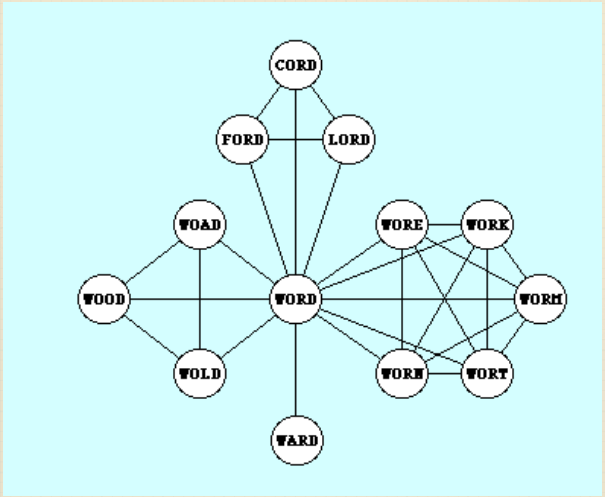
Admin

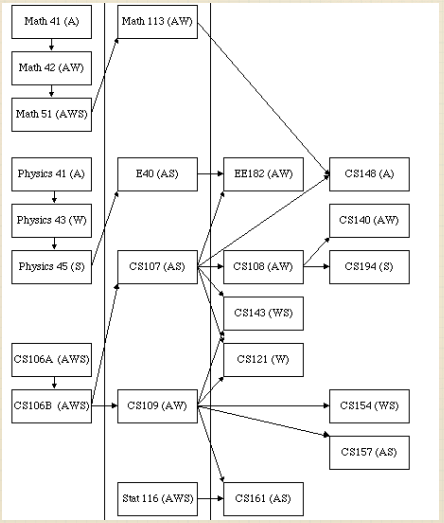
- PQueue due, Pathfinder out
  - Joy poll
- Today's topics
  - Graphs and graph algorithms
- Reading
  - Ch 11 hashing (next)

Airline routes

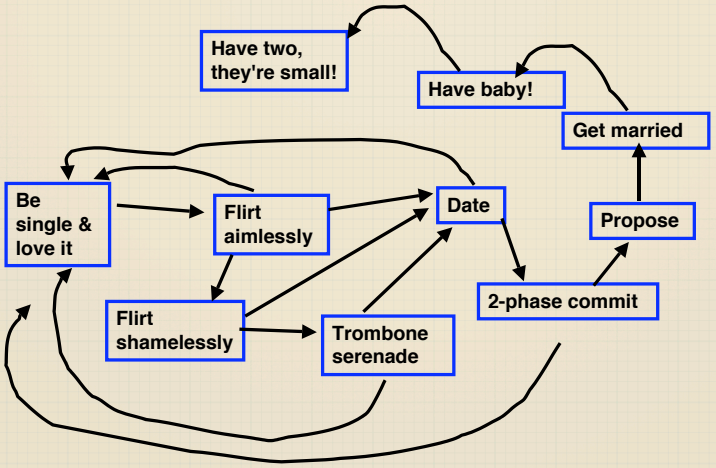
Word ladders



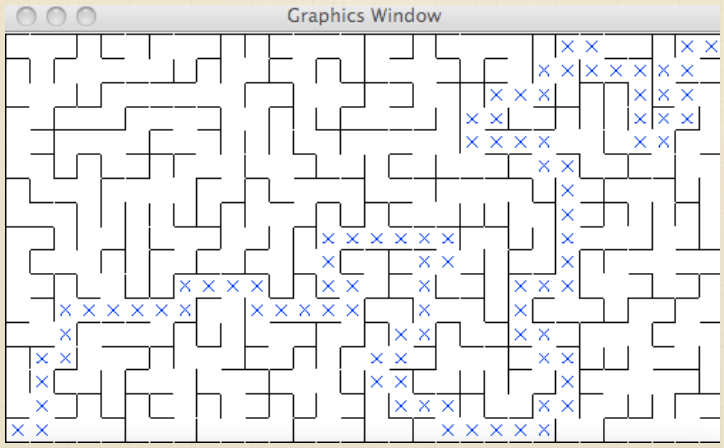
Course prerequisites



Flowcharts



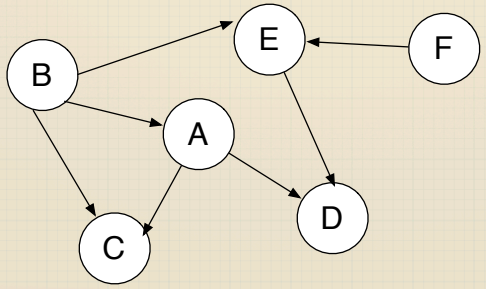
Mazes



Social networks

Graphs

- Another recursive data structure
  - Node with any number of pointers to other nodes
  - No restrictions on connectivity (allows disconnected, multiple paths, cycles)
- Terminology
  - Node, arc, directed, connected, path, cycle



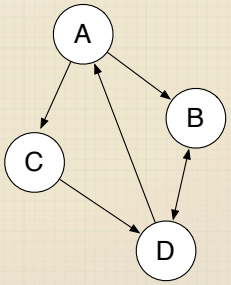
Implementation strategies

- Set of all arcs

```
{a->b, a->c, b->d, c->d, d->a, d->b}
```

- Adjacency list

```
a: {b, c}
b: {d}
c: {d}
d: {a, b}
```



- Adjacency matrix

	a	b	c	d
a		x	x	
b				x
c				x
d	x	x		

- Consider: sparse vs dense, space vs time

Representing graphs in C++

```
struct nodeT {
    // data for node goes here
    Vector<nodeT *> connected; // or Set
};
```

- Graph itself is Set/Vector of node *
  - Why not just pointer to root, like tree?
  - Could you designate an arbitrary node as root?

Nodes and arcs in C++

- Often graphs have data associated with arc itself
  - Unlike lists/trees where links are only for wiring up nodes
  - Arcs may have distance, cost, etc information
    - So add struct to hold arc information
  - Arc has pointers to start/end node, Node has collection of arcs
    - Uh-oh, circular reference!

```
struct nodeT;     // this is a forward reference

struct arcT {
     // arc fields (distance, cost, etc.)
    nodeT *start, *end;
};

struct nodeT {
     // node fields (name, etc.)
    Vector<arcT *> outgoing;
};
```

Graph traversals

- Traverse *reachable* nodes
  - Start from a node and follow arcs to other nodes
  - Some graphs not fully connected, not all nodes reachable
- Depth-first and breadth-first
  - Akin to tree's post/pre/in-order
  - Both visit all reachable nodes, but different order
  - Possibility of cycles means must track "visited" to avoid infinite loop
    - Could maintain visited flag per node or Set of visited nodes

Depth-first traversal

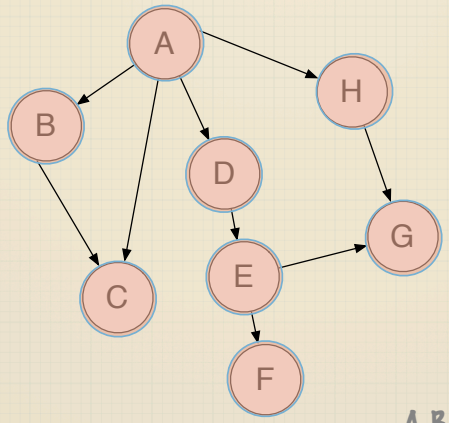
- Choose starting node
  - No "root" node in graph, may have specific start in mind or just choose one arbitrarily
- Go deep
  - Pick a neighbor, explore all reachable from there
- Backtrack
  - After fully exploring everything reachable from first neighbor, choose another neighbor and go again
- Continue until all neighbors exhausted
- What is base case for the recursion?

Depth-first cde

```
void DepthFirstSearch(nodeT *cur, Set<nodeT *> & visited)
{
   if (visited.contains(cur)) return;
   // do something with cur
   visited.add(cur);
   for (int i = 0; i < cur->outgoing.size(); i++)
      DepthFirstSearch(cur->outgoing[i]->end, visited);
}
```

- Using Set to track which nodes visited

Trace DFS



A B C D E F G H

Breadth-first traversal

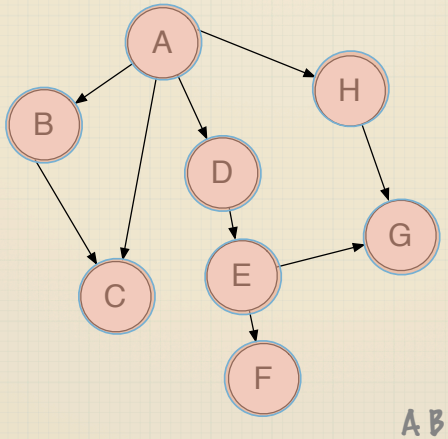
- Choose starting node
- Visit all immediate neighbors
  - Those directly connected to start node
- Branch out to all neighbors 2 hops away
- Again to 3 hops, and so on
  - Until all reachable nodes visited
- How to manage nodes to visit?
  - Perfect use for Queue!
- What about cycles/multiple paths?
  - Need to track visited status

Breadth-first traversal

```
void BreadthFirstSearch(nodeT *start)
{
   Queue<nodeT *> q;
   Set<nodeT *> visited;
   q.enqueue(start);

   while (!q.isEmpty()) {
       nodeT *cur = q.dequeue();
       if (!visited.contains(cur))) {
           visited.add(cur);
           for (int i = 0; i < cur->outgoing.size(); i++)
               q.enqueue(cur->outgoing[i]->end);
       }
   }
}
```

Trace BFS



A B C D H E G F

Graph search algorithms

- Many interesting questions are just graph search
  - Which nodes are reachable from this node?
  - Does the graph have a cycle?
  - Is the graph fully connected?
  - Longest path without a cycle?
  - Is there a continuous path that visits all nodes once and exactly once?
- Searching for word ladders
  - What are the nodes? What are the arcs?
  - What kind of traversal might work?
- Spell-checking suggestions
  - What are the nodes? What are the arcs?
  - How to support wildcards: desp.rate

Weighted arcs

- What if hops not all equal?