S-72.2420/T-79.5203 Graph Theory (5 cr) P(=L)

Lectures: Wednesdays and Fridays 9–12, room T4

Teachers: Dr. Petteri Kaski, Dept. of Computer Science, University of Helsinki, http://www.cs.helsinki.fi/u/pkaski/; and Prof. Patric Östergård, Dept. of Communications and Networking, TKK, http://www.tkk.fi/u/pat/

Tutorials: Mondays and Thursdays 10–12, room T4, first tutorial on March 27

Assistant: Olli Pottonen, M.Sc.(Tech), room SI436 (Otakaari 5), tel. 451 2350, e-mail: olli.pottonen(at)tkk.fi

Home page: http://www.tcs.hut.fi/Studies/T-79.5203/

Contents

- ▷ Graph theory
- ▷ Graph algorithms
- ▷ Applications of graph theory and algorithms

This course combines very naturally with the courses T-79.4201 Search Problems and Algorithms and T-79.5202 Combinatorial Algorithms. In the current course, less emphasis is put on those graph-theoretical problems that are considered in T-79.5202.

Prerequisites

Prerequisites: Basic courses in mathematics and computer science.

Graph theory forms a rather self-sufficient part of mathematics, so a capability of (abstract) mathematical reasoning is more important than a solid mathematical background.

Literature (1)

There is a vast literature on (basic and more advanced) graph theory.

[Wes] D. B. West, *Introduction to Graph Theory*, 2nd ed., Prentice Hall, Upper Saddle River, 2001. [**Course literature**.]

[Jun] D. Jungnickel, *Graphs, Networks and Algorithms*, 2nd ed., Springer, Berlin, 2005. [**Course literature**.]

Robin J. Wilson, *Introduction to Graph Theory*, 4th ed., Addison-Wesley, Reading, 1997. [**Easy reading**.]

B. Bollobás, *Modern Graph Theory*, Springer, New York, 1998.

Literature (2)

R. Diestel, *Graph Theory*, 3rd ed., Springer, New York, 2005. [**Electronically available at** `http://www.math.uni-hamburg.de/home/diestel/books/graph.theory/`]

C. D. Godsil and G. F. Royle, *Algebraic Graph Theory*, Springer, New York, 2001.

J. Gross and J. Yellen, *Graph Theory and Its Applications*, CRC Press, Boca Raton, 1998.

More books are listed at `http://www.ericweisstein.com/encyclopedias/books/GraphTheory.html`

Journals and Conferences

Graph theory is perhaps the most important—at least most popular—area of discrete mathematics. There is a wide range of journals publishing results on graph theory and algorithms;

Main journals: *Journal of Graph Theory*; *Journal of Combinatorial Theory (B)*.

Other journals: *Discrete Mathematics*; *Graphs and Combinatorics*; *Journal of Graph Algorithms and Applications*.

Outline of the Course

Part I. Graph Theory (lectured by P. Östergård).

1. Introduction
2. Basic concepts
3. Trees and distance; Graph parameters
4. Connectivity; Coloring
5. Planarity; Edges and cycles
6. Ramsey theory; Random graphs

Part II. Graph Algorithms (lectured by P. Kaski).

To pass the course: (A) Exam (14.5.2008, 9–12, T1) and (B) project. $A, B \in \{0, 1, 2, 3, 4, 5\}$. Mark $= 0.6A + 0.4B$ rounded to the nearest integer; A, B must be ≥ 1 to pass the course. Information on extra points for the exam through the tutorials is given separately.

Graphs (1)

The Königsberg Bridge Problem [Wes, Example 1.1.1] is said to have been the birth of graph theory (Königsberg = Kaliningrad).

A **graph** G is a triple consisting of a **vertex set** $V(G)$ (or simply V), an **edge set** $E(G)$ (or simply E), and a relation that associates with each edge two vertices (not necessarily distinct) called its **endpoints**. If $v \in V$ is the endpoint of $e \in E$, then v and e are said to be **incident**.

Example. In the graph in [Wic, Example 1.1.1], the vertex set is $\{x, y, z, w\}$, the edge set is $\{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$, and the assignments of endpoints to edges can be read from the picture.

Graphs (2)

loop An edge whose endpoints are equal.
multiple edges Edges having the same endpoints.
simple graph A graph with neither loops nor multiple edges.
adjacent vertices, neighbors Vertices connected by an edge.

The edges of simple graphs are here treated as unordered pairs of vertices, written $e = uv$ (or vu). Adjacency is denoted by $u \leftrightarrow v$.

Infinite graphs ($|V| = \infty$ or $|E| = \infty$) and the *null graph* ($|V| = |E| = 0$) are special types of graphs not considered here.

Graphs (3)

complement The complement of a simple graph G is the simple graph $\overline{G}$ with vertex set $V(G)$ and $uv \in E(\overline{G})$ iff $uv \notin E(G)$.
clique A set of pairwise adjacent vertices.
independent set, stable set A set of pairwise nonadjacent vertices.

Example. Does every set of six people contain three mutual acquaintances or three mutual strangers? In graph-theoretic terms: Does every 6-vertex graph have a clique of size 3 or an independent set of size 3? (Here edges identify pairwise acquaintances.)

Graphs (4)

bipartite A graph is bipartite if its vertices can be partitioned into the union of two disjoint independent (*partite*) sets.
vertex coloring Assignment of colors to the vertices of a graph so that no two adjacent vertices have the same color.
chromatic number The minimum number of colors in a vertex coloring of a graph G; denoted by $\chi(G)$.
k-partite The vertices of a graph G can be partitioned into the union of k disjoint independent sets (this holds iff $\chi(G) \leq k$).

Example. *Map coloring.* Create a graph with one vertex for each region, connecting regions sharing a boundary with edges. A famous problem in graph theory: Does every *planar* graph G have $\chi(G) \leq 4$?

Graphs (5)

walk A sequence of alternating vertices and edges in a graph, $v_0e_0v_1e_1\cdots$, such that the endpoints of e_i are v_i and v_{i+1}.
path A walk where all vertices are distinct.
cycle A walk where the first and the last vertex (called *endpoints*) coincide and the other vertices are distinct (also called an *n-cycle*, if it has n vertices).

The terms *path* and *cycle* also mean graphs with the aforementioned vertices and edges; such graphs are denoted by P_n and C_n, respectively, where $n = |V|$.

Example. See the graphs drawn in [Wes, Example 1.1.15].

Graphs (6)

subgraph A subgraph of a graph G is a graph H for which $V(H) \subseteq V(G)$, $E(H) \subseteq V(G)$, and the endpoint assignments to edges in H is the same as in G.

induced subgraph Given a graph G, the (unique) subgraph induced by $V' \subseteq V(G)$ has vertex set V' and contains all edges of G whose both endpoints are in V'.

connected graph Every pair of vertices belongs to a path. A graph that is not connected is said to be *disconnected*.

Specifying Graphs

We now consider *loopless* graphs, that is, multiple edges are allowed but loops are not. Let $V(G) = \{v_1, v_2, \ldots, v_n\}$ and $E(G) = \{e_1, e_2, \ldots, e_m\}$.

adjacency matrix The $n \times n$ matrix $A(G)$ in which the entry $a_{i,j}$ is the number of edges in G with endpoints v_i and v_j ($a_{i,i} = 0$ for all i).

incidence matrix The $n \times m$ matrix $M(G)$ in which the entry $m_{i,j}$ is 1 (0) if v_i is (not) an endpoint of e_j.

vertex degree, valency The number of incident edges.

Example. See [Wes, Example 1.1.19].

Graph Isomorphism (1)

So far we have considered labelled graphs: the vertices and edges have names. But structural properties of graphs do not depend on the labelling.

An **isomorphism** from a simple graph G to a simple graph H is a bijection $f : V(G) \to V(H)$ such that $uv \in E(G)$ iff $f(u)f(v) \in E(H)$. If there exists such an isomorphism, then we say that G *is isomorphic to* H and write $G \cong H$.

Example. The graphs in [Wes, Example 1.1.21] are isomorphic; for example, $f(w) = a$, $f(x) = d$, $f(y) = b$, and $f(z) = c$.

Graph Isomorphism (2)

Theorem 1.1.24. The isomorphism relation is an equivalence relation on the set of (simple) graphs.

General algorithms for isomorphism testing are considered in the algorithm part. Note, however, that if we are given the function f, it is easy to check that two graphs are indeed isomorphic. To prove that graphs are not isomorphic, one may try to find structural properties in which they differ: the number of edges, subgraphs, etc.

Note: For *dense* graphs, it may be useful to utilize the fact that $G \cong H$ iff $\overline{G} \cong \overline{H}$ [Wes, Exercise 1.1.4].

Important Graphs

We already introduced the graph classes P_n and C_n; two other important classes are as follows:

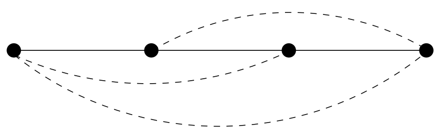
complete graph A simple graph whose vertices are pairwise adjacent; denoted by K_n, where $n = |V|$.

complete bipartite graph A simple bipartite graph whose vertices are adjacent iff they are in different partite sets; denoted by $K_{r,s}$ when the partite sets have sizes r and s.

Example. The graphs K_5 and $K_{2,3}$ are depicted in [Wes, Definition 1.1.27].

Decomposition (1)

It turns out that $P_4 \cong \overline{P_4}$.



self-complementary graph A graph G for which $G \cong \overline{G}$.

decomposition A collection of subgraphs such that each edge appears in exactly one of these subgraphs.

An n-vertex graph G is self-complementary iff K_n has a decomposition consisting of two copies of G.

Decomposition (2)

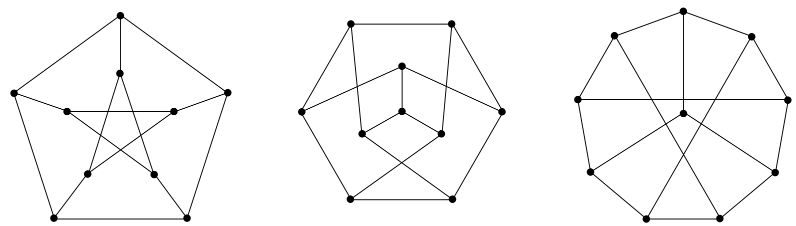
The question of which complete graphs decompose into copies of K_3 is exactly that of finding Steiner triple systems in design theory!

The graph K_3 is called a **triangle**. Several other small graphs have been given names, for example, $K_{1,3}$ is called a **claw**. See [Wes, Example 1.1.35] for further names of small graphs.

Note: A necessary (but not sufficient) condition for decomposition H into copies of G is that $|E(G)|$ divides $|E(H)|$.

The Petersen Graph (1)

The **Petersen Graph** can be constructed by taking one vertex for each 2-element subset of a 5-element set and inserting edges between vertices whose corresponding subsets are disjoint.



The Petersen Graph (2)

Theorem 1.1.38. Two nonadjacent vertices in the Petersen graph have exactly one common neighbor.

Proof: Nonadjacent vertices are 2-sets sharing one element, so their union has size 3. Then there are exactly two elements that are not in the union; these form a 2-set, which is the unique neighbor of both vertices. □

Also adjacent vertices have the same number of common neighbors (how many?), and since all vertices have the same degree, the Petersen graph is a **strongly regular graph**.

The Petersen Graph (3)

The **girth** of a graph is the length of its shortest cycle (or ∞ if no cycles occur).

Theorem 1.1.40. The Petersen graph has girth 5.

Proof: The graph is simple, so it has neither 1- nor 2-cycles. A 3-cycle would mean that there are three pairwise disjoint 2-sets out of a 5-set, which is not possible. A 4-cycle in the absence of 3-cycles would require nonadjacent vertices with two common neighbors, which is not possible by Theorem 1.1.38. By construction (many) 5-cycles can be found. □

Automorphisms

An **automorphism** is an isomorphism from G to G. A graph G is **vertex-transitive** if for all $u, v \in V(G)$ there is an automorphism mapping u to v.

Example. By permuting the values in the set $\{1, 2, 3, 4, 5\}$, we get $5! = 120$ different permutations of the vertices (2-subsets), which are (the only [Wes, Exercise 1.1.43]) automorphisms for the Petersen graph.

Induction

Induction is a common proof method in graph theory.

Theorem 1.2.1. (Strong Principle of Induction) Let $P(n)$ be a statement with an integer parameter n. If the following two conditions hold, then $P(n)$ is true for each positive integer n.

1. (basis step) $P(1)$ is true.
2. (induction step) For all $n > 1$, "$P(k)$ is true for $1 \leq k < n$" (induction hypothesis) implies "$P(n)$ is true".

Connection (1)

The concepts of *walk*, *path*, and *cycle* were introduced earlier. A **trail** is a walk with no repeated edges. The **length** of a walk is its number of edges. A walk is **closed** if its endpoints are the same.

Theorem 1.2.5. If there is a walk from u to v, then there is a path from u to v.

Proof: We use induction on the length l of a walk W.
Basis step: $l = 0$. Obvious.
Induction step: $l \geq 1$. If W has no repeated vertices, then W is a path. If W has a repeated vertex w, then delete the part of the walk between two occurrences of w to get a shorter walk. By the induction hypothesis, the theorem follows. □

Connection (2)

component A component of a graph G is a connected subgraph that is *maximal*, that is, it is not contained in any other connected subgraph of G.
trivial component A component with no edges.
isolated vertex A vertex of degree 0, which is always a (trivial) component.

Example. The graph of [Wes, Example 1.2.9] has four components.

Connection (3)

Adding (deleting) an edge decreases (increases) the number of components by 0 or 1. Deleting a vertex, however, can cause a considerable increase in the number of components (consider $K_{1,m}$). If deleting a edge or an vertex increases the number of components, we have a **cut-edge** or a **cut-vertex**, respectively.

We write $G - e$ or $G - M$ for the subgraph obtained by deleting an edge e or a set of edges M; and analogously for a vertex v or a set of vertices S: $G - v$, $G - S$.

Example. The graph of [Wes, Example 1.2.9] has cut-vertices v and y, and it has cut-edges qr, vw, xy, and yz.