EDGE COLORING

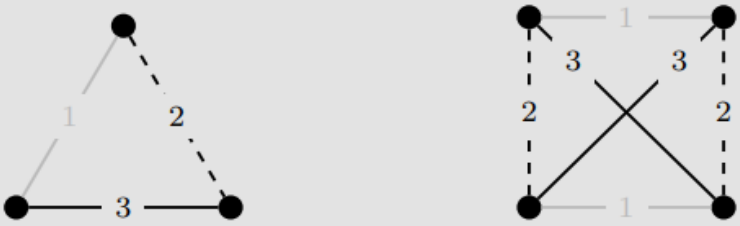
- This section focuses on a different aspect of graph coloring where instead of assigning colors to the vertices of a graph, we will instead assign colors to the edges of a graph.
- Such colorings are called **edge-colorings** and have their own set of definitions and notations, many of which are analogous to those for vertex colorings from above.
- Like our previous study of vertex colorings, we will only consider simple graphs, that is graphs without multi-edges. (Note that a graph with a loop cannot be edge-colored).

Definition 6.18 Given a graph $G = (V, E)$ an ***edge-coloring*** is an assignment of colors to the edges of G so that if two edges share an endpoint, then they are given different colors. The minimum number of colors needed over all possible edge-colorings is called the ***chromatic index*** and denoted $\chi'(G)$.

Example 6.9 Recall that the chromatic number for any complete graph is equal to the number of vertices. Find the chromatic index for K_n for all n up to 6.

Solution: Since K_1 is a single vertex with no edges and K_2 consists of a single edge, we have $\chi'(K_1) = 0$ and $\chi'(K_2) = 1$. Due to their simplicity, a drawing is omitted for these two graphs.

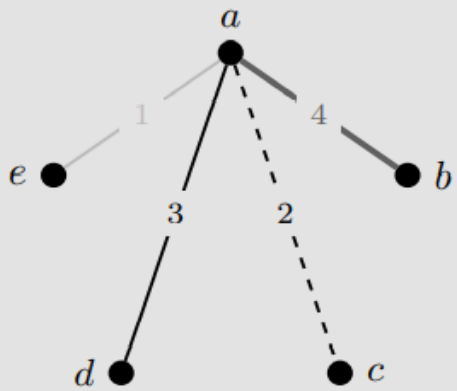
For K_3 since any two edges share an endpoint, we know each edge needs its own color and so $\chi'(K_3) = 3$. For K_4 we can color opposite edges with the same color, thus requiring only 3 colors. Optimal edge-colorings for K_3 and K_4 are shown below.

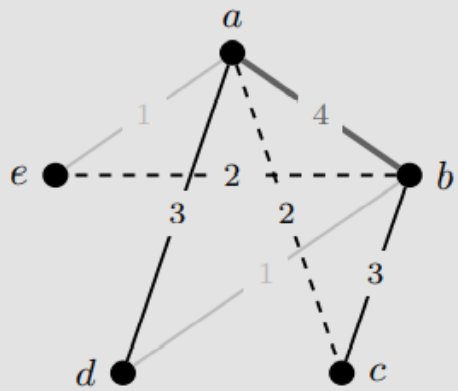


Edge Coloring for K_5 & K_6:

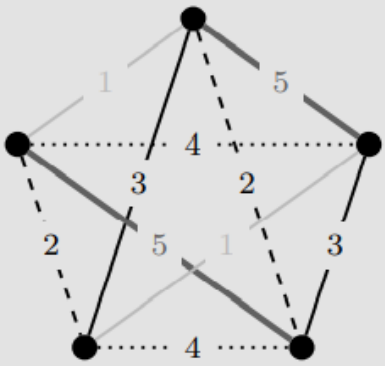
Edge-coloring K_5 and K_6 is not quite so obvious as those from above.

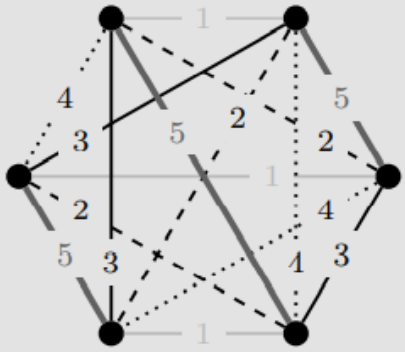
- Since no two adjacent edges can be given the same color, we know every edge out of a vertex must be given different colors.
- Since every vertex in K_5 has degree 4, we know at least 4 colors will be required.
- Start by using 4 colors out of one of the vertices of the K_5. As shown in the next graph on left, we started at a.
- Moving to the edges incident to b, we attempt to use our pool of 4 previously used colors; however, one of these is unavailable since it has already been used on the **ab** edge.
- A possible coloring of the edges incident to b is given in the next graph on the right.





At this point c is left with two incident edges to color but only one color available, namely the one used on edge ab, thus requiring a fifth color to be used. A proper edge-coloring of K_5 is shown below on the left. A similar procedure can be used to find a coloring of K_6 using only 5 colors, as shown below on the right.





In general, $\chi'(K_n) = n-1$ when n is even and $\chi'(K_n) = n$ when n is odd.

Explanation: From the example above, we conclude that if a vertex x has degree k, then the entire graph will need at least k colors since each of the edges incident to x will need a different color.

- Thus, the chromatic index must be at least the maximum degree, $\Delta(G)$, of the graph.
- But notice that for odd values of n, that K_n required one more color than the maximum degree (for example, $\chi'(K_5) = 5$ and $\Delta(K_5) = 4$).
- In fact, any graph will either require $\Delta(G)$ or $\Delta(G) + 1$ colors to color its edges.
- This is a much tighter bound than we were able to find for the chromatic number of a graph and was proven in 1964 by the Ukrainian mathematician, Vadim Vizing.

Theorem 6.19 (Vizing's Theorem) $\Delta(G) \leq \chi'(G) \leq \Delta(G) + 1$ for all simple graphs G.

Results:

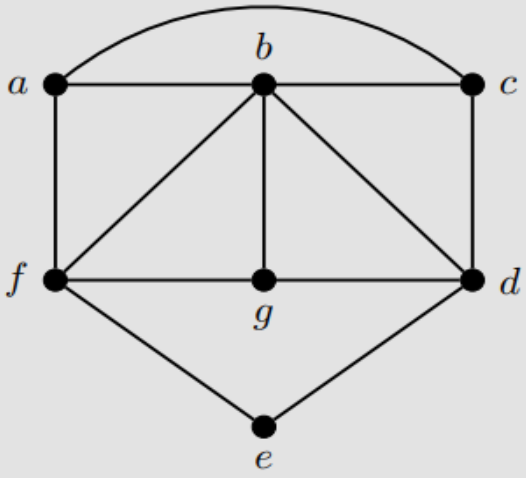
- Graphs are referred to as **Class 1** if $\chi'(G) = \Delta(G)$ and **Class 2** if $X'(G) = \Delta(G) + 1$.
- Some graph types are known to be in each class (for example, **bipartite graphs** are Class 1 and **regular graphs** with an odd number of vertices are Class 2).

Remarks:

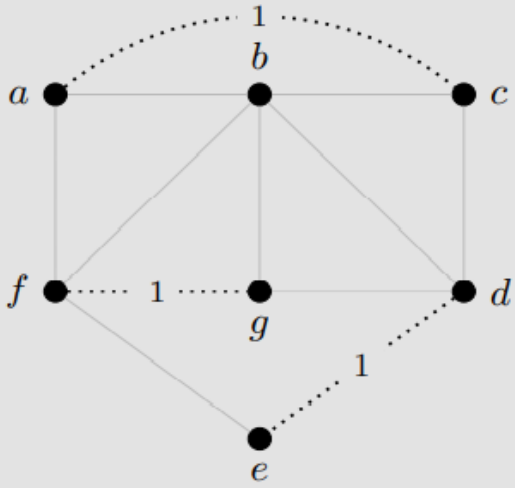
- However, determining which class a graph belongs to is a nontrivial problem.
- For graphs of small size, it is not difficult to find an optimal or nearly optimal edge-coloring.
- Using a **greedy algorithm** (where the first color available is used) will produce an edge-coloring with at most $2\Delta(G) - 1$ colors on any simple graph.
- More complex algorithms exist that improve on this bound, and if color shifting is allowed then an algorithm exists that will produce an edge-coloring with at most $\Delta(G) + 1$ colors.

The example below investigates a suboptimal edge-coloring using a greedy algorithm and explains a better procedure for finding an optimal (or nearly optimal) edge-coloring.

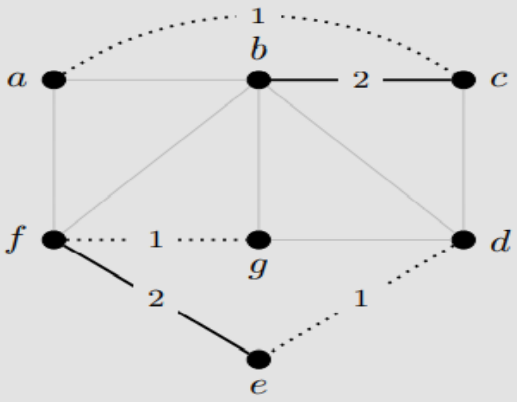
Example 6.10 Consider the graph G_4 below and color the edges in the order $ac, fg, de, ef, bc, cd, dg, af, bd, bg, bf, ab$ using a greedy algorithm.



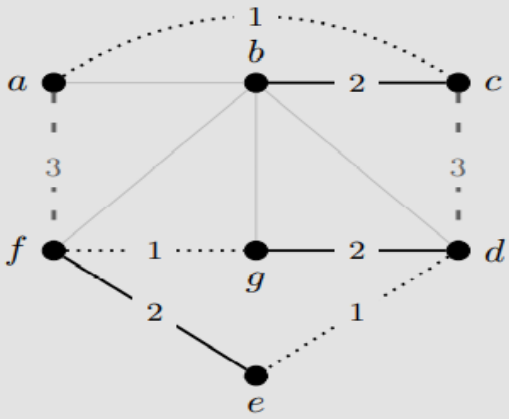
Step 1: Since the first three edges ac, fg, and de are not adjacent, we give each of them the first color.



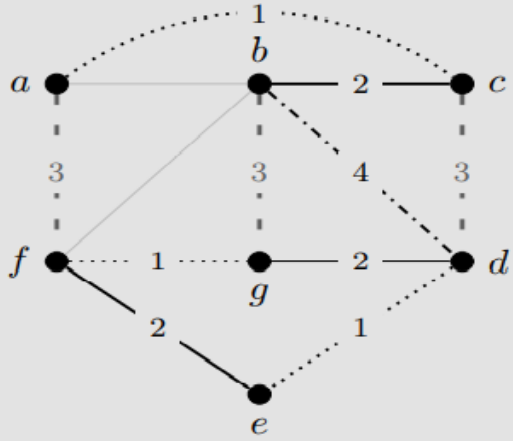
Step 2: Now since ef is adjacent to a previously colored edge, we need a second color for it. Moreover, bc must also use a second color and since these are not adjacent they can have the same color.



Step 3: Edge cd is adjacent to edges using the first two colors, so a third color is needed. Edge dg can use the second color and edge af must use the third color.

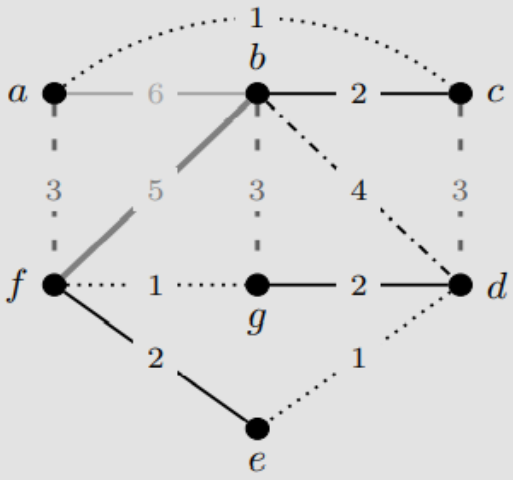


Step 4: Edge bd needs a fourth color since it is adjacent to edges using each of the previous three colors. However, bg can be given the third color.



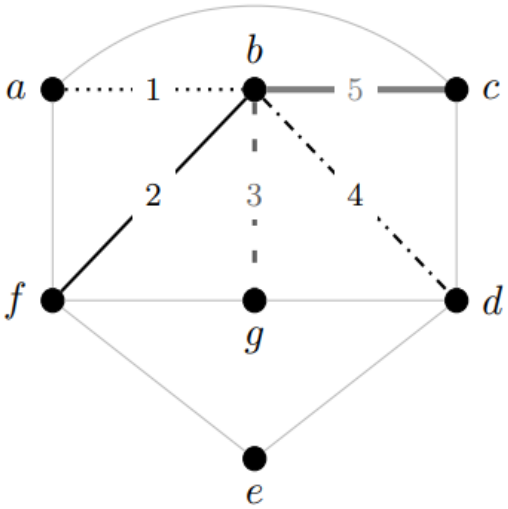
Step 5: Edge bf needs a fifth color since it is adjacent to the first three colors through f and adjacent to the fourth color through b. Moreover, ab

needs a sixth color since it is adjacent to the first and third colors through a and the second through fifth colors from b.

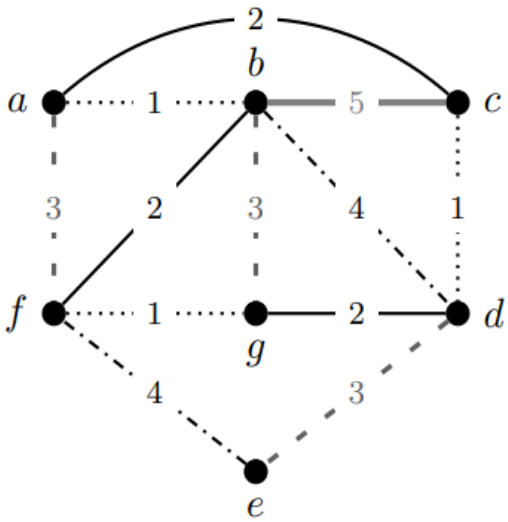


- In the example above Δ(G4) = 5, and the edge-coloring above uses 6 colors; however χ'(G4) = 5.
- In general, starting with the vertex of highest degree and coloring its edges has a better chance of success in avoiding unnecessary colors.

Once we have the minimum number of colors established, we attempt to fill in the remaining edges without introducing an extra color; one possible solution is shown below:



initial coloring of G_4



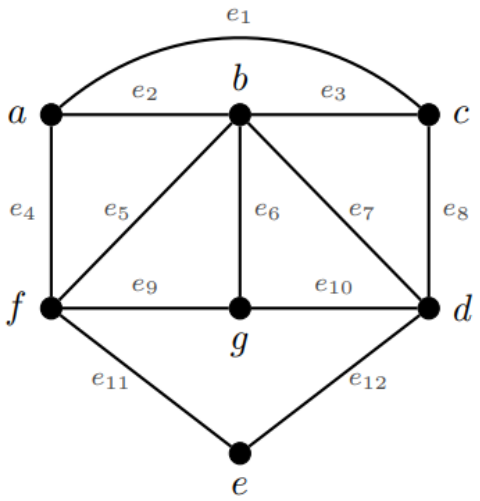
optimal edge-coloring of G_4

- ✓ Beyond the analogous definitions and procedures between edge-coloring and vertex-coloring, there is a very direct relationship between the two by the use of a line graph.

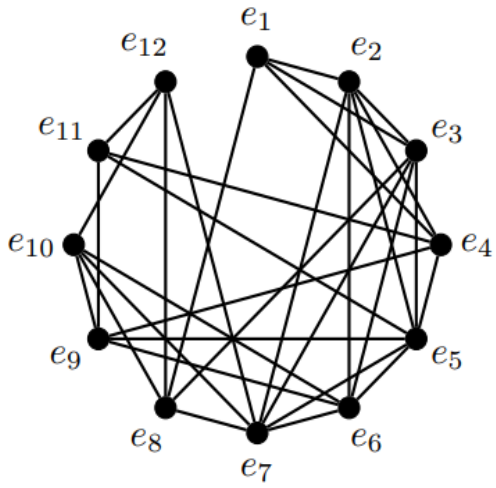
Definition 6.20 Given a graph $G = (V, E)$, the ***line graph*** $L(G) = (V', E')$ is the graph formed from G where each vertex x' in $L(G)$ represents the edge x' from G and $x'y'$ is an edge of $L(G)$ if the edges x' and y' share an endpoint in G.

Example:

Below is the graph G_4 from Example 6.10 and its line graph. Notice that the vertex e_1 in $L(G_4)$ is adjacent to e_2 and e_4 through the vertex a in G_4 and e_1 is adjacent to e_3 and e_8 through the vertex c in G_4.



G_4



$L(G_4)$

Remark:

From this definition, it should be clear how edge-coloring and vertex coloring are related.

- If edge e_1 is given the color blue in G, this would correspond to coloring vertex e_1 in L(G) with blue.
- This correspondence provides the following result.

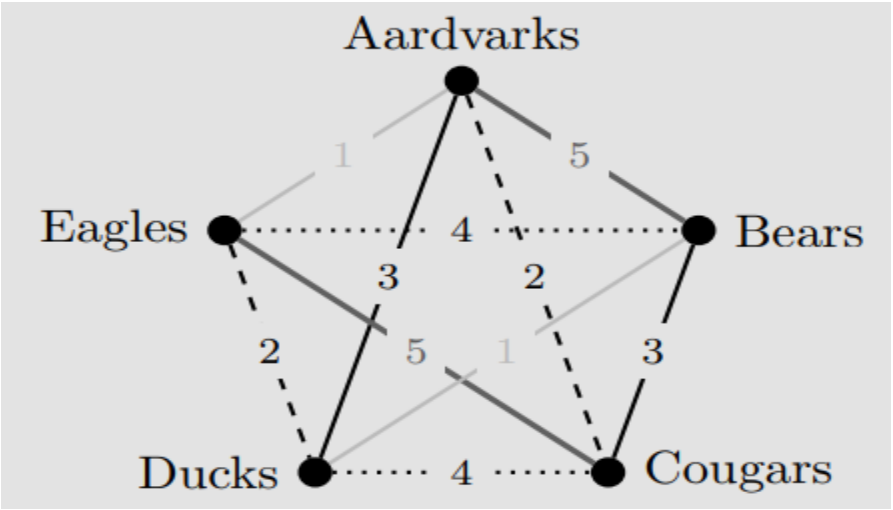
Theorem 6.21 Given a graph G with line graph $L(G)$, we have $\chi'(G) = \chi(L(G))$.

- The result above shows we can find an edge-coloring of any graph by simply vertex-coloring its line graph.
- However, other results on line graphs provide some interest, namely if G is eulerian then L(G) is hamiltonian!
- Applications of edge-coloring abound, in particular scheduling independent tasks onto machines and communicating data through a fiber-optic network.

Example:

Example 6.11 The five teams from Section 1.1 (Aardvarks, Bears, Cougars, Ducks, and Eagles) need to determine the game schedule for the next year. If each team plays each of the other teams exactly once, determine a schedule where no team plays more than one game on a given weekend.

Solution: Represent each team by a vertex and a game to be played as an edge between the two teams. Then exactly one edge exists between every pair of vertices and so K_5 models the system of games that must be played. Assigning a color to an edge corresponds to assigning a time to a game. By the discussion from Example 6.9, we know $\chi'(K_5) = 5$ and so 5 weeks are needed to schedule the games. One such solution is shown below.



Week	Games	
1	Aardvarks vs. Bears	Cougars vs. Eagles
2	Aardvarks vs. Cougars	Ducks vs. Eagles
3	Aardvarks vs. Ducks	Bears vs. Cougars
4	Aardvarks vs. Eagles	Bears vs. Ducks
5	Bears vs. Eagles	Cougars vs. Ducks

Although the example above is fairly easy to solve as the graph model is a complete graph, the same procedure can be extended to a larger number of teams where each team only plays a subset of all teams in the league.

- ✓ Edge-coloring can be used to determine team schedules in the National Football League!

RAMSEY NUMBER:

- As with vertex-coloring, when investigating edge-colorings we are often concerned with finding an optimal coloring (using the least number of colors possible).
- Other problems exist, where optimality is no longer the goal.
- One such edge-coloring problem relaxes some of the restrictions on coloring the edges, and is named for the British mathematician and economist, Frank P. Ramsey.
- Although a simple concept, it birthed an area of mathematics now known as Ramsey Theory.
- We will discuss Ramsey numbers as they relate to coloring the edges of a graph.

Unlike our edge-colorings above in which no two edges can be given the same color if they have a common endpoint, here we will be concerned with specific monochromatic structures within the larger graph.

Definition 6.23 Given positive integers m and n, the ***Ramsey number*** $R(m, n)$ is the minimum number of vertices r so that all simple graphs on r vertices contain either a clique of size m or an independent set of size n.

- Recall that a clique of size m refers to a subgraph with m vertices in which there exists an edge between every pair of vertices.
- An independent set of size n is a group of n vertices in which no edge exists between any two of these n vertices.
- To get a better handle on this technical definition, Ramsey numbers are often described in terms of guests at a party.

Question: If you wanted to find R(3, 2), then you would be asking how many guests must be invited so that at least 3 people all know each other or at least 2 people do not know each other. Try it!

Intuitive Idea:

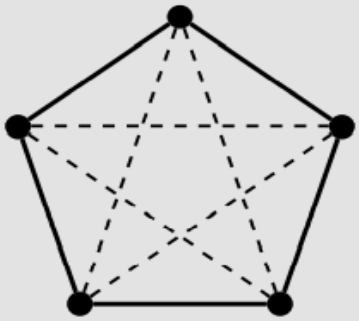
- Ramsey numbers can be viewed as coloring the edges of a complete graph using two colors, say red and blue.
- So that either a red clique of size m exists, or a blue clique of size n exists.
- You can view the blue clique as the edges that would not exist in the graph, thus making their endpoints an independent set of vertices.

Task: Proving R(m, n) = r requires two steps:

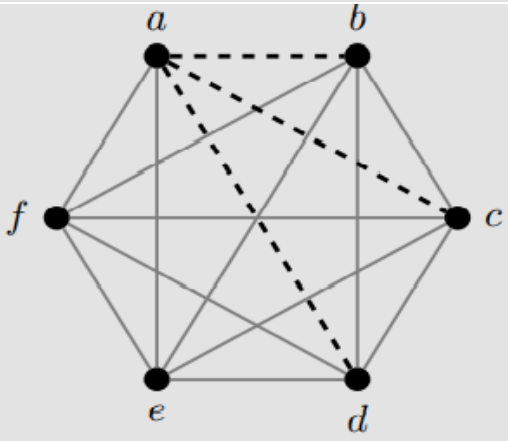
- first, we find an edge coloring of $K_r - 1$ without a red m-clique and without a blue n-clique;
- second, we must show that any edge-coloring of K_r will have either a red m-clique or a blue n-clique.

Example 6.12 Determine $R(3,3)$.

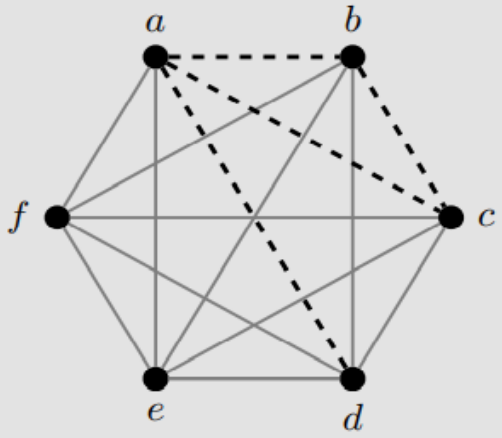
Solution: First note that we are searching for a 3-clique of either color, also known as a monochromatic triangle. However, $R(3,3) > 5$ as the edge-coloring of K_5 below does not contain a monochromatic triangle.



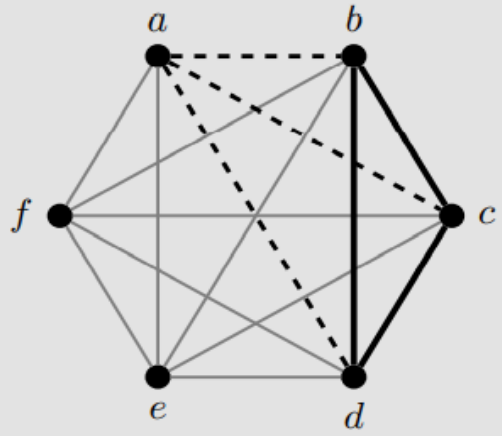
Next, consider K_6. Since each vertex has degree 5 and we are coloring the edges using only two colors, we know every vertex must have at least 3 adjacent edges of the same color. Suppose a has three adjacent dashed edges, as shown below. All other edges are shown in gray to indicate we do not care (yet) which color they have.



If any one of the edges between b, c, and d is dashed, then a dashed triangle exists using the edges back to a. One possibility is shown in the next graph.



Otherwise, none of the edges between b, c and d are dashed, creating a bold triangle among these vertices.



Although the argument above used vertex a with three adjacent dashed edges, a similar argument would hold for any vertex of K_6 and for a vertex with three black adjacent edges. Thus $R(3, 3) = 6$.

Results:

Using the discussion at the end of the previous example for inspiration, we note two Ramsey number relationships:

- $R(m, n) = R(n, m)$
- $R(2, n) = n$

Although the solution for R(3, 3) was not too difficult to determine, increasing m and n both by one value greatly increases the complexity of a solution.

- In fact, R(4, 4) = 18 was proven in 1955 and yet R(5, 5) is still unknown.

The following table lists some known values and bounds for small values of m and n at the time of publication. A single value in a column indicates the exact value is known; otherwise, the two values given are the upper and lower bounds as stated.

m	3	4	5		6		7	
n			lower	upper	lower	upper	lower	upper
3	6	9	14		18		23	
4		18	25		36	41	49	61
5			43	48	58	87	80	143
6					102	165	115	298
7							205	540

COLORING VARIATION:

- The previous sections should show just how varied and deep of a field is graph coloring.
- This section will highlight additional variations of graph coloring, specifically vertex colorings.
- Within each of these we will see applications of graph coloring that can inform why this new version of coloring is worthy of study.

On-Line Coloring:

- On-line coloring differs from a general coloring in that the vertices are examined one at a time (hence they are seen in a linear manner, or "on a line").
- The notion of an on-line coloring relies on a specific type of subgraph, called an induced subgraph.
- We need induced subgraphs for coloring problems since if we only took any subgraph and colored it, we may be missing edges that would indicate two vertices need different colors in the larger graph.
- On-line coloring algorithms require a vertex to be colored based only upon the induced subgraph containing that vertex and the previously

colored vertices.

Definition 6.24 Consider a graph G with the vertices ordered as $x_1 \prec x_2 \prec \cdots \prec x_n$. An ***on-line algorithm*** colors the vertices one at a time where the color for x_i depends on the induced subgraph $G[x_1, \ldots, x_i]$ which consists of the vertices up to and including x_i. The maximum number of colors a specific algorithm $\mathcal{A}$ uses on any possible ordering of the vertices is denoted $\chi_{\mathcal{A}}(G)$.

- Many different on-line algorithms exist, some of which can be quite complex.
- Mathematicians are often interested in finding an on-line algorithm that works well on a specific type of graph, or in showing how the underlying structure of specific types of graphs limits the performance of any on-line algorithm.
- We will focus on a **greedy algorithm** called **First-Fit** that uses the first available color for a new vertex.

First-Fit Coloring Algorithm

Input: Graph G with vertices ordered as $x_1 \prec x_2 \prec \cdots \prec x_n$.

Steps:

1. Assign x_1 color 1.
2. Assign x_2 color 1 if x_1 and x_2 are not adjacent; otherwise, assign x_2 color 2.
3. For all future vertices, assign x_i the least number color available to x_i in $G[x_1, \ldots, x_i]$; that is, give x_i the first color not used by any neighbor of x_i that has already been colored.

Output: Coloring of G.

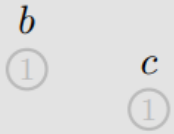
- ✓ One of the benefits of First-Fit is the ease with which it is applied. When a new vertex is encountered, we simply need to examine its neighbors that have already been colored and give the new vertex the least color available.

Example 6.13 Apply the First-Fit Algorithm to the graph from Example 6.3 if the vertices are ordered alphabetically.

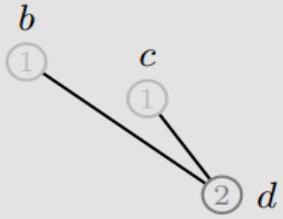
Solution: To emphasize that only some of the graph is available at each step of the algorithm, only the edges to previously considered vertices will be drawn.

Step 1: Color b with 1.

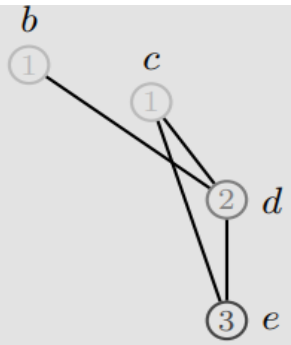
Step 2: Color c with 1 since b and c are not adjacent.



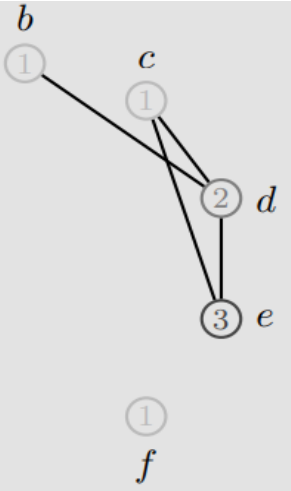
Step 3: Color d with 2 since d is adjacent to a vertex of color 1.



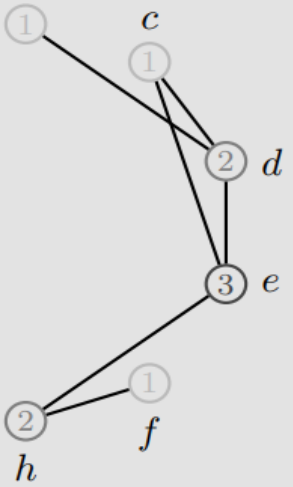
Step 4: Color e with 3 since e is adjacent to a vertex of color 1 (c) and a vertex of color 2 (d).



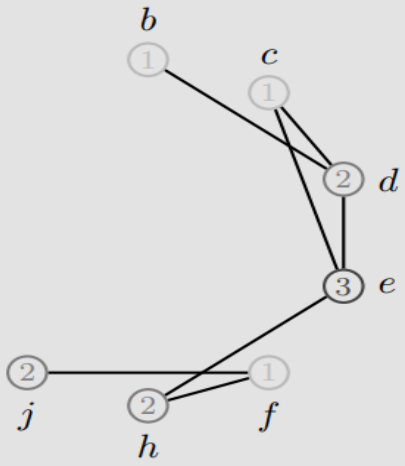
Step 5: Color f with 1 since f is not adjacent to any previous vertices.



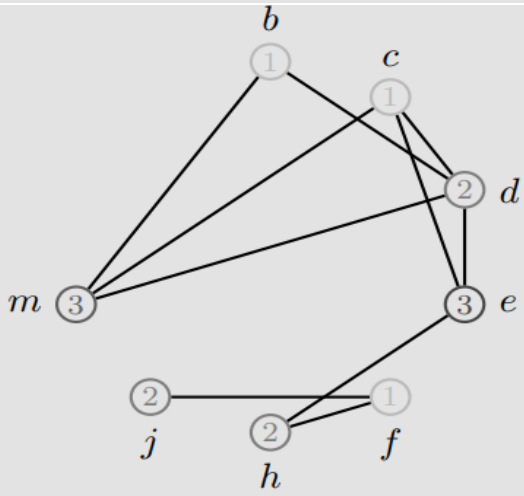
Step 6: Color h with 2 since h is adjacent to a vertex of color 1 (f).



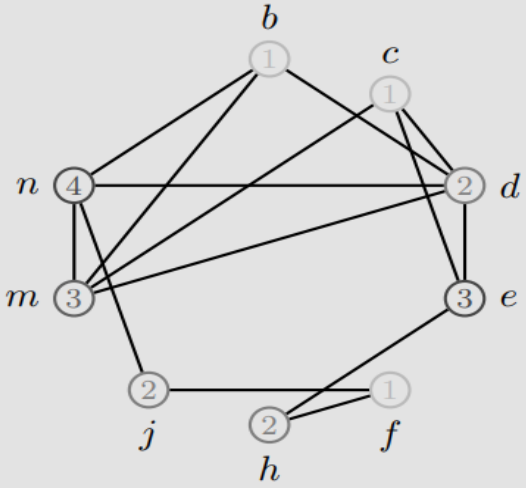
Step 7: Color j with 2 since j is adjacent to a vertex of color 1 (f).



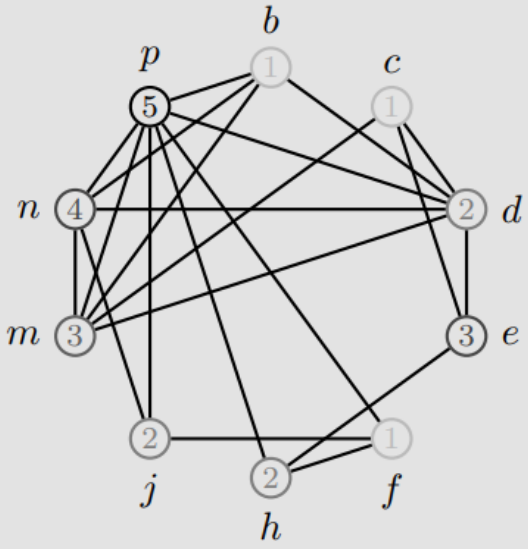
Step 8: Color m with 3 since m is adjacent to vertices of color 1 (b, c) and a vertex of color 2 (d).



Step 9: Color n with 4 since n is adjacent to vertices of color 1, 2, and 3.



Step 10: Color p with 5 since p is adjacent to vertices of color 1, 2, 3, and 4.



- As the example above shows, First-Fit can perform quite well given the proper ordering.
- Unfortunately, given the right graph and wrong order of the vertices, First-Fit can perform remarkably poorly, as seen below.

Example 6.14 A vertex will be revealed one at a time, along with any edges to previously seen vertices. The First-Fit Algorithm will be applied in the order the vertices are seen.

Step 1: The first vertex is v_1. It is given color 1.

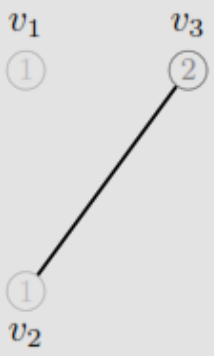
v_1

Step 2: The second vertex is v_2. Since there is no edge to v_1, it is also given color 1.

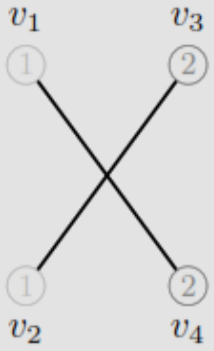
v_1
1

1
v_2

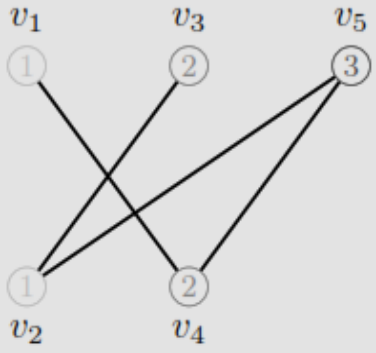
Step 3: The third vertex, v_3, has an edge to v_2 and so must be assigned color 2.



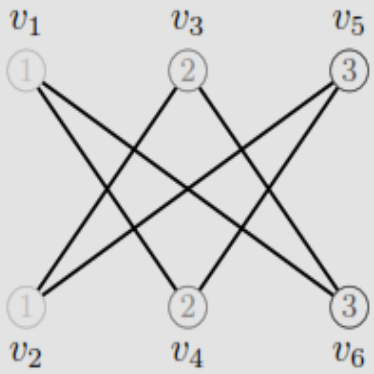
Step 4: The fourth vertex, v_4, has an edge to v_1 but not v_3 and so is also given color 2.



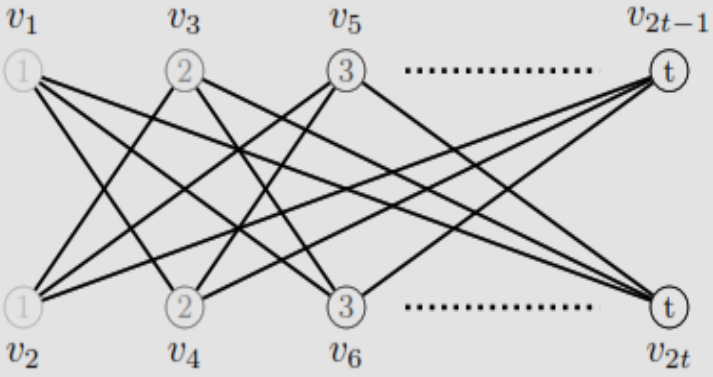
Step 5: The fifth vertex, v_5, is adjacent to a vertex of color 1 (v_2) and a vertex of color 2 (v_4). It must be assigned color 3.



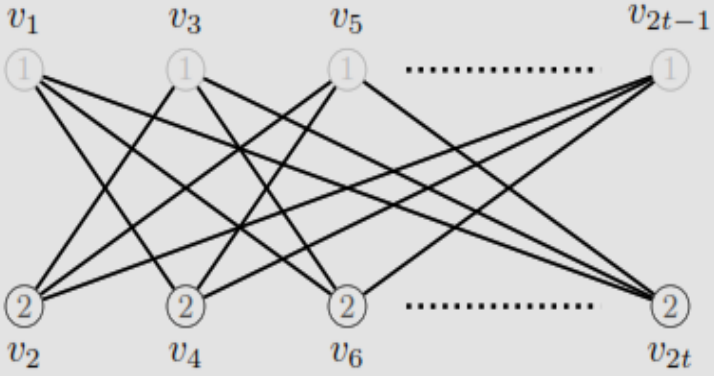
Step 6: The sixth vertex, v_6, is also adjacent to a vertex of color 1 (v_1) and a vertex of color 2 (v_3) but not a vertex of color 3 so it can also be given color 3.



If we continue in this fashion, after $2t$ steps we will have used t colors.



However, this graph is bipartite and every bipartite graph can be colored using 2 colors, as shown below.



- First-Fit is in fact one of the worst performers for on-line algorithms since it uses very little knowledge of how the previous vertices were treated.

- When evaluating a coloring algorithm's performance, we look at the worst case scenario, that is we are asking what is the most number of colors the algorithm could use on any possible ordering of the vertices, denoted $\chi_{FF}(G)$.
- It is possible for an on-line algorithm (even First-Fit) to provide an optimal coloring, but in many cases this does not occur.

On-line coloring algorithms perform nicely on interval graphs.

- In particular, when the interval representation is known and the vertices are ordered by starting time of their intervals, then First-Fit will produce a coloring using exactly $\chi(G)$.
- However, if the intervals are not ordered by their starting time or a different on-line algorithm is used, the optimal coloring may not be found.

Example 6.15 (H.W).

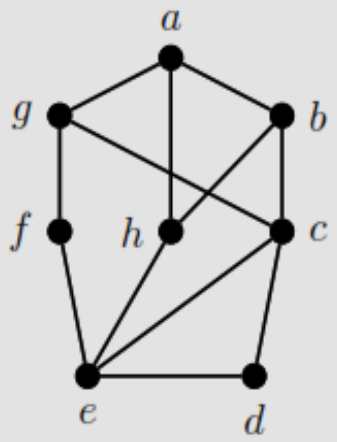
Remarks:

- Even with all its limitations, First-Fit is a useful strategy when attempting to prove bounds on the chromatic number of a graph.
- In particular, we will use it below in our proof of Brooks' Theorem (from Section 6.2.2).
- On-line coloring, and more specifically First-Fit, can be applied to other graph coloring variations (even edge-coloring).
- One can view these algorithms from a more applied perspective, as an additional strategy for tackling a graph coloring problem, or through a theoretical perspective, where we want to think about the limiting behavior.
- Below we include one more example of each, though these just scratch the surface of the abundance of problems surrounding on- line coloring.

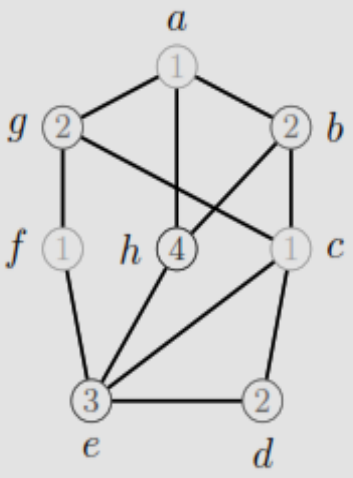
Example 6.16 Color the graph below using First-Fit using the two different vertex orders listed and determine if either finds the optimal coloring for the graph.

(a) $a \prec b \prec c \prec d \prec e \prec f \prec g \prec h$

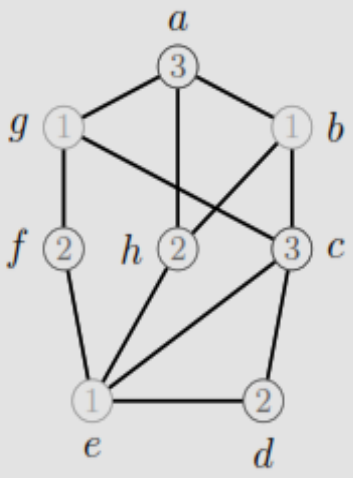
(b) $e \prec h \prec b \prec d \prec c \prec g \prec f \prec a$



Solution: Below are the two First-Fit colorings of the graph based on the order given. Only the vertex order from (b) gives the optimal coloring since $\omega(G) = 3$.



vertex order (a)



vertex order (b)

WEIGHED COLORING:

Consider the following scenario:

Ten families need to buy train tickets for an upcoming trip. The families vary in size but each of them needs to sit together on the train. Determine the minimum number of seats needed to accommodate the ten family trips.

Explanation:

This problem should sound very similar to Example 6.15 where colors were representing seats and the vertices were intervals of time indicative of when a person was on the train. Here, we are still interested in a graph coloring, but now each vertex represents a family and so has a size associated with it.

- In previous chapters, we used weights on the edges of a graph to indicate distance, time, or cost.
- For graph coloring models, weighted edges would have very little meaning.
- Instead, we will be assigning a weight to each vertex, and finding a proper coloring will be referred to as a weighted coloring.

Definition 6.25 Given a weighted graph $G = (V, E, w)$, where w assigns each vertex a positive integer, a proper ***weighted coloring*** of G assigns each vertex a set of colors so that

(i) the set consists of consecutive colors (or numbers);

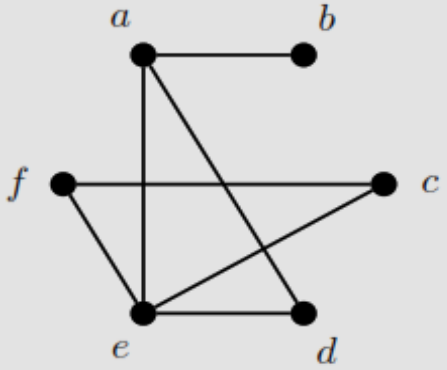
(ii) the number of colors assigned to a vertex equals its weight; and

(iii) if two vertices are adjacent, then their set of colors must be disjoint.

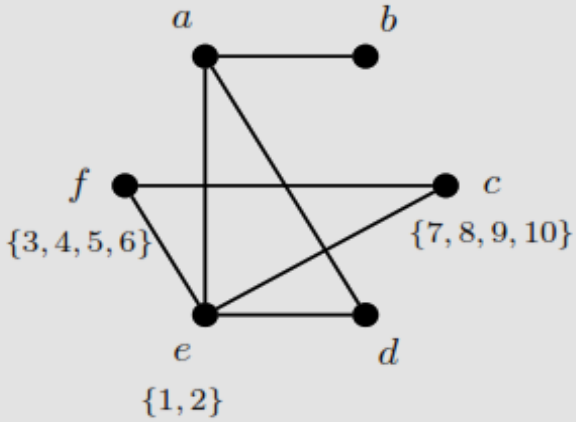
- Note that in some publications, weighted colorings are referred to as interval colorings (since an interval of colors is being assigned to each vertex).
- To avoid the confusion between interval colorings and interval graphs, we use the term weighted coloring.

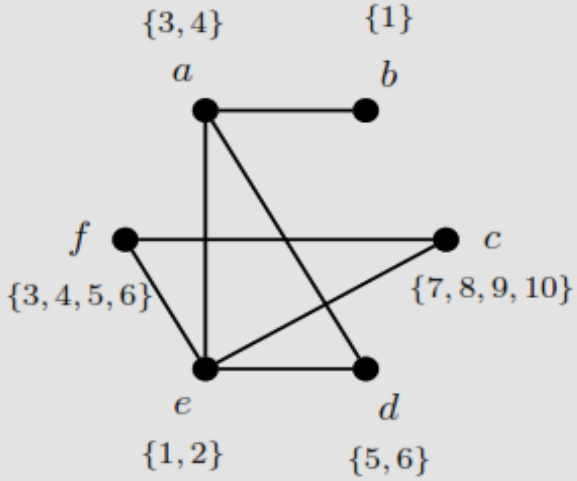
Example 6.18 Find an optimal weighted coloring for the graph below where the vertices have weights as shown below.

$w(a) = 2$
$w(b) = 1$
$w(c) = 4$
$w(d) = 2$
$w(e) = 2$
$w(f) = 4$



Solution: Note that a, d, and e form a K_3 with total weight 8, and c, e, and f form a K_3 with total weight 10. We begin by assigning weights to the vertices from the second K_3, and since e appears twice we assign it the first set of colors $\{1, 2\}$. From there we are forced to use another 4 colors on f ($\{3, 4, 5, 6\}$) and another 4 colors on c ($\{7, 8, 9, 10\}$), as shown on the left below. We can color the remaining three vertices using colors 1 through 10 as shown on the right.





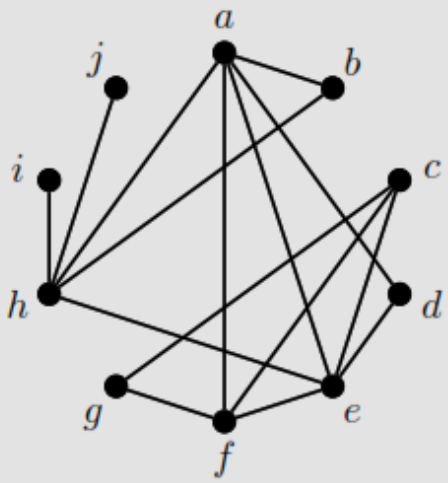
Remarks:

- we focus less on a vertex degree and more on the vertex weight when we are searching for a minimal weighted coloring.
- This is in part due to the need for the set of colors to be consecutive.

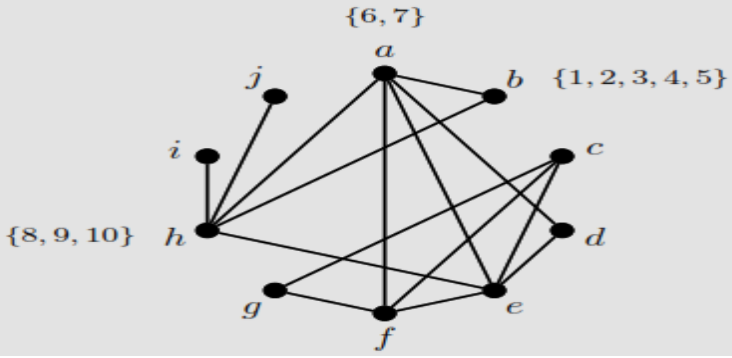
- If a vertex has **high weight**, then it needs a larger range from which to pick the set of colors.
- Whereas a vertex with large degree but a small weight may be able to squeeze in between the sets of colors of its neighbors.

Example 6.19 Suppose the ten families needing train tickets have the same underlying graph as that from Example 6.15 and the size of each family is noted below. Determine the minimum number of seats needed to accommodate everyone's travels.

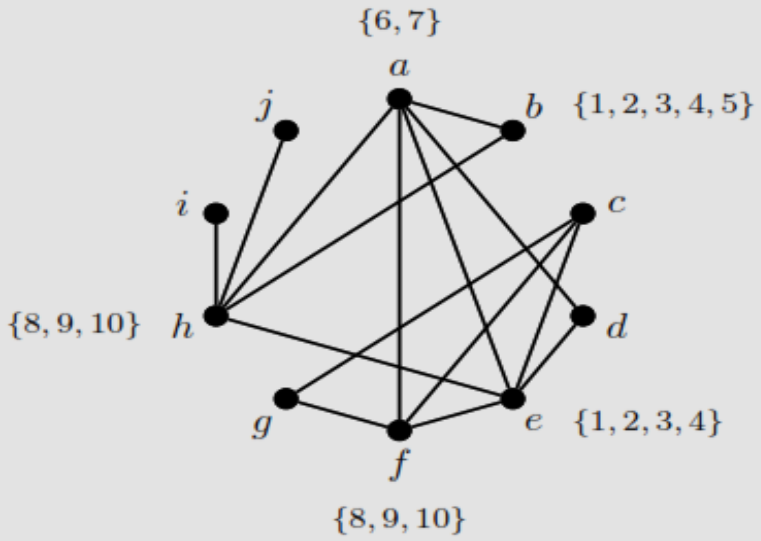
$$
\begin{aligned}
w(a) &= 2 \\
w(b) &= 5 \\
w(c) &= 2 \\
w(d) &= 1 \\
w(e) &= 4 \\
w(f) &= 3 \\
w(g) &= 1 \\
w(h) &= 3 \\
w(i) &= 4 \\
w(j) &= 5
\end{aligned}
$$



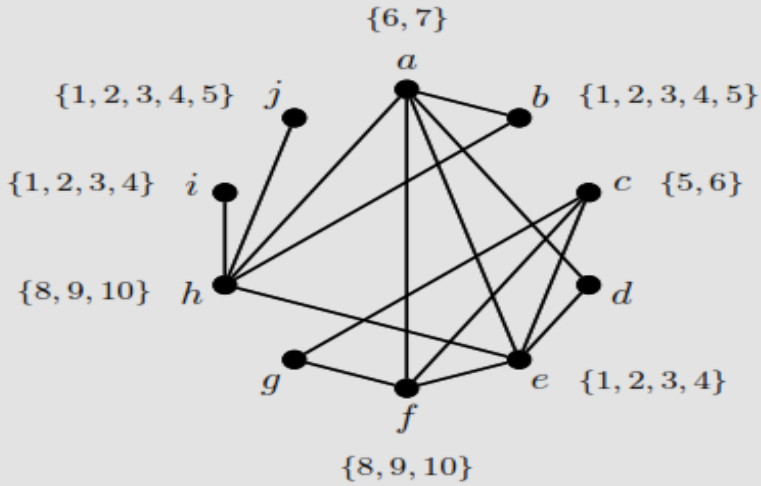
Solution: As with the example above, we begin by looking for the largest total weight for a clique. Since the clique size is three, we want to find K_3 subgraphs with high total weight. The largest of these is with a, b, and h with total 10. Since b has the largest weight among these, we give it colors 1 through 5, assign a colors 6 and 7, and colors 8, 9, and 10 to h.



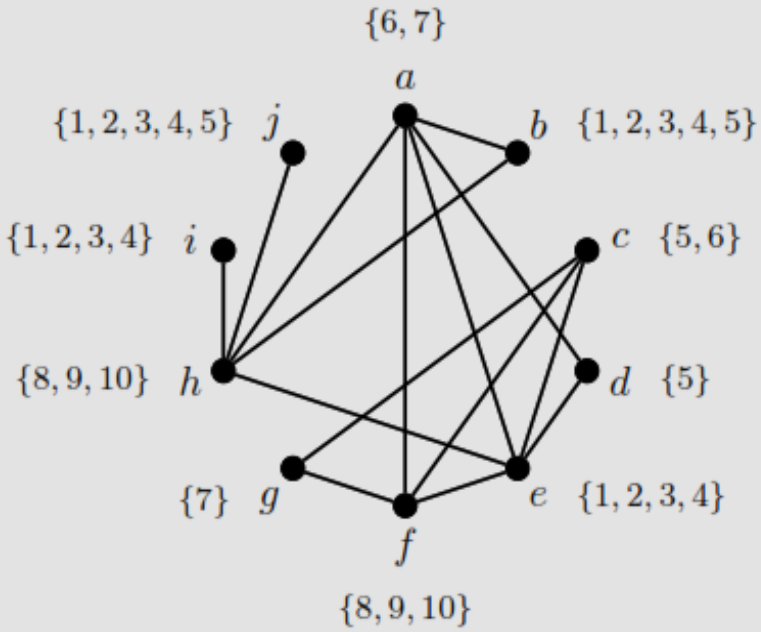
The next two cliques with a high total weight (of 9) are formed by a, e, f and c, e, f. Since a has already been assigned colors, we begin with the first clique. We can fill in the colors for e and f without introducing new colors, as shown below. Note that since e is adjacent to both a and h it could only use four consecutive colors chosen from those used on b.



At this point we can fill in the colors for c by choosing two consecutive colors from those available (5, 6, and 7). We also fill in the colors for i and j by choosing the correct number of consecutive colors from any of 1 to 7.



Finally, we need one color each for d and g. These can be any color not already used by one of their neighbors.



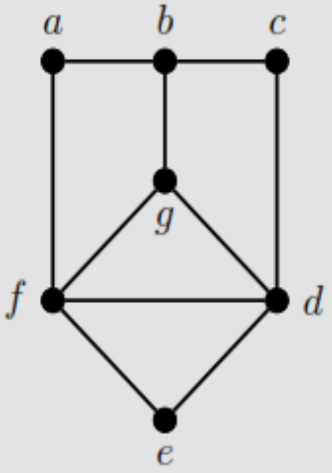
- On-line algorithms, and specifically First-Fit, can also be used for weighted colorings.
- It should not come to much surprise that on-line algorithms generally use more colors than when the entire graph can be seen.

LIST COLORING:

- Like weighted coloring, list coloring is a more modern approach to graph coloring and was introduced independently by **Vizing** in 1976.
- In list coloring we are concerned with assigning colors to vertices (or edges) with the added restriction that the colors must come from some predefined list.

Before we dive into the theory and some surprising results, let us consider the following example.

Example 6.20 The table on the right below shows two different lists for the vertices in the graph on the left. Find a proper coloring for each version of the lists or explain why none exists.



Vertex	List 1	List 2
a	$\{1,2\}$	$\{1\}$
b	$\{1,3,5\}$	$\{1,4,5\}$
c	$\{3,4\}$	$\{3,4\}$
d	$\{3,4\}$	$\{3,4\}$
e	$\{2,3\}$	$\{2,3\}$
f	$\{1,2,3\}$	$\{1,2,3\}$
g	$\{4,5\}$	$\{4,5\}$

Solution: A possible coloring from the first set of lists can be given as

$$a-1, b-4, c-3, d-4, e-3, f-2, g-5.$$

However, there is no proper coloring possible from the second set of lists. First note a must be given color 1 and so f can only be colored 2 or 3. But since those are the same colors available for e, we know one will be given color 2 and the other color 3. In either case, since d is adjacent to both f and e we know it cannot be given color 3, and so must be given color 4, forcing g to be color 5 and c to be color 3. But now b is adjacent to vertices of color 1, 3, and 5, and so cannot be given a color from its list.

The lists for the vertices need not be the same size nor contain the same items. The example above shows that finding a proper coloring can largely depend on the lists given for the vertices; the difference between the first and second set of lists above was minor (only those for a and b changed). Before we discuss some results on list coloring, we formally define what we mean by measuring a graph's ability to be list colored.

Definition 6.26 Let G be a graph where each vertex x is given a list $L(x)$ of colors. A proper ***list coloring*** assigns to x a color from its list $L(x)$ so that no two adjacent vertices are given the same color.

If for every collection of lists, each of size k, a proper list coloring exists then G is ***k-choosable***. The minimum value for k for which G is k-choosable is called the ***choosability*** of G and denoted $ch(G)$.

- First, we should note that choosability is not based on if there exists a singular collection of lists, each of a given size, that exhibit a proper coloring.
- Every possible collection of lists of a given size can produce a proper coloring. This should intuitively make sense as otherwise we could find a proper coloring if each vertex has a list of size one with the lists being disjoint.
- However, there is a direct relationship between the choosability of a graph and its chromatic number.

Proposition 6.27 For any simple graph G, $ch(G) \geq \chi(G)$.

Proof: Let G satisfy $\chi(G) = k$ and give each vertex of G the list $\{1, 2, \ldots, k\}$. Then there is a proper coloring for G from these lists, namely the one exhibited by the fact that $\chi(G) = k$. However, if we remove the same one element from each of these lists, then G cannot be colored since otherwise $\chi(G) < k$.

Proposition 6.28 For any simple graph G, $ch(G) \leq \Delta(G) + 1$.

Conclusion: Graph coloring provides avenues for studying limitations and optimal strategies for resource management. As such, many additional variations of graph coloring exist that cannot be included here, such as total coloring, path coloring, and acyclic coloring.