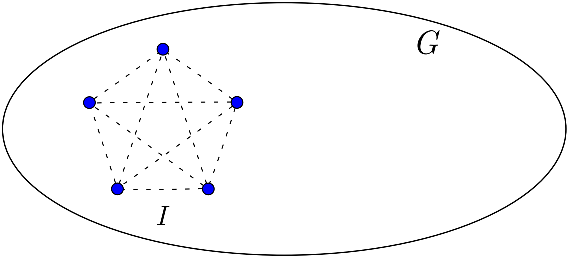
Graph colorings

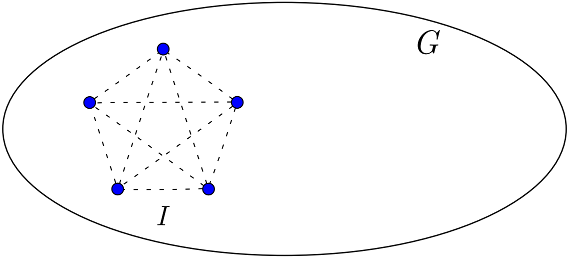
Graph theory
for MSc students in Computer Science

University of Szeged
Szeged, 2024.

Definition. In a multigraph G, the set $I \subseteq V(G)$ is an independent set, if no two vertices of I are adjacent in G (and no loop is indident to a vertex of I).



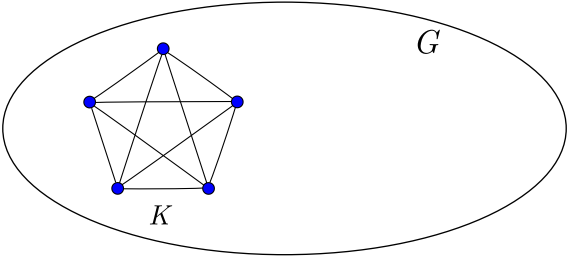
Definition. In a multigraph G, the set $I \subseteq V(G)$ is an independent set, if no two vertices of I are adjacent in G (and no loop is indident to a vertex of I).



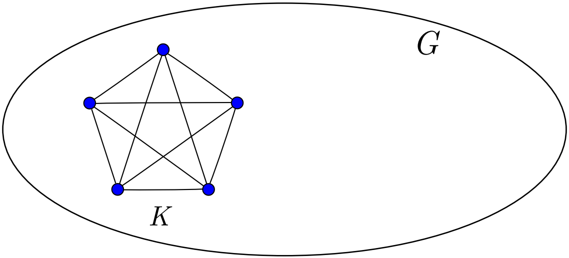
Definition. $\alpha(G)$ denotes the cardinality of an independent set of maximum size in G:

$$\alpha(G) = \max\{|I| : I \text{ is an independent set in } G\}.$$

Definition. In a multigraph G, the set $K \subseteq V(G)$ a clique, if every two distinct vertices in K are adjacent.



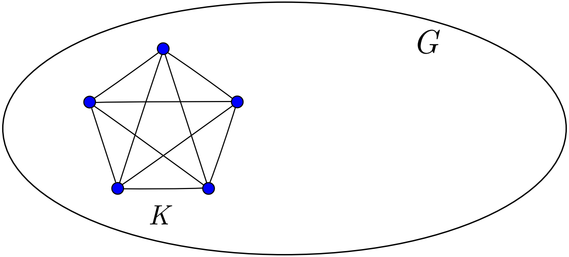
Definition. In a multigraph G, the set $K \subseteq V(G)$ a clique, if every two distinct vertices in K are adjacent.



Definition. $\omega(G)$ denotes the cardinality of a clique of maximum size in G:

$$\omega(G) = \max\{|K| : K \text{ is a clique in } G\}.$$

Definition. In a multigraph G, the set $K \subseteq V(G)$ a clique, if every two distinct vertices in K are adjacent.

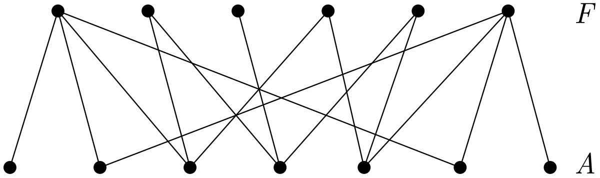


Definition. $\omega(G)$ denotes the cardinality of a clique of maximum size in G:

$$\omega(G) = \max\{|K| : K \text{ is a clique in } G\}.$$

Remark. There is no "efficient" (polynomial-time) algorithm known for computing either $\alpha(G)$ or $\omega(G)$ for an input graph G. (It is conjectured that such an algorithm does not exists.)

Bipartite graphs.

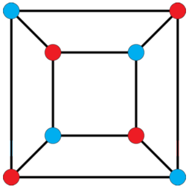


A multigraph is G bipartite, if its vertices can be partitioned into two disjoint classes A and B such that each edge of G has one end vertex in A and one end vertex in B.

Bipartite graphs.

A multigraph is G bipartite, if its vertices can be partitioned into two disjoint classes A and B such that each edge of G has one end vertex in A and one end vertex in B.

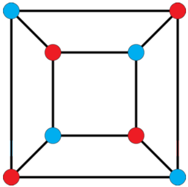
Example. The cube graph (in the figure) is bipartite. A good partitioning of the vertices is indicated by the vertex colors: One class is the set of red vertices, and the other class is the set of blue vertices. (We have to check that the end vertices of each edge have different colors.)



Bipartite graphs.

A multigraph is G bipartite, if its vertices can be partitioned into two disjoint classes A and B such that each edge of G has one end vertex in A and one end vertex in B.

Example. The cube graph (in the figure) is bipartite. A good partitioning of the vertices is indicated by the vertex colors: One class is the set of red vertices, and the other class is the set of blue vertices. (We have to check that the end vertices of each edge have different colors.)



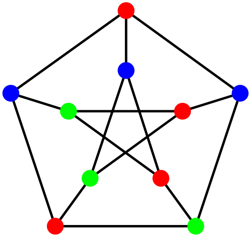
T. A multigraph is bipartite if and only if it does not contain odd cycles.

A vertex coloring of a graph G is a function $c\colon V(G) \to \mathcal{P}$, where $\mathcal{P}$ is a given finite set (the „palette”).

A vertex coloring of a graph G is a function $c\colon V(G) \to \mathcal{P}$, where $\mathcal{P}$ is a given finite set (the „palette”).

Now we generalize the coloring in the previous slide to more colors.

Definition. A proper coloring of a graph G is vertex coloring c of G, such that for all $xy \in E(G)$, $c(x) \neq c(y)$ holds.



A vertex coloring of a graph G is a function $c\colon V(G) \to \mathcal{P}$, where $\mathcal{P}$ is a given finite set (the „palette").

Now we generalize the coloring in the previous slide to more colors.

Definition. A proper coloring of a graph G is vertex coloring c of G, such that for all $xy \in E(G)$, $c(x) \neq c(y)$ holds.

Remark. We consider (simple) graphs only in case of vertex colorings. (In the presence of loops, no proper coloring exists. Multiple edges can be replaced to single edges, as parallel edges between x and y give the same $c(x) \neq c(y)$ condition.)

A vertex coloring of a graph G is a function $c\colon V(G) \to \mathcal{P}$, where $\mathcal{P}$ is a given finite set (the „palette”).

Now we generalize the coloring in the previous slide to more colors.

Definition. A proper coloring of a graph G is vertex coloring c of G, such that for all $xy \in E(G)$, $c(x) \neq c(y)$ holds.

Remark. We consider (simple) graphs only in case of vertex colorings. (In the presence of loops, no proper coloring exists. Multiple edges can be replaced to single edges, as parallel edges between x and y give the same $c(x) \neq c(y)$ condition.)

Remark. A simple graph always has a proper coloring, for example, we can assign pairwise different colors to the vertices of the graph. An important optimization problem is to minimize the number of colors used by the coloring.

Definition. The chromatic number $\chi(G)$ of a graph G is the smallest number of colors needed to color (the vertices of) G properly.

Definition. The chromatic number $\chi(G)$ of a graph G is the smallest number of colors needed to color (the vertices of) G properly.

So „$\chi(G) = k$" means TWO things: There exists a proper coloring of G using k colors, AND it is not possible to properly color G with less colors!

Definition. The chromatic number $\chi(G)$ of a graph G is the smallest number of colors needed to color (the vertices of) G properly.

So „$\chi(G) = k$" means TWO things: There exists a proper coloring of G using k colors, AND it is not possible to properly color G with less colors!

Convention. If G has a proper coloring with k colors, we say that G is k-colorable. In other words: G is k-colorable $\iff \chi(G) \leq k$.

Observation. Bipartite graphs are precisely the 2-colorable graphs.

Definition. The chromatic number $\chi(G)$ of a graph G is the smallest number of colors needed to color (the vertices of) G properly.

So „$\chi(G) = k$" means TWO things: There exists a proper coloring of G using k colors, AND it is not possible to properly color G with less colors!

Convention. If G has a proper coloring with k colors, we say that G is k-colorable. In other words: G is k-colorable $\iff \chi(G) \leq k$.

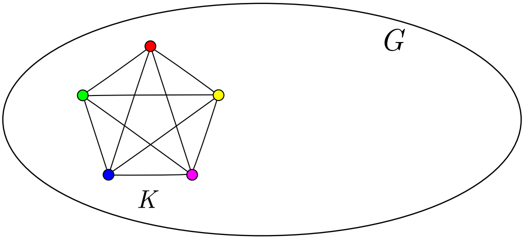
Observation. Bipartite graphs are precisely the 2-colorable graphs.

Remark. There is no polynomial-time algorithm known for determining the chromatic number of an input graph. It is even NP-complete to decide whether a graph G is 3-colorable or not. (This means and it is an open question if there exists a polynomial-time algorithm for the 3-coloring problem, and this question is equivalent to the $P \neq NP$ problem, a major unsolved problem of complexity theory.)

Proposition. $\chi(G) \geq \omega(G)$.

Proposition. $\chi(G) \geq \omega(G)$.

Proof. Consider a largest clique K in G, whose number of vertices is $\omega(G)$. It is clear that every proper coloring of G must assign (pairwise) different colors to the vertices of K. This means that an optimal coloring of G also requires at least $|K| = \omega(G)$ colors. $\square$



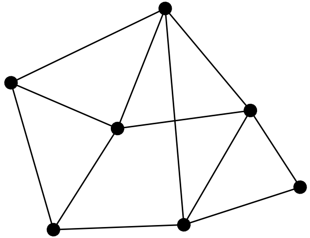
Proposition. $\chi(G) \geq \omega(G)$.

Proof. Consider a largest clique K in G, whose number of vertices is $\omega(G)$. It is clear that every proper coloring of G must assign (pairwise) different colors to the vertices of K. This means that an optimal coloring of G also requires at least $|K| = \omega(G)$ colors. □

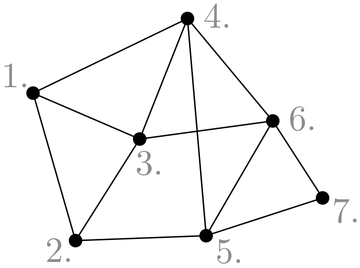
Remark. $\chi(G) > \omega(G)$ is possible, see the Petersen graph, for example. Moreover,

Thm. For an arbitrary (large) integer $k \geq 2$, there exists a graph G whose chromatic number is k, but G does not contain a triangle, i.e. $\omega(G) = 2$.

Greedy coloring algorithm. The input is a simple graph G which we want to color properly. (The assigned colors will be positive integers.)

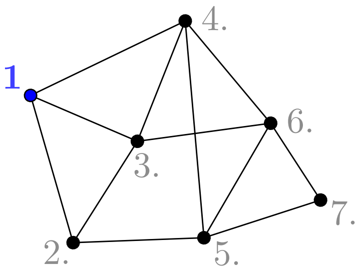


Greedy coloring algorithm. The input is a simple graph G which we want to color properly. (The assigned colors will be positive integers.)



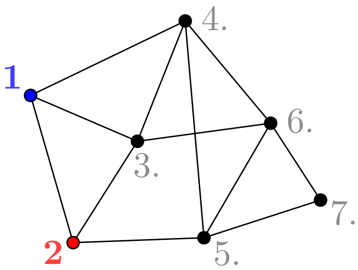
- Fix an order π of vertices of G. (See the grey numbers in the figure.)

Greedy coloring algorithm. The input is a simple graph G which we want to color properly. (The assigned colors will be positive integers.)



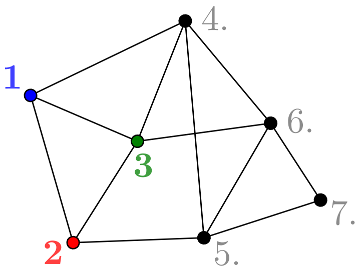
- Fix an order π of vertices of G. (See the grey numbers in the figure.)
- According to the order π, color the vertices of G one by one in the following way: To the current vertex v, assign the smallest positive integer (as color $c(v)$) which does not appear on the already colored neighbors of v.

Greedy coloring algorithm. The input is a simple graph G which we want to color properly. (The assigned colors will be positive integers.)



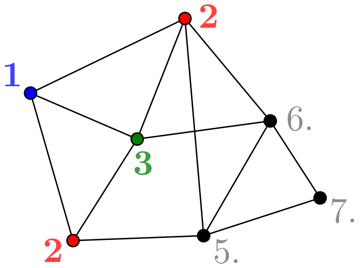
- Fix an order π of vertices of G. (See the grey numbers in the figure.)
- According to the order π, color the vertices of G one by one in the following way: To the current vertex v, assign the smallest positive integer (as color $c(v)$) which does not appear on the already colored neighbors of v.

Greedy coloring algorithm. The input is a simple graph G which we want to color properly. (The assigned colors will be positive integers.)



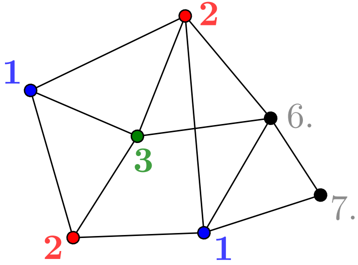
- Fix an order π of vertices of G. (See the grey numbers in the figure.)
- According to the order π, color the vertices of G one by one in the following way: To the current vertex v, assign the smallest positive integer (as color $c(v)$) which does not appear on the already colored neighbors of v.

Greedy coloring algorithm. The input is a simple graph G which we want to color properly. (The assigned colors will be positive integers.)



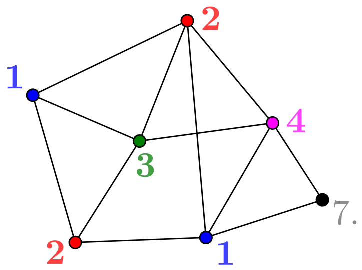
- Fix an order π of vertices of G. (See the grey numbers in the figure.)
- According to the order π, color the vertices of G one by one in the following way: To the current vertex v, assign the smallest positive integer (as color $c(v)$) which does not appear on the already colored neighbors of v.

Greedy coloring algorithm. The input is a simple graph G which we want to color properly. (The assigned colors will be positive integers.)



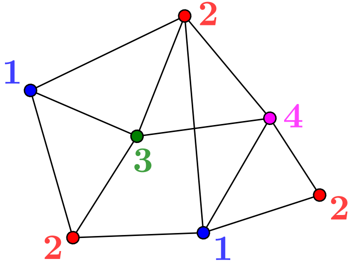
- Fix an order π of vertices of G. (See the grey numbers in the figure.)
- According to the order π, color the vertices of G one by one in the following way: To the current vertex v, assign the smallest positive integer (as color $c(v)$) which does not appear on the already colored neighbors of v.

Greedy coloring algorithm. The input is a simple graph G which we want to color properly. (The assigned colors will be positive integers.)



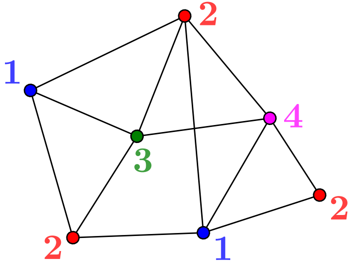
- Fix an order π of vertices of G. (See the grey numbers in the figure.)
- According to the order π, color the vertices of G one by one in the following way: To the current vertex v, assign the smallest positive integer (as color $c(v)$) which does not appear on the already colored neighbors of v.

Greedy coloring algorithm. The input is a simple graph G which we want to color properly. (The assigned colors will be positive integers.)



- Fix an order π of vertices of G. (See the grey numbers in the figure.)
- According to the order π, color the vertices of G one by one in the following way: To the current vertex v, assign the smallest positive integer (as color $c(v)$) which does not appear on the already colored neighbors of v.

Greedy coloring algorithm. The input is a simple graph G which we want to color properly. (The assigned colors will be positive integers.)

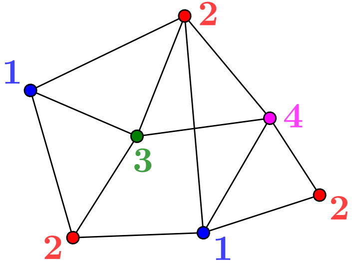


- Fix an order π of vertices of G. (See the grey numbers in the figure.)
- According to the order π, color the vertices of G one by one in the following way: To the current vertex v, assign the smallest positive integer (as color $c(v)$) which does not appear on the already colored neighbors of v.

Proposition. The algorithm constructs a proper coloring of G.

Proof. Consider an arbitrary edge $e \in E(G)$. Let u_e be the end vertex of e which has been colored earlier, and let v_e be the other end vertex of e. When v_e was colored, its assigned color was different from the color of u_e (since then u_e was an already colored neighbor of v_e, c.f. the second • above). $\square$

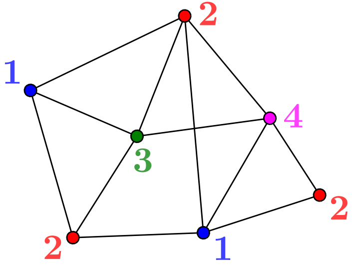
Greedy coloring algorithm. The input is a simple graph G which we want to color properly. (The assigned colors will be positive integers.)



- Fix an order π of vertices of G. (See the grey numbers in the figure.)
- According to the order π, color the vertices of G one by one in the following way: To the current vertex v, assign the smallest positive integer (as color $c(v)$) which does not appear on the already colored neighbors of v.

Remark. The coloring provided by the greedy coloring algorithm – and so the number of used colors – depends on vertex order π!

Greedy coloring algorithm. The input is a simple graph G which we want to color properly. (The assigned colors will be positive integers.)

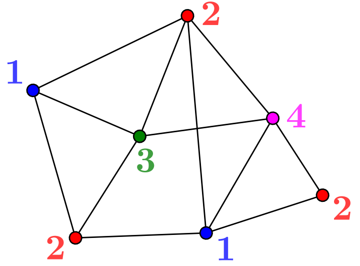


- Fix an order π of vertices of G. (See the grey numbers in the figure.)
- According to the order π, color the vertices of G one by one in the following way: To the current vertex v, assign the smallest positive integer (as color $c(v)$) which does not appear on the already colored neighbors of v.

Remark. The coloring provided by the greedy coloring algorithm – and so the number of used colors – depends on vertex order π!

Remark. So it can happen the obtained coloring is not optimal, e.g. the graph in the figure can also be properly colored using only 3 colors. The obtained number of colors is just an upper bound for $\chi(G)$.

Greedy coloring algorithm. The input is a simple graph G which we want to color properly. (The assigned colors will be positive integers.)



- Fix an order π of vertices of G. (See the grey numbers in the figure.)
- According to the order π, color the vertices of G one by one in the following way: To the current vertex v, assign the smallest positive integer (as color $c(v)$) which does not appear on the already colored neighbors of v.

However. For every graph G, there exists a vertex order π, such that the greedy vertex coloring algorithm with respect to π yields an optimal coloring of G with $\chi(G)$ colors. (We omit the proof.)

But this does not help us to determine $\chi(G)$ efficiently: If G has n vertices, there are $n!$ possible orders π. :(

The largest vertex degree in the multigraph G is denoted by $\Delta(G)$.

The largest vertex degree in the multigraph G is denoted by $\Delta(G)$.

Theorem. $\chi(G) \leq \Delta(G) + 1$.

The largest vertex degree in the multigraph G is denoted by $\Delta(G)$.

Theorem. $\chi(G) \leq \Delta(G) + 1$.

Proof. Color the vertices of G properly using the greedy coloring algorithm (with an arbitrary order π). We show that every vertex got its color from the set $\{1, \ldots, \Delta(G) + 1\}$, which means that at most $\Delta(G) + 1$ colors are used.

- An arbitrary vertex v has $d(v)$ neighbors, some of them are already colored when coloring v. So when coloring v, the number of already colored neighbors of v is at most $d(v)$, so at most $\Delta(G)$.
- This implies that at most $\Delta(G)$ different colors appear on the neighbors of v (even if those colors are pairwise different).
- So at least one color from $\{1, 2, \ldots, \Delta(G)+1\}$ is not appears on the neighbors of v. So when the algorithm assigns the smallest "free" color to v, it must come from this set, as stated. $\square$

The largest vertex degree in the multigraph G is denoted by $\Delta(G)$.

Theorem. $\chi(G) \leq \Delta(G) + 1$.

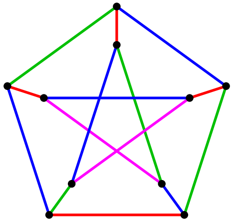
In case of complete graphs and odd cycle graphs, equality holds in the above theorem. (Why?) For any other connected graphs, the upper bound can be improved:

Brooks' theorem. If G is connected graph that is neither a complete graph nor an odd cycle, then

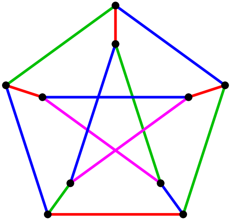
$$\chi(G) \leq \Delta(G).$$

We omit the proof.

Definition. Let G be a multigraph, and $\mathcal{P}$ is a finite set of colors ($\mathcal{P}$ is the "palette"). The function $c\colon E \to \mathcal{P}$ is a proper edge coloring of G, if for every $xy, xz \in E(G)$ we have $c(xy) \neq c(xz)$ that is, if two edges have a common endpoint, then their colors must be different.



Definition. Let G be a multigraph, and $\mathcal{P}$ is a finite set of colors ($\mathcal{P}$ is the "palette"). The function $c\colon E \to \mathcal{P}$ is a proper edge coloring of G, if for every $xy, xz \in E(G)$ we have $c(xy) \neq c(xz)$ that is, if two edges have a common endpoint, then their colors must be different.



Remark. „no two edges of the same color have common end vertex" = „the edges incident to an arbitrary fixed vertex got pairwise different colors" = „the edges of color c form a matching in G, for each color $c \in \mathcal{P}$".

Definition. Let G be a multigraph, and $\mathcal{P}$ is a finite set of colors ($\mathcal{P}$ is the "palette"). The function $c\colon E \to \mathcal{P}$ is a proper edge coloring of G, if for every $xy, xz \in E(G)$ we have $c(xy) \neq c(xz)$ that is, if two edges have a common endpoint, then their colors must be different.

Definition. The smallest number of colors needed in a proper edge coloring of G is called the edge chromatic number of G, and it is denoted by $\chi'(G)$.

$$\chi'(G) := \min\{k : \text{There exists a proper edge coloring } E(G) \to \{1, \ldots, k\}\}.$$

Definition. Let G be a multigraph, and $\mathcal{P}$ is a finite set of colors ($\mathcal{P}$ is the "palette"). The function $c\colon E \to \mathcal{P}$ is a proper edge coloring of G, if for every $xy, xz \in E(G)$ we have $c(xy) \neq c(xz)$ that is, if two edges have a common endpoint, then their colors must be different.

Definition. The smallest number of colors needed in a proper edge coloring of G is called the edge chromatic number of G, and it is denoted by $\chi'(G)$.

$$\chi'(G) := \min\{k : \text{There exists a proper edge coloring } E(G) \to \{1, \ldots, k\}\}.$$

Remark.

$\chi'(G) = k$: „G can be properly edge-colored using k colors, but less colors are not enough."

$\chi'(G) \leq k$: „G can be properly edge-colored using k colors."

$\chi'(G) \geq k$: „Every proper edge-coloring of G contains at least k colors."

Definition. Let G be a multigraph, and $\mathcal{P}$ is a finite set of colors ($\mathcal{P}$ is the "palette"). The function $c\colon E \to \mathcal{P}$ is a proper edge coloring of G, if for every $xy, xz \in E(G)$ we have $c(xy) \neq c(xz)$ that is, if two edges have a common endpoint, then their colors must be different.

Definition. The smallest number of colors needed in a proper edge coloring of G is called the edge chromatic number of G, and it is denoted by $\chi'(G)$.

$$\chi'(G) := \min\{k : \text{There exists a proper edge coloring } E(G) \to \{1, \dots, k\}\}.$$

Remark.

$\chi'(G) = k$: „G can be properly edge-colored using k colors, but less colors are not enough."

$\chi'(G) \leq k$: „G can be properly edge-colored using k colors."

$\chi'(G) \geq k$: „Every proper edge-coloring of G contains at least k colors."

Note. In case of the presense of loops no proper edge-coloring exists, so we always assume that G is a loopless multigraph.

Remark. $\Delta(G)$ denotes the largest vertex degree in G.

Observation. For any multigraph G,

$$\chi'(G) \geq \Delta(G).$$

Remark. $\Delta(G)$ denotes the largest vertex degree in G.

Observation. For any multigraph G,

$$\chi'(G) \geq \Delta(G).$$

Proof. This is obvious, because if we consider a vertex v whose degree is $\Delta(G)$, then every proper edge coloring of G must assign $\Delta(G)$ (pairwise) different colors to the edges incident to v. $\square$

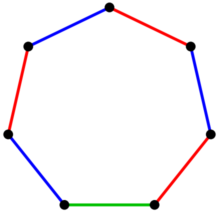
Remark. $\Delta(G)$ denotes the largest vertex degree in G.

Observation. For any multigraph G,

$$\chi'(G) \geq \Delta(G).$$

Proof. This is obvious, because if we consider a vertex v whose degree is $\Delta(G)$, then every proper edge coloring of G must assign $\Delta(G)$ (pairwise) different colors to the edges incident to v. $\square$

Remark. $\chi'(G) > \Delta(G)$ may occur, for example, by considering the odd cycles (where $\Delta(C_{2k+1}) = 2$, but $\chi'(C_{2k+1}) = 3$.)



Theorem (Dénes Kőnig). If G is a bipartite multigraph, then

$$\chi'(G) = \Delta(G).$$

Theorem (Dénes Kőnig). If G is a bipartite multigraph, then

$$\chi'(G) = \Delta(G).$$

Theorem (Vizing). If G is a simple graph, then

$$\Delta(G) \leq \chi'(G) \leq \Delta(G) + 1.$$

Theorem (Dénes Kőnig). If G is a bipartite multigraph, then

$$\chi'(G) = \Delta(G).$$

Theorem (Vizing). If G is a simple graph, then

$$\Delta(G) \leq \chi'(G) \leq \Delta(G) + 1.$$

Theorem (Shannon). For any (loopless) multigraph G, we have

$$\Delta(G) \leq \chi'(G) \leq \frac{3}{2}\Delta(G).$$