
Common Graphs, Models

10 March 2010
Lecture 2

Slide credits: R. H. Mak and P. Velkamp

Topics for Today

- Graph Definitions
 - Common Graph Types and their properties
- Models
 - Why, which
- Transition systems
 - Executions

Source: Tel App B, 2

Undirected graphs

An *undirected graph* G is a pair (V, E) , where

- V is the *node set* of G
- $E \subseteq \{ \{u, v\} \mid u, v \in V \}$ is the *edge set* of G .
 - edges are unordered pairs
 - instead of $\{u, v\}$ we also write uv , or equivalently, vu
 - no autoloops $\{u, u\}$
 - if $uv \in E$, then nodes u and v are *adjacent*, and uv is an *incident* edge of both u and v

For $v \in V$ we define the set of *neighbors* of v by

- $Neigh_v = \{ u \mid uv \in E \}$
- $|Neigh_v|$ is the *degree* of v

Directed graphs

A *directed graph* G is a pair (V, E) , where

- V is the *node set* of G .
- $E \subseteq V \times V$ is the *edge (arc, arrow) set* of G .
 - edges are ordered pairs
 - instead of (u, v) we also write uv
 - $uv \in E$ is an *outgoing edge* of u and an *incoming edge* of v
 - no autoloops (u, u)

For $v \in V$ we define the sets

- $In_v = \{ u \mid uv \in E \}$ of *in-neighbors*, or *predecessors*,
- $Out_v = \{ u \mid vu \in E \}$ of *out-neighbors*, or *successors*.
- $|In_v|$ is the *in-degree* and $|Out_v|$ is the *out-degree* of v ,
- $Deg(v) = |In_v| + |Out_v|$ is the *degree* of v .

Subgraphs, Weighted graphs

1. A graph $G' = (V', E')$ is a *subgraph* of graph $G = (V, E)$, if $V' \subseteq V$ and $E' \subseteq E$.
2. A subgraph G' is an *induced* subgraph (induced by its vertex set)
 - if $E' = \{ uv \in E \mid u \in V' \wedge v \in V' \}$ (undirected case)
 - or $E' = E \cap V' \times V'$, (directed case)
- If $V' = V$, then G' is a *spanning* subgraph.
3. A *weighted* graph is a (directed or undirected) graph in which each pair $uv \in E$ is assigned a numerical value ω_{uv} .
 - A weight assignment is symmetric when $\omega_{uv} = \omega_{vu}$.

Paths, cycles

A *path* P is of length k is a sequence $\langle v_0, \dots, v_k \rangle$ of $k+1$ nodes such that $v_i v_{i+1}$ is an edge for $0 \leq i < k$.

- If $\forall_{0 \leq i < j \leq k} v_i \neq v_j$ then P is a *simple path*.
- If $v_0 = v_k$, then P is a *cycle*.
- If $v_0 = v_k$ and $\forall_{1 \leq i < j \leq k} v_i \neq v_j$, then P is a *simple cycle*.

An directed (undirected) graph is *acyclic*, when it has no simple cycle of length 2 (3) or more.

The *distance* from u to v is the length of a shortest path from u to v .

The *diameter* of a graph is the largest distance between any pair of nodes.

Connectivity

1. An undirected graph is *connected*, if there exists a path between each pair of nodes.
2. A directed graph is *strongly connected*, if there exists a path from each node to every other node.
3. A *connected component* of an undirected graph G is a maximal connected induced subgraph.
 - i.e. if you add any other node, the subgraph is no longer connected
4. A *strongly connected component* of a directed graph is a maximal strongly connected induced subgraph.
 - i.e. if you add any other node, the subgraph is no longer strongly connected

The *diameter* of a graph that is not connected is infinite (∞).

A *Hamilton cycle* is a simple cycle that contains all nodes.

Theorem. An undirected graph that has a Hamilton cycle is connected.

Regular graphs

An undirected graph is *regular*, when all its nodes have the same degree. If that degree is d , then the graph is called d -regular.

Fact. For all graphs

$$\sum_{0 < i < N} \deg(v_i) = 2|E|$$

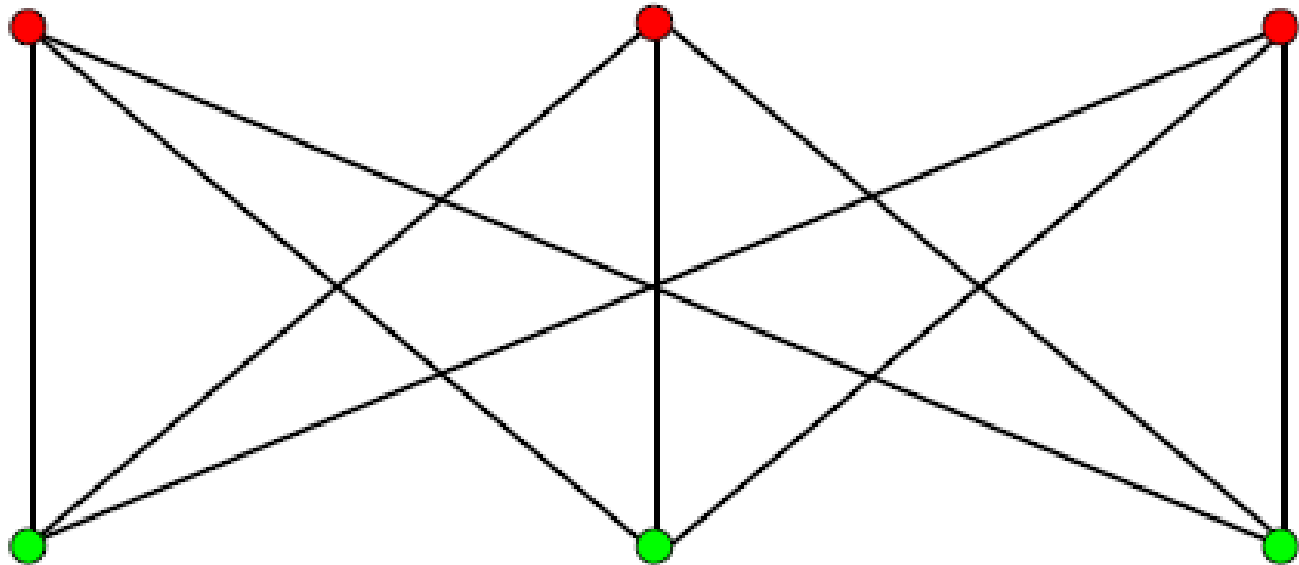
Fact.

For a d -regular graph we have $d|V| = 2|E|$

Bipartite graphs

An undirected graph is *bipartite* when there exist a partition $\{V_1, V_2\}$ of its nodes such that each edge has one endpoint in V_1 and the other in V_2 . A bipartite graph is called *complete* when $E = \{ \{v_1, v_2\} \mid v_1 \in V_1 \wedge v_2 \in V_2 \}$

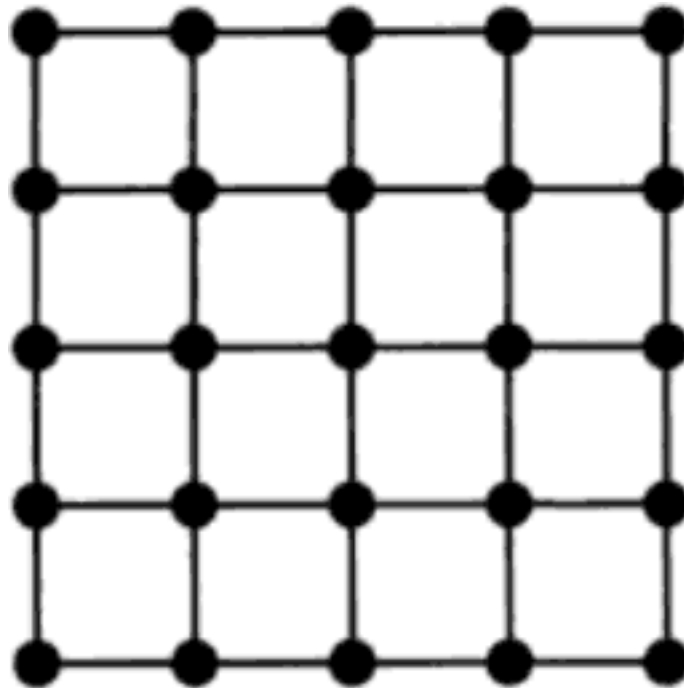
$K_{3,3}$



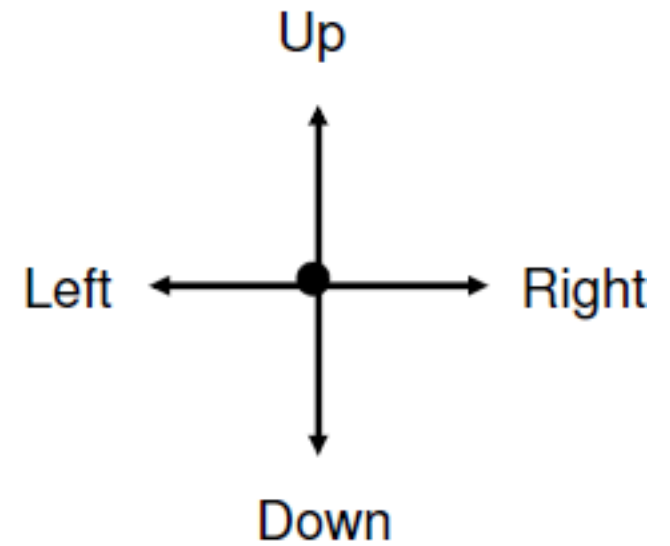
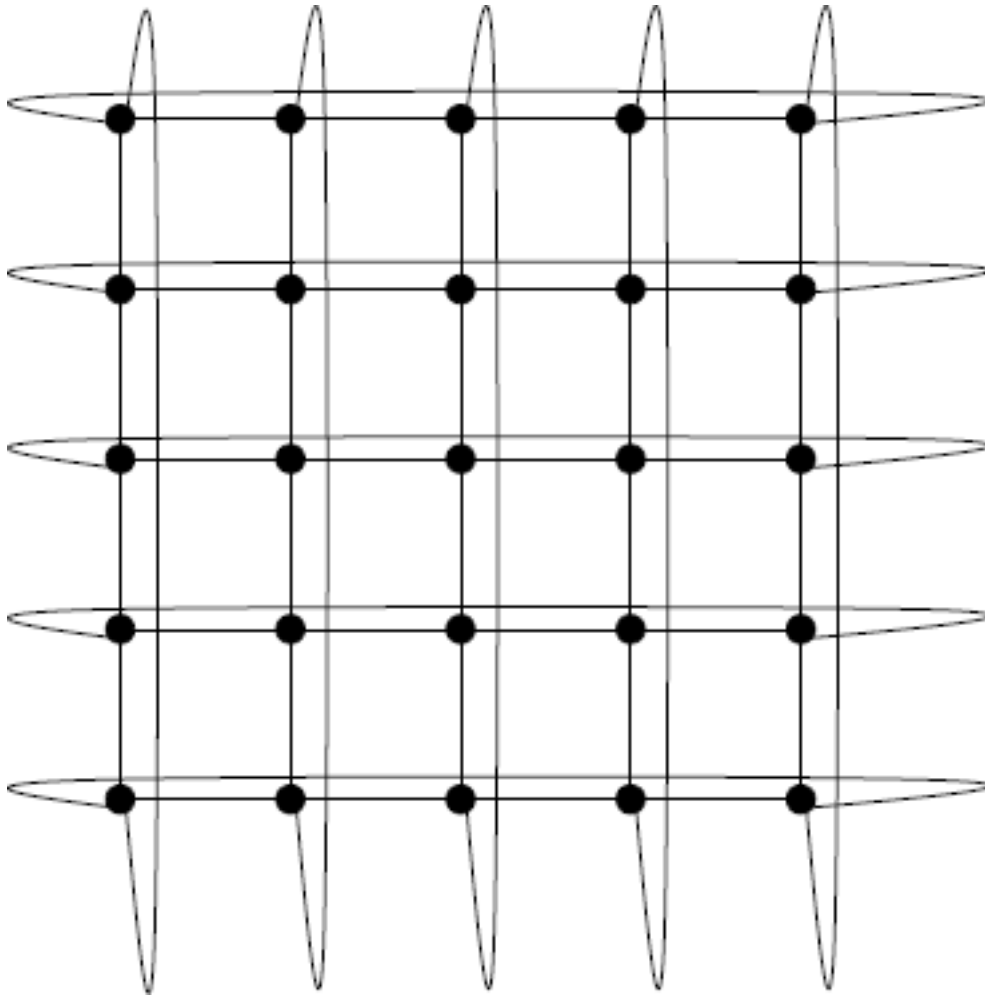
Frequently used graphs

1. Line , grid
2. Ring, torus
3. Star, Tree, Forest
4. Hypercube,
5. Clique

5x5 - Grid

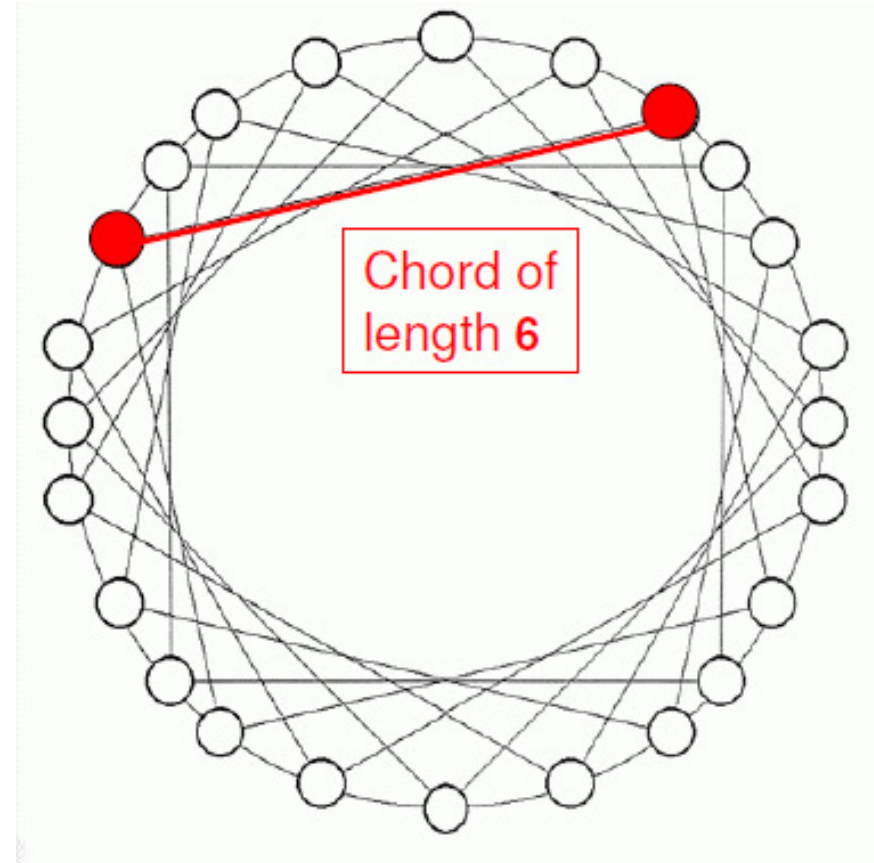


5x5 Torus



Chordal rings

For $0 \leq 2k \leq N$ a chordal ring of size N with chord length k is a ring of size N with additional edges between every pair of nodes with distance k .



Trees, forests and stars

A *tree* is an undirected, connected, acyclic graph.

A *forest* is an undirected, acyclic graph.

The following statements are equivalent:

1. G is tree
2. Between any pair of nodes there is a unique simple path
3. G is connected and $|E| = N-1$
4. G is acyclic and $|E| = N-1$

A *rooted tree* is a tree with a designated node r , the root.

If u is a node on the path from v to r , then

- u is *ancestor* of v ; *father* if $uv \in E$
- v is *descendant* of u ; *son* or *child* if $uv \in E$

A *star* is a rooted tree of depth one. The root of the star is called the center.

Spanning trees

Spanning tree: An spanning subgraph (V, E') $E' \subseteq E$ such that (V, E') is a tree.

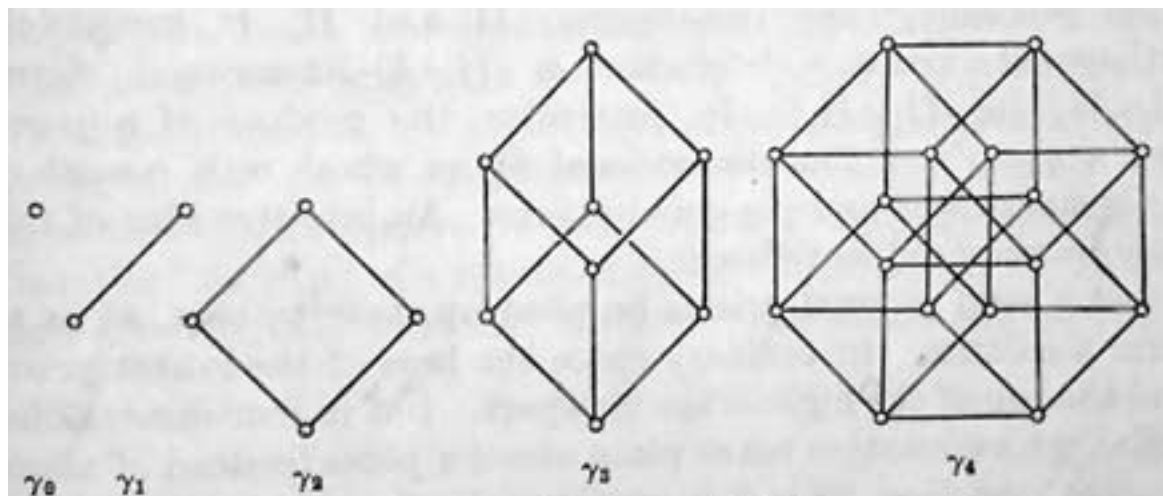
- Edges in $E' \rightarrow$ Tree Edges
- Edges not in E' ($E - E'$) \rightarrow Frond Edges

Thm. Every connected graph contains a spanning tree

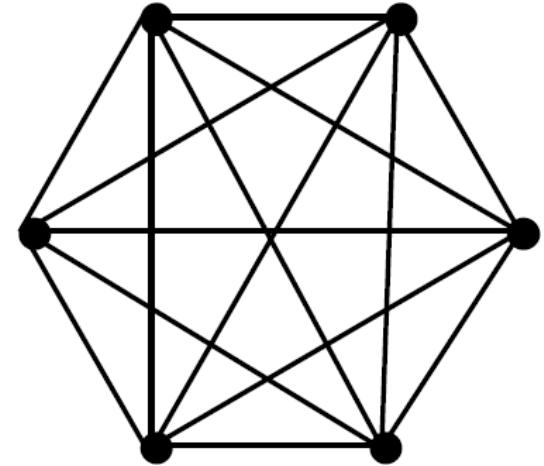
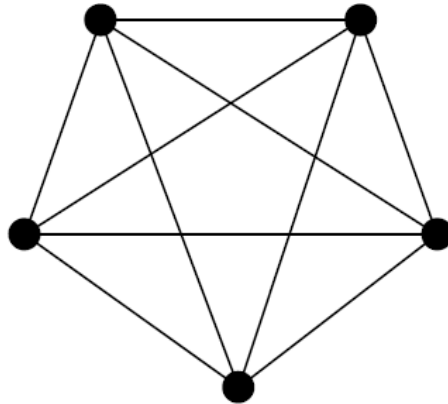
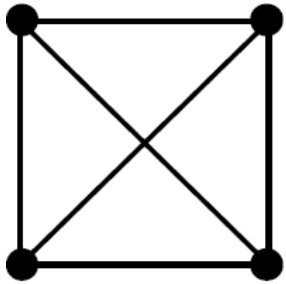
Important kinds of spanning trees:

- Low diameter spanning trees
 - Minimizes total computation time
- Minimal-weight spanning tree
 - Minimizes total communication cost
- Restricted-degree spanning tree
 - Minimizes maximum computation overhead per node
- Depth-first search tree
 - Useful structure to compute many properties of graphs
 - Every frond edge links a node and its descendent

Hypercube



Cliques



A clique or complete graph is a graph with diameter 1

The complete graph with n nodes is known as K_n

For a clique $|E| = \frac{1}{2} |V| (|V|-1)$

Properties of graph families

Graph $G=(V,E)$	Degree	$N= V $	$ E $	Diameter
Line (n)	1,2	n	n	n-1
Ring (n)	2	n	n	n/2
Grid (n x n)	2, 3, 4	n^2	$2n(n-1)$	$2(n-1)$
Torus (n x n)	4	n^2	$2n^2$	(odd n ? n-1 : n)
Hypercube (dim d)	d	2^d	$2^{d-1}d$	d
Tree (n)	varies	n	n-1	1..(n-1)
Forest (n)	varies	n	0..n-1	inf
Star (n)	1, n-1	n	n-1	(n=2 ? 1 : 2)
Clique (n)	n-1	n	$n(n-1)/2$	1

So far

- Graph Definitions
 - Common Graph Types and their properties
- Models
 - Why, which
- Transition systems
 - Executions

Computational models

Computation models are used to show

1. Feasibility
2. Correctness
3. Complexity

Computational models need to be

1. Concise
2. Precise
3. Tuned to the problem area

Some frequently used models are

1. Transition systems
2. I/O-automata
3. Actors

Distributed algorithm characteristics

- Lack of knowledge (existence) of a global state
 - Due to local connectivity, asynchronous mode of cooperation
- Lack of a global time frame
 - Also called asynchronous mode of cooperation
 - Synchronous mode of cooperation useful for theoretical purposes; can be enforced in practice, but at considerable cost
- Non-determinism
 - Differences in execution speed
 - Varying communication delays, caused by congestion, multipath routing, atmospheric conditions (for wireless links), mobility of processors
 - Extreme cases of the above are link or processor failures
- Inter-process communication via message passing: no shared variables

Synchrony

1. Mode of communication

- Asynchronous:
 - non-blocking send, blocking receive
 - deadlock, when both parties wait for input
- Synchronous:
 - first participant (sender or receiver) blocks until its counterpart is ready; simultaneous completion
 - deadlock, when parties disagree on order of communications

2. Mode of cooperation

- Synchronous network model
 - there is a notion of global time
 - processes operate in lockstep
 - reduction of number of communications

So far

- Graph Definitions
 - Common Graph Types and their properties
- Models
 - Why, which
- Transition systems
 - Executions

Labeled transition system

Labeled transition system

1. Configurations (system state)

- for each node a local (process) state
- virtual concept: when transitions take positive amount of time, then the system may never be in a stable configuration.

2. Labeled transitions

- represent the state changes caused by the execution of an event by one of the nodes of the system.
- event is added as a label to the transition
 - Communications
 - computations

Transition systems

A *transition system* is a triple (C, \rightarrow, I) where

1. C is a set of *configurations*
2. $\rightarrow \subseteq C \times C$ is a binary *transition relation*
 1. We write $\gamma \rightarrow \delta$ instead of $(\gamma, \delta) \in \rightarrow$
 2. We write \rightsquigarrow for the reflexive transitive closure of \rightarrow
3. $I \subseteq C$ is the subset of *initial configurations*

A configuration γ is *terminal*, when there is no δ such that $\gamma \rightarrow \delta$. A configuration δ is *reachable* from γ when $\gamma \rightsquigarrow \delta$.

Executions

Let $S = (C, \rightarrow, I)$ be a transition system. Let C^* denote the set of finite sequences of configurations. Let C^ω denote the set of infinite sequences of configurations.

A sequence $s \in C^* \cup C^\omega$ is *maximal*, when it is infinite, or when it is finite and ends in a configuration γ for which there is no configuration δ such that $\gamma \rightarrow \delta$.

An *execution* E of S is a maximal sequence $(\gamma_0, \gamma_1, \gamma_2, \dots)$ such that $\gamma_0 \in I$ and $\gamma_i \rightarrow \gamma_{i+1}$, for all $0 \leq i$.

Conclusion

- Graph Definitions
 - Common Graph Types and their properties
- Models
 - Why, which
- Transition systems
 - Executions