Programming Abstractions

CS106B

Cynthia Lee

Upcoming Topics

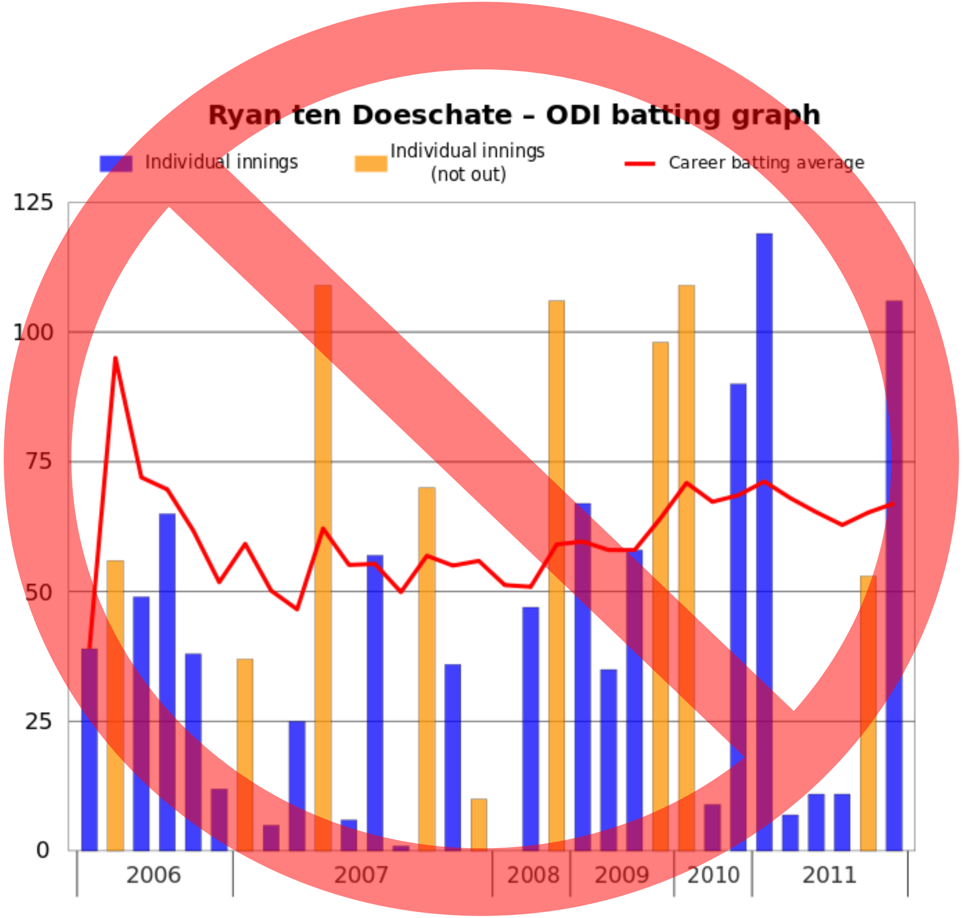
Graphs!

1. **Basics**
   - What are they? How do we represent them?
2. **Theorems**
   - What are some things we can prove about graphs?
3. **Breadth-first search** on a graph
   - Spoiler: just a very, very small change to tree version
4. **Dijkstra’s shortest paths algorithm**
   - Spoiler: just a very, very small change to BFS
5. **A* shortest pathsalgorithm**
   - Spoiler: just a very, very small change to Dijkstra’s
6. **Minimum Spanning Tree**
   - Kruskal’s algorithm

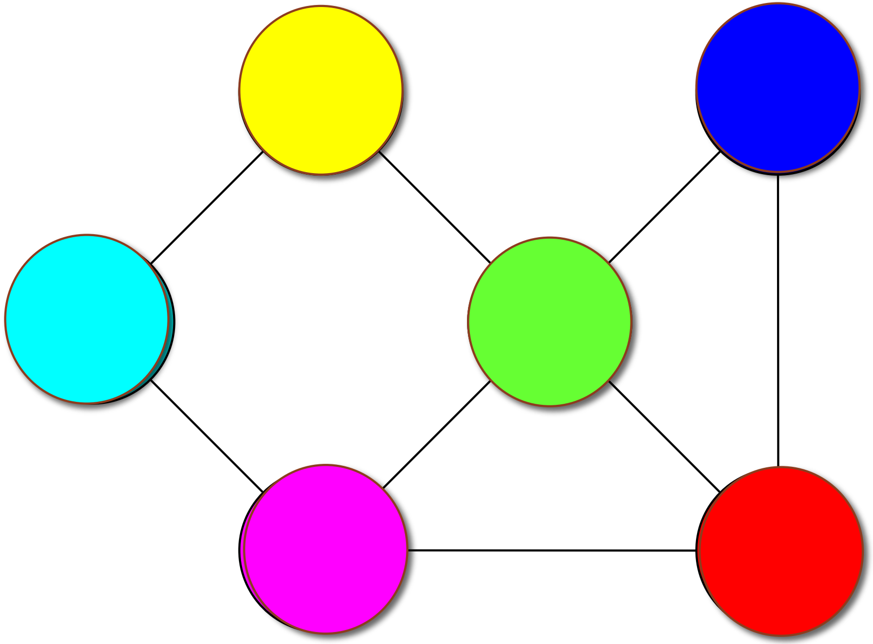
Graphs

What are graphs? What are they good for?

Graph


Ryan ten Doeschate – ODI batting graph
Individual innings
Individual innings (not out)
Career batting average
125
100
75
50
25
0
2006
2007
2008
2009
2010
2011

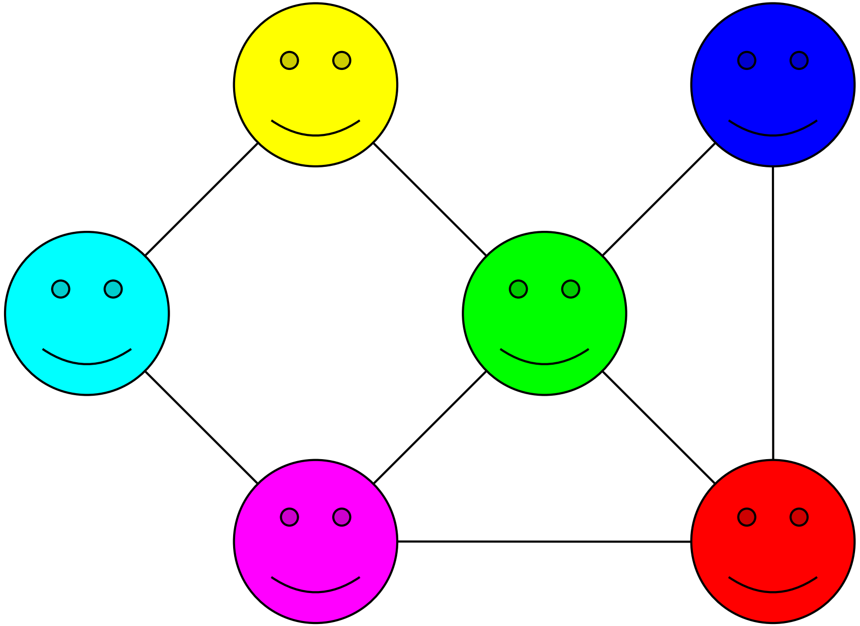
Graphs in Computer Science

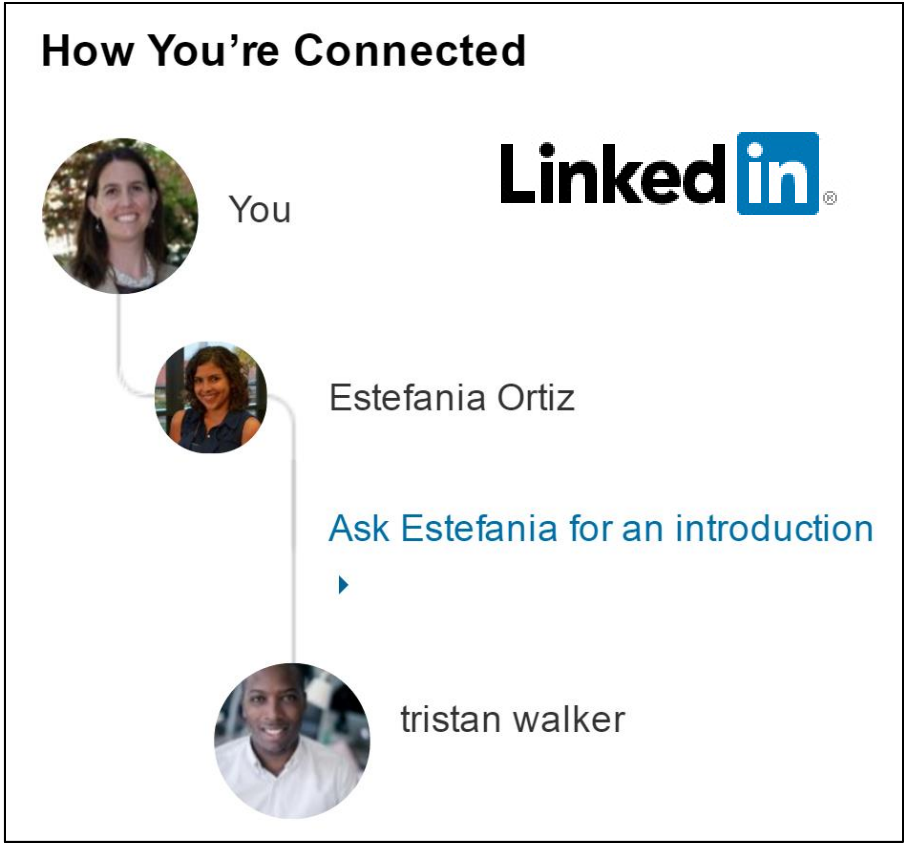


A graph is a mathematical structure for representing relationships

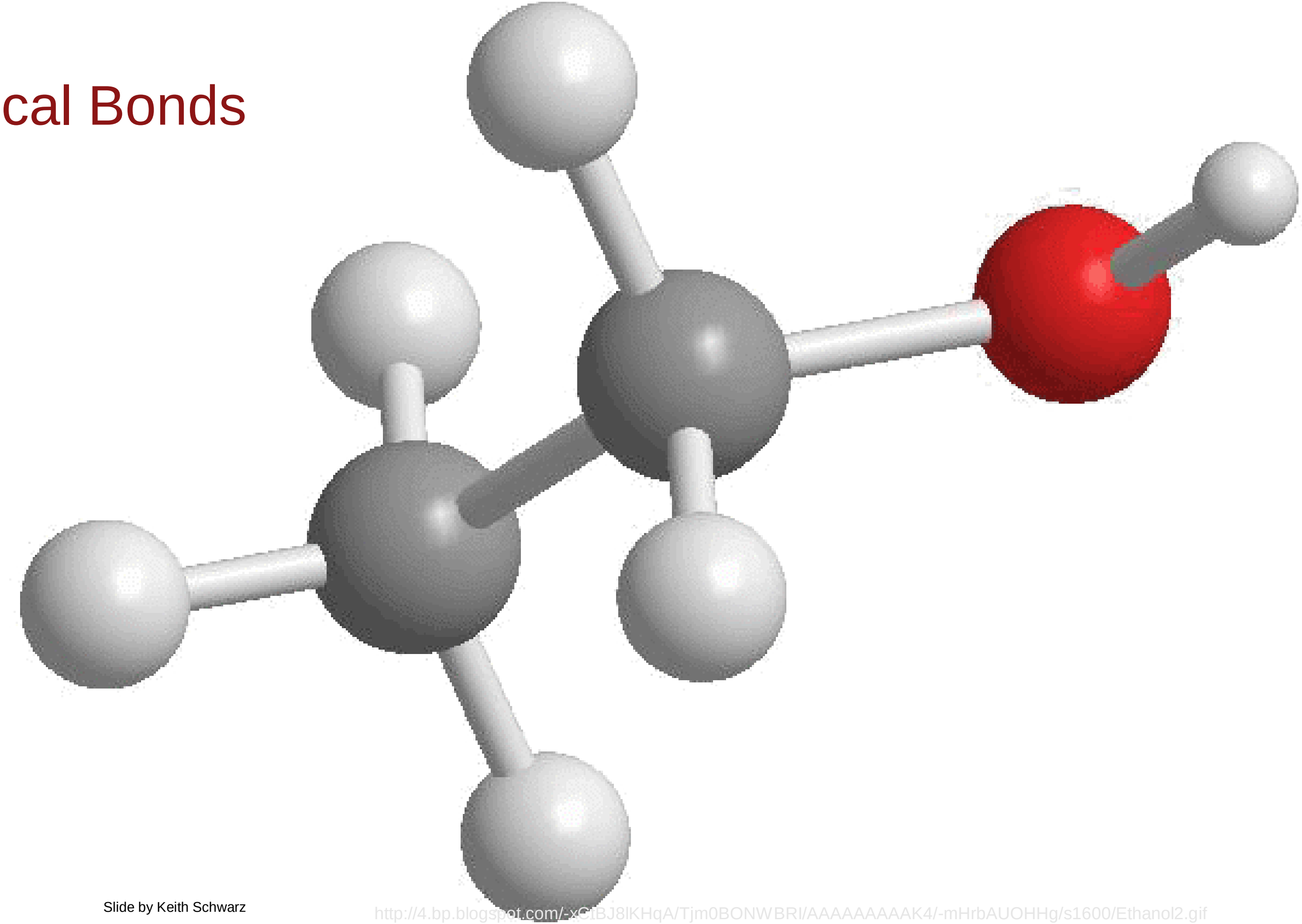
- A set V of **vertices** (or *nodes*)
- A set E of **edges** (or *arcs*) connecting a pair of vertices

A Social Network



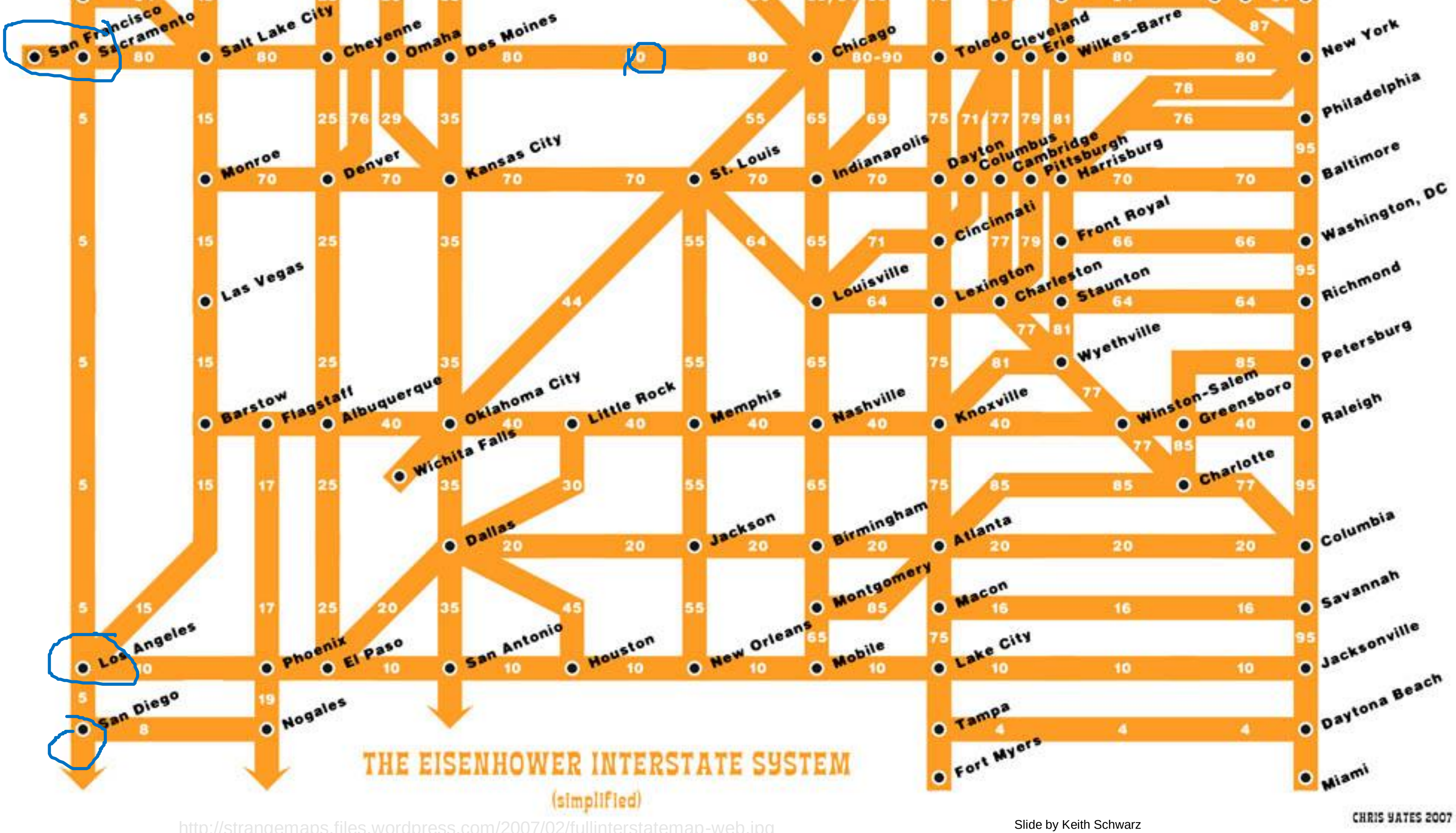


Chemical Bonds



Slide by Keith Schwarz

http://4.bp.blogspot.com/-xCtBJ8lKHqA/Tjm0BONWBRI/AAAAAAAAAK4/-mHrbAUOHHg/s1600/Ethanol2.gif



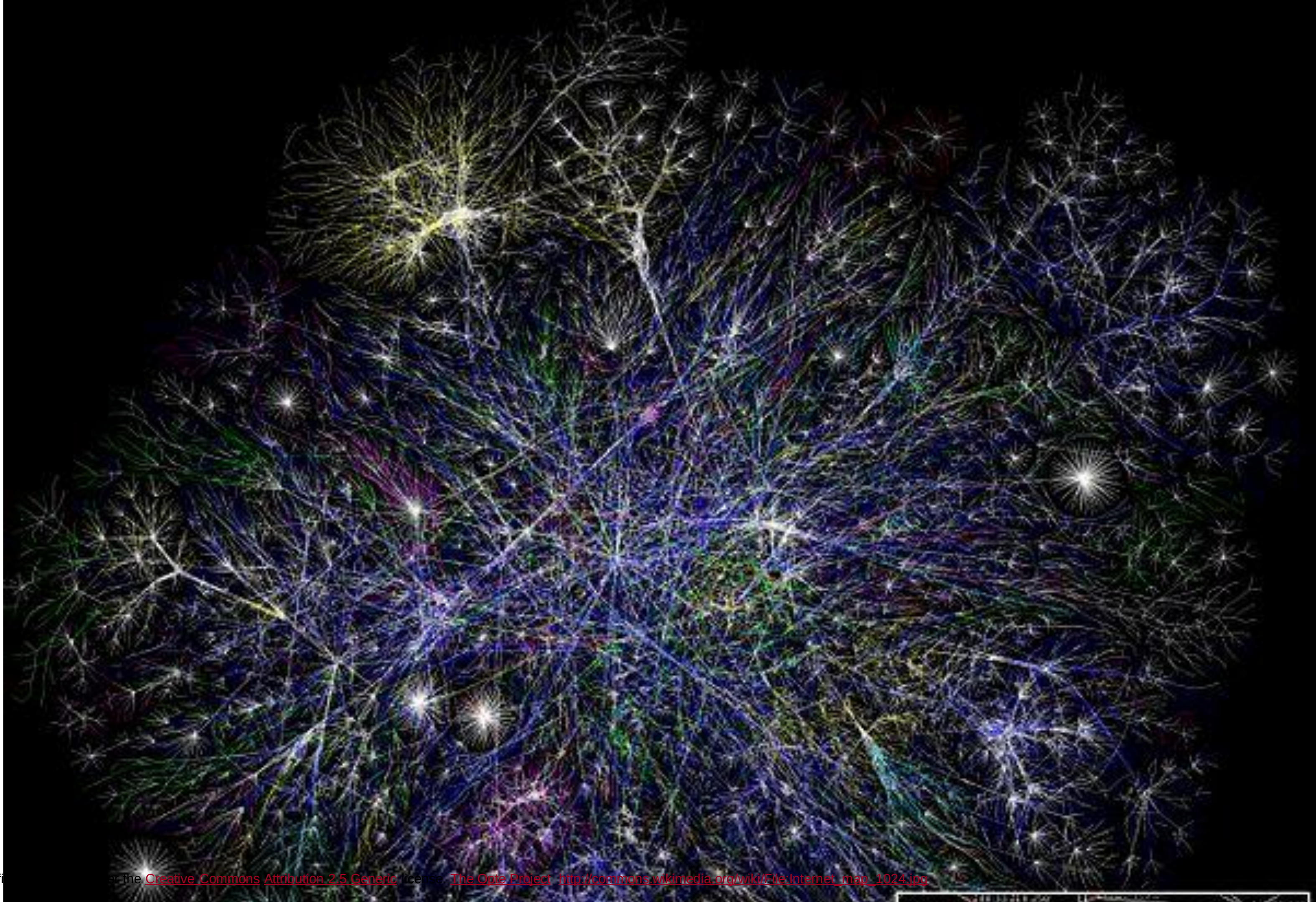
THE EISENHOWER INTERSTATE SYSTEM

(simplified)

http://strangemaps.files.wordpress.com/2007/02/fullinterstatemap-web.jpg

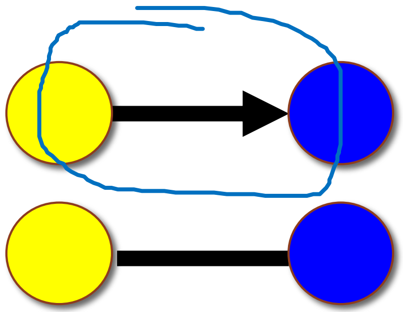
Slide by Keith Schwarz

Internet



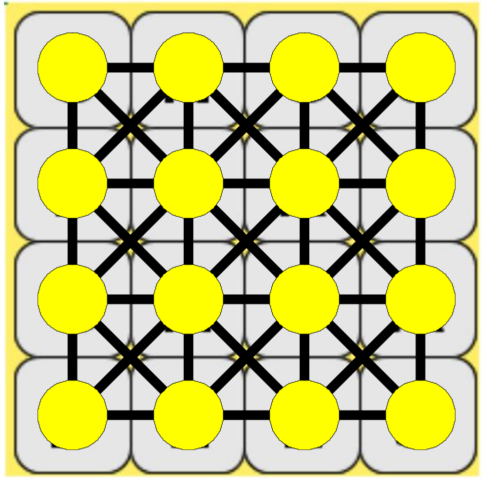
A graph is a mathematical structure for representing relationships

- A set V of **vertices** (or *nodes*)
  - Often have an associated label
- A set E of **edges** (or *arcs*) connecting a pair of vertices
  - Often have an associated cost or weight
- A graph may be **directed** (an edge from A to B only allow you to go from A to B, not B to A)
- or **undirected** (an edge between A and B allows travel in both directions)
- We talk about the number of vertices or edges as the *size of the set*, using the set theory notation for size: **|V|** and **|E|**



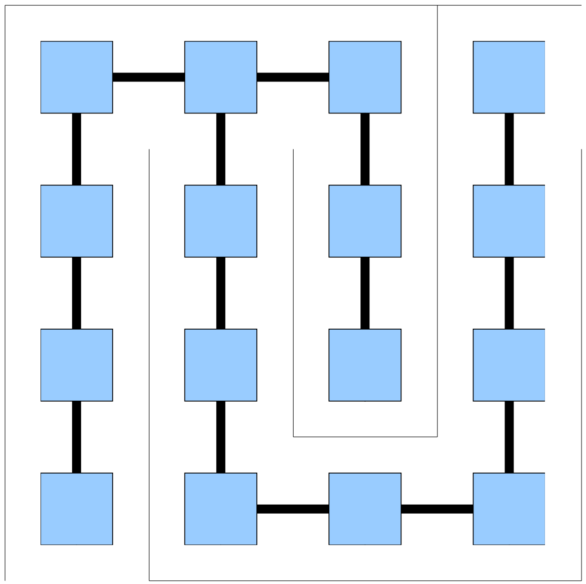
Boggle as a graph

Vertex = letter cube; Edge = connection to neighboring cube



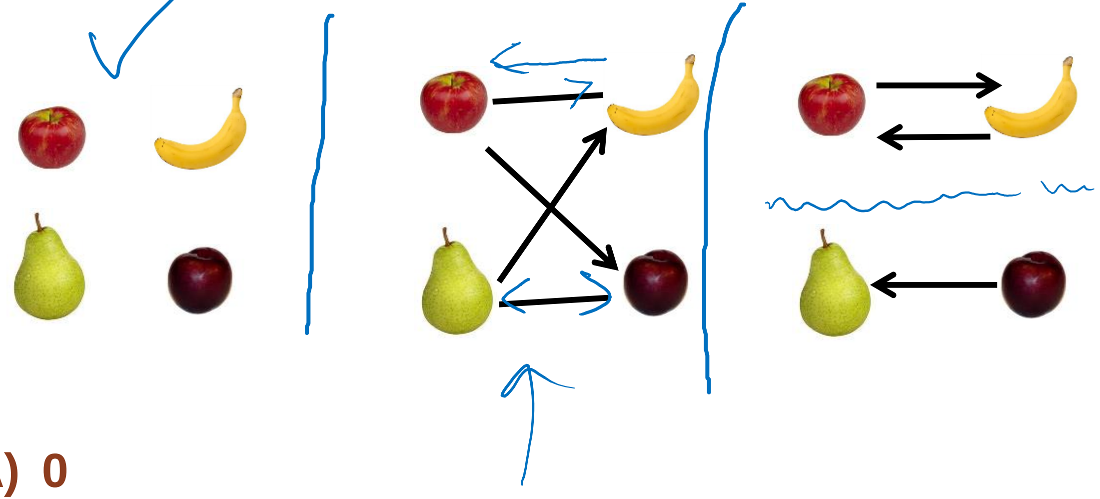
Maze as graph

If a maze is a graph, what is a vertex and what is an edge?



Graphs

How many of the following are valid graphs?



A) 0
B) 1
C) 2
D) 3

Graph Terminology

Graph terminology: **Paths**

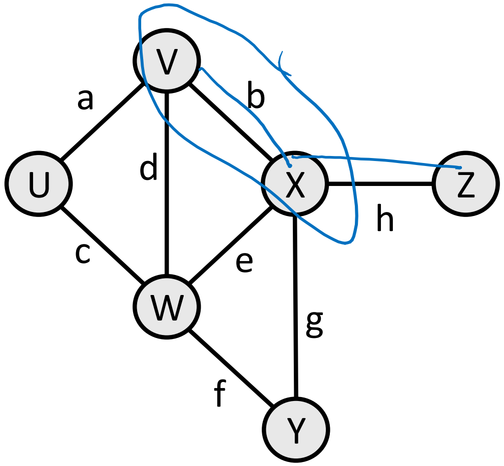
path: A path from vertex *a* to *b* is a sequence of edges that can be followed starting from *a* to reach *b*.

- can be represented as vertices visited, or edges taken
- Example: one path from *V* to *Z*: {b, h} or {V, X, Z}

path length: Number of vertices or edges contained in the path.

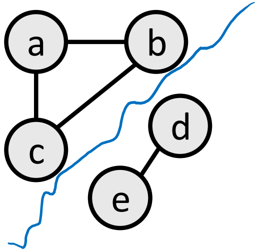
neighbor or **adjacent**: Two vertices connected directly by an edge.

- example: V and X



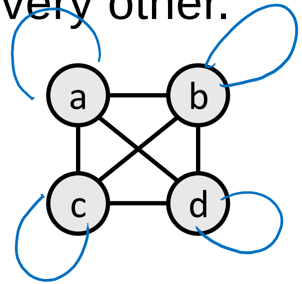
Graph terminology: **Reachability, connectedness**

reachable: Vertex *a* is *reachable* from *b* if a path exists from *a* to *b*.



connected: A graph is *connected* if every vertex is reachable from every other.

complete: If every vertex has a direct edge to every other.



Graph terminology: **Loops and cycles**

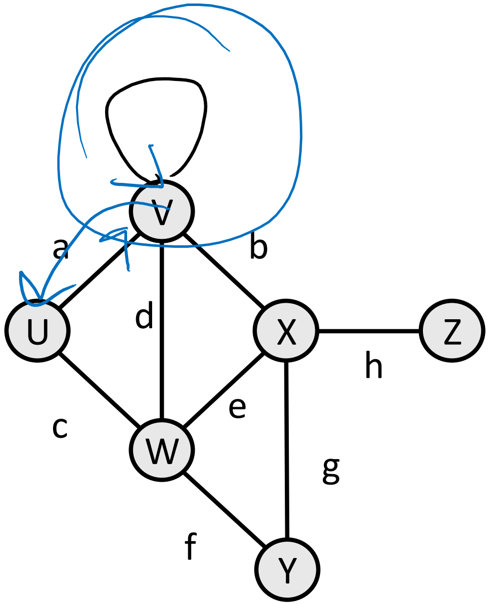
cycle: A path that begins and ends at the same node.

- example: {V, X, Y, W, U, V}.
- example: {U, W, V, U}.

- **acyclic graph**: One that does not contain any cycles.

loop: An edge directly from a node to itself.

- Many graphs don't allow loops.

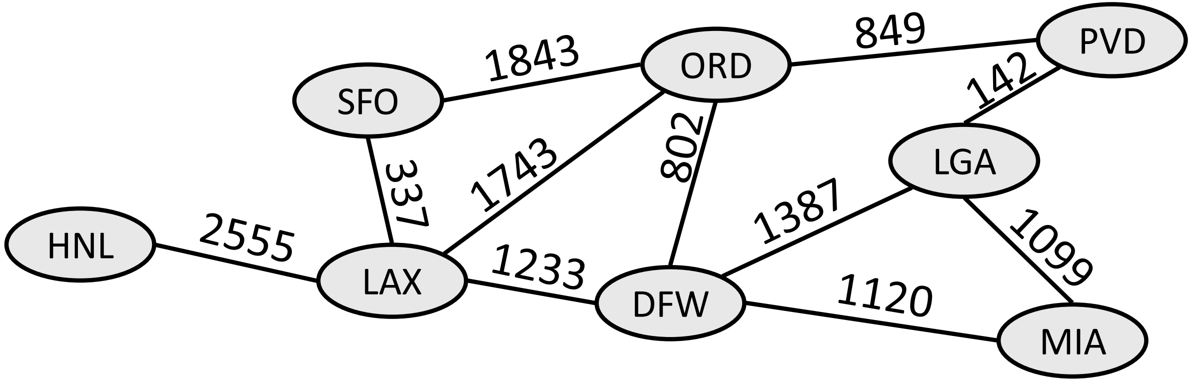


Graph terminology: **Weighted graphs**

weight: Cost associated with a given edge.

- Some graphs have weighted edges, and some are unweighted.
- Edges in an unweighted graph can be thought of as having equal weight (e.g. all 0, or all 1, etc.)
- Most graphs do not allow negative weights.

example: graph of airline flights, weighted by miles between cities:



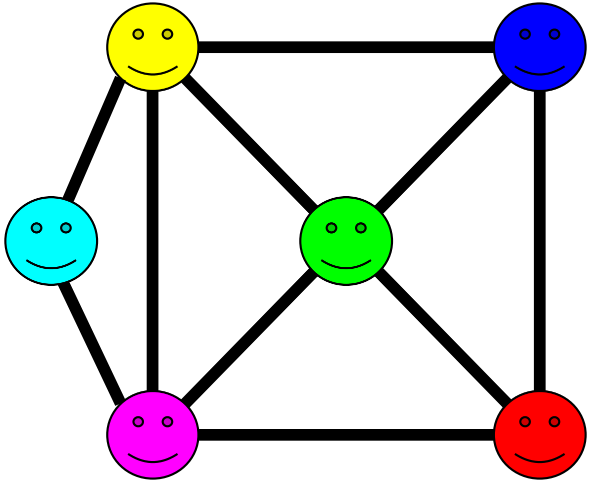
Representing Graphs

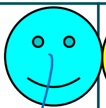
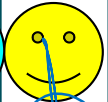
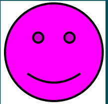
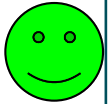
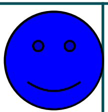
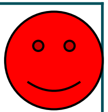
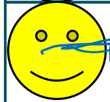
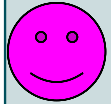
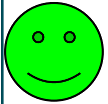
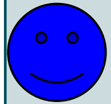
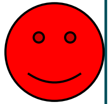
Ways we could implement a Graph class

Representing Graphs: **Adjacency matrix**

We can represent a graph as a Grid<bool> (unweighted)
or
Grid<int> (weighted)

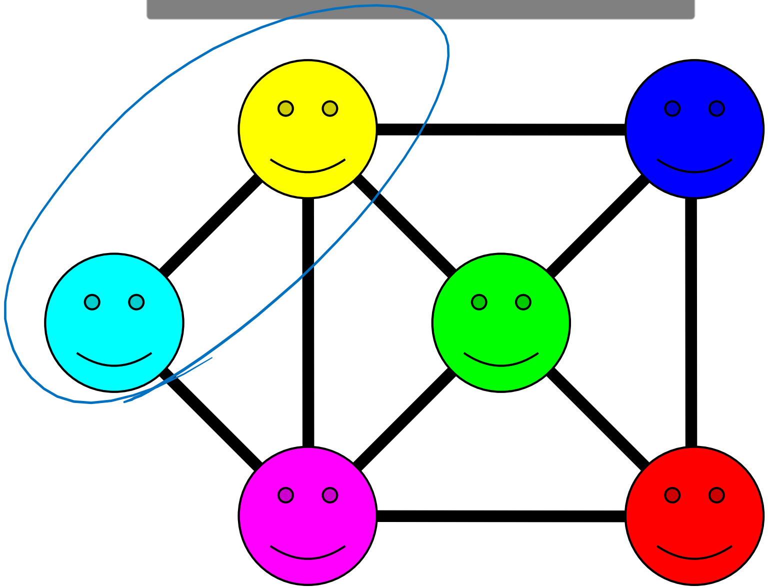
Facebook



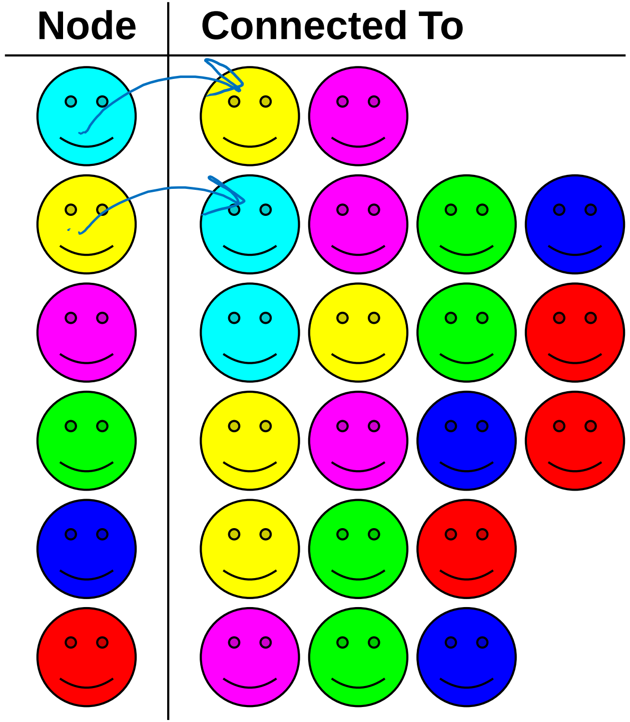
						
	0	1	1	0	0	0
	1	0	1	1	1	0
	1	1	0	1	0	1
	0	1	1	0	1	1
	0	1	0	1	0	1
	0	0	1	1	1	0

Representing Graphs: **Adjacency list**

We can represent a graph as a map from nodes to the set of nodes each node is connected to.



`Map<Node*, Set<Node*>>`



Common ways of representing graphs

Adjacency list:

- Map<Node*, Set<Node*>>

Adjacency matrix:

- Grid<bool> unweighted
- Grid<int> weighted

How many of the following are true?

- Adjacency **list** can be used for **directed** graphs
- Adjacency **list** can be used for **undirected** graphs
- Adjacency **matrix** can be used for **directed** graphs
- Adjacency **matrix** can be used for **undirected** graphs

(A) 0 (B) 1 (C) 2 (D) 3 (E) 4

Graph Theory

Just a little taste of theorems about graphs

Stanford University

Graphs lend themselves to fun theorems and proofs of said theorems!

Any graph with 6 vertices contains either a triangle (3 vertices with all pairs having an edge) or an empty triangle (3 vertices no two pairs having an edge)

also called "clique"

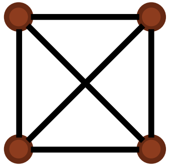
also called "independent set"

Eulerian graphs

Let G be an **undirected graph**

A graph is **Eulerian** if it can
drawn without lifting the pen
and without repeating edges

Is this graph Eulerian?

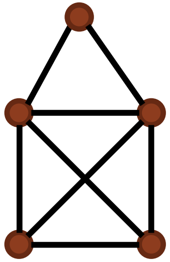


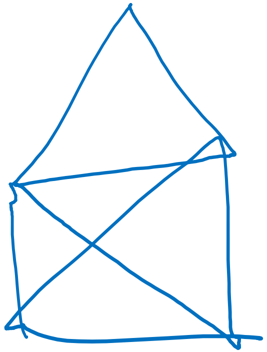
Eulerian graphs

Let G be an **undirected graph**

A graph is **Eulerian** if it can
drawn without lifting the pen
and without repeating edges

What about this graph?

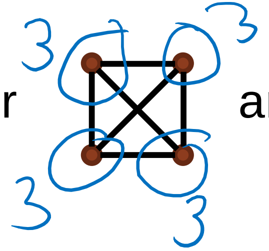


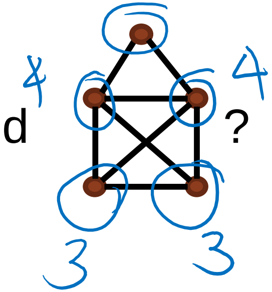


Our second graph theorem

Definition: **Degree** of a vertex: number of edges adjacent to it

Euler's theorem: a connected graph is Eulerian iff the number of vertices with odd degrees is either 0 or 2 (eg all vertices or all but two have even degrees)

Does it work for 

and 

?