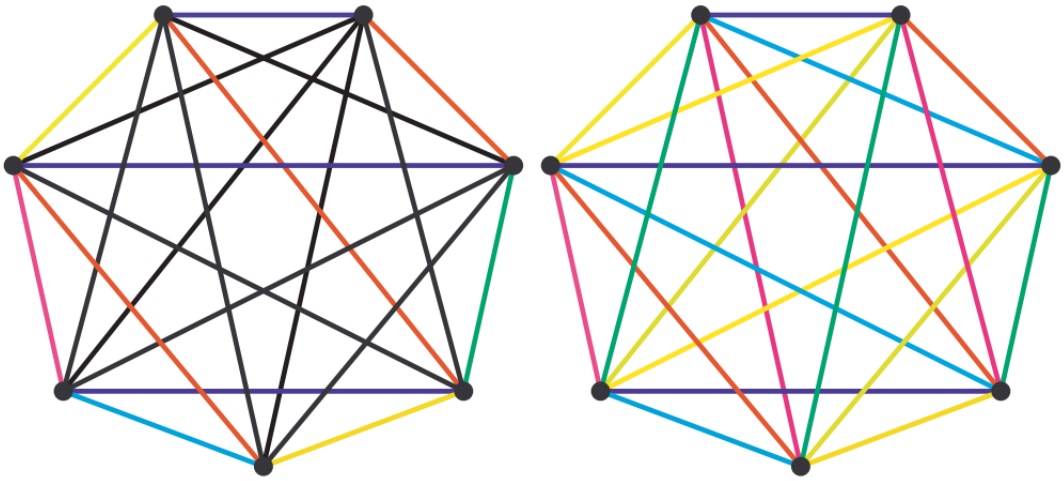

Figure 6: Coloring the first set of parallel edges red and blue (left) and the complete K_7 (right).

To clarify Step 2, the horizontal edge between the top vertices is assigned blue in Figure 6. The next lower horizontal edge is colored blue (the vertices adjacent to the initial pair) and so forth. The same parallel coloring is done for each outer edge of the polygon. This takes N colors and since $\Delta = N - 1$, this graph is Class 2.

If N is even:
1) Color a K_{N-1} using the algorithm shown above (N – 1 will be odd).
2) Add a new outer vertex and an edge between it and each vertex in K_{N-1}.
3) Color each edge with the missing color from its connected K_{N-1} vertex.

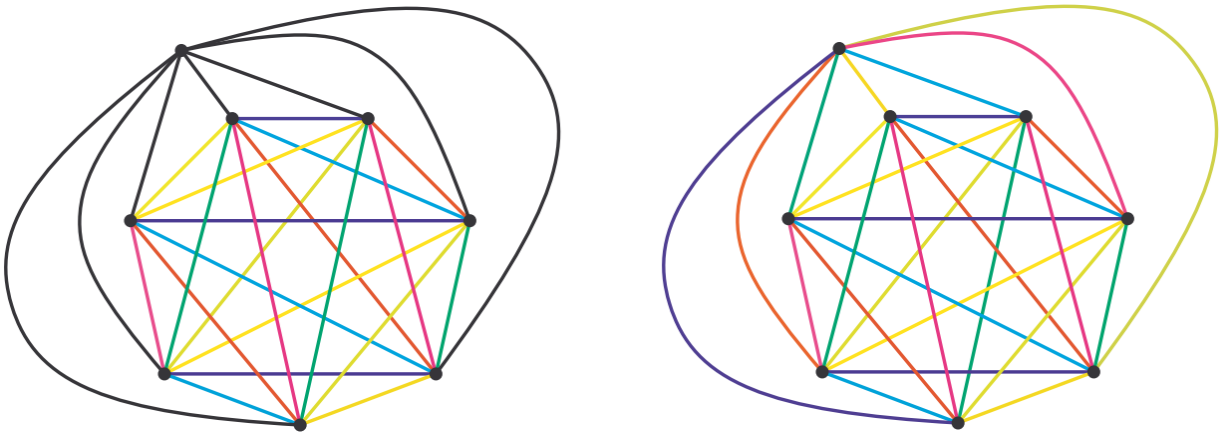

Figure 7: Coloring a K_8 using the inner K_7 from Figure 6.

Each vertex in the inner K_{N-1} has a missing color because its edge chromatic number is Δ + 1 but each vertex has degree Δ. By introducing the new vertex for K_N, a different color can be used for each edge and satisfy the edge coloring conditions.

4.3 Bipartite Graphs

Bipartite graphs are all Class 1; they can all be edge colored with Δ colors. By extension, all trees are bipartite graphs so all trees are also Class 1.

One definition of a bipartite graph is that it does not contain an odd cycle, agreeing with the results above. Another method of proving this involves Hall's Marriage Theorem (perfect matching) and König's theorem with induction. This paper will not formally explore this proof but the idea is that every bipartite graph can be extended to a k-regular bipartite graph (by adding edges and vertices) and because there is a perfect matching for this type of graph, it can be colored with k colors, where k is equal to Δ. These steps are the framework of a basic algorithm to color a bipartite graph in $O(\Delta|E|)$ time.

1) Extend a given bipartite graph G to a k-regular bipartite graph, G', where k = Δ(G). (Once G' is colored, the added edges and vertices can be removed to show a coloring for the original graph G.)

2) If k is odd, find a perfect matching, M, for graph G' (proven to exist via Hall's Marriage Theorem). Assign all the edges belonging to M as one new color and remove M from the graph. The graph is still k-regular but k is now even since one edge has been removed from every vertex. Proceed to Step 3.

3) If k is even, split the graph G' into two subgraphs G_1 and G_2 where half the edges from each vertex (from one "side" or independent set) are assigned to G_1 and the other half are assigned to G_2. Recursively apply the algorithm on each k'-regular subgraph from Step 2 or 3 where k' = ½k.

4) The base case is that when k = 1, all edges can be colored with one color and finish. If k = 2, then the resulting edges form an even cycle and can be colored with two colors.

This algorithm is a bit hard to visualize and future work is planned to be done to provide an animated demonstration to show the steps of this bipartite edge coloring algorithm. Further studies have been done to reduce the complexity of finding a perfect matching to push towards O (|E| log Δ) and other more efficient algorithms (Schrijver [5]).

4.4 Planar Graphs

A planar graph is defined as a graph drawn in a two-dimensional plane where no edges intersect or overlap each other except at their endpoints, which are the vertices in the graph. The areas bounded by the edges are called “faces” and the outer bounded area is also considered one face.

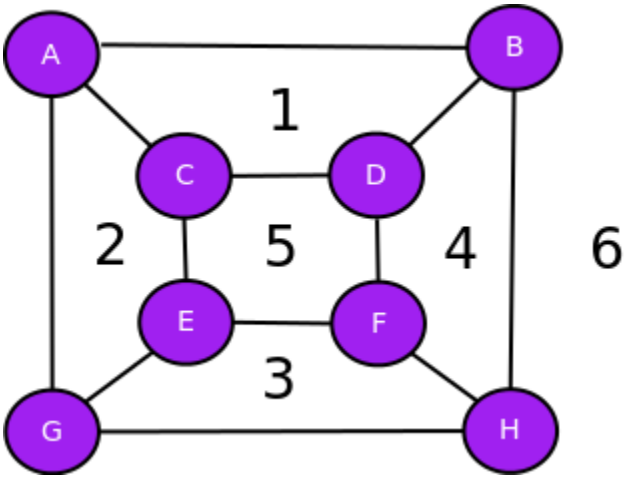

Figure 8: A planar graph with the vertices (letters) and faces (numbers) labelled.

For finite, connected, and simple planar graphs, Euler’s theorem states that the number of vertices (V) minus the number of edges (E) plus the number of faces (F) must always equal 2 or **V – E + F = 2**. Using this formula and substituting F with 2E (since each edge belongs to 1 or 2 faces), it can be shown through algebra that the average degree of planar graphs is less than 6. An equivalent statement is that all finite connected simple planar graphs have at least **one vertex of degree 5 or less**.

Vizing (1965) used Euler’s results to prove that all such planar graphs where **Δ ≥ 8** are Class 1. Sanders and Zhao (2001) later proved that planar graphs of **Δ ≥ 7** are also Class 1 (Bonamy [6]). Vizing was able to construct Class 2 graphs for Δ values between 2 and 5. Classifying planar graphs that have Δ = 6 is still an open problem.

Future work is planned to be done to demonstrate how a general Δ + 1 color algorithm for all simple graphs can be used for planar graphs. An algorithm using Δ colors was shown by He (1988) for planar graphs of Δ ≥ 33 that runs in linear O(n) time [7].

4.5 Multigraphs

Future Work

- Adding **Section 4.6 Line Graphs**, framing the edge coloring problem as a dual vertex coloring problem and proving other results
- An animation or visual demonstration of a general algorithm that uses Δ + 1 colors
- Other Algorithms and Comparisons

References & Further Reading

[1] https://en.wikipedia.org/wiki/Edge_coloring
Wikipedia entry, a basic overview of edge coloring

[2] http://www.sci.brooklyn.cuny.edu/~amotz/GC-ALGORITHMS/PRESENTATIONS/edge-coloring.pdf
Amotz Bar-Noy, Brooklyn College lecture notes on edge coloring, contains detailed breakdowns and helpful visuals that were used for many of the figures in this report

[3] http://web.iyte.edu.tr/~tinabeseri/tina-tez.pdf
Tina Beseri, thesis on edge coloring, a wide range of topics with formal proofs and a section on using Mathmatica for implementation of algorithms

[4] http://math.uchicago.edu/~may/REU2015/REUPapers/Green.pdf
Robert Green, more on edge coloring theory and Vizing

[5] http://homepages.cwi.nl/~lex/files/edgecol.pdf
Alexander Schrijver, bipartite graph edge coloring algorithms

[6] http://www.labri.fr/perso/mbonamy/delta8court.pdf
Marthe Bonamy, proofs for edge coloring planar graphs

[7] http://www.sciencedirect.com/science/article/pii/030439759090079W
Xin He, an algorithm for planar graphs using Δ colors

[8] http://www.jhuapl.edu/techdigest/TD/td2302/Gilbert.pdf
Jeffrey Gilbert, multigraph edge coloring and strategies