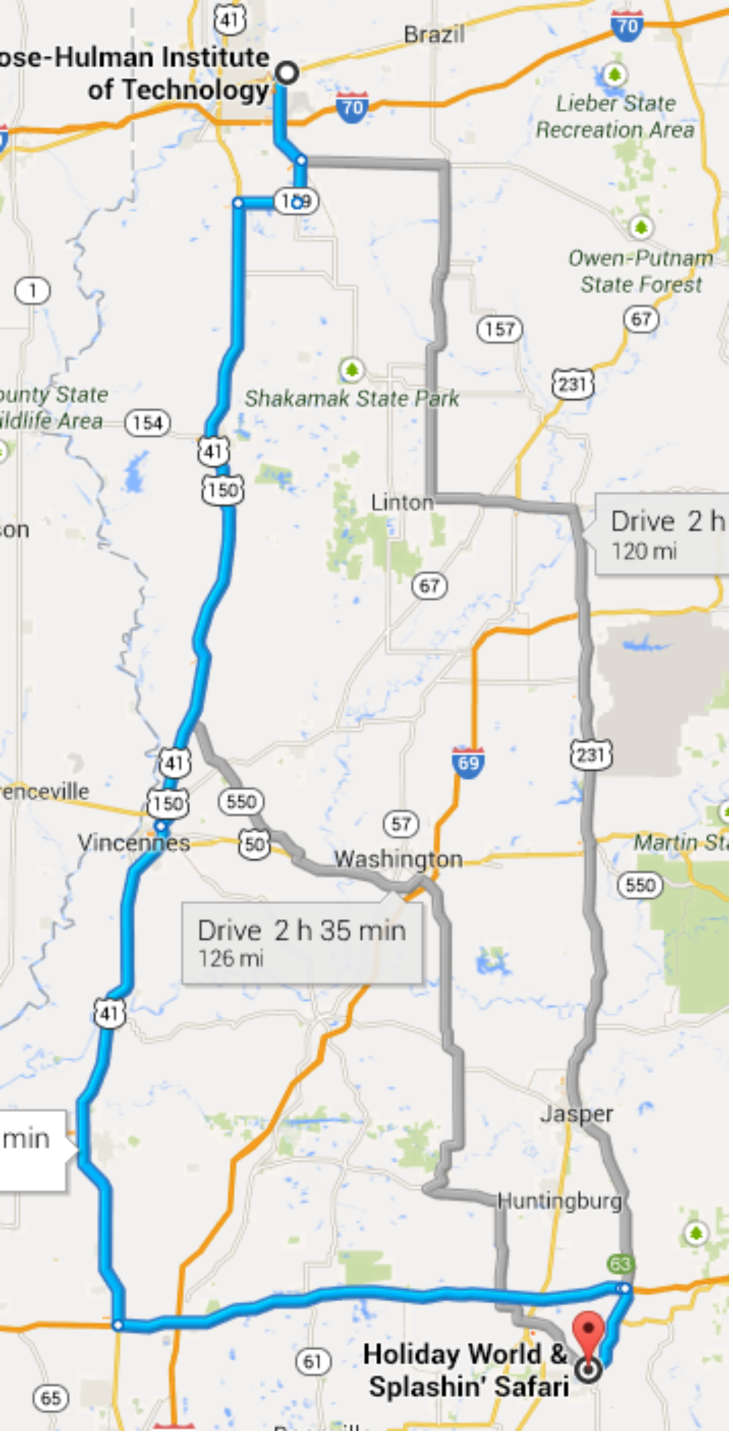
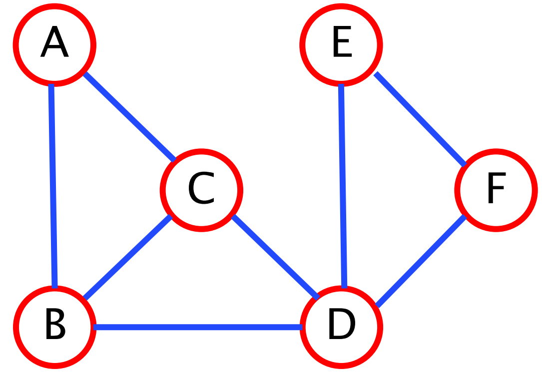


Review



CSSE 230 Day 22

Graphs and their representations

After this lesson, you should be able to …
… define the major terminology relating to graphs
… implement a graph in code, using various conventions

https://www.google.com/maps/dir/Rose-Hulman+Institute+of+Technology,+Wabash+Avenue,+Terre+Haute,+IN/Holiday+World+%26+Splashin'+Safari,+452+E+Christmas+Blvd,+Santa+Claus,+IN+47579/@38.7951117,-88.3071855,8z/data=!3m1!4b1!4m13!4m12!1m5!1m1!1s0x886d6e421b703737:0x96447680305ae1a4!2m2!1d-87.3234824!2d39.4830622!1m5!1m1!1s0x886e581193468c21:0x50d781efa416e09b!2m2!1d-86.9128116!2d38.1208766

Graphs

Terminology
Representations
Algorithms

Graph Definitions

A graph G = (V,E) is composed of:

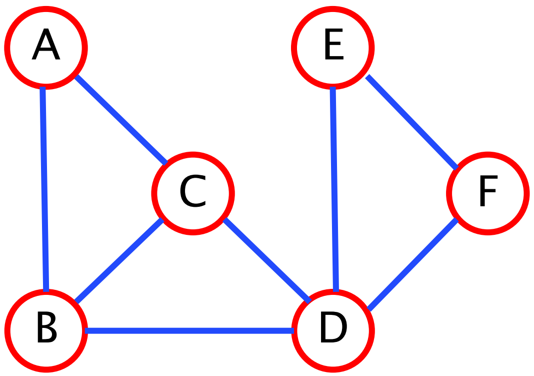
V: set of *vertices* (singular: vertex)

E: set of *edges*

An edge is a pair of vertices. Can be

unordered: e = {u,v} (*undirected* graph)

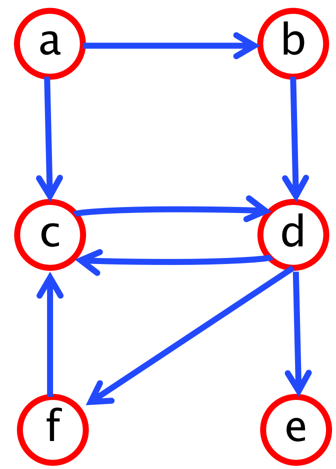
ordered: e = (u,v) (*directed* graph / *digraph*)



Undirected

V = {A,B,C,D,E,F}

E = {{A,B},{A,C},{B,C},{B,D},
{C,D},{D,E},{D,F},{E,F}}



Directed

V = {a,b,c,d,e,f}

E = {(a,b),(a,c),(b,d),(c,d),
(d,c),(d,e),(d,f),(f,c)}

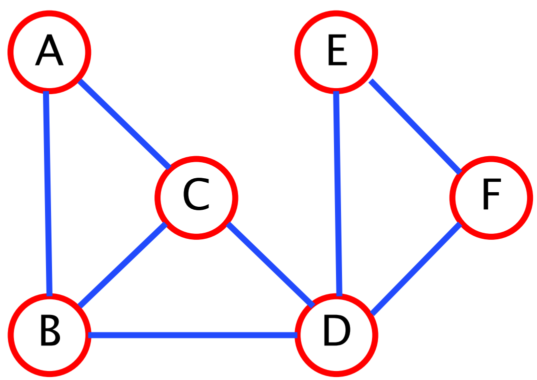
Graph Terminology

- Size? Edges or vertices?
- Usually take size to be $n = |V|$ (# of vertices)
- But the runtime of graph algorithms often depend on the number of edges, $|E|$
- Relationships between $|V|$ and $|E|$?

Undirected Graphs: adjacency, degree

- If {u,v} is an edge, then u and v are *neighbors* (also: u is *adjacent* to v)

- *degree* of v = number of neighbors of v



Fact:

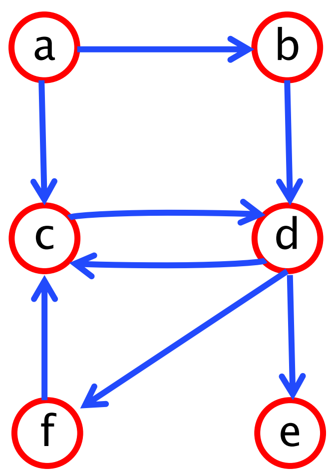
$$\sum_{v \in V} \deg(v) = 2|E|$$

(Why?)

Directed Graphs: adjacency, degree

- If (u,v) is an edge, then v is a *successor* of u and u is a *predecessor* of v

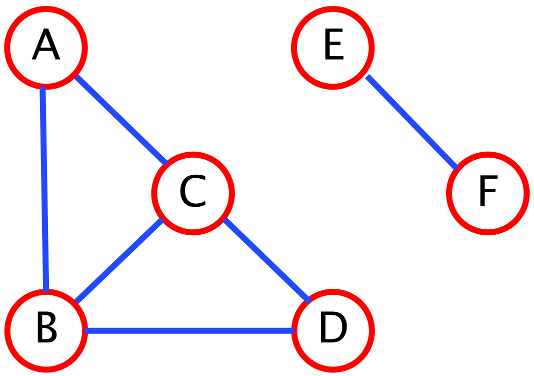
- *Out-degree* of v = number of successors of v
- *In-degree* of v = number of predecessors of v



Undirected Graphs: paths, connectivity

- A *path* is a list of unique vertices joined by edges.
  - For example, [a, c, d] is a path from a to d.

- A subgraph is *connected* if every pair of vertices in the subgraph has a path between them.



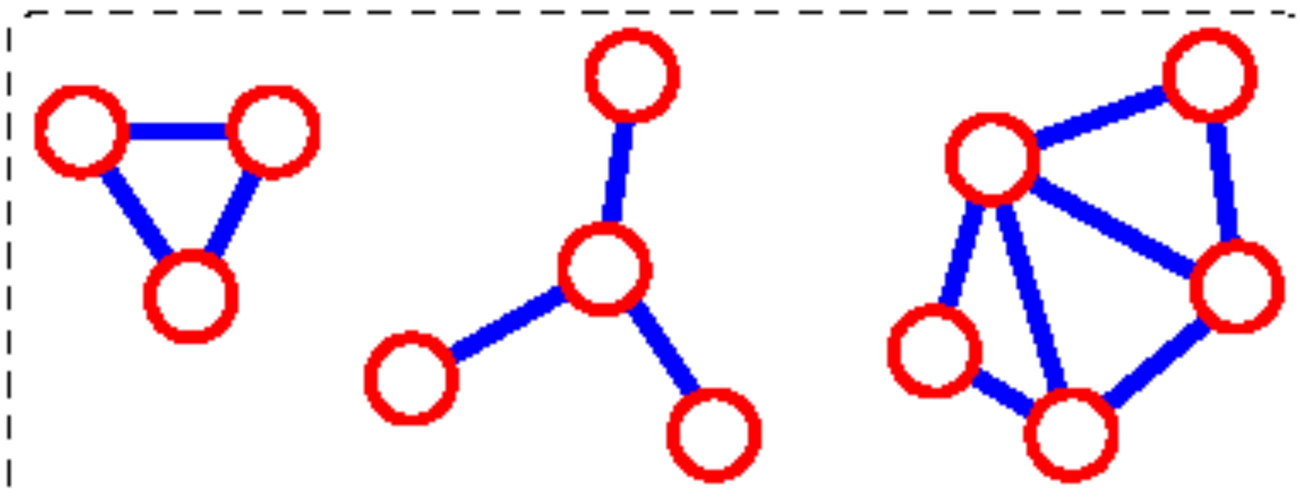
Not a connected graph.

Subgraph	Connected?
{A,B,C,D}	Yes
{E,F}	Yes
{C,D,E}	No
{A,B,C,D,E,F}	No

Undirected Graphs: components

(Connected) *component*: a maximal connected subgraph.

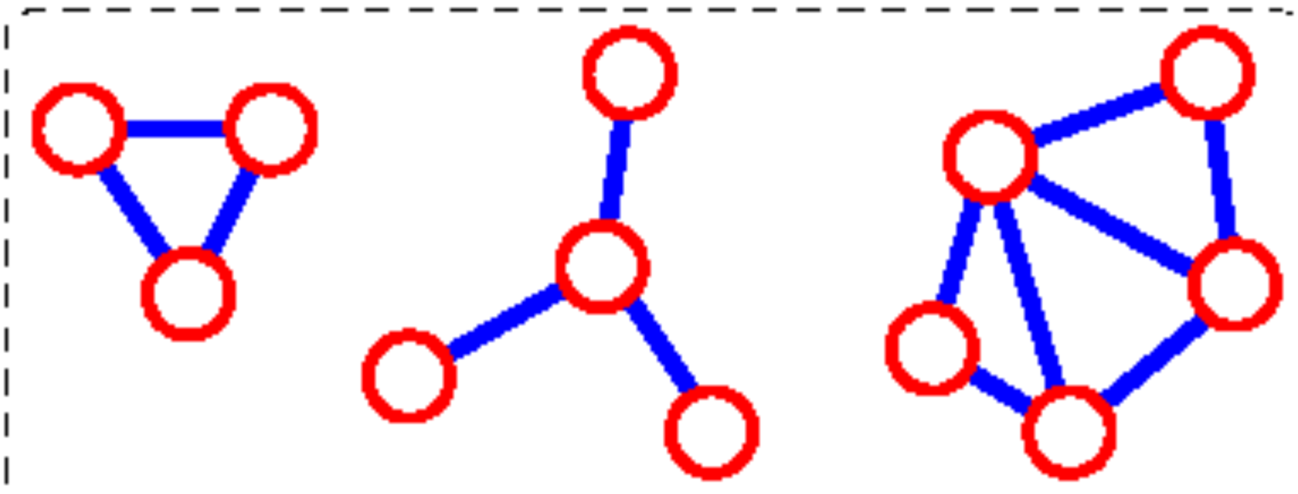
For example, this graph has 3 connected components:



Undirected Graphs: (mathematical) tree

Tree: connected acyclic graph (no cycles)

Example. Which component is a tree?



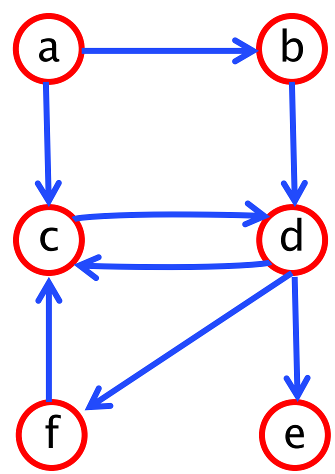
Question: for a tree, what is the relationship between
m = #edges and n = #vertices?

$m = n - 1$

Directed Graphs: paths, connectivity

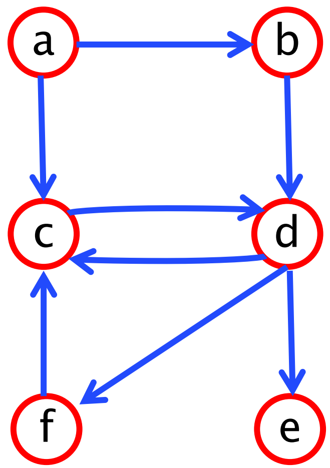
- A *directed path* is a list of unique vertices joined by directed edges.
  - For example, [a, c, d, f] is a directed path from a to f. We say f is *reachable* from a.

- A subgraph is *strongly connected* if for every pair (u,v) of its vertices, v is reachable from u and u is reachable from v.



Directed graphs: components

- *Strongly-connected component*: maximal strongly connected subgraph



Strongly connected components
{a}
{b}
{c,d,f}
{e}

Viewing a graph as a data structure

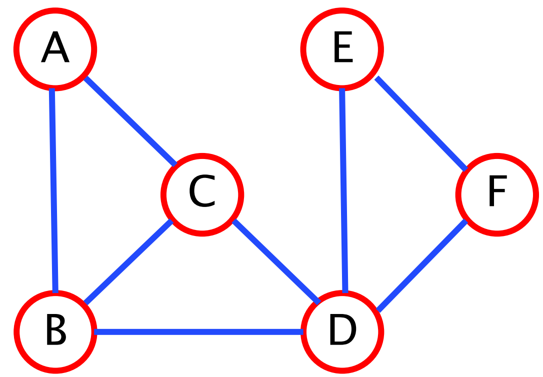
- Each vertex associated with a name (key)
- Examples:
  - City name
  - IP address
  - People in a social network

- An edge (undirected/directed) represents a link between keys

- Graphs are flexible: edges/nodes can have *weights*, *capacities*, or other attributes

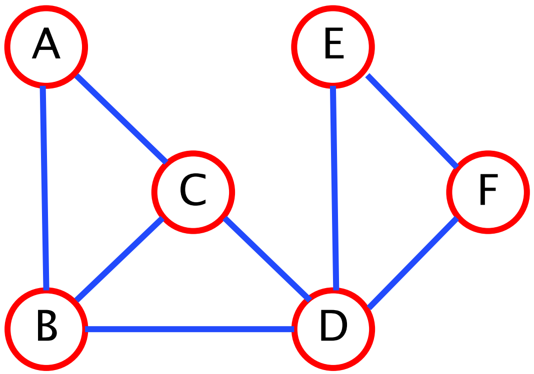
There are several alternatives for representing edges of a graph

- Edge list
  - A collection of vertices and a collection of edges

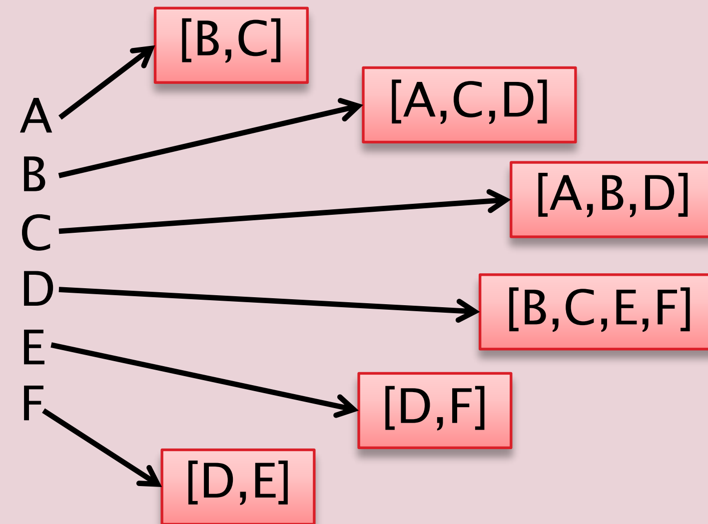


- Adjacency matrix
  - Each key is associated with an index from 0, …, (n-1)
    - Map from keys to ints?
  - Edges denoted by 2D array (#V x #V) of 0's and 1's
- Adjacency list
  - Collection of vertices
    - Map from keys to Vertex objects?
  - Each Vertex stores a List of adjacent vertices

Implementation tradeoffs



Adjacency list



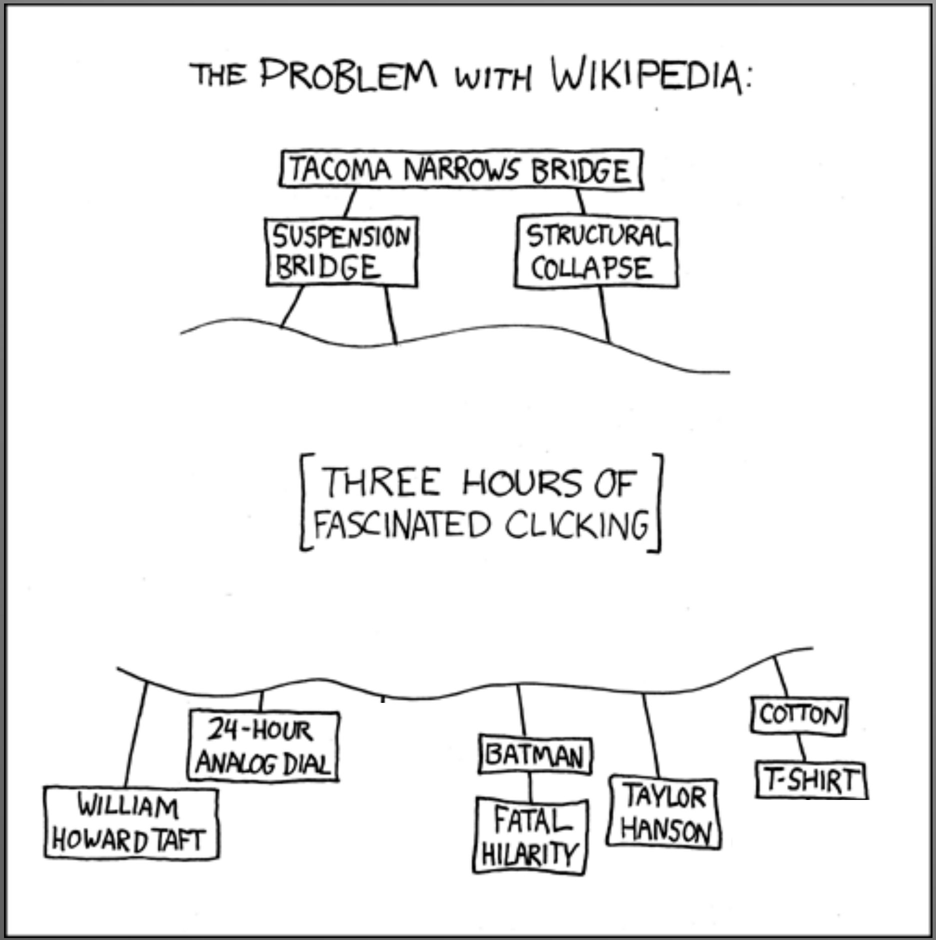
Adjacency matrix

A→0
B→1
C→2
D→3
E→4
F→5

	0	1	2	3	4	5
0	0	1	1	0	0	0
1	1	0	1	1	0	0
2	1	1	0	1	0	0
3	0	1	1	0	1	1
4	0	0	0	1	0	1
5	0	0	0	1	1	0

- Running time of degree(v)?
- Running time of deleteEdge(u,v)?
- Space efficiency?

GraphSurfing Project



GraphSurfing assignment

- Milestone 1: Implement AdjacencyListGraph<T> and AdjacencyMatrixGraph<T>
  - both extend the given ADT, Graph<T>.
- Milestone 2: Write methods
  - stronglyConnectedComponent(v)
  - shortestPath(from, to)

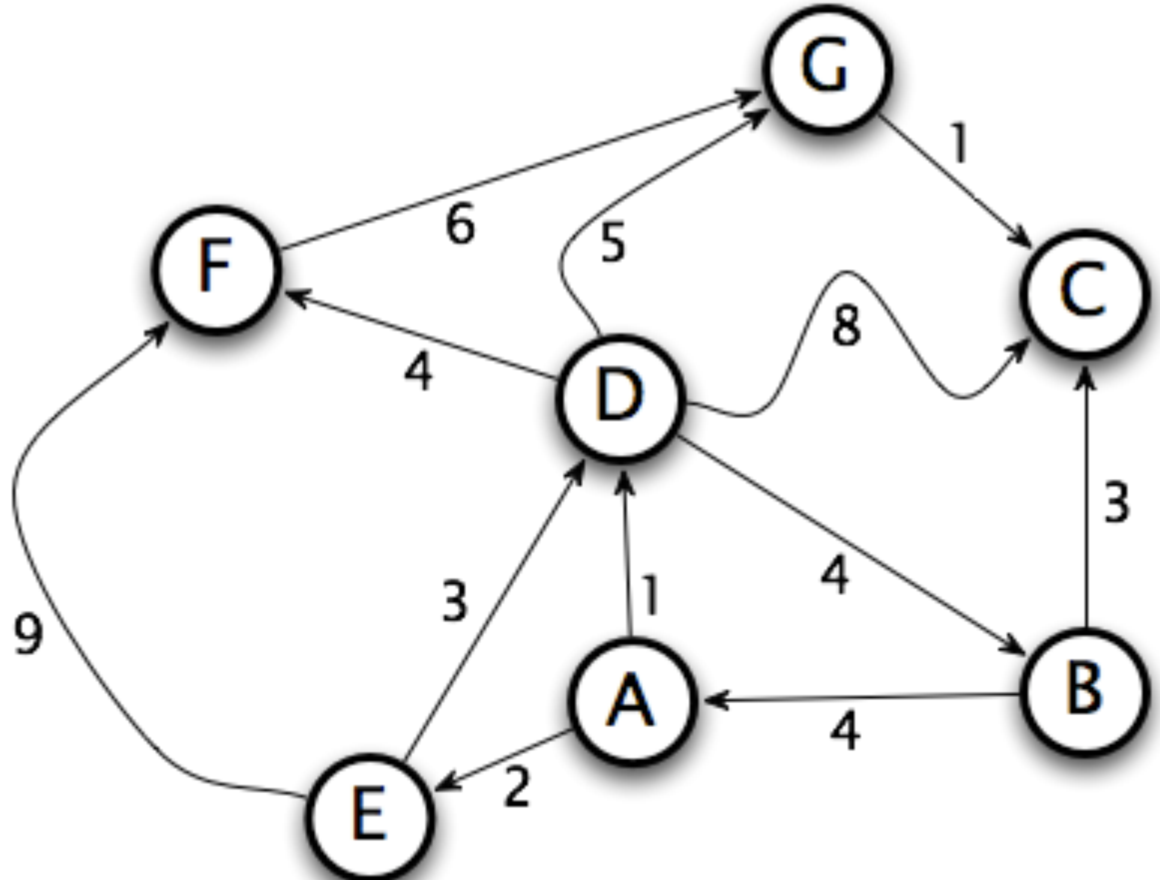
  and use them to go WikiSurfing!

Sample Graph Problems

To discuss algorithms, take MA/CSSE473 or MA477

Weighted Shortest Path

- What's the cost of the shortest path from A to each of the other nodes in the graph?

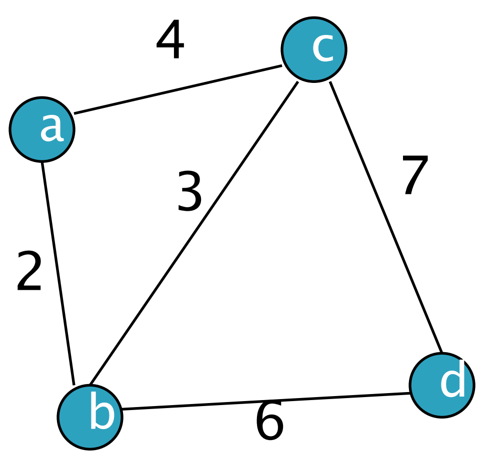


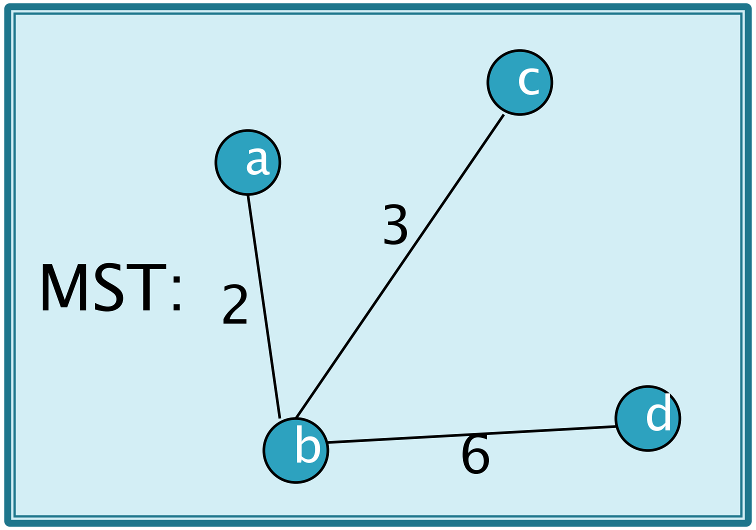
For much more on graphs, take MA/CSSE 473 or MA 477

Minimum Spanning Tree

- *Spanning tree*: a connected acyclic subgraph that includes all of the graph's vertices
- *Minimum spanning tree* of a weighted, connected graph: a spanning tree of minimum total weight

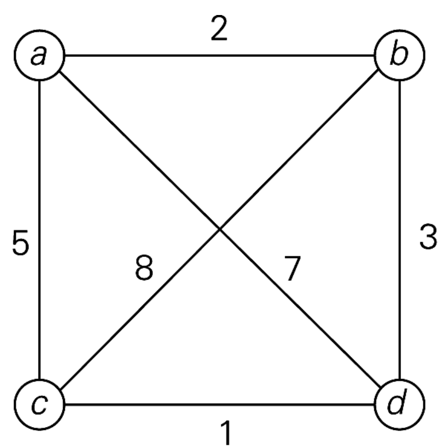
Example:





Traveling Salesman Problem (TSP)

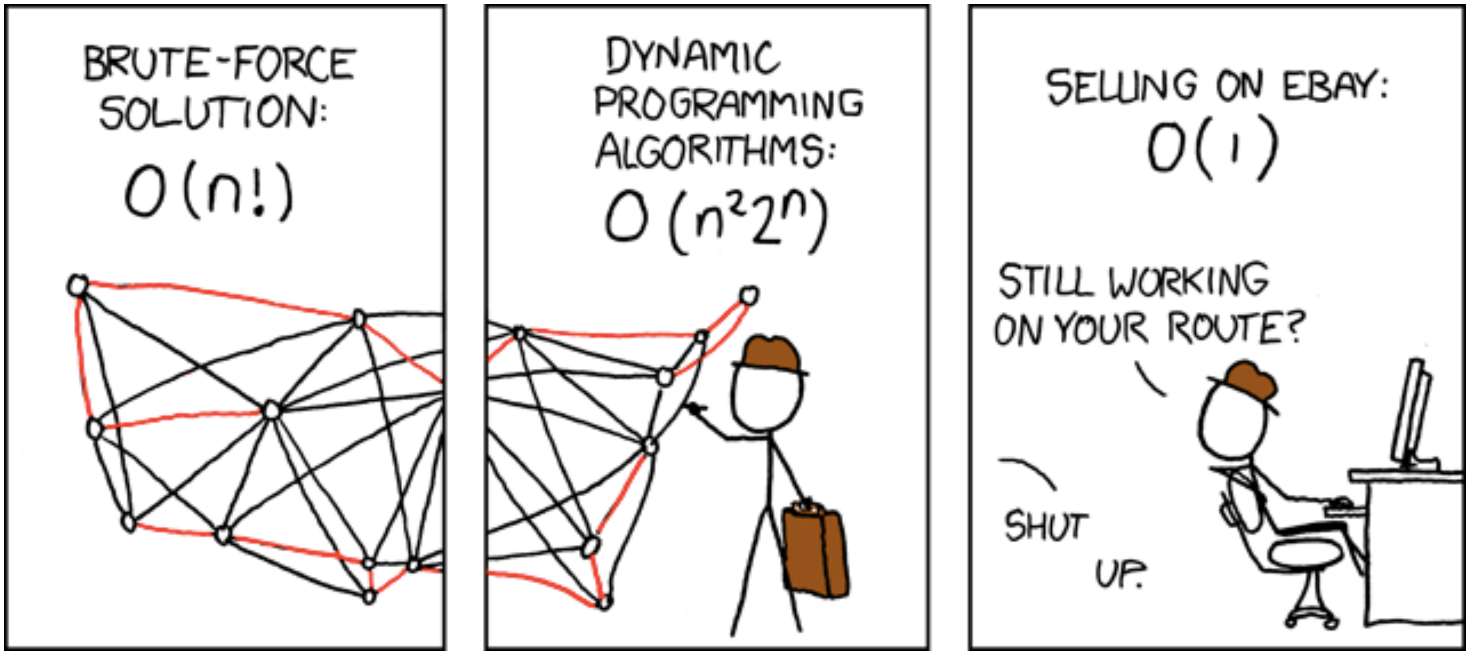
- *n* cities, weights are travel distance
- Must visit all cities (starting & ending at same place) with shortest possible distance



Tour	Length	
$a \to b \to c \to d \to a$	$l = 2 + 8 + 1 + 7 = 18$	
$a \to b \to d \to c \to a$	$l = 2 + 3 + 1 + 5 = 11$	optimal
$a \to c \to b \to d \to a$	$l = 5 + 8 + 3 + 7 = 23$	
$a \to c \to d \to b \to a$	$l = 5 + 1 + 3 + 2 = 11$	optimal
$a \to d \to b \to c \to a$	$l = 7 + 3 + 8 + 5 = 23$	
$a \to d \to c \to b \to a$	$l = 7 + 1 + 8 + 2 = 18$	

- Exhaustive search: how many routes?
- $(n-1)!/2 \in \Theta((n-1)!)$

Traveling Salesman Problem



- Online source for all things TSP:
  - http://www.math.uwaterloo.ca/tsp/

Example graphs for project