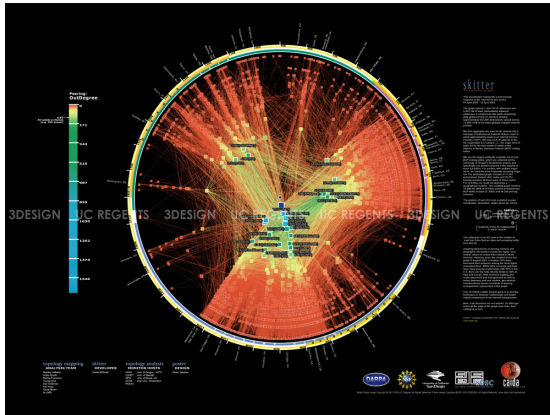


Graphs, the Internet, and Everything



<http://www.caida.org/>

Is there a Science of Networks?

- What kinds of networks are there?
- From Bacon numbers to random graphs to Internet
 - From FOAF to Selfish Routing: apparent similarities between many human and technological systems & organization
 - Modeling, simulation, and hypotheses
 - Compelling concepts
 - Metaphor of viral spread
 - Properties of connectivity has qualitative and quantitative effects
 - Computer Science?
- From the facebook to tomogravity
 - How do we model networks, measure them, and reason about them?
 - What mathematics is necessary?
 - Will the real-world intrude?

Jon Kleinberg



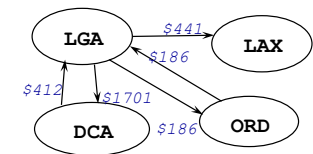
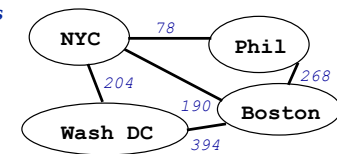
- 2005 MacArthur Fellow, 2008 Infosys Award, 2008 Discover "20 Best Brains under 40"
- Networks course and book
 - CompSci 96 Spring 2010

- "...Try to keep an open mind about topics and areas going on....It's much easier to make progress on a problem when you are enjoying what you are doing. In addition to finding work that is important, find work that has some personal interest for you....I've benefited from a lot of mentoring throughout my career. I think it's important to pass it on to the next generation and work in a mentoring capacity or a teaching capacity with people entering the field...."

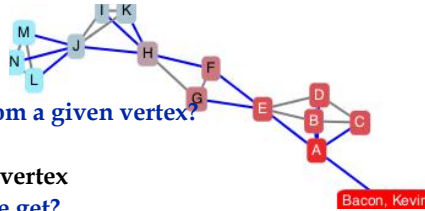
ACM Infosys Interview

Vocabulary

- Graphs are collections of *vertices* and *edges* (vertex also called node)
 - Edge connects two *vertices*
 - Direction can be important, *directed edge*, *directed graph*
 - Edge may have associated weight/cost
- A vertex sequence v_0, v_1, \dots, v_{n-1} is a *path* where v_k and v_{k+1} are connected by an edge.
 - If some vertex is repeated, the path is a *cycle*
 - A graph is *connected* if there is a path between any pair of vertices
- What vertices are reachable from a given vertex?
 - Traverse the graph...



Traversals



- What vertices are reachable from a given vertex?
 - > Connected components?
 - > Degree: # edges incident a vertex
- Starting at Bacon where can we get?
 - > *Random search*, choose a neighboring vertex at random
 - Can we move in circles?
 - > *Depth-first search*, envision each vertex as a room, with doors leading out
 - Go into a room, mark the room, choose an unused door, exit
 - Don't go into a room you've already been in (see mark)
 - *Backtrack* if all doors used (to room with unused door)
 - Used in Percolation assignment
 - > Rooms are stacked up, backtracking is really recursion
 - > One alternative uses a queue: *breadth-first search*

Depth-first search on Graphs

```
public Set<Graph.Vertex> dfs(Graph.Vertex start) {
    Set<Graph.Vertex> visited = new TreeSet<Graph.Vertex>();
    Stack<Graph.Vertex> qu = new Stack<Graph.Vertex>();
    visited.add(start);
    qu.push(start);

    while (qu.size() > 0) {
        Graph.Vertex v = qu.pop();
        for (Graph.Vertex adj : myGraph.getAdjacent(v)) {
            if (! visited.contains(adj)) {
                visited.add(adj);
                qu.push(adj);
            }
        }
    }
    return visited;
}
```

BFS compared to DFS

```
public Set<Graph.Vertex> bfs(Graph.Vertex start) {
    Set<Graph.Vertex> visited = new TreeSet<Graph.Vertex>();
    Queue<Graph.Vertex> qu = new LinkedList<Graph.Vertex>();
    visited.add(start);
    qu.add(start);

    while (qu.size() > 0) {
        Graph.Vertex v = qu.remove();
        for (Graph.Vertex adj : myGraph.getAdjacent(v)) {
            if (! visited.contains(adj)) {
                visited.add(adj);
                qu.add(adj);
            }
        }
    }
    return visited;
}
```

Graph implementations

- Typical operations on graph:
 - > Add vertex
 - > Add edge (parameters?)
 - > getAdjacent(vertex)
 - > getVertices(..)
 - > String->Vertex (vice versa)
- Different kinds of graphs
 - > Lots of vertices, few edges, *sparse graph*
 - Use adjacency list
 - > Lots of edges (max # ?) *dense graph*
 - Use adjacency matrix

