Programa de Pós-graduação em INFORMÁTICA

PUC Minas

IM SCIENCE

Teoria dos Grafos e Computabilidade

— Planar graphs —

Silvio Jamil F. Guimarães

Graduate Program in Informatics – PPGINF
Image and Multimedia Data Science Laboratory – IMScience
Pontifical Catholic University of Minas Gerais – PUC Minas

Aug 2022

Teoria dos Grafos e Computabilidade

— Some concepts —

Silvio Jamil F. Guimarães

Graduate Program in Informatics – PPGINF
Image and Multimedia Data Science Laboratory – IMScience
Pontifical Catholic University of Minas Gerais – PUC Minas

Aug 2022

Bipartite graphs

If it is possible to partition the vertex set, V, into two disjoint sets, V_1 and V_2, such that there are no edges between any two vertices in the same set, then the graph is Bipartite.

Bipartite graphs

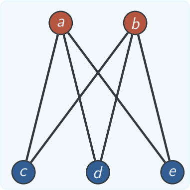
If it is possible to partition the vertex set, V, into two disjoint sets, V_1 and V_2, such that there are no edges between any two vertices in the same set, then the graph is Bipartite.

When the bipartite graph is such that every vertex in V_1 is connected to every vertex in V_2 (and vice versa) the graph is called Complete Bipartite Graph. If $|V_1| = m$, and $|V_2| = n$, we denote it $K_{m,n}$.

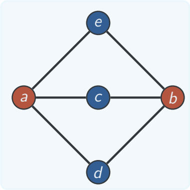
Bipartite graphs

If it is possible to partition the vertex set, V, into two disjoint sets, V_1 and V_2, such that there are no edges between any two vertices in the same set, then the graph is Bipartite.

When the bipartite graph is such that every vertex in V_1 is connected to every vertex in V_2 (and vice versa) the graph is called Complete Bipartite Graph. If $|V_1| = m$, and $|V_2| = n$, we denote it $K_{m,n}$.



$K_{2,3}$

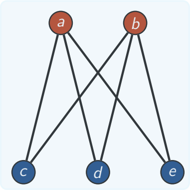


$K_{2,3}$

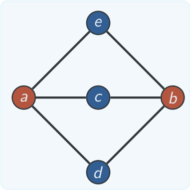
Bipartite graphs

If it is possible to partition the vertex set, V, into two disjoint sets, V_1 and V_2, such that there are no edges between any two vertices in the same set, then the graph is Bipartite.

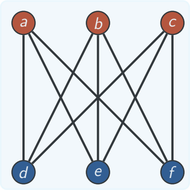
When the bipartite graph is such that every vertex in V_1 is connected to every vertex in V_2 (and vice versa) the graph is called Complete Bipartite Graph. If $|V_1| = m$, and $|V_2| = n$, we denote it $K_{m,n}$.



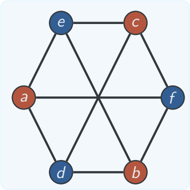
$K_{2,3}$



$K_{2,3}$



$K_{3,3}$

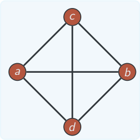


$K_{3,3}$

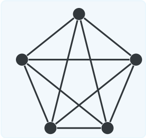
Some named graphs

K_n

Complete graph of n vertices



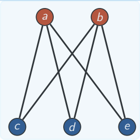
K_4



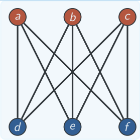
K_5

$K_{m,n}$

Complete bipartite graph of m and n vertices



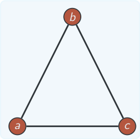
$K_{2,3}$



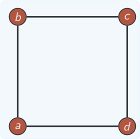
$K_{3,3}$

C_n

The cycle with n vertices



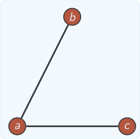
C_3



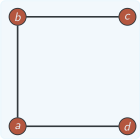
C_4

P_n

The path with n vertices



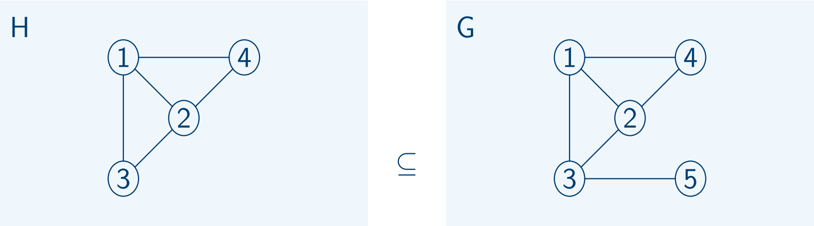
C_3



P_4

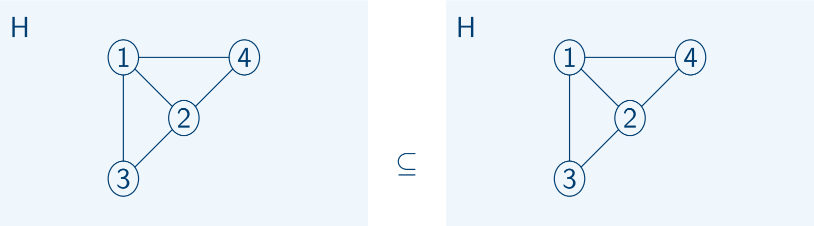
Sub-grafo

- Um grafo H é dito ser um subgrafo de um grafo G ($H \subseteq G$) se **todos** os **vértices** e todas as **arestas** de g estão em G



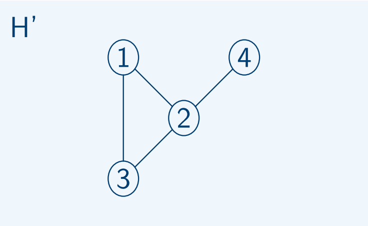
Sub-grafo

- Um grafo H é dito ser um subgrafo de um grafo G ($H \subseteq G$) se **todos** os **vértices** e todas as **arestas** de g estão em G
  - todo grafo é subgrafo de si próprio

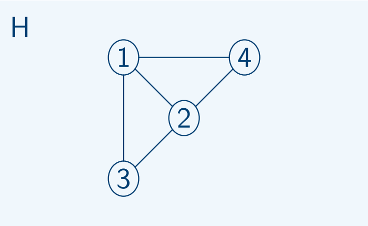


Sub-grafo

- Um grafo H é dito ser um subgrafo de um grafo G ($H \subseteq G$) se **todos** os **vértices** e todas as **arestas** de g estão em G
  - todo grafo é subgrafo de si próprio
  - o subgrafo de um subgrafo de G é subgrafo de G

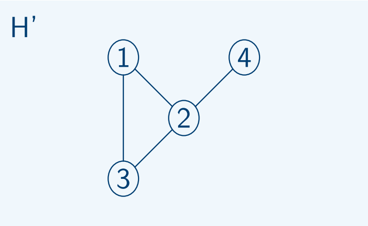


$\subseteq$

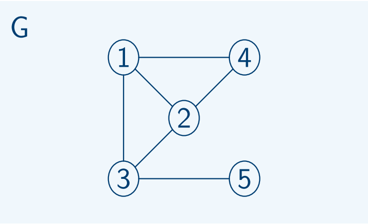


Sub-grafo

- Um grafo H é dito ser um subgrafo de um grafo G ($H \subseteq G$) se **todos** os **vértices** e todas as **arestas** de g estão em G
  - todo grafo é subgrafo de si próprio
  - o subgrafo de um subgrafo de G é subgrafo de G

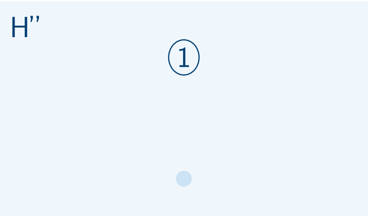


$\subseteq$

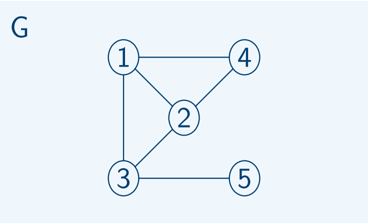


Sub-grafo

- Um grafo H é dito ser um subgrafo de um grafo G ($H \subseteq G$) se **todos** os **vértices** e todas as **arestas** de g estão em G
  - todo grafo é subgrafo de si próprio
  - o subgrafo de um subgrafo de G é subgrafo de G
  - um vértice simples de G é um subgrafo de G

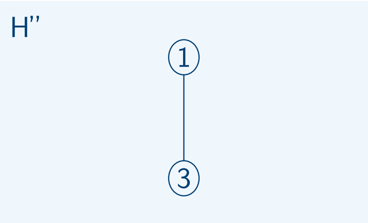


$\subseteq$

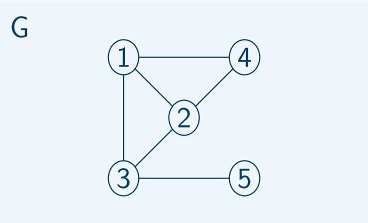


Sub-grafo

- Um grafo H é dito ser um subgrafo de um grafo G ($H \subseteq G$) se **todos** os **vértices** e todas as **arestas** de g estão em G
  - todo grafo é subgrafo de si próprio
  - o subgrafo de um subgrafo de G é subgrafo de G
  - um vértice simples de G é um subgrafo de G
  - uma aresta simples de G (juntamente com suas extremidades) é subgrafo de G



$\subseteq$



Questions?

Planar graphs
– Some concepts –

Programa de Pós-graduação em INFORMÁTICA

PUC Minas

IM SCIENCE

Teoria dos Grafos e Computabilidade

— Planar graphs —

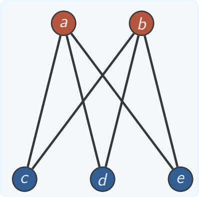
Silvio Jamil F. Guimarães

Graduate Program in Informatics – PPGINF
Image and Multimedia Data Science Laboratory – IMScience
Pontifical Catholic University of Minas Gerais – PUC Minas

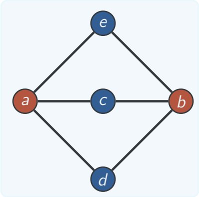
Aug 2022

Planar graphs

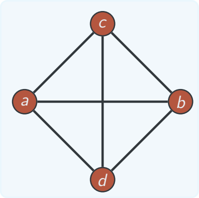
If you can sketch a graph so that none of its edges cross, then it is a planar graph.



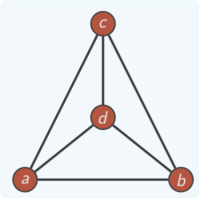
$K_{2,3}$



$K_{2,3}$



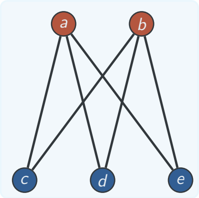
K_4



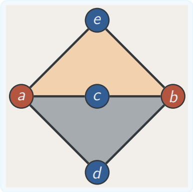
K_4

Planar graphs

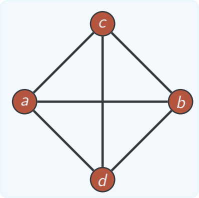
When a planar graph is drawn without edges crossing, the edges and vertices of the graph divide the plane into regions. Each region is called a face.



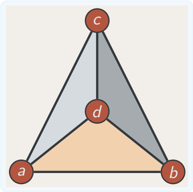
$K_{2,3}$



$K_{2,3}$ – 3 faces



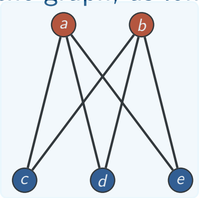
K_4



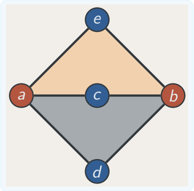
K_4 – 4 faces

Planar graphs

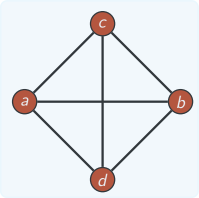
When a planar graph is drawn without edges crossing, the edges and vertices of the graph divide the plane into regions. Each region is called a face. The number of faces does not change no matter how you draw the graph, as long as no edges cross.



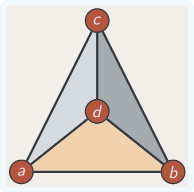
$K_{2,3}$



$K_{2,3}$ – 3 faces



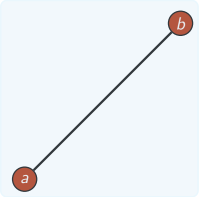
K_4



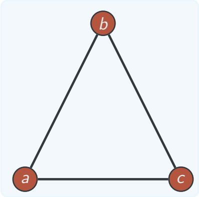
K_4 – 4 faces

Planar graphs

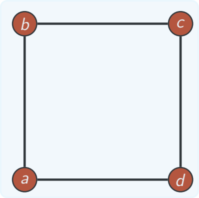
Count the number of edges, faces and vertices in the cycle graphs C_3, C_4 and C_5. What about C_k?



C_2

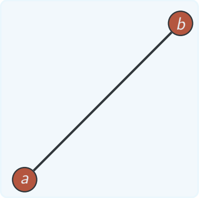


C_3

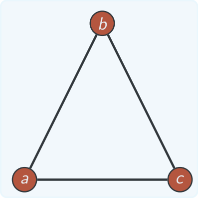


C_4

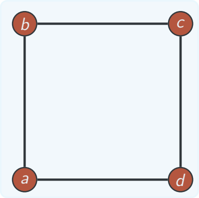
Count the number of edges, faces and vertices in the cycle graphs C_3, C_4 and C_5. What about C_k? And what about P_k?



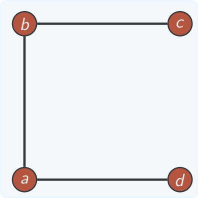
C_2



C_3



C_4



P_4

Planar graphs and Euler's formula

Let a list of some planar graphs, and count their vertices, edges, and faces, for example, K_3, K_4 and C_5

Planar graphs and Euler's formula

Let a list of some planar graphs, and count their vertices, edges, and faces, for example, K_3, K_4 and C_5 Is there a pattern?

Planar graphs and Euler's formula

Let a list of some planar graphs, and count their vertices, edges, and faces, for example, K_3, K_4 and C_5 Is there a pattern?

For any (connected) planar graph with v vertices, e edges and f faces, we have

$$v - e + f = 2$$

Planar graphs and Euler's formula

Let a list of some planar graphs, and count their vertices, edges, and faces, for example, K_3, K_4 and C_5 Is there a pattern?

For any (connected) planar graph with v vertices, e edges and f faces, we have

$$v - e + f = 2$$

Outline of the proof:

Planar graphs and Euler's formula

Let a list of some planar graphs, and count their vertices, edges, and faces, for example, K_3, K_4 and C_5 Is there a pattern?

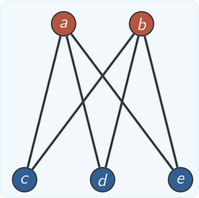
For any (connected) planar graph with v vertices, e edges and f faces, we have

$$v - e + f = 2$$

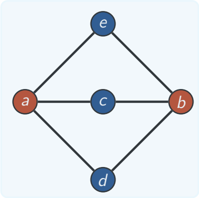
Outline of the proof:

Consider the graph with a single vertex and no edges. So v=1, e=0 and f=1.
We can construct any other planar connected graph from this as follows:
(i) – Let a K_3 be a complete graph with 3 vertices. Add one vertex and one edge. This will increase the number of vertices and edges by 1, and the number of faces will stay the same. So, v - e + f is the same.
(ii) – Let the graph of (i). Add one edge but no new vertex. So, the number of vertices is unchanged, but the number of edges and faces will increase by 1. So, v - e + f is the same.
So, by induction, v - e + f = 2

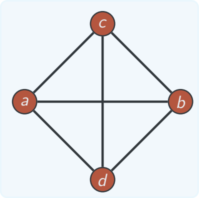
According to Fáry theorem (1947), every (simple) planar graph admits a straight line planar embedding (no edge crossings).



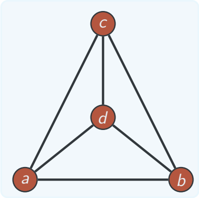
$K_{2,3}$



$K_{2,3}$



K_4



K_4

Questions?

Planar graphs

– Planar graphs –

Teoria dos Grafos e Computabilidade

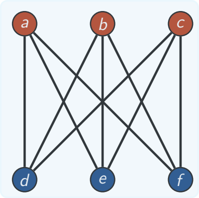
— Non-planar graphs —

Silvio Jamil F. Guimarães

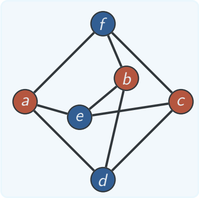
Graduate Program in Informatics – PPGINF
Image and Multimedia Data Science Laboratory – IMScience
Pontifical Catholic University of Minas Gerais – PUC Minas

Aug 2022

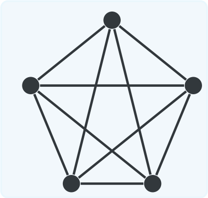
Most graphs do not have a planar representation. For example, the following two graphs cannot be drawn so no edges cross: K_5 and $K_{3,3}$.



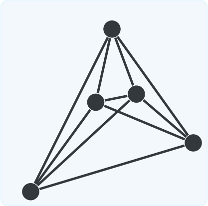
$K_{3,3}$



$K_{3,3}$



K_5

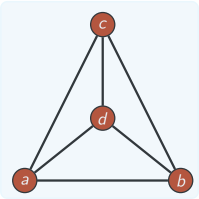


K_5

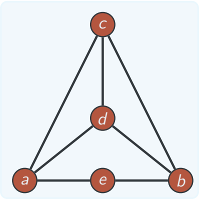
Homeomorphic graphs

Recall that a graph G' is a subgraph of G if it can be obtained by deleting some vertices and/or edges of G.

- A subdivision of an edge is obtained by adding a new vertex of degree 2 to the middle of the edge.
- A subdivision of a graph is obtained by subdividing one or more of its edges.



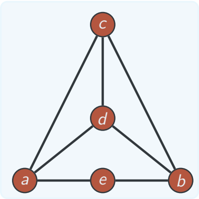
K_4

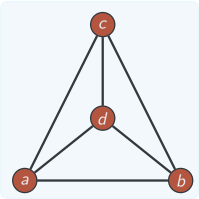


Homeomorphic graphs

Recall that a graph G' is a subgraph of G if it can be obtained by deleting some vertices and/or edges of G.

- Smoothing of the pair of edges $\{a, b\}$ and $\{b, c\}$, in which the degree of vertex b is equal to 2, means to remove these two edges, and add $\{a, c\}$.



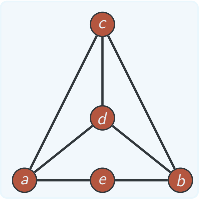


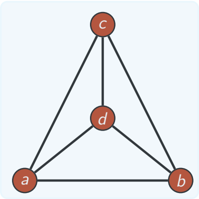
K_4

Homeomorphic graphs

Recall that a graph G' is a subgraph of G if it can be obtained by deleting some vertices and/or edges of G.

- The graphs G_1 and G_2 are homeomorphic if there is some subdivision of G_1 that is isomorphic to some subdivision of G_2.

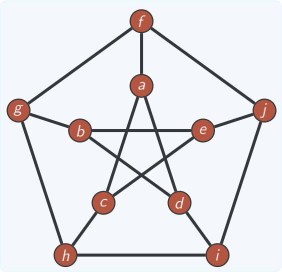




K_4

Kuratowski's theorem

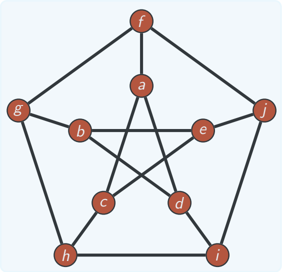
The Kuratowski's theorem says that a graph is planar if and only if it does not contain a subgraph that is homeomorphic to K_5 or $K_{3,3}$.



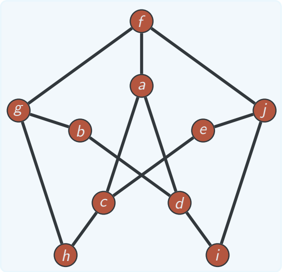
Petersen Graph

Kuratowski's theorem

The Kuratowski's theorem says that a graph is planar if and only if it does not contain a subgraph that is homeomorphic to K_5 or $K_{3,3}$.



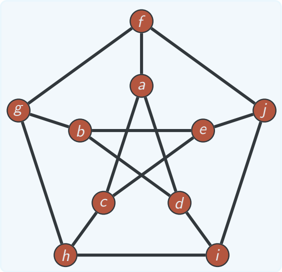
Petersen Graph



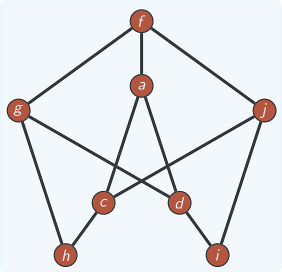
Subgraph of Petersen Graph

Kuratowski's theorem

The Kuratowski's theorem says that a graph is planar if and only if it does not contain a subgraph that is homeomorphic to K_5 or $K_{3,3}$.



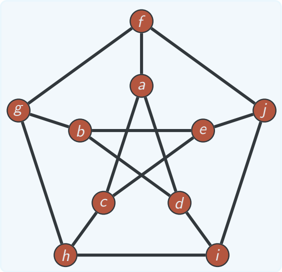
Petersen Graph



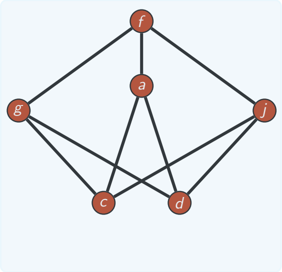
Petersen Graph – smoothing out

Kuratowski's theorem

The Kuratowski's theorem says that a graph is planar if and only if it does not contain a subgraph that is homeomorphic to K_5 or $K_{3,3}$.



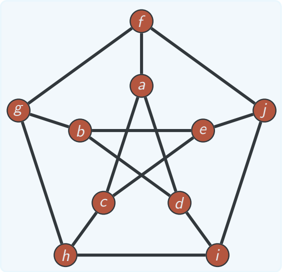
Petersen Graph



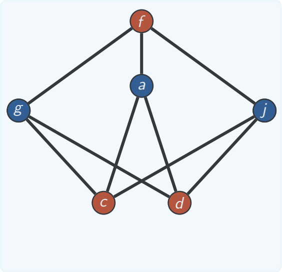
Petersen Graph – smoothing out

Kuratowski's theorem

The Kuratowski's theorem says that a graph is planar if and only if it does not contain a subgraph that is homeomorphic to K_5 or $K_{3,3}$.



Petersen Graph

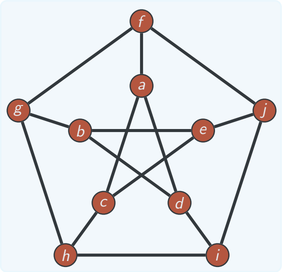


$K_{3,3}$

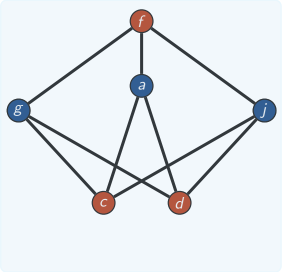
Kuratowski's theorem

The Kuratowski's theorem says that a graph is planar if and only if it does not contain a subgraph that is homeomorphic to K_5 or $K_{3,3}$.

- What this really means is that every non-planar graph has some smoothing that contains a copy of K_5 or $K_{3,3}$ somewhere inside it.



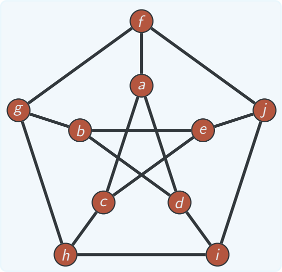
Petersen Graph



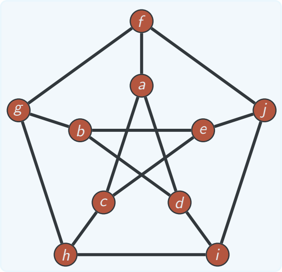
$K_{3,3}$

Wagner's theorem

The Wagner's therorem says that a graph has planar embedding, if, and only if, it contains no minor isomorphic to K_5 or $K_{3,3}$.



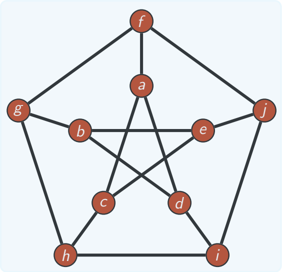
Petersen Graph



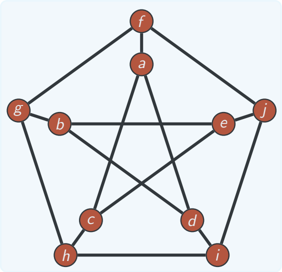
Petersen Graph

Wagner's theorem

The Wagner's therorem says that a graph has planar embedding, if, and only if, it contains no minor isomorphic to K_5 or $K_{3,3}$.



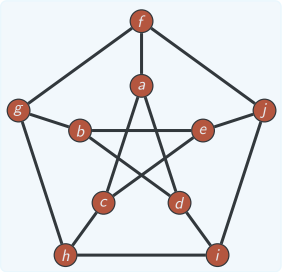
Petersen Graph



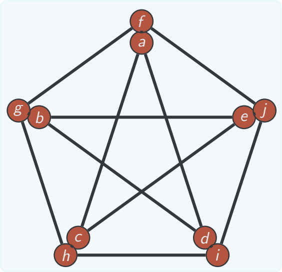
Petersen Graph

Wagner's theorem

The Wagner's therorem says that a graph has planar embedding, if, and only if, it contains no minor isomorphic to K_5 or $K_{3,3}$.



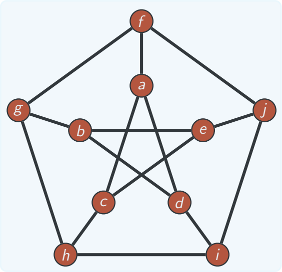
Petersen Graph



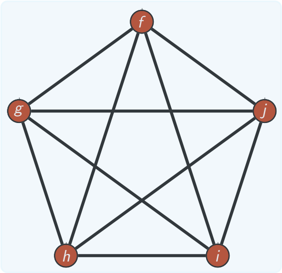
Petersen Graph

Wagner's theorem

The Wagner's therorem says that a graph has planar embedding, if, and only if, it contains no minor isomorphic to K_5 or $K_{3,3}$. A contraction of G is a graph obtained from G by repeated edge contractions. A minor of G is any subgraph of a contraction of G.



Petersen Graph

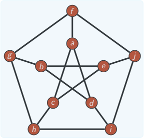


Petersen Graph – K_5

Wagner's theorem

Let $G = (V, E)$ be a graph and let $\{x, y\} \in E$. The graph G/xy, called the edge xy-contraction of G, consists of

- the vertex set $V' = V \backslash \{y\}$ and the edge set E' consisting of pairs $\{w, v\} \in E$ such that $y \notin \{w, v\}$

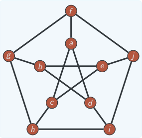


Petersen Graph

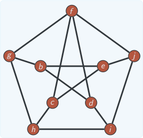
Wagner's theorem

Let $G = (V, E)$ be a graph and let $\{x, y\} \in E$. The graph G/xy, called the edge xy-contraction of G, consists of

- the vertex set $V' = V \backslash \{y\}$ and the edge set E' consisting of pairs $\{w, v\} \in E$ such that $y \notin \{w, v\}$
- plus all pairs $\{w, x\}, w \neq x$, with $\{w, y\} \in E$. No loops or multiple edges are allowed.



Petersen Graph

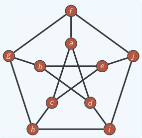


Petersen Graph – contraction

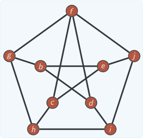
Wagner's theorem

Let $G = (V, E)$ be a graph and let $\{x, y\} \in E$. The graph G/xy, called the **edge xy-contraction of G**, consists of

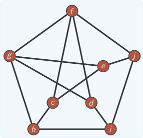
- the vertex set $V' = V \backslash \{y\}$ and the edge set E' consisting of pairs $\{w, v\} \in E$ such that $y \notin \{w, v\}$
- plus all pairs $\{w, x\}, w \neq x$, with $\{w, y\} \in E$. No loops or multiple edges are allowed.



Petersen Graph



Petersen Graph – contraction

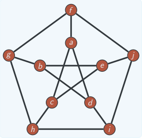


Petersen Graph – contraction

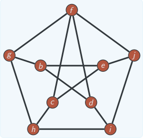
Wagner's theorem

Let $G = (V, E)$ be a graph and let $\{x, y\} \in E$. The graph G/xy, called the **edge xy-contraction of G**, consists of

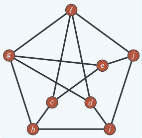
- the vertex set $V' = V \backslash \{y\}$ and the edge set E' consisting of pairs $\{w, v\} \in E$ such that $y \notin \{w, v\}$
- plus all pairs $\{w, x\}, w \neq x$, with $\{w, y\} \in E$. No loops or multiple edges are allowed.



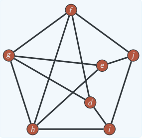
Petersen Graph



Petersen Graph – contraction



Petersen Graph – contraction

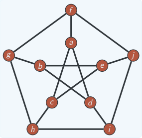


Petersen Graph – contraction

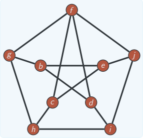
Wagner's theorem

Let $G = (V, E)$ be a graph and let $\{x, y\} \in E$. The graph G/xy, called the **edge xy-contraction of G**, consists of

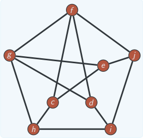
- the vertex set $V' = V \backslash \{y\}$ and the edge set E' consisting of pairs $\{w, v\} \in E$ such that $y \notin \{w, v\}$
- plus all pairs $\{w, x\}, w \neq x$, with $\{w, y\} \in E$. No loops or multiple edges are allowed.



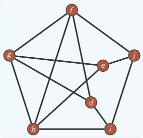
Petersen Graph



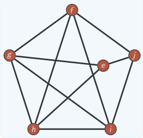
Petersen Graph – contraction



Petersen Graph – contraction



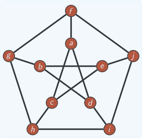
Petersen Graph – contraction



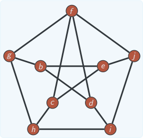
Petersen Graph – contraction

Let $G = (V, E)$ be a graph and let $\{x, y\} \in E$. The graph G/xy, called the **edge xy-contraction of G**, consists of

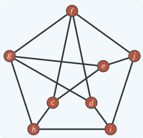
- the vertex set $V' = V \backslash \{y\}$ and the edge set E' consisting of pairs $\{w, v\} \in E$ such that $y \notin \{w, v\}$
- plus all pairs $\{w, x\}, w \neq x$, with $\{w, y\} \in E$. No loops or multiple edges are allowed.



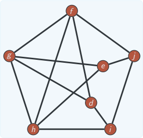
Petersen Graph



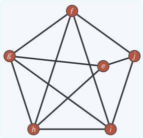
Petersen Graph – contraction



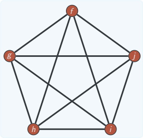
Petersen Graph – contraction



Petersen Graph – contraction



Petersen Graph – contraction



Petersen Graph – K_5

Questions?

Planar graphs

– Non-planar graphs –

Teoria dos Grafos e Computabilidade

— Geometric duality —

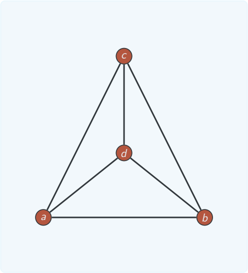
Silvio Jamil F. Guimarães

Graduate Program in Informatics – PPGINF
Image and Multimedia Data Science Laboratory – IMScience
Pontifical Catholic University of Minas Gerais – PUC Minas

Aug 2022

Geometric duality

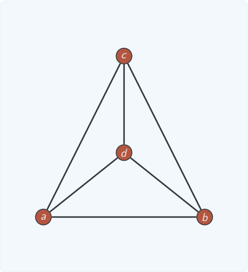
Let G = (V,E). A geometric dual $G^* = (V^*, E^*)$ of a planar representation of G – no crossing edges – is computed as follows:

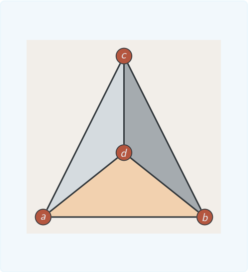


Geometric duality

Let G = (V,E). A geometric dual $G^* = (V^*, E^*)$ of a planar representation of G – no crossing edges – is computed as follows:

- For each face of G, pick one point $v\hat{}*$ inside the face. These are the the set of vertices V^* of G^* .

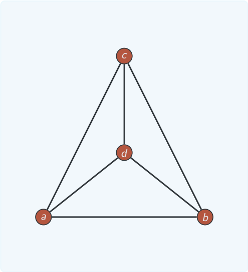


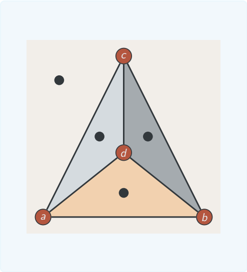


Geometric duality

Let G = (V,E). A geometric dual $G^* = (V^*, E^*)$ of a planar representation of G – no crossing edges – is computed as follows:

- For each face of G, pick one point $v\hat{}*$ inside the face. These are the the set of vertices V^* of G^* .

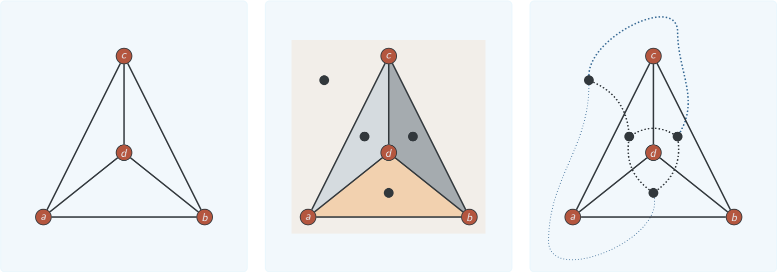




Geometric duality

Let G = (V,E). A geometric dual $G^* = (V^*, E^*)$ of a planar representation of G – no crossing edges – is computed as follows:

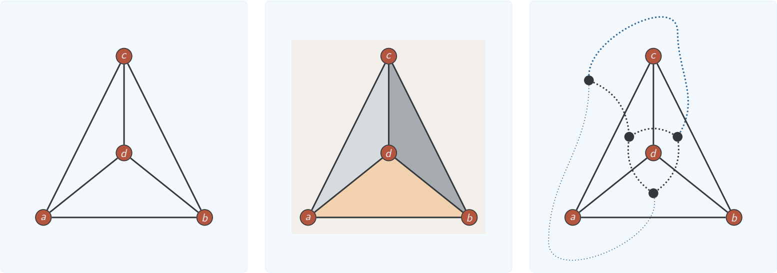
- For each face of G, pick one point $v\hat{}*$ inside the face. These are the the set of vertices V^* of G^* .
- Any edge $e \in E$ of G that divides two faces of G and hence two vertices of $G*$, so let e^* be the edge of G^*.



Geometric duality

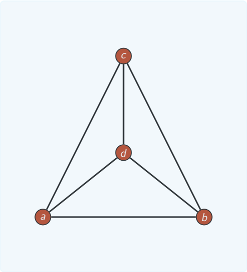
Let G = (V,E) be a planar connected graph.

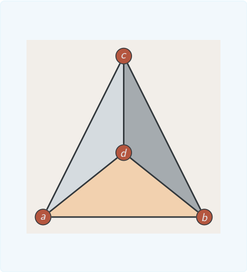
1. Is the number of edges which encloses the region *a* equal to the degree of the vertex correspondent to the region *a*?

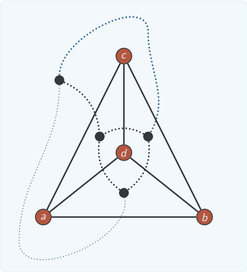


Let G = (V,E) be a planar connected graph.

1. Is the number of edges which encloses the region *a* equal to the degree of the vertex correspondent to the region *a*? Yes



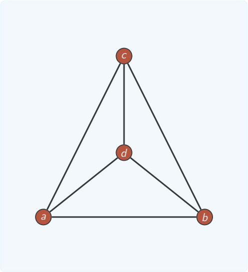


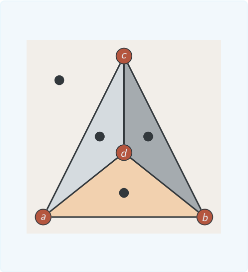


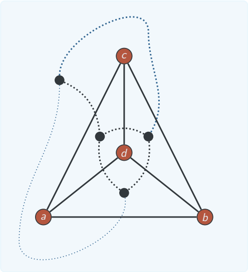
Geometric duality

Let G = (V,E) be a planar connected graph.

1. Is the number of edges which encloses the region *a* equal to the degree of the vertex correspondent to the region *a*? Yes
2. Is the dual graph a planar one?

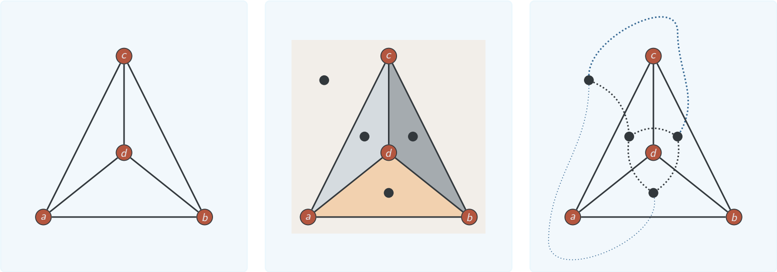






Let G = (V,E) be a planar connected graph.

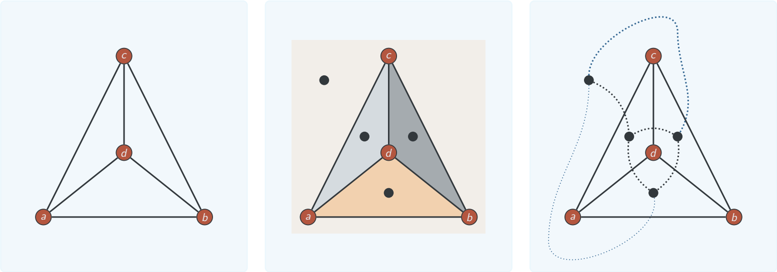
1. Is the number of edges which encloses the region *a* equal to the degree of the vertex correspondent to the region *a*? Yes
2. Is the dual graph a planar one? Yes



Geometric duality

Let G = (V,E) be a planar connected graph.

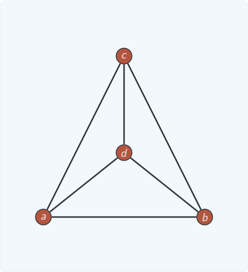
1. Is the number of edges which encloses the region *a* equal to the degree of the vertex correspondent to the region *a*? Yes
2. Is the dual graph a planar one? Yes
3. Is the dual graph of the dual graph G equal to G?

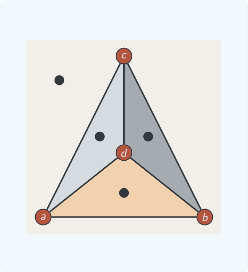


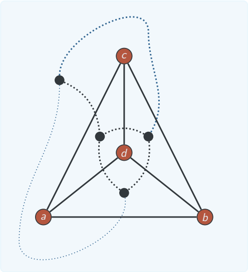
Geometric duality

Let G = (V,E) be a planar connected graph.

1. Is the number of edges which encloses the region *a* equal to the degree of the vertex correspondent to the region *a*? Yes
2. Is the dual graph a planar one? Yes
3. Is the dual graph of the dual graph G equal to G? No. They are isomorphic







Questions?

Planar graphs
– Geometric duality –

Programa de Pós-graduação em INFORMÁTICA

PUC Minas

IM SCIENCE

Teoria dos Grafos e Computabilidade

— Graph coloring —

Silvio Jamil F. Guimarães

Graduate Program in Informatics – PPGINF
Image and Multimedia Data Science Laboratory – IMScience
Pontifical Catholic University of Minas Gerais – PUC Minas

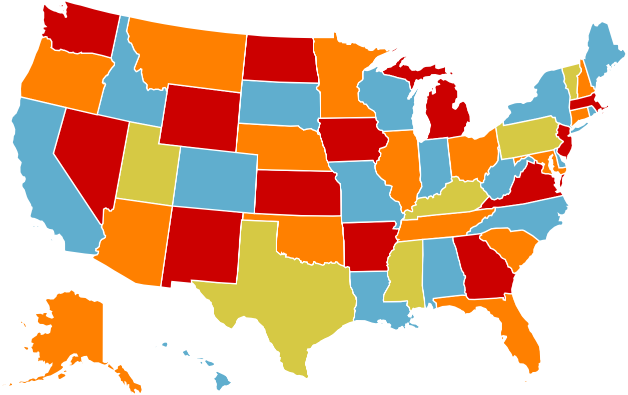
Aug 2022

Graph coloring

Here is a map of the USA country. Color it so that adjacent regions are colored differently. What is the fewest colors required?

Graph coloring

Here is a map of the USA country. Color it so that adjacent regions are colored differently. What is the fewest colors required?



Graph coloring

Here is a map of the USA country. Color it so that adjacent regions are colored differently. What is the fewest colors required?

There are maps can be colored with: (i) one color; (ii) two colors; (iii) three colors; (iv) four colors.

It turns out that the is no map that needs more than 4 colors. This is the famous Four Colour Theorem, which was originally conjectured by the British/South African mathematician and botanist, Francis Guthrie who at the time was a student at University College London

Graph coloring

Thanks to the geometric duality, a map can be seen as a graph in which:

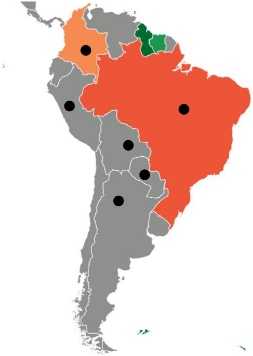
Graph coloring

Thanks to the geometric duality, a map can be seen as a graph in which:

Graph coloring

Thanks to the geometric duality, a map can be seen as a graph in which:

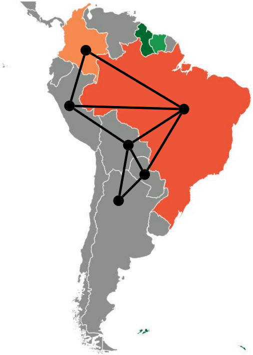
- A vertex in the graph corresponds to a region (face) in the map;



Graph coloring

Thanks to the geometric duality, a map can be seen as a graph in which:

- A vertex in the graph corresponds to a region (face) in the map;
- There is an edge between two vertices in the graph if the corresponds regions share a border.



Graph coloring

Thanks to the geometric duality, coloring regions of a map corresponds to coloring vertices of the graph.

Graph coloring

Thanks to the geometric duality, coloring regions of a map corresponds to coloring vertices of the graph.

- Vertex Coloring An assignment of colors to the vertices of a graph;

Graph coloring

Thanks to the geometric duality, coloring regions of a map corresponds to coloring vertices of the graph.

- Vertex Coloring An assignment of colors to the vertices of a graph;
- Proper Coloring If the vertex coloring has the property that adjacent vertices are colored differently.

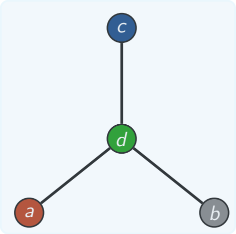
Graph coloring

Thanks to the geometric duality, coloring regions of a map corresponds to coloring vertices of the graph.

- **Vertex Coloring** An assignment of colors to the vertices of a graph;
- **Proper Coloring** If the vertex coloring has the property that adjacent vertices are colored differently.

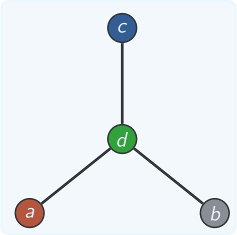
If the graph has v vertices, the clearly at most v colours are needed. However, usually, we need far fewer.

Graph coloring

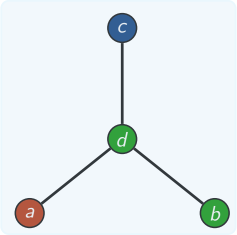


4 colors

Graph coloring

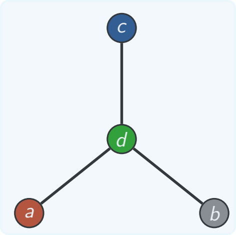


4 colors

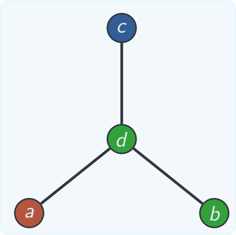


3 colors – no proper

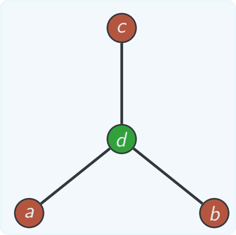
Graph coloring



4 colors

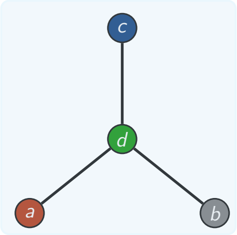


3 colors – no proper

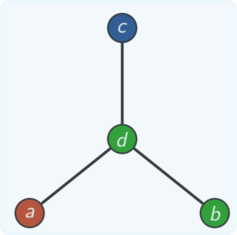


2 colors – proper and minimal

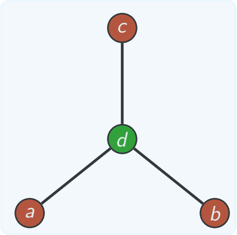
Graph coloring



4 colors



3 colors – no proper



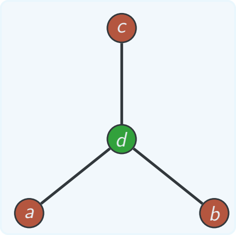
2 colors – proper and minimal

From now, the vertex coloring will be also proper coloring.

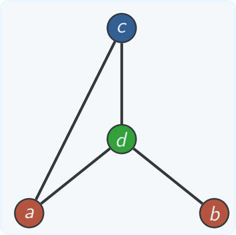
Graph coloring

The smallest number of colors needed to get a proper vertex coloring of a graph G=(V,E) is called the chromatic number of the graph, written $\chi(G)$ in which $1 \leq \chi(G) \leq |V|$.

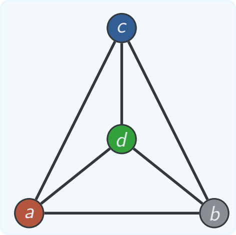
We said that a graph is K-colorable if K colors are sufficient to compute a vertex coloring.



$\chi(G) = 2$



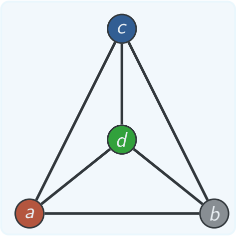
$\chi(G) = 3$



$\chi(G) = 4$

Graph coloring

If the graph $G = (V, E)$ is a complete one, then $\boxed{\chi(G) = |V|}$. If it is not complete the we can look at cliques in the graph.

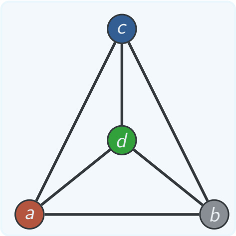


$\chi(G) = 4$

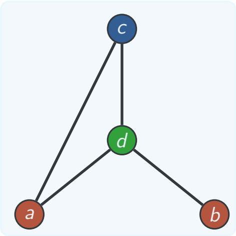
Graph coloring

If the graph $G = (V, E)$ is a complete one, then $\chi(G) = |V|$. If it is not complete the we can look at cliques in the graph.

A clique is a subgraph of a graph all of whose vertices are connected to each other.



$\chi(G) = 4$



$\chi(G) = 3$

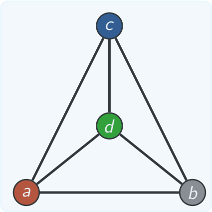
Graph coloring

The clique number of a graph, $G = (V, E)$, is the number of vertices in the largest clique in G.

Graph coloring

The clique number of a graph, $G = (V, E)$, is the number of vertices in the largest clique in G.

- The chromatic number of a graph G, called $\chi(G)$, is at least its clique number

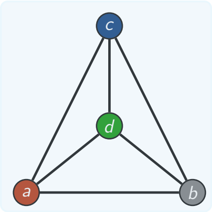


$\chi(G) = 4$

Graph coloring

The clique number of a graph, $G = (V, E)$, is the number of vertices in the largest clique in G.

- The chromatic number of a graph G, called $\chi(G)$, is at least its clique number Lower bound

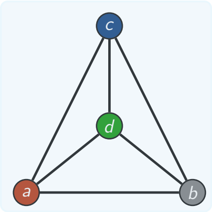


$\chi(G) = 4$

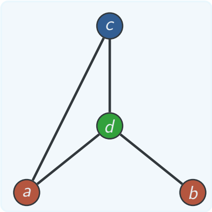
Graph coloring

The clique number of a graph, $G = (V, E)$, is the number of vertices in the largest clique in G.

- The chromatic number of a graph G, called $\chi(G)$, is at least its clique number Lower bound
- Let $\Delta(G)$ be the largest degree of any vertex in the graph, G. Thus $\chi(G) \leq \Delta(G) + 1$



$\chi(G) = 4$

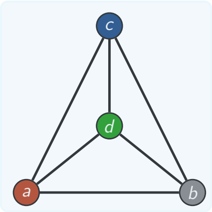


$\chi(G) = 3$

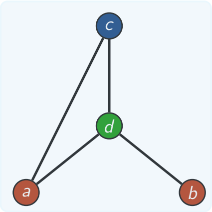
Graph coloring

The clique number of a graph, $G = (V, E)$, is the number of vertices in the largest clique in G.

- The chromatic number of a graph G, called $\chi(G)$, is at least its clique number Lower bound
- Let $\Delta(G)$ be the largest degree of any vertex in the graph, G. Thus $\chi(G) \leq \Delta(G) + 1$ Upper bound



$\chi(G) = 4$



$\chi(G) = 3$

Graph coloring

There are some algorithms that are efficient, but **not optimal** to compute a vertex coloring (that is proper too as defined).

Graph coloring

There are some algorithms that are efficient, but not optimal to compute a vertex coloring (that is proper too as defined).

1. The Greedy algorithm: simple and efficient

1. Number all the vertices and number your colors;
2. Give a color to the first vertex;
3. Take the remaining vertices in order. Assign each one the lowest numbered color, that is different from the colours of its neighbours.

Graph coloring

There are some algorithms that are efficient, but not optimal to compute a vertex coloring (that is proper too as defined).

1. The Greedy algorithm: simple and efficient, but the result can depend on the ordering of the vertices.

1. Number all the vertices and number your colors;
2. Give a color to the first vertex;
3. Take the remaining vertices in order. Assign each one the lowest numbered color, that is different from the colours of its neighbours.

Graph coloring

There are some algorithms that are efficient, but not optimal to compute a vertex coloring (that is proper too as defined).

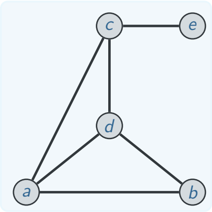
2. The Welsh-Powell algorithm: slightly more complicated, but can give better colorings.

1. Sort the vertices in non-increasing order of their degree;
2. Colour to the first vertex;
3. Take the next sorted vertice, giving that new or old color to the vertice depending if it is connected to one previously colorred or not.

Graph coloring

There are some algorithms that are efficient, but not optimal to compute a vertex coloring (that is proper too as defined).

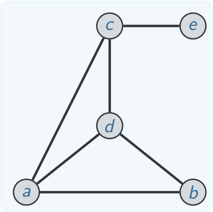
1. The Greedy algorithm: simple and efficient, but the result can depend on the ordering of the vertices.



Graph coloring

There are some algorithms that are efficient, but not optimal to compute a vertex coloring (that is proper too as defined).

1. The Greedy algorithm: simple and efficient, but the result can depend on the ordering of the vertices.

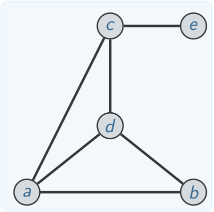


b–e–c–d–a

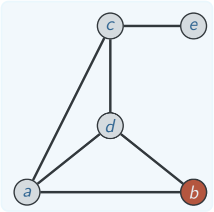
Graph coloring

There are some algorithms that are efficient, but not optimal to compute a vertex coloring (that is proper too as defined).

1. The Greedy algorithm: simple and efficient, but the result can depend on the ordering of the vertices.



b–e–c–d–a

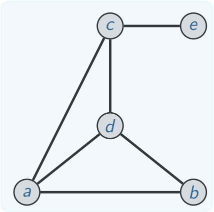


b–e–c–d–a

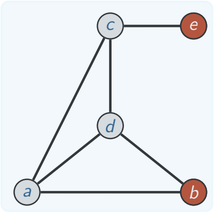
Graph coloring

There are some algorithms that are efficient, but not optimal to compute a vertex coloring (that is proper too as defined).

1. The Greedy algorithm: simple and efficient, but the result can depend on the ordering of the vertices.



b–e–c–d–a

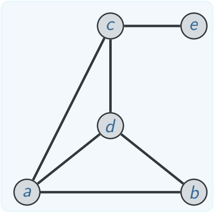


b–e–c–d–a

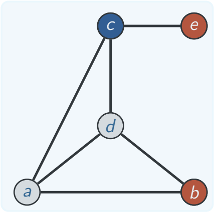
Graph coloring

There are some algorithms that are efficient, but not optimal to compute a vertex coloring (that is proper too as defined).

1. The Greedy algorithm: simple and efficient, but the result can depend on the ordering of the vertices.



b–e–c–d–a

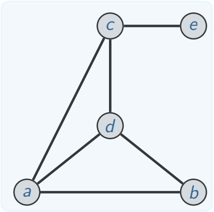


b–e–c–d–a

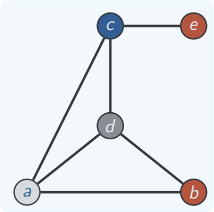
Graph coloring

There are some algorithms that are efficient, but not optimal to compute a vertex coloring (that is proper too as defined).

1. The Greedy algorithm: simple and efficient, but the result can depend on the ordering of the vertices.



b–e–c–d–a

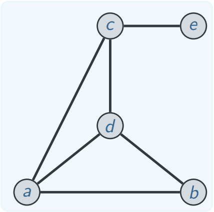


b–e–c–d–a

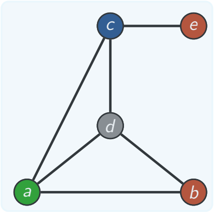
Graph coloring

There are some algorithms that are efficient, but not optimal to compute a vertex coloring (that is proper too as defined).

1. The Greedy algorithm: simple and efficient, but the result can depend on the ordering of the vertices.



b–e–c–d–a

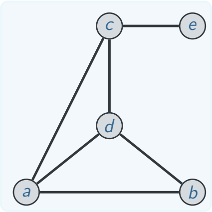


b–e–c–d–a

Graph coloring

There are some algorithms that are efficient, but not optimal to compute a vertex coloring (that is proper too as defined).

2. The Welsh-Powell algorithm: slightly more complicated, but can give better colorings.

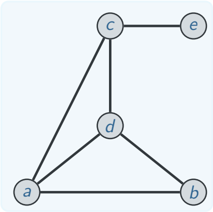


a–d–c–b–e

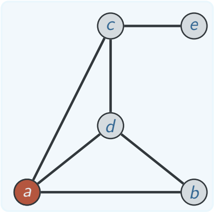
Graph coloring

There are some algorithms that are efficient, but **not optimal** to compute a vertex coloring (that is proper too as defined).

2. The Welsh-Powell algorithm: slightly more complicated, but can give better colorings.



a–d–c–b–e

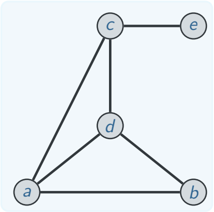


a–d–c–b–e

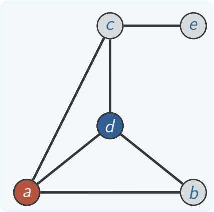
Graph coloring

There are some algorithms that are efficient, but not optimal to compute a vertex coloring (that is proper too as defined).

2. The Welsh-Powell algorithm: slightly more complicated, but can give better colorings.



a–d–c–b–e

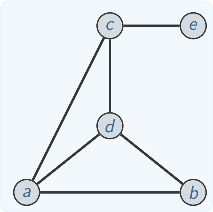


a–d–c–b–e

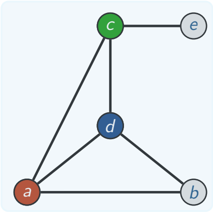
Graph coloring

There are some algorithms that are efficient, but **not optimal** to compute a vertex coloring (that is proper too as defined).

2. The Welsh-Powell algorithm: slightly more complicated, but can give better colorings.



a–d–c–b–e

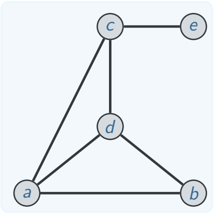


a–d–c–b–e

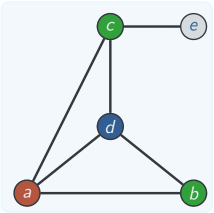
Graph coloring

There are some algorithms that are efficient, but **not optimal** to compute a vertex coloring (that is proper too as defined).

2. The Welsh-Powell algorithm: slightly more complicated, but can give better colorings.



a–d–c–b–e

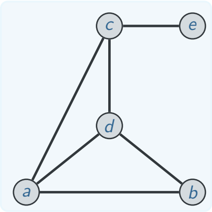


a–d–c–b–e

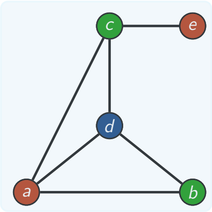
Graph coloring

There are some algorithms that are efficient, but not optimal to compute a vertex coloring (that is proper too as defined).

2. The Welsh-Powell algorithm: slightly more complicated, but can give better colorings.



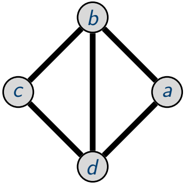
a–d–c–b–e

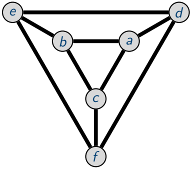


a–d–c–b–e

Edge coloring

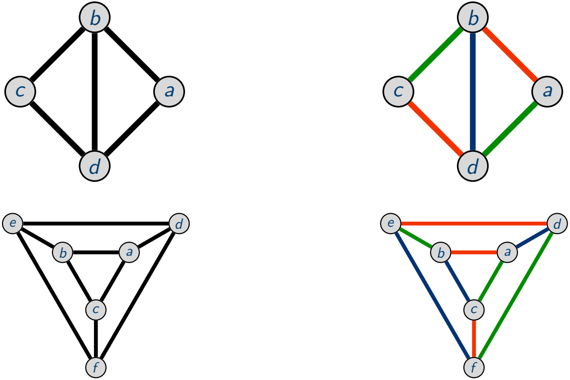
Let $G = (V, E)$ be a undirected connected graph.





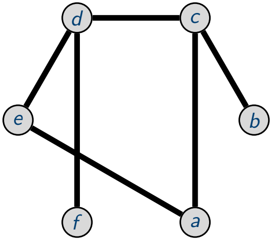
Let $G = (V, E)$ be a undirected connected graph.

- **Edge Coloring** is an assignment of colors to the edges of G in which adjacent edges are colored differently.



Edge coloring

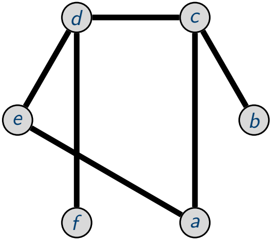
Let $G = (V, E)$ be a undirected connected graph.

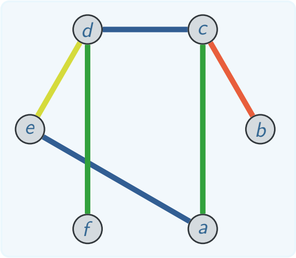


Edge coloring

Let $G = (V, E)$ be a undirected connected graph.

- The graph G is **K-edge-colorable** if the edges can be colored by using K colors;



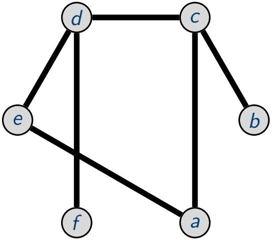


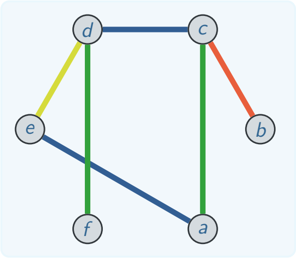
$K = 4$

Edge coloring

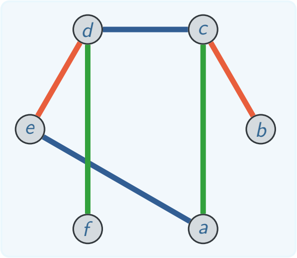
Let $G = (V, E)$ be a undirected connected graph.

- The graph G is K-edge-colorable if the edges can be colored by using K colors;
- The chromatic number $\chi'(G)$ is equal to the smallest number of K for coloring the edges of G.





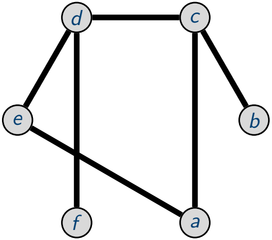
$K = 4$



$\chi'(G) = 3$

Line graph

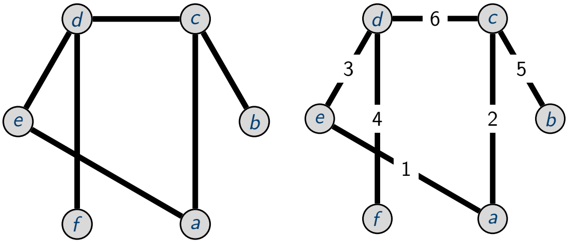
Let $G = (V, E)$ be a undirected connected graph.



Line graph

Let $G = (V, E)$ be a undirected connected graph. A **line graph** $L(G)$ is defined as follows:

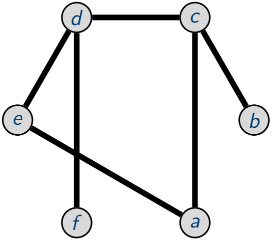
- The vertices of $L(G)$ are the edges of G;

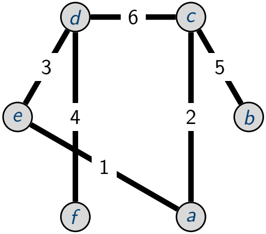


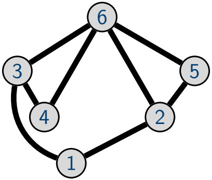
Line graph

Let $G = (V, E)$ be a undirected connected graph. A **line graph** $L(G)$ is defined as follows:

- The vertices of $L(G)$ are the edges of G;
- Two vertices are adjacent in $L(G)$ if their corresponding edges in G are adjacent.





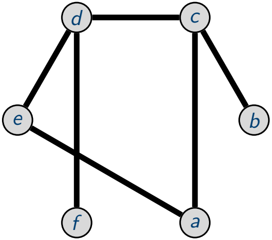


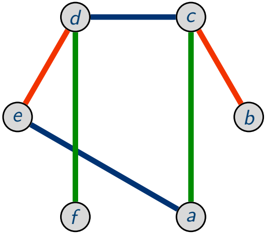
$\chi'(G) = \chi(L(G))$

Line graph

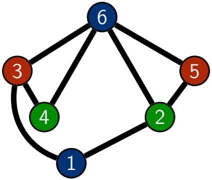
Let $G = (V, E)$ be a undirected connected graph. A **line graph** $L(G)$ is defined as follows:

- The vertices of $L(G)$ are the edges of G;
- Two vertices are adjacent in $L(G)$ if their corresponding edges in G are adjacent.





$\chi'(G) = 3$



$\chi(L(G)) = 3$

Questions?

Planar graphs

– Graph coloring –

Slots of time for exams – an example

A university is preparing a selection process for its n courses. How to organize the exams in order to minimize the number of days for the process in which each candidate can make just one exam per day. It's known that for the candidates will be applied specific exams depending on the course.

1. Computer Science – Math, Physics
2. Nutrition – Chemical, Biology, History
3. Architecture – Physics, Math, History
4. Biological Science - Chemical, Biology, Math

Slots of time for exams – an example

A university is preparing a selection process for its n courses. How to organize the exams in order to **minimize** the number of days for the process in which each candidate can make just one exam per day. It's known that for the candidates will be applied specific exams depending on the course.

1. Computer Science – Math, Physics
2. Nutrition – Chemical, Biology, History
3. Architecture – Physics, Math, History
4. Biological Science - Chemical, Biology, Math

How to model this selection process as a graph problem?

Task completion – an example

An industry has N tasks to be done and M employees. Each employee was assigned to a set of tasks, and the lenght of each task is by one day. Thus, how many days are needed to finish all tasks?

Task completion – an example

An industry has N tasks to be done and M employees. Each employee was assigned to a set of tasks, and the lenght of each task is by one day. Thus, how many days are needed to finish all tasks?

How to model this process as a graph problem?

Selection process – an example

A software house is hiring. For the positions, there are N software developers with different skills that will participate for the selection process. From a list of projects, each candidate must indicate just one project in which it wish to work. The interview for a specific candidate will be conducted by manager of the chosen project. How many slots of time are needed to the whole selection process?

Selection process – an example

A software house is hiring. For the positions, there are N software developers with different skills that will participate for the selection process. From a list of projects, each candidate must indicate just one project in which it wish to work. The interview for a specific candidate will be conducted by manager of the chosen project. How many slots of time are needed to the whole selection process?

How to model this process as a graph problem?