Intelligent and Adaptable Software Systems

Advanced Algorithms: Graph Theory

Dr. Arno Formella

Computer Science Department
University of Vigo

11/12

Course organization

course notes

- Homepage:
  `http://www.ei.uvigo.es/~formella/doc/ssia11`
- almost everything will be accessible on our moodle-platform:
  `http://postgrado.ei.uvigo.es/tadsi-online/login/index.php`
- whiteboard illustrations (notations, ideas for proofs, algorithms)
- very short introduction to some specific aspects of graph theory and their applications

Course organization

class room hours

Graph Theory, WednesDay, 16:00–18:00

14.09. (class)	21.09. (lab)	28.10. (Arno)	05.10. (Arno)	19.10. (Arno)
26.10. (Arno)	02.11. (Arno)	09.11. (Marta)	16.11. (Marta)	23.11. (Marta)
30.11. (Marta)	07.12. (Marta)	19.12. (Marta)	21.12. (Marta)	11.01. (??)
18.01. eval	25.01. eval			

Course organization

office hours

- Dra. Marta Pérez Rodríguez
  office hours: ??, ??-??
- Dr. Arno Formella
  office hours: tuesdays, 10-13 and 17-20

Bibliography

books

- Reinhard Diestel. *Graph Theory.* 3rd edition, Springer Verlag, 2005. ISBN 3-540-26183-4.
  Existe una versión electrónica entre–enlazada (no imprimible):
  `http://diestel-graph-theory.com/index.html`
- Thomas H. Cormen, Charles E. Leiseron, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition.* Especially Part VI. McGraw Hill, 2001. ISBN 0-262-03292-7.

(working in september 2011)

- http://www.graphtheory.com
- http://www.ericweisstein.com/encyclopedias/books/GraphTheory.html
- http://mathworld.wolfram.com/Graph.html
- http://en.wikipedia.org/wiki/Graph_theory

Bibliography

course notes (examples...)

(working in september 2011)

- Gregorio Hernández Peñalver, Universidad Politécnica de Madrid,
  `http://www.dma.fi.upm.es/docencia/segundociclo/teorgraf`
  (in Spanish)
- Steven C. Locke, Florida Atlantic University,
  `http://www.math.fau.edu/locke/graphthe.htm`
  (alphabetically ordered)

Your work

homework, lab hours, presentations

graph theory: study the material

graph programming libraries: analyze and use of programming tools and libraries that work with graphs (Leda, GraphBase, Boost, etc.)

graph visualization: analyze and use of tools to visualize graphs and the information they contain (OGDF, Graphviz, yED, etc.)

applications: search for applications that use graph algorithms or graphs as data structures (e.g., network planification, route optimization) graphs can be found, e.g., in the following situations:

- street plans or maps
- networks (data, fluids, traffic etc.)
- transport systems
- chemical connexions in a large molecule
- neighborhood relations in a worldmap
- interference relations between antennas in a wireless communication system
- links between WWW pages
- closeness relation in arrangments

Hence, a graph is an abstract concept behind the representation of relations (edges) between entities (nodes or vertices)

Motivation

usage to resolve problems

- Determine the order how to dress cloths.
- Is it possible to design a journey through a city that passes through every street exactly once?
- What is the shortest distance a postman must walk to visit (pass along) each street or the necessary ones at least once?

Motivation

more examples

- How should we direct the street using one–direction signs such that it is still possible to drive from everywhere to everywhere?
- What are the necessary conditions to organize a group dance at a party such that the pairs consist of partners who knew each other beforehand?
- How should we place the chips on a board to minimize the interconnection length?
- Where should be place the firebrigades to shorten their maximum distances to any house?

Notations

basic issues

V	set of nodes or vertices
$[V]^r$	set of subsets of V of size r
$E \subseteq [V]^2$	set of edges
$v \in V$	vertex or node
$e = \{x, y\} \in E$	edge
$\{x, y\} \Longleftrightarrow: xy$	xy is edge
$G = (V, E)$	graph
$V(G), E(G)$	vertices and edges of the graph G
$v \in G :\Longleftrightarrow v \in V(G)$	v is vertex of the graph G
$e \in G :\Longleftrightarrow e \in E(G)$	e is edge of the graph G
$\|V\| =: n$	number of vertices
$\|V\| = \|V(G)\| = \|G\|$	equivalent notations
$\|E\| =: m$	number of edges
$\|E\| = \|E(G)\| = \\|G\\|$	equivalent notations

trivial	if $\|G\| = 0$ or $\|G\| = 1$, the graph is trivial
over	if $G = (V, E)$, G is a graph over V
incident	a vertex v is incident to an edge e, if $v \in e$
	an edge e is incident to a vertex v, if $v \in e$
adjacent	two vertices v and w are adjacent, if $\{v, w\} \in E$
	two edges e and f are adjacent, if $e \cap f \neq \emptyset$
connected	an edge connects its vertices
$X - Y$–edge	if $x \in X \subseteq V$ and $y \in Y \subseteq V$, xy is $X - Y$–edge
$E(X, Y)$	set of $X - Y$–edges

Notations

degree related

$E(v) := E(v, V \setminus \{v\})$	set of edges incident to v
$N(v)$	set of vertices adjacent to v (neighbors)
$d(v) := \lvert E(v)\rvert = \lvert N(v)\rvert$	degree of vertex v
$d_G(v)$	degree of vertex $v \in G$
$\delta(G)$	minimum degree of the vertices in G
$\Delta(G)$	maximum degree of the vertices in G
$d(G) := 2\lvert E\rvert/\lvert V\rvert$	mean degree of the vertices en G
$\epsilon(G) := \lvert E\rvert/\lvert V\rvert = \frac{d(G)}{2}$	mean number of edges per vertex of G

Notations

substractions

$$
\begin{array}{ll}
G \setminus e & \text{graph } (V, E \setminus \{e\}) \\
G \setminus v & \text{graph } (V \setminus \{v\}, E \setminus E(v)) \\
G \setminus E' & \text{graph } (V, E \setminus E') \\
G \setminus V' & \text{graph } (V \setminus V', E \setminus E(V'))
\end{array}
$$

neighbor node	v is neighbor of w, if $vw \in E(v)$, i.e., if v and w are adjacent
neighbor edge	e is neighbor of f, if $e \cap f \neq \emptyset$ i.e., e and f are incident to the same vertex
independent	vertices/edges non adjacent, a set of vertices (edges) mutually independent is an independent set
complete	a graph is complete, if all its vertices are neighbors
partition	the set of set $\{V_0, ..., V_{r-1}\}$ is a partition of V, if $V = \bigcup_i V_i$, $V_i \neq \emptyset$, and $\forall i \neq j : V_i \bigcap V_j = \emptyset$

$\alpha(G)$ size of the largest independent set of vertices

digraph	the edges are directed, i.e., instead of the sets $\{v, w\}$ we use pairs (v, w) or (w, v) i.e., $E \subseteq V \times V$
multigraph	permits more than one edge between vertices
pseudograph	permits loops on vertices

A graph can be stored with three basic methods:

- adjacency matrix
  - square matrix, and in the simple case, binary (and symmetric if not digraph) that codes whether there exists an edge between vertices
  - space complexity $\Omega(n^2)$
- adjacency lists
  - list or array of vertices which contain in each entry a list to its adjacent vertices
  - space complexity $\Theta(n+m)$

- hashtables
  - list or array of vertices which contain in each entry a hashtable to its adjacent vertices
  - space complexity $\Theta(n + m)$

What are the principal advantages and disadvantages of each method?

There are more data structures available which are useful to implement certain algorithms more efficiently (especially for planar graphs).