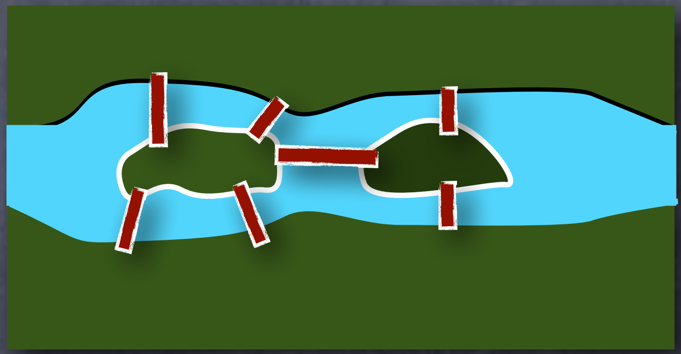
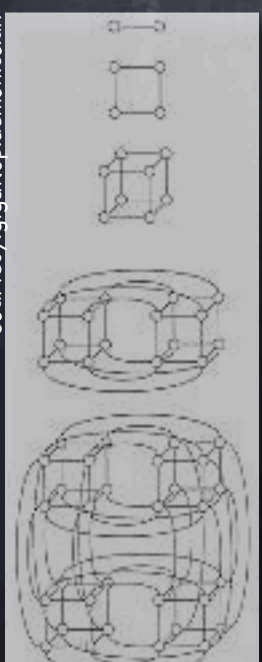
Graphs

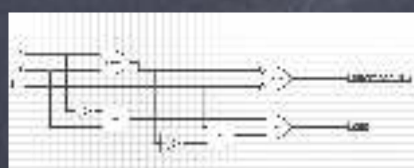


Graphs

- What is "connected" to what
- Many things we deal with in computer science are graphs
  - Networks: humans, communication, computation, transportation, knowledge



Courtesy:gigaflop.demon.co.uk



Courtesy: Microsoft Academic Search

Courtesy: cablemap.info

Courtesy: New Scientist

Courtesy: Digital Humanities Specialist @ Stanford

Graphs in action

Many Applications

- Often want to design graphs with "good properties"
  - Connecting processors in a super-computer
  - Data structures (e.g., "trees") to keep data in an easy-to-search/manipulate fashion
    - Typically want graphs with few connections (i.e., edges), but good "connectivity" -- i.e., (possibly many) short paths between any two nodes
- Very efficient algorithms known for relevant graph problems
  - e.g., breadth/depth-first search, shortest path algorithm...
- But many other graph problems are known to be "NP-hard"
  - e.g., Traveling Salesperson Problem (TSP): visit all cities, by traveling the least distance

Simple Graphs

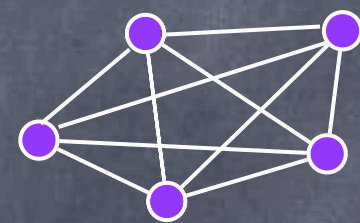
- A simple graph $G = (V,E)$, where
  - $E \subseteq \{ \{a,b\} \mid a,b \in V, a \neq b \}$
- V is the set of nodes, E the set of edges
- V non-empty and finite (for us)
- Note: the "drawing" is not part of the graph, only the connectivity is

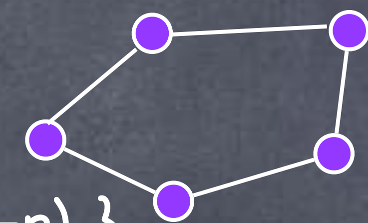
Simple Graphs

- Recall graphs for relations: directed graphs with self-loops
  - Each element in the domain forms a node
  - Each ordered pair (a,b) in the relation forms an edge
    - Edges of the form (a,a) are "self-loops"
- A simple graph is essentially a symmetric, irreflexive relation
  - Symmetric: An undirected edge {a,b} can be modelled as two directed edges (a,b) and (b,a)
  - Irreflexive: No self-loops
- In a "non-simple" graph, can allow more than one edge between any pair (multigraphs), or more generally, allow weights on edges (weighted graphs)

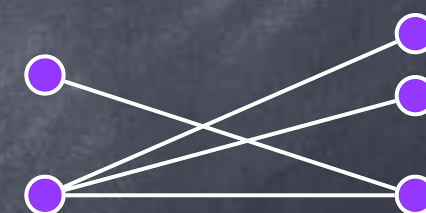
Examples

- Complete graph K_n : n nodes, with all possible edges between them
  - $E = \{ \{a,b\} \mid a,b \in V, a \neq b \}$
  - # edges, $|E| = n(n-1)/2$

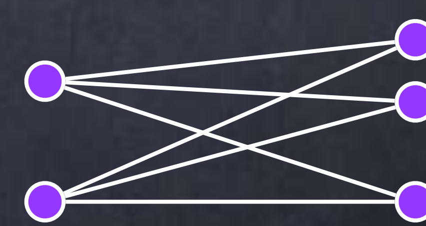




- Cycle C_n : $V = \{ v_1,\ldots,v_n \}$, $E = \{ \{v_i,v_j\} \mid j=i+1 \text{ or } (i=1 \text{ and } j=n) \}$
- Bipartite graph : $V = V_1 \cup V_2$, where $V_1 \cap V_2 = \emptyset$ (i.e., a partition), and no edge between two nodes in the same "part": $E \subseteq \{ \{a,b\} \mid a \in V_1, b \in V_2 \}$
  - e.g., C_n where n is even



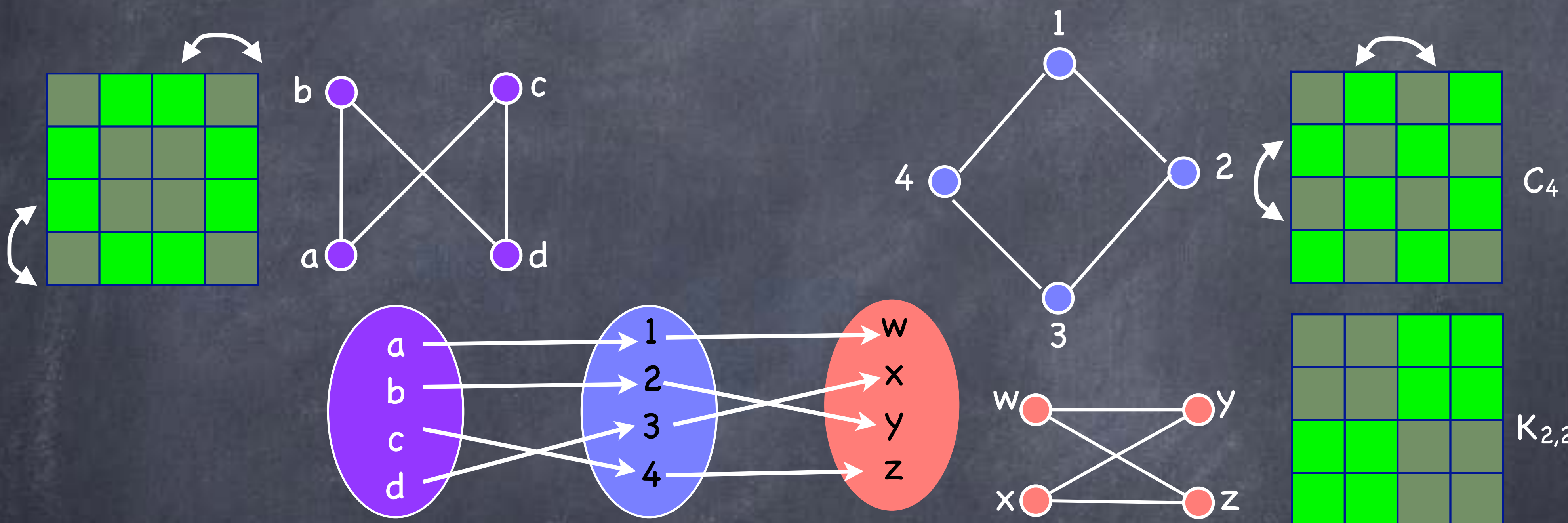
- Complete bipartite graph $K_{n1,n2}$: Bipartite graph, with $|V_1|=n_1$, $|V_2|=n_2$ and $E = \{ \{a,b\} \mid a \in V_1, b \in V_2 \}$
  - # edges, $|E| = n_1 \cdot n_2$



- Later: Hypercube, Trees

Graph Isomorphism

- $G_1 = (V_1,E_1)$ and $G_2 = (V_2,E_2)$ are isomorphic if there is a bijection $f:V_1 \rightarrow V_2$ such that $\{u,v\} \in E_1$ iff $\{f(u),f(v)\} \in E_2$



- Computational problem: check if two graphs (given as adjacency matrices) are isomorphic
  - Can rule out if certain "invariants" are not preserved (e.g. |V|,|E|)
- In general, no "efficient" algorithm known, when graph is large
  - Some believe no efficient algorithm exists!

Subgraphs

- A subgraph of G = (V,E) is a graph G′ = (V′,E′) such that V′ ⊆ V and E′ ⊆ E

- To get a subgraph: Remove zero or more vertices along with the edges incident on them, and further remove zero or more edges
    - Induced subgraph: omit the last step