Graph Theory

Marco Patrignani

K.U.Leuven

19 October 2012

Application Area

graph

Application Area

graph old problems

Application Area

bridges of Konigsberg

graph old problems 4 colors

Application Area

networks

bridges of Konigsberg

graph

old problems

4 colors

Application Area

websites

networks

bridges of Konigsberg

graph old problems 4 colors

Application Area

websites

networks social 6 degrees

bridges of Konigsberg

graph old problems 4 colors

Application Area

websites

networks

social

6 degrees

bridges of Konigsberg

graph

old problems

4 colors

chemistry

Application Area

websites

networks social 6 degrees

bridges of Konigsberg

graph old problems 4 colors

display

chemistry search

Application Area

websites

networks social 6 degrees

bridges of Konigsberg

graph old problems 4 colors

display

chemistry search

trees

Application Area

websites

networks social 6 degrees

bridges of Konigsberg

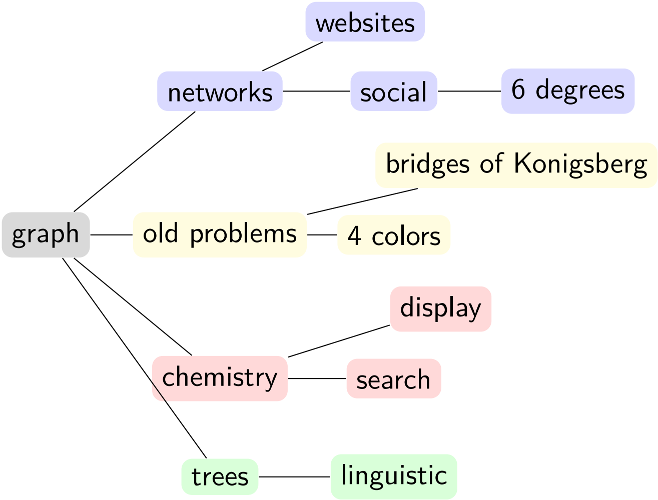
graph old problems 4 colors

display

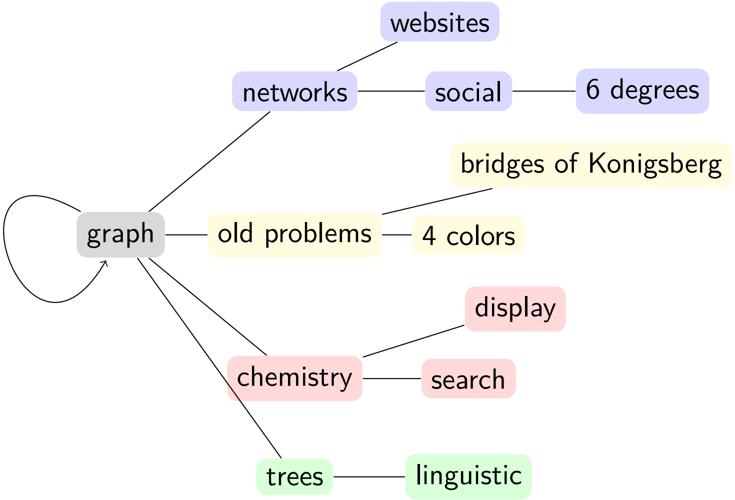
chemistry search

trees linguistic

Application Area



Application Area



Basic definitions

Definition (Graph)

A *graph* is an ordered triple $G = (V, E, \phi)$, where

1. $V \neq \emptyset$
2. $V \cap E = \emptyset$
3. $\phi : E \to \mathcal{P}(V)$ is a map such that $|\phi(e)| \in \{1, 2\}$ for each $e \in E$.

Basic definitions

Definition (Graph)

A *graph* is an ordered triple $G = (V, E, \phi)$, where

1. $V \neq \emptyset$
2. $V \cap E = \emptyset$
3. $\phi : E \rightarrow \mathcal{P}(V)$ is a map such that $|\phi(e)| \in \{1, 2\}$ for each $e \in E$.

Definition (Directed Graph)

A *directed graph* or *digraph* is an ordered triple $\vec{G} = (V, E, \eta)$, where

- $V \neq \emptyset$.
- $V \cap E = \emptyset$.
- $\eta : E \rightarrow V \times V$ is a map.

Types of graphs

Definition (Simple Graph)

$G = (V, E)$, where $V \neq \emptyset$ and E is a set of 2-elements from V such that $E \subseteq \{X \mid X \subseteq V, |X| = 2\} = \{\{u, v\} \mid u, v \in V, u \neq v\}$.

Types of graphs

Definition (Simple Graph)

$G = (V, E)$, where $V \neq \emptyset$ and E is a set of 2-elements from V such that $E \subseteq \{X \mid X \subseteq V, |X| = 2\} = \{\{u, v\} \mid u, v \in V, u \neq v\}$.

Definition (Complete Graph)

$G = (V, E)$, where $V \neq \emptyset$ and E is a set of 2-elements from V such a vertex is connected to all other vertices.

Types of graphs

Definition (Simple Graph)

$G = (V, E)$, where $V \neq \emptyset$ and E is a set of 2-elements from V such that $E \subseteq \{X \mid X \subseteq V, |X| = 2\} = \{\{u, v\} \mid u, v \in V, u \neq v\}$.

Definition (Complete Graph)

$G = (V, E)$, where $V \neq \emptyset$ and E is a set of 2-elements from V such a vertex is connected to all other vertices.

Definition (Bipartite graph)

A *bipartite graph* (or bigraph) is a graph whose vertices can be divided into two disjoint sets U and V such that every edge connects a vertex in U to one in V.

Types of graphs

Definition (Subgraph)

For graphs $G' = (V', E', \phi')$ and $G = (V, E, \phi)$, we say that G' is a *subgraph* of G if

1. $V' \subseteq V$,
2. $E' \subseteq E$,
3. $\phi'(e) = \phi(e)$ for all $e \in E'$.

Types of graphs

Definition (Subgraph)

For graphs $G' = (V', E', \phi')$ and $G = (V, E, \phi)$, we say that G' is a *subgraph* of G if

1. $V' \subseteq V$,
2. $E' \subseteq E$,
3. $\phi'(e) = \phi(e)$ for all $e \in E'$.

Definition (Clique)

A *clique* in a graph $G = (V, E, \phi)$ is a subset of the vertex set $C \subseteq V$ such that for every two vertices in C there is an edge connecting the two.

1. A *walk* in a graph $G = (V, E, \phi)$ is an alternating sequence

$$(u_0, e_1, u_1, e_2, \ldots, e_k, u_k)$$

of vertices and edges that begins and ends with a vertex.

1. A *walk* in a graph $G = (V, E, \phi)$ is an alternating sequence

$$(u_0, e_1, u_1, e_2, \ldots, e_k, u_k)$$

of vertices and edges that begins and ends with a vertex.

2. A *trail* in G is a walk with all of its edges $e_1, e_2, \ldots, e_k$ distinct.

1. A *walk* in a graph $G = (V, E, \phi)$ is an alternating sequence

$$(u_0, e_1, u_1, e_2, \ldots, e_k, u_k)$$

   of vertices and edges that begins and ends with a vertex.
2. A *trail* in G is a walk with all of its edges $e_1, e_2, \ldots, e_k$ distinct.
3. A *path* in G is a walk with all of its nodes $u_0, e_1, \ldots, u_k$ distinct.

1. A *walk* in a graph $G = (V, E, \phi)$ is an alternating sequence

$$(u_0, e_1, u_1, e_2, \dots, e_k, u_k)$$

of vertices and edges that begins and ends with a vertex.

2. A *trail* in G is a walk with all of its edges $e_1, e_2, \dots, e_k$ distinct.
3. A *path* in G is a walk with all of its nodes $u_0, e_1, \dots, u_k$ distinct.
4. A walk or trail of length at least one is *closed* if its initial vertex and final vertex are the same. A closed trail is also called a *circuit*.

1. A *walk* in a graph $G = (V, E, \phi)$ is an alternating sequence
$$(u_0, e_1, u_1, e_2, \ldots, e_k, u_k)$$
of vertices and edges that begins and ends with a vertex.
2. A *trail* in G is a walk with all of its edges $e_1, e_2, \ldots, e_k$ distinct.
3. A *path* in G is a walk with all of its nodes $u_0, e_1, \ldots, u_k$ distinct.
4. A walk or trail of length at least one is *closed* if its initial vertex and final vertex are the same. A closed trail is also called a *circuit*.
5. A *cycle* is a closed walk with distinct vertices except for the initial and final vertices, which are the same.

Types of graphs

Definition (Connected)

A graph G is *connected* if for every pair of distinct vertices $u, v \in V(G)$, there is a path from u to v. Otherwise we say that the graph is *disconnected*.

Types of graphs

Definition (Connected)

A graph G is *connected* if for every pair of distinct vertices $u, v \in V(G)$, there is a path from u to v. Otherwise we say that the graph is *disconnected*.

Definition (Connector components)

Let G be a graph. Let $H_1, \ldots, H_k$ be connected subgraphs of G whose vertex sets and edge sets are pairwise disjoint and such that they *cover* all the vertices and edges of G. That is,

$$
\begin{aligned}
V(G) &= V(H_1) \cup \cdots \cup V(H_k), \\
E(G) &= E(H_1) \cup \cdots \cup E(H_k),
\end{aligned}
$$

where $V(H_i) \cap V(H_j) = \emptyset = E(H_i) \cap E(H_j)$, for each distinct i, j.

Trees

Definition (Tree, Forest)

A *tree* is a connected graph that has no cycle as a subgraph. A *forest* is a graph in which every component is a tree.

Trees

Definition (Tree, Forest)

A *tree* is a connected graph that has no cycle as a subgraph. A *forest* is a graph in which every component is a tree.

Definition (Leaf)

A vertex u of a simple graph G is called a *leaf* if $d_G(u) = 1$. A vertex that is not a leaf is called an *internal vertex*.

Spanning Trees

Definition

Let G be a graph.

1. A subtree T of G is called a *spanning tree* of G if $V(T) = V(G)$.
2. A subforest F of G is called a *spanning forest* of G if for each component H of G, the subgraph $F \cap H$ is a spanning tree of H.

Kruskal's Algorithm

Kruskal's Algorithm
INPUT: A connected weighted graph (G, W) on n vertices.
OUTPUT: A minimum cost spanning tree T on G.
begin
$T_1 = \emptyset$.
for $i = 1$ **to** $n - 1$ **do** {
let $e_i \in E(G) \setminus E(T_i)$ be a minimum weight edge such that $T_i \cup \{e_i\}$ is a forest;
$T_{i+1} = T_i \cup \{e_i\}$; // that is, e_i along with its other endpoint added
}
output $T = T_n$.
end

Prim's Algorithm

Prim's Algorithm

INPUT: A connected weighted graph (G, W) on n vertices.

OUTPUT: A minimum cost spanning tree T on G.

begin

$T_1 = (\{u_1\}, \emptyset)$. // u_i arbitrary initial vertex

for $i = 1$ **to** n **do** {

let $e_i \in E(G) \setminus E(T_i)$ be a minimum weight edge such that $|V(T_i) \cap e_i| = 1$;

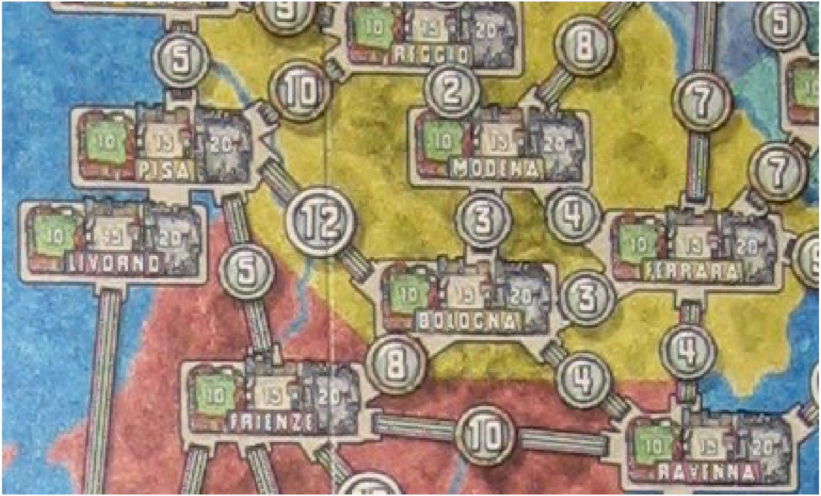
$T_{i+1} = T_i \cup \{e_i\}$; // e_i with its other endpoint added

}

output $T = T_n$.

end

Example



Eulerian Graphs

Definition

Let G be a graph. A trail of G that contains each edge of G is called an *Eulerian trail* of G. A circuit of G that contains each edge of G is called an *Eulerian circuit* of G. If G has an Eulerian circuit, then G is called an *Eulerian graph*.

Properties of Eulerian Graphs

Theorem

A connected graph G is Eulerian if and only if each vertex in G has even degree.

Properties of Eulerian Graphs

Theorem

A connected graph G is Eulerian if and only if each vertex in G has even degree.

Corollary

A connected graph G has an Eulerian trail if and only if all except two vertices in G have an even degree.

Hamiltonian Graphs

Definition

Let G be a graph. A path in G that includes every vertex of G is called a *Hamiltonian path* of G. A cycle that includes every vertex in G is called a *Hamiltonian cycle* of G. If G contains a Hamiltonian cycle (that is a path), then G is called a *Hamiltonian graph*.

Observations on Hamiltonian Graphs

- Every Hamiltonian graph must be connected.

Observations on Hamiltonian Graphs

- Every Hamiltonian graph must be connected.
- No tree is Hamiltonian.

Observations on Hamiltonian Graphs

- Every Hamiltonian graph must be connected.
- No tree is Hamiltonian.
- For each $n \geq 3$, the cycle graph C_n is Hamiltonian.

Observations on Hamiltonian Graphs

- Every Hamiltonian graph must be connected.
- No tree is Hamiltonian.
- For each $n \geq 3$, the cycle graph C_n is Hamiltonian.
- For each $n \geq 3$, the complete graph K_n is Hamiltonian.

Observations on Hamiltonian Graphs

- Every Hamiltonian graph must be connected.
- No tree is Hamiltonian.
- For each $n \geq 3$, the cycle graph C_n is Hamiltonian.
- For each $n \geq 3$, the complete graph K_n is Hamiltonian.
- For each $n \geq 2$, the complete bipartite graph $K_{n,n}$ is Hamiltonian.

Properties of Hamiltonian Graphs

Theorem

If G is a simple Hamiltonian graph, then for each $S \subseteq V(G)$, the number of components of $G - S$ is at most $|S|$.

Example

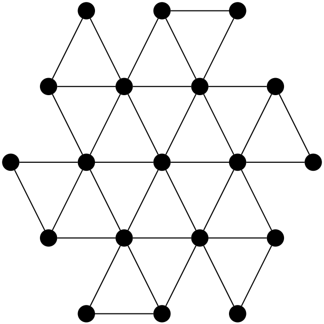
Example

Let the complete bipartite graph $K_{2,3}$ be presented on the vertices

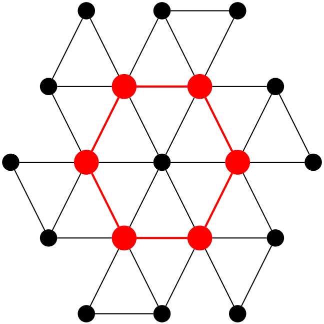
$$V(K_{2,3}) = \{u_1, u_2, u_3\} \cup \{v_1, v_2\}.$$

where each u_i is connected to each v_j. If we let $S = \{v_1, v_2\}$, then $|S| = 2$ but $G - S$ is a graph consisting of three isolated vertices u_1, u_2 and u_3, and hence $G - S$ has three components, one more than the elements of S.

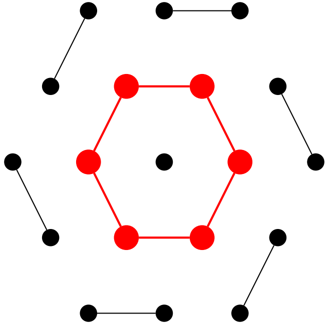
Example



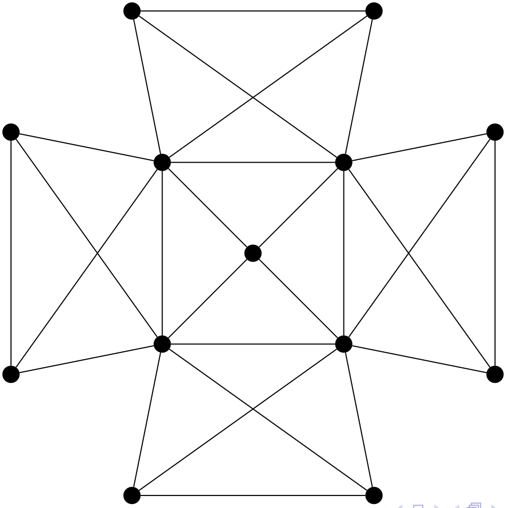
Example



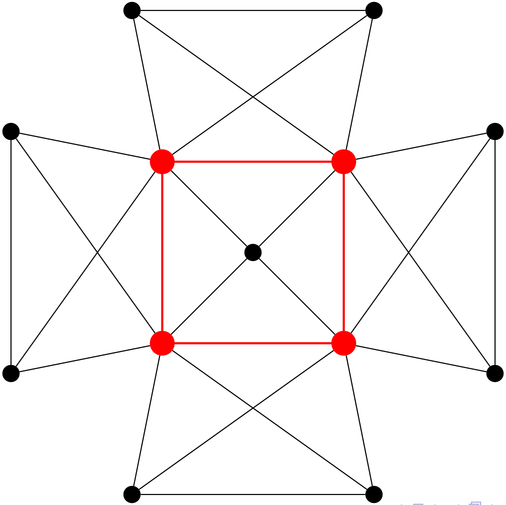
Example



Example



Example



Example