Graphs II

Discrete Mathematics

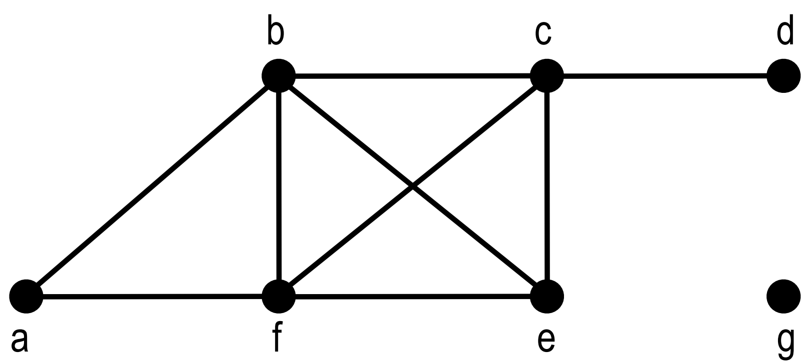
Evgeny Skvortsov

Previous Lecture

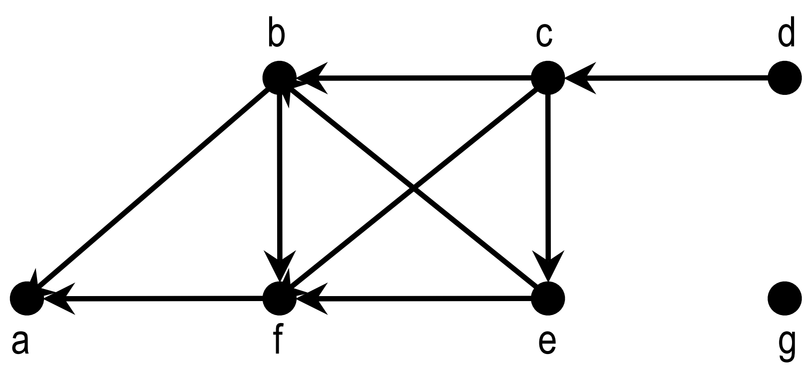
- Graphs, directed and undirected
- Adjacency, incidence, endpoints
- Degree of vertices, isolated and pendant vertices
- Walks, trails, and paths

Representation of Graphs: Lists of Vertices and Edges

- Lists of vertices and edges



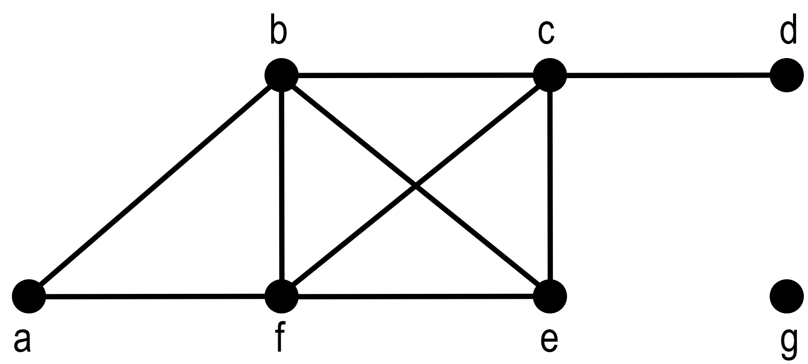
{a, b, c, d, e, f, e, g}
{ ab, bc, af, bf, fc, be, ce, cd }



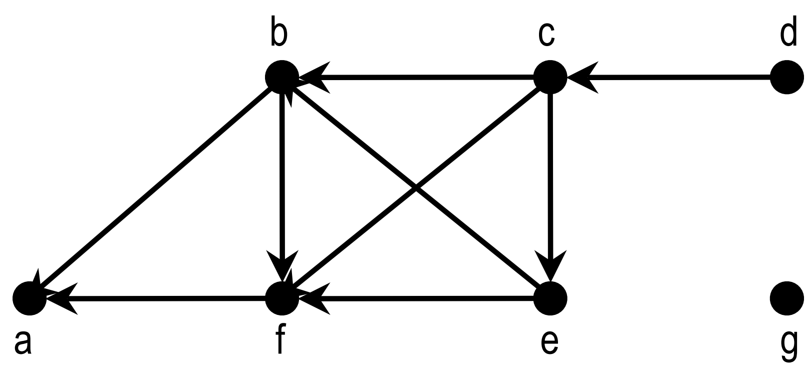
{a, b, c, d, e, f, e, g}
{ (b,a), (c,b), (f,a), (b,f), (c,f), (e,b), (c,e), (d,c) }

Representation of Graphs: Adjacency Lists

- Adjacency list is a table containing for each vertex a lists of vertices adjacent to it.



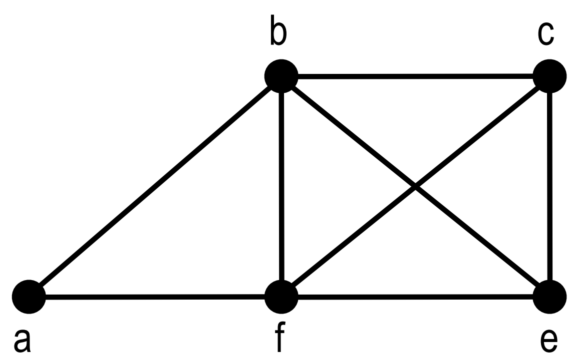
vert.	list	vert.	list
a	b, f	e	c, b, f
b	a, f, e, c	f	a, b, c, e
c	b, d, e, f	g	
d	c		



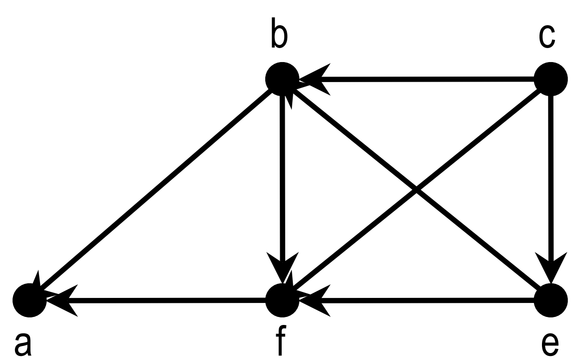
vert.	list	vert.	list
a		e	b, f
b	a, f	f	a
c	b, e, f	g	
d	c		

Representation of Graphs: Adjacency Matrix

- The adjacency matrix of a graph is a 0-1-matrix whose rows and columns are labeled by the vertices of the graph. The entry in ith row and jth column is 1 if there is an edge from vertex i to vertex j. Otherwise it is 0.



$$\begin{array}{c c} & \begin{array}{ccccc} a & b & c & e & f \end{array} \\ \begin{array}{c} a \\ b \\ c \\ e \\ f \end{array} & \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix} \end{array}$$



$$\begin{array}{c c} & \begin{array}{ccccc} a & b & c & e & f \end{array} \\ \begin{array}{c} a \\ b \\ c \\ e \\ f \end{array} & \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

Representation of Graphs: Adjacency Matrix (cntd)

- Draw a graph with the adjacency matrices

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

- The adjacency matrix of a multigraph shows how many edges connect every pair of vertices.
- Draw a multigraph with the adjacency matrix

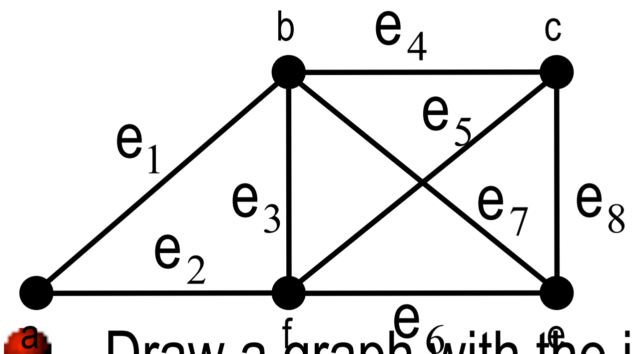
$$\begin{pmatrix} 0 & 3 & 0 & 2 & 1 \\ 3 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 2 & 2 \\ 2 & 1 & 2 & 0 & 3 \\ 2 & 0 & 2 & 3 & 0 \end{pmatrix}$$

Adjacency List vs. Adjacency Matrix

- Which method is better?
- It depends on a graph and the problem we are going to solve.
- If a graph is sparse, that is, has few edges (usually this means O(number of vertices)), then the adjacency list is more concise.
- If a graph is dense, that is, contains many edges, then adjacency matrix is more concise.
- Some problems and algorithms may require particular type of representation.
- In the problem CONNECTIVITY the objective is to find the connected components of a given graph. This problem can be solved very efficiently using a bunch of parallel processors. For this we enhance the connectivity of a graph by multiplying its adjacency matrix.

Representation of Graphs: Incidence Matrix

- The incidence matrix of a graph G = (V, E) is a 0-1-matrix whose rows are labeled by vertices of G and columns are labeled by edges of G. The entry in ith row and jth column is 1 if the vertex i is incident to the edge j.



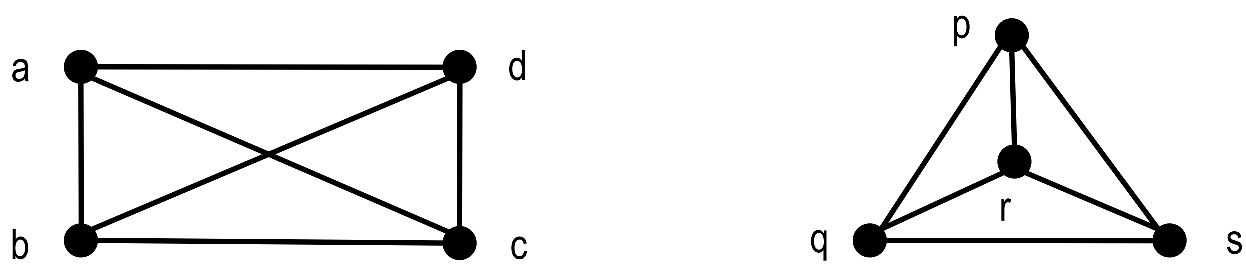
$$\begin{array}{c} \\ a \\ b \\ c \\ e \\ f \end{array} \begin{pmatrix} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

- Draw a graph with the incidence matrix

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Isomorphism of Graphs

- We often need to know whether it is possible to draw two graphs in the same way
- The simple graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic if there is a one-to-one and onto function f from V_1 to V_2 with the property that vertices a and b are adjacent in G_1 if and only if $f(a)$ and $f(b)$ are adjacent in G_2, for all a and b in V_1. Such a function f is called an isomorphism.



x	a	b	c	d
f(x)	r	q	s	p

Graph Invariants

- In general, it is very hard to decide whether two graphs are isomorphic or not.
- However, sometimes we can easily see that two graphs are not isomorphic. In particular, we can show that two graphs are not isomorphic if we find a property only one of the two graphs has, but that is preserved by isomorphism.
- A property preserved by isomorphism of graphs is called a graph invariant.
- Example invariants:

  number of vertices

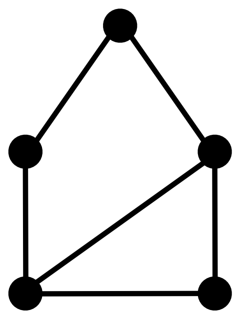
  number of edges

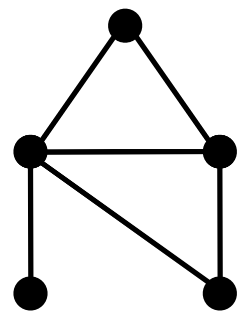
  degrees of vertices

  the length of paths and cycles

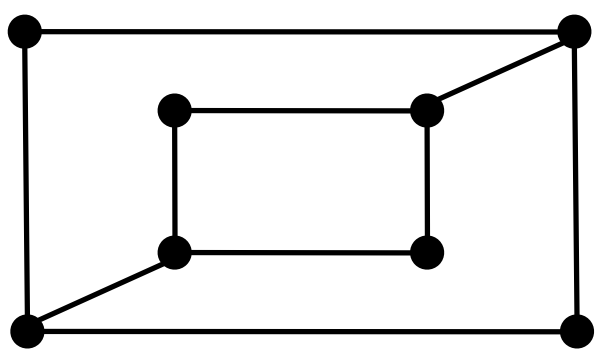
Graph Invariants (cntd)

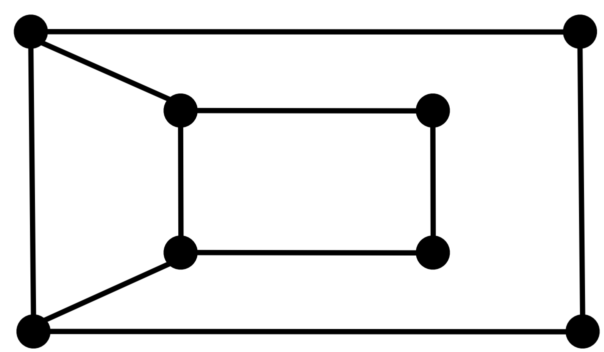
- Are these graphs isomorphic?





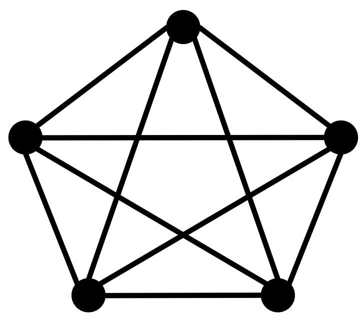
- What about these?



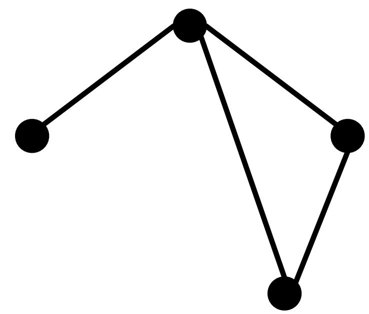


Subgraphs

- Sometimes we need only a part of a big graph.
- A subgraph of a graph G = (V,E) is a graph H = (W,F), where $W \subseteq V$ and $F \subseteq E$.



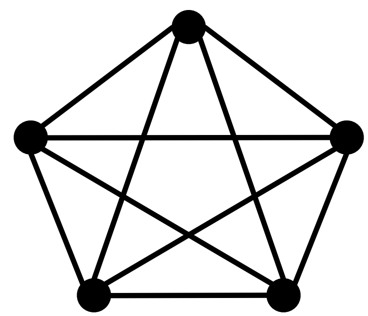
G



H

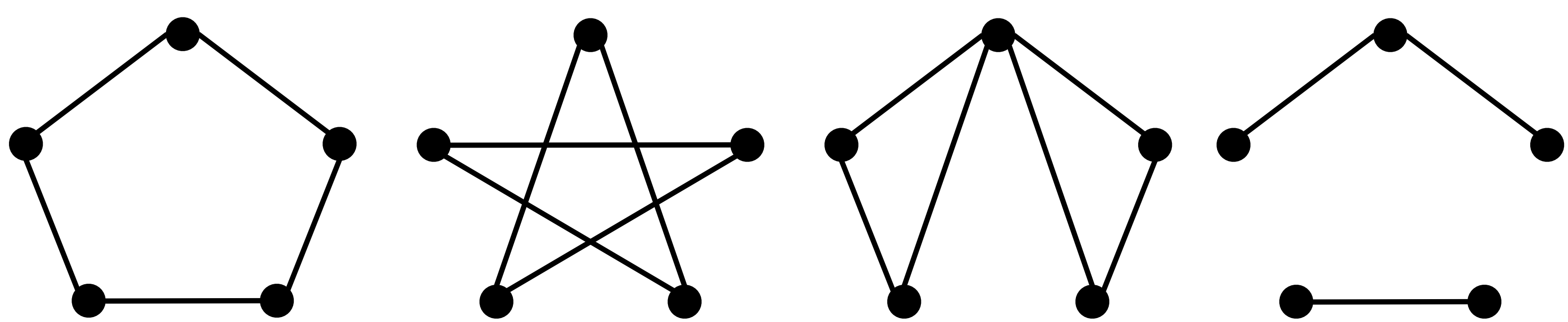
Subgraphs (cntd)

- If H = (W,F) is a subgraph of G = (V,E) and W = V then H is called a spanning subgraph



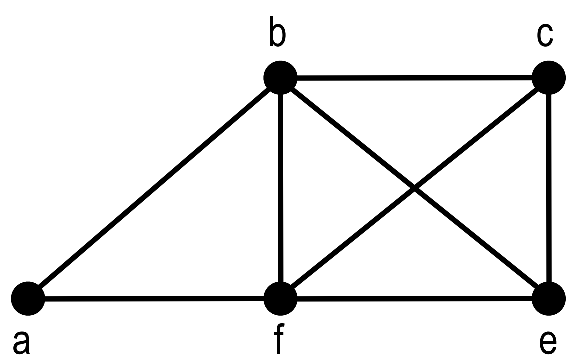
G

Spanning subgraphs:

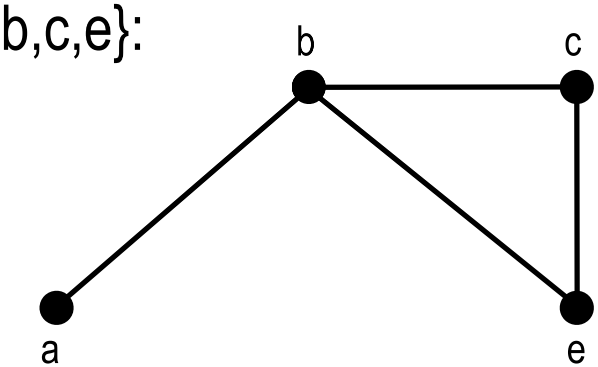


Subgraphs (cntd)

- Let $G = (V,E)$ be a graph and $W \subseteq V$ a nonempty set of vertices. The subgraph of G induced by W is the subgraph whose vertex set is W and which contains all edges (from G) whose endpoints are in W.
- A subgraph H of a graph $G = (V,E)$ is called an induced subgraph if there is $W \subseteq V$ such that H is the subgraph unduced by W.

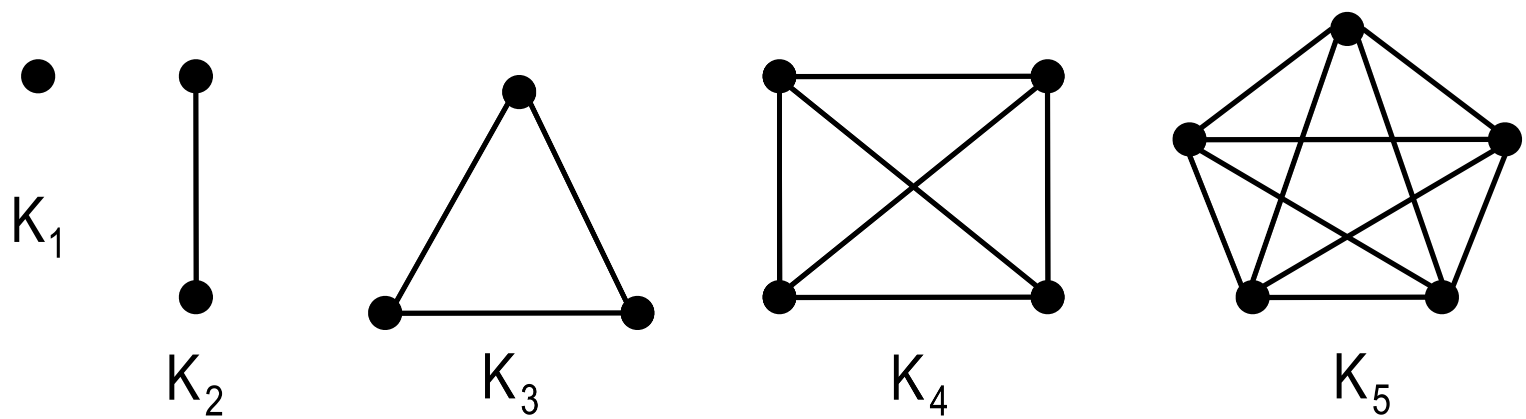


$W = \{a,b,c,e\}$:



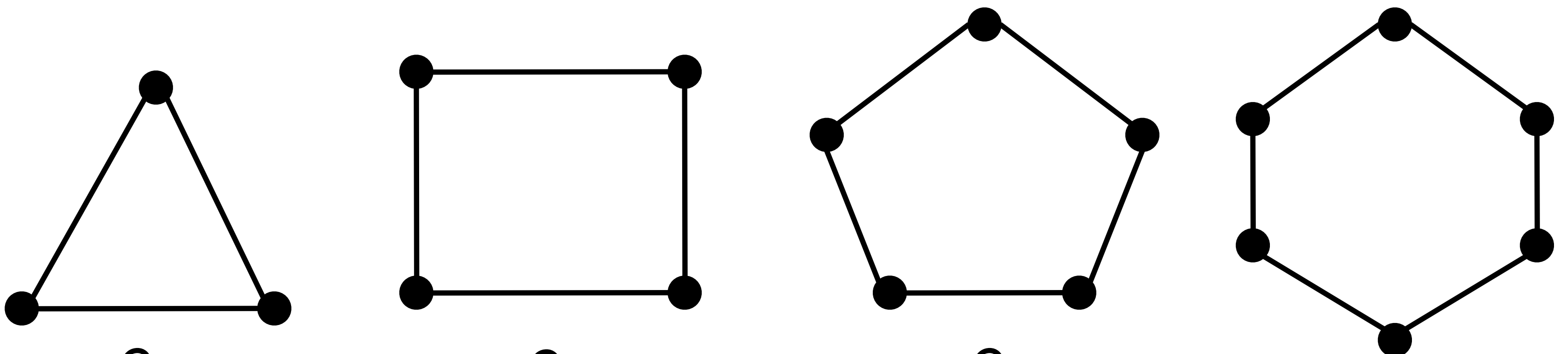
Some Special Graphs

- Complete Graphs. The complete graph on n vertices, denoted by K_n, is the simple graph that contains exactly one edge between each pair of distinct vertices



Some Special Graphs (cntd)

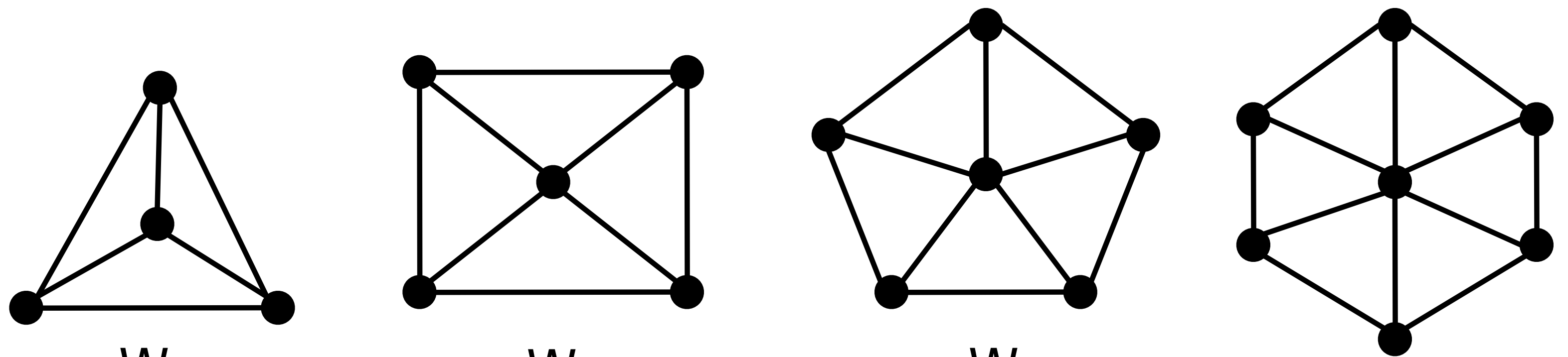
- Cycles. The cycle C_n, $n \geq 3$, consists of n vertices $v_1, v_2, \ldots, v_n$ and edges $\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-1}, v_n\}$, and $\{v_n, v_1\}$.



C_3 C_4 C_5 C_6

Some Special Graphs (cntd)

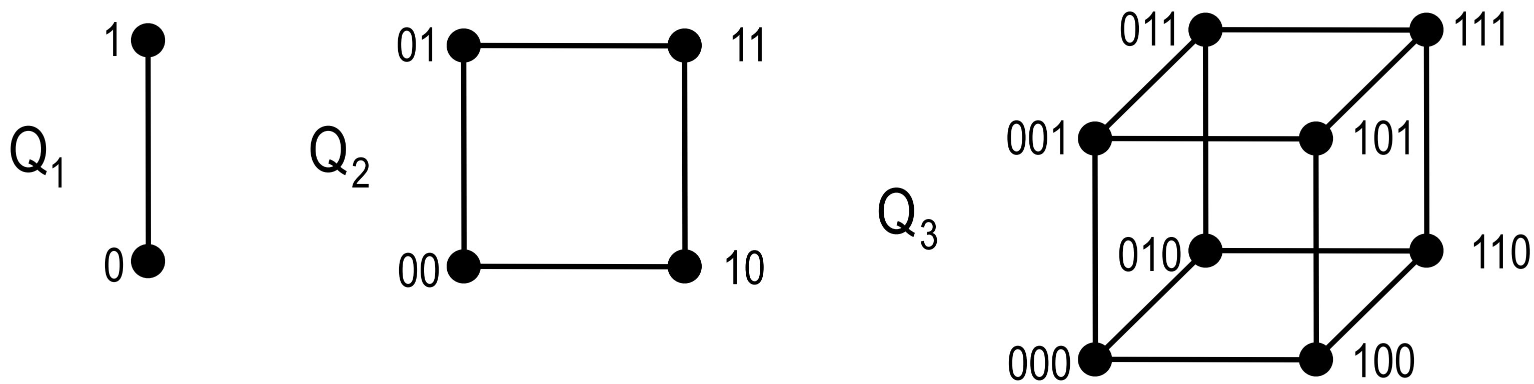
- Wheels. The wheel, W_n, is obtained from the cycle C_n when we add an additional vertex to it and connect this vertex to each of the n vertices in C_n by new edges



W_3 W_4 W_5 W_6

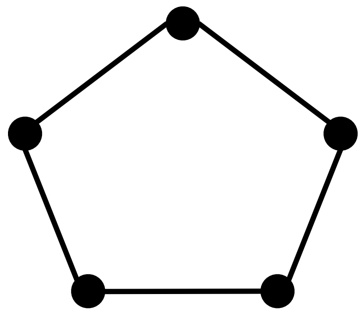
Some Special Graphs (cntd)

- Hypercubes. The n-dimensional hypercube, or n-cube, denoted by Q_n, is the graph that has vertices representing the bit strings of length n. Two vertices are adjacent if and only if the bit strings that they represent differ in exactly one bit position.

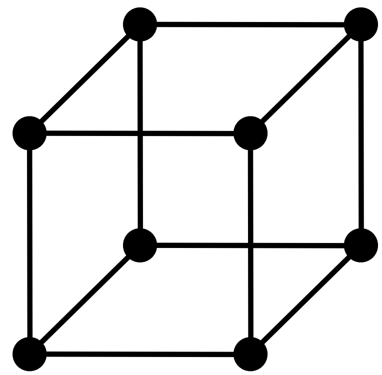


Regular Graphs

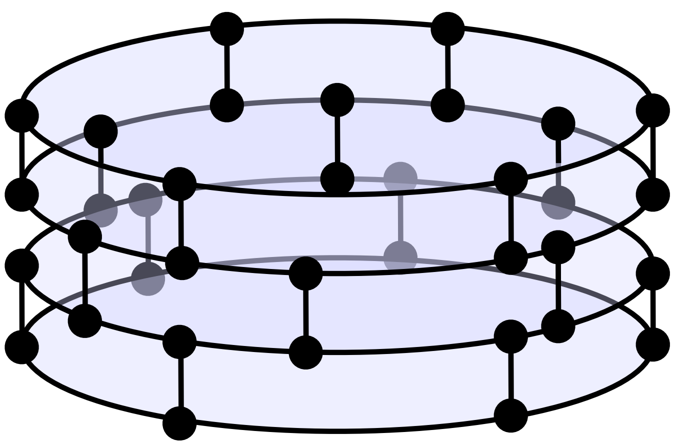
- A graph is called k-regular if all its vertices have degree k.



2-regular graphs are cycles



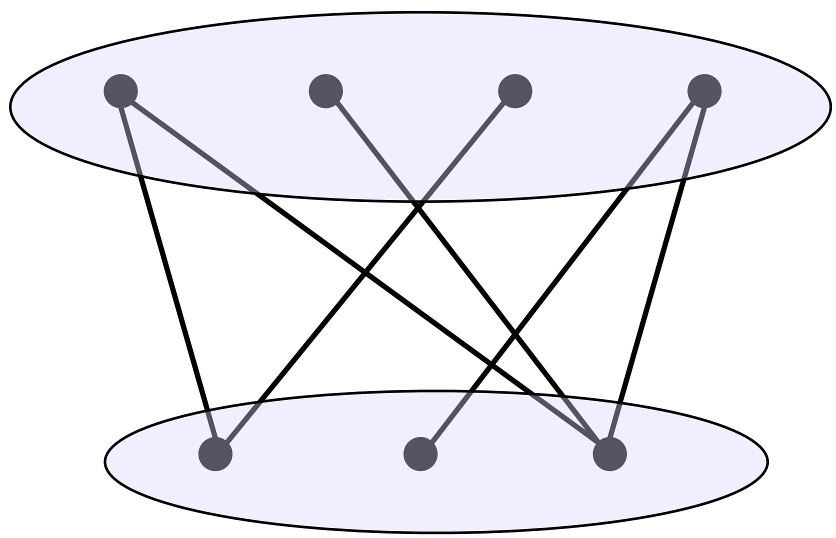
A k-cube is a k-regular graph



A brick graph, another example of a 3-regular graph

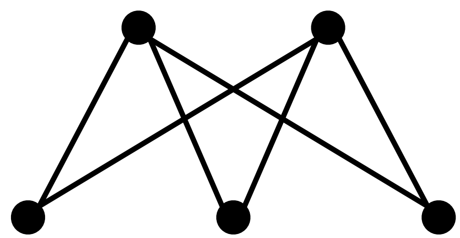
Bipartite Graphs

- A simple graph G is bipartite if its vertex set V can be partitioned into two disjoint sets V_1 and V_2 such that every edge in the graph connects a vertex in V_1 and a vertex in V_2.
  When this condition holds, we call the pair V_1, V_2 a bipartition of the vertex set V of G

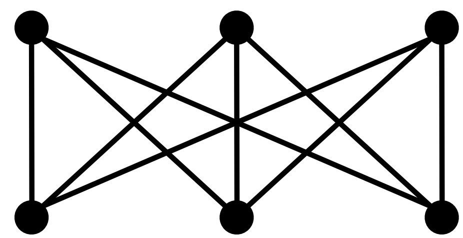


Complete Bipartite Graphs

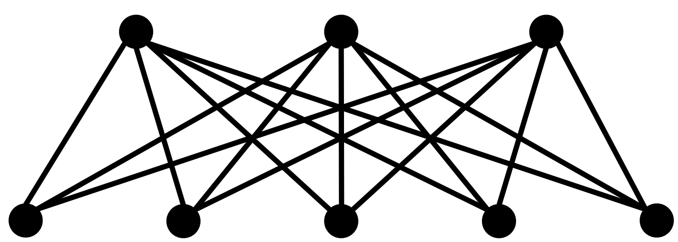
- The complete bipartite graph $K_{m,n}$ is the graph that has its vertex set partitioned into two subsets of m and n vertices, respectively. There is an edge between two vertices if and only if one vertex is in the first subset and the other vertex is in the second subset.



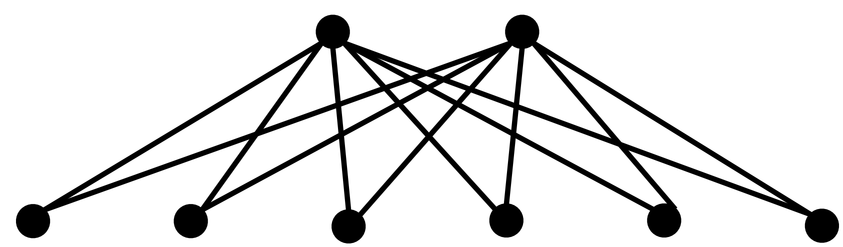
$K_{2,3}$



$K_{3,3}$



$K_{3,5}$



$K_{2,6}$

More Bipartite Graphs

- **Theorem.**
  For a simple graph $G = (V,E)$ the following conditions are equivalent:
  (1) G is bipartite,
  (2) G does not contain a cycle of odd length,
  (3) it is possible to color each vertex of G in one of two different colors so that no two adjacent vertices are colored the same color.

- Proof:
  We prove implications (1) $\rightarrow$ (2), (2) $\rightarrow$ (3), and (3) $\rightarrow$ (1).

More Bipartite Graphs (cntd)

- (1) $\to$ (2)

  Suppose that G is bipartite, and U, W is a bipartition.

  Let $u = u_0, e_1, u_1, \ldots, e_{n-1}, u_{n-1}, e_n, u_n = u$ be a cycle. Without loss of generality we may assume that $u \in U$.

  Then $u_1 \in W, u_2 \in U, u_3 \in W, \ldots$ and so on. Since $u_n = u \in U$, n is even.

More Bipartite Graphs (cntd)

- (2) → (3)

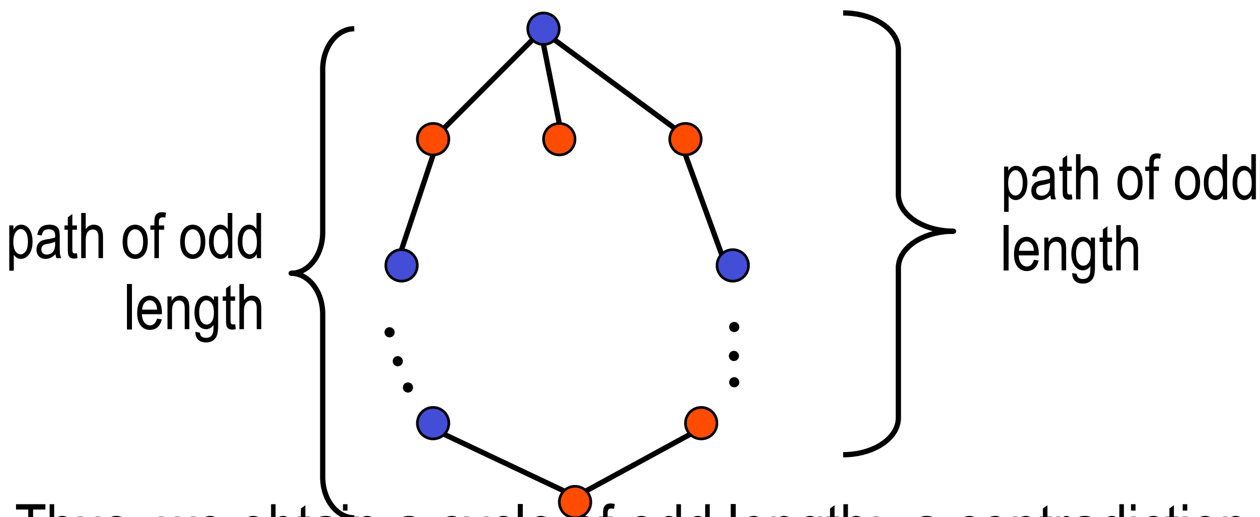
  Suppose that every cycle in G has even length.

  We show that the following algorithm constructs a proper coloring of G in two colors

take any vertex and **color** it blue
repeat the following **while** there is an uncolored vertex
  take any vertex v that is already colored
  if v is colored blue **then color** all its neighbors red
  if v is colored red **then color** all its neighbors blue
end repeat

More Bipartite Graphs (cntd)

- To prove the correctness of this algorithm, we have to show that it never attempts to color red a vertex already colored blue, and that it never attempts to color blue a vertex already colored red
- Suppose that the algorithm attempts to color red a vertex previously colored red



- Thus, we obtain a cycle of odd length; a contradiction.

More Bipartite Graphs (cntd)

- (3) $\to$ (1)

  Suppose that the vertices of the graph G can be colored in two colors, red and blue, such that no two adjacent vertices receives the same color.

  We construct a bipartition by including all blue vertices into one part of the bipartition, and all red vertices into the other part of the bipartition.

  Q.E.D.

Homework

Exercises from the Book:

No. 1a, 2b, 3ab, 9, (page 529)

No. 3, 33 (page 539)

- Draw all simple non-isomorphic graphs with 5 or fewer vertices