Week 6 Lecture 18

Theory

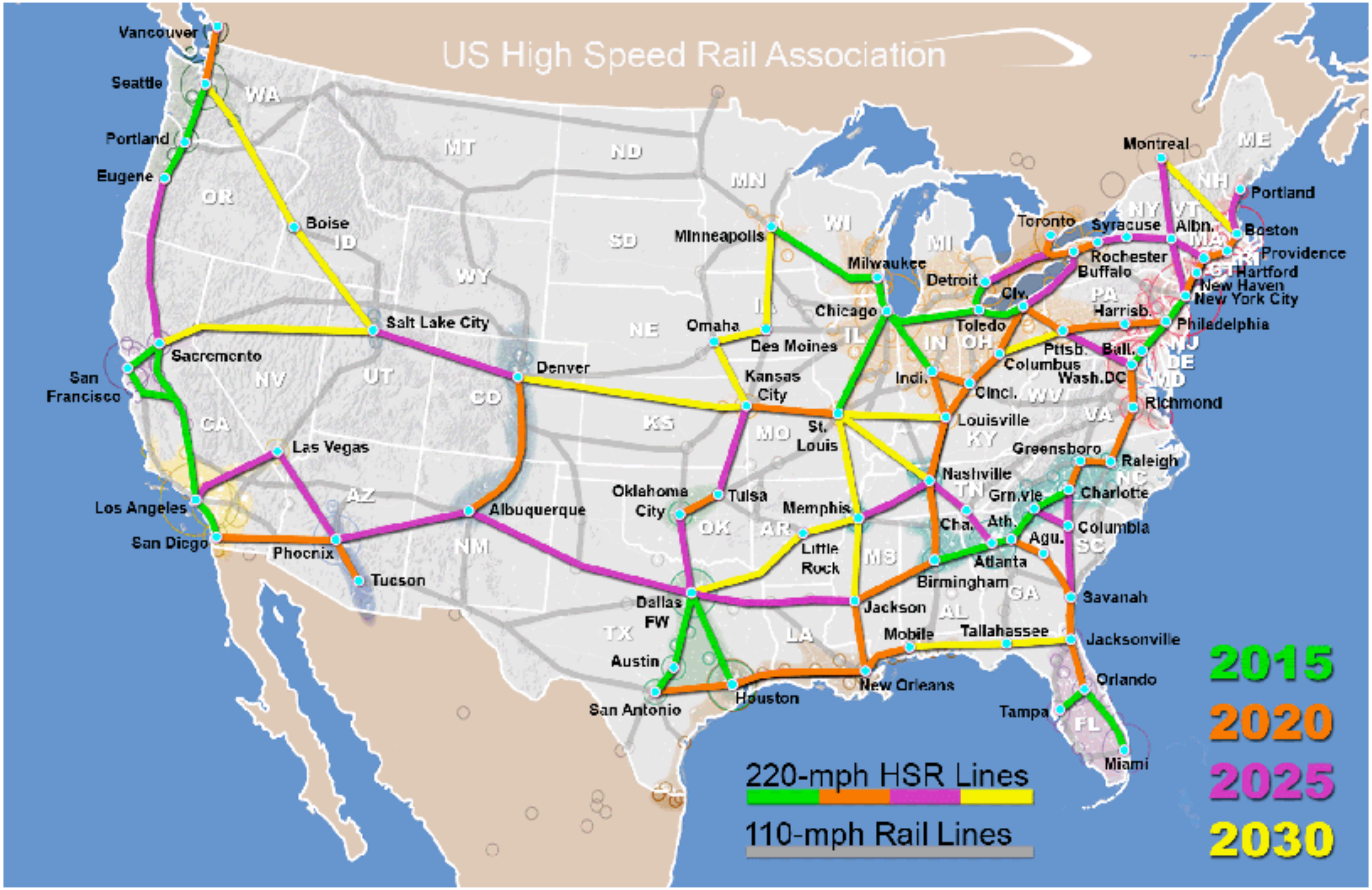
What’s in this lecture?

- Graph Algorithms
  - Breadth-First Search
  - Depth-First Search

The Graph

- A graph is used to model connections between things
- For example, a graph may be used to model transit connections between cities
- Facebook manages a graph representing friendship connections between people

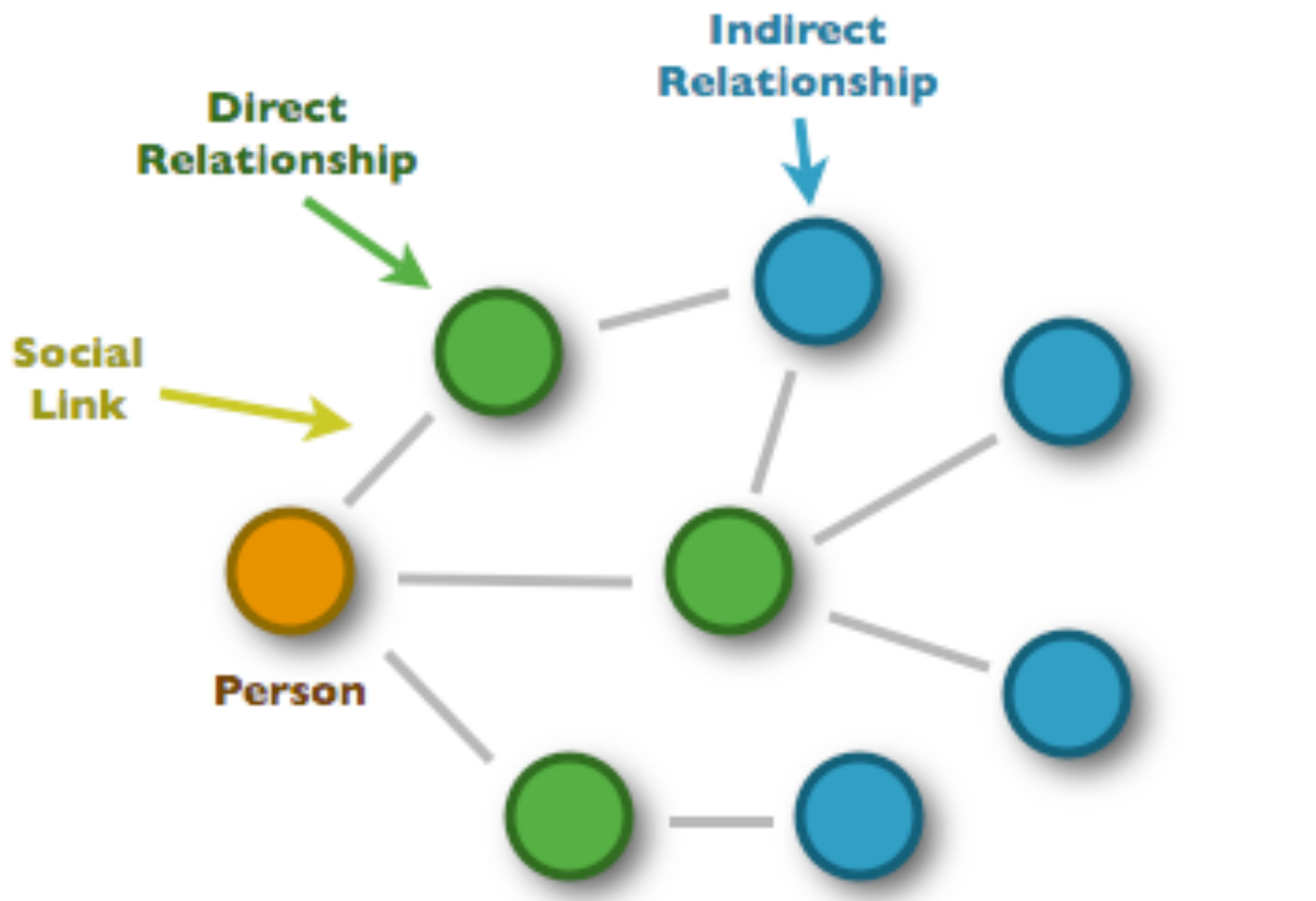
Transit Graph



Social Graph

Social Graphs:

The pattern of social relationships between people



Source: Dion Hinchcliffe. http://web2.socialcomputingmagazine.com

Modeling Graphs

- A Graph consists of nodes (or vertices) and edges
- Each node has an identifier, typically an integer (1, 2, 3...) or a string (“node4”, “chicago”, ...)
- Each edge consists of a “from” and “to” identifier, for example: (1, 2), (3, 4), ...

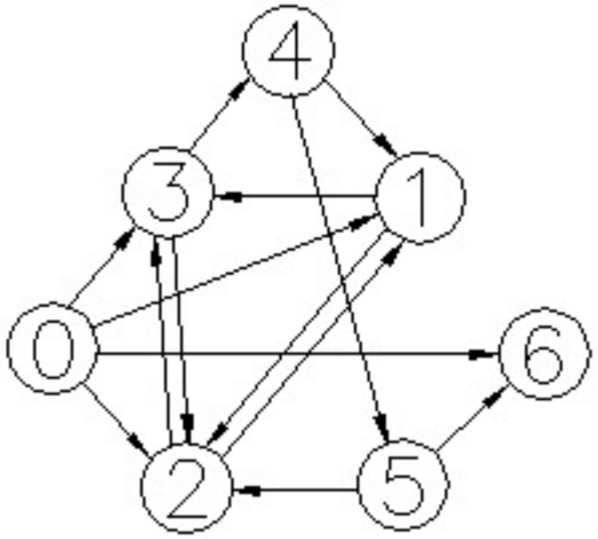
Modeling Graphs

- Edges may be directed (typically used for modeling asymmetric relationships) or undirected (used for symmetric relationships)
- An undirected graph may be modeled by a directed graph with an edge for each direction
- In more advanced applications (not covered here), edges may also have weights; for example distances on a road map

Cyclic Graphs

- A graph where some subset of nodes forms a path such that the first is also the last
- A graph can have multiple cycles contained
- Simple rule: when you have your pen on a single node, can you traverse the graph back to your starting point?

A Simple Graph

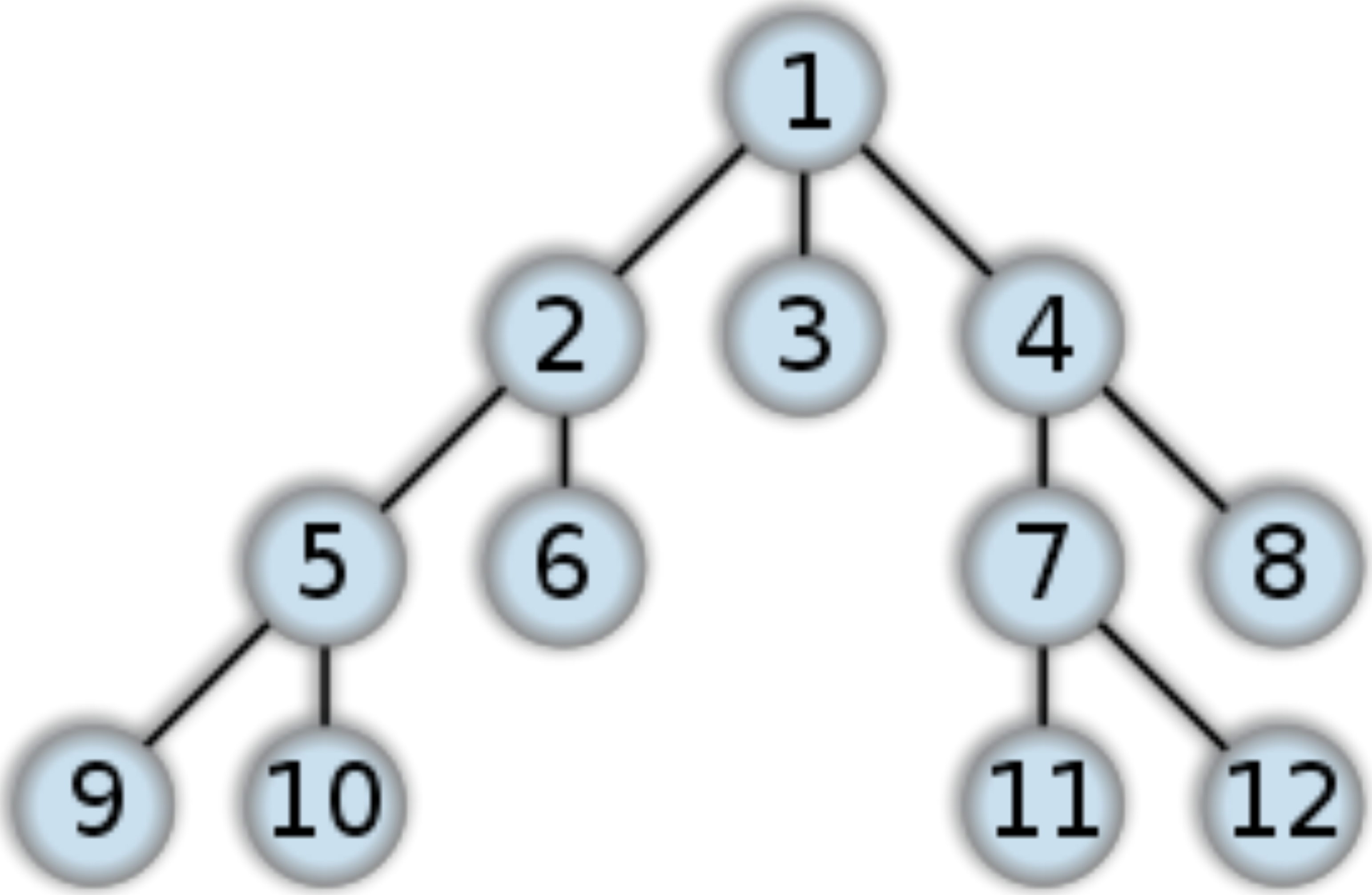


- V = [0, 1, 2, 3, 4, 5, 6]
- E = [(0, 1), (0, 2), (0, 3), (0, 6),
  (1, 3), (1, 2), (2, 1), (2, 3) ...]
- The degree of a node is the number of inbound edges (node 5 is degree 1 above, node 2 is degree 3)

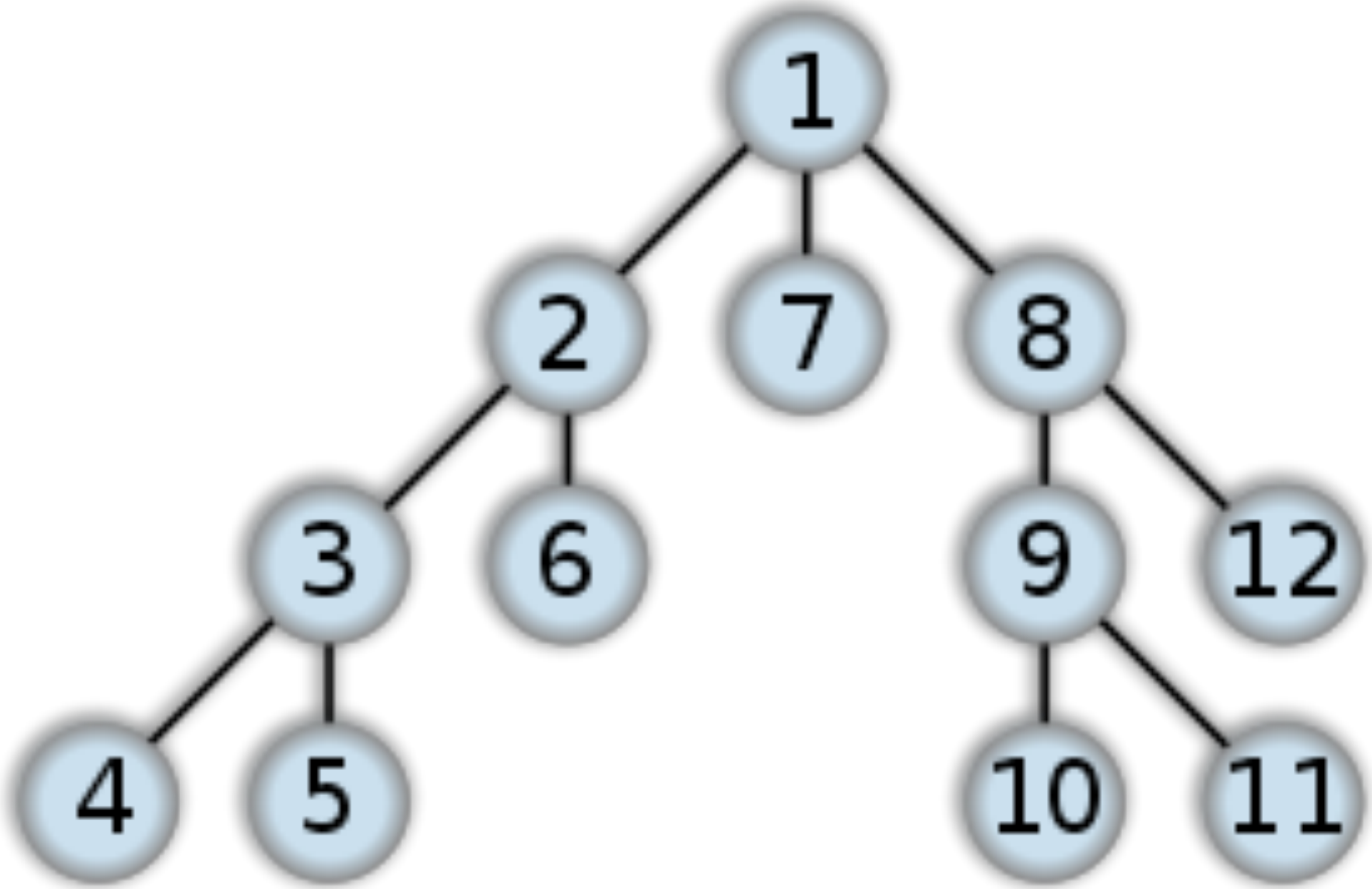
Searching Graphs

- The 2 common ways of searching graphs are Breadth-First Search (BFS) and Depth-First Search
- In Breadth-First Search, the traversal moves among peer vertexes with increasing distance from the starting node
- In Depth-First Search, the traversal moves to the greatest depth possible before backtracking among peers

Breadth-First Search



Depth-First Search



Directed Graph Impl

```
function Graph() {
  this.g = {};
}

Graph.prototype.add_edge = function(from, to) {
  if (!this.g[from]) { this.g[from] = []; }
  if (!this.g[to])     { this.g[to] = []; }
  if (this.g[from].indexOf(to) == -1) {
    this.g[from].push(to);
  }
}
```

BFS Impl

```
// outer ‘driver’ function
Graph.prototype.bfs = function(from, maxdepth) {
  return this._bfs_impl(
    [from], 1, maxdepth, {}, [{ node : from, depth : 0}]);
};

// NOTES: for the _bfs_impl call
// arg0 : [from] is the list of nodes we need to check
// arg1  : depth is the current depth
// arg2 : maxdepth is the depth to continue from here
// arg3 : visited is a hash of nodes that we’ve visited
// arg4 : accum builds up a list of nodes and depths
// return : the accum list of nodes to specified maxdepth
```

BFS Impl

```
Graph.prototype._bfs_impl = function(from, depth, maxdepth, visited, accum) {
  if (maxdepth == 0 || from.length < 1) { return accum; }

  var current = from.pop();
  visited[current] = 1;
  var neighbors = this.g[current];

  for (var i = 0; i < neighbors.length; i++) {
    var next = neighbors[i];
    accum.push({ node : next, depth : depth });
    if (!visited[next]) {
      from.push(next);
    }
  }

  return this._bfs_impl(from, depth + 1, maxdepth - 1, visited, accum);
};
```

Exercises

- Implement Depth-First Search “dfs” using the Intro to Algorithms Book and Wikipedia as a guide
- Create a cyclic graph and test whether your DFS function terminates on it
- Create another version of DFS that terminates (or fails to terminate) on the cyclic graph
- Create a social graph based on your Facebook or LinkedIn friends that contains at least 50 nodes and goes to depth 3 or 4, and run a couple searches on it starting from different people