Networks
Graph Data Type

- **Graph** = data type that represents pairwise connections.
- **Vertex** = element.
- **Edge** = connection between two vertices.
Graph API

Graph data type.

public class Graph (graph with String vertices)

Graph()
Graph(In in)
void addEdge(String v, String w)
Iterable<String> adjacentTo(String v)

create an empty graph
read graph from input stream
add edge v-w
neighbors of v

to support use with foreach

% more tiny.txt
A/B/I
B/A/F
C/D/G/H
D/C
E/F/I
F/B/E/G
G/C/F/H
H/C/G
I/A/E/F
Kevin Bacon Game

**Game.** Find (shortest) chain of movies connecting a performer to Kevin Bacon.

<table>
<thead>
<tr>
<th>performer</th>
<th>was in</th>
<th>with</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kevin Kline</td>
<td>French Kiss</td>
<td>Meg Ryan</td>
</tr>
<tr>
<td>Meg Ryan</td>
<td>Sleepless in Seattle</td>
<td>Tom Hanks</td>
</tr>
<tr>
<td>Tom Hanks</td>
<td>Apollo 13</td>
<td>Kevin Bacon</td>
</tr>
<tr>
<td>Kevin Bacon</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Q. How to represent the movie-performer relationships?
A. Use a graph.

- Vertex: performer or movie.
- Edge: connect performer to movie.
Computing Bacon Numbers

How to compute. Find shortest path in performer-movie graph.
To compute shortest paths:
- Source vertex is at distance 0.
- Its neighbors are at distance 1.
- Their remaining neighbors are at distance 2.
- Their remaining neighbors are at distance 3.
- ...
Breadth First Search

Goal. Given a vertex $s$, find shortest path to every other vertex $v$.

BFS from source vertex $s$

Put $s$ onto a FIFO queue.
Repeat until the queue is empty:
  - dequeue the least recently added vertex $v$
  - add each of $v$’s unvisited neighbors to the queue, and mark them as visited.

Key observation. Vertices are visited in increasing order of distance from $s$ because we use a FIFO queue.
Path Finder API

Path finder API.

```
public class PathFinder  // (data type to compute shortest paths)

  PathFinder(Graph G, String s)  // process graph G with source s
  int distanceTo(String v)  // return shortest distance between s and v
  void showPath(String v)  // print shortest path between s and v
```

Design principles.
- Decouple graph algorithm from graph data type.
- Avoid feature creep.
### Computing Bacon Numbers: Java Implementation

```java
public class Bacon {
    public static void main(String[] args) {
        In in = new In(args[0]); // read in the graph from a file
        Graph G = new Graph(in);

        String s = "Bacon, Kevin";
        PathFinder finder = new PathFinder(G, s); // create object to return shortest paths

        while (!StdIn.isEmpty()) {
            String performer = StdIn.readLine();
            finder.showPath(performer);
        }
    }
}
```

To process queries, you can use the `main` method with the input file `top-grossing.txt`.

```
% java Bacon top-grossing.txt
Stallone, Sylvester
Rocky III (1982)
Tamburro, Charles A.
Berkeley, Xander
Apollo 13 (1995)
Bacon, Kevin

Goldberg, Whoopi
Sister Act (1992)
Grodenchik, Max
Apollo 13 (1995)
Bacon, Kevin

Tilghman, Shirley
```
public class PathFinder {
    private ST<String, String> prev = new ST<String, String>();
    private ST<String, Integer> dist = new ST<String, Integer>();

    public PathFinder(Graph G, String s) {
        Queue<String> q = new Queue<String>();
        q.enqueue(s);
        dist.put(s, 0);
        while (!q.isEmpty()) {
            String v = q.dequeue();
            for (String w : G.adjacentTo(v)) {
                if (!dist.contains(w)) {
                    q.enqueue(w);
                    dist.put(w, 1 + dist.get(v));
                    prev.put(w, v);
                }
            }
        }
    }
}

Breadth First Searcher: Preprocessing
Breadth First Searcher: Printing the Path

To print shortest path: follow $\text{prev}[]$ from vertex $v$ back to source $s$.

- Print $v, \text{prev}[v], \text{prev}[\text{prev}[v]], \ldots, s$.
- Ex: shortest path from $C$ to $A$: $C \leftarrow G \leftarrow F \leftarrow B \leftarrow A$

```java
public void showPath(String v) {
    while (prev.contains(v)) {
        StdOut.println(v);
        v = prev.get(v);
    }
}
```

symbol tables

<table>
<thead>
<tr>
<th>key</th>
<th>prev</th>
<th>dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>G</td>
<td>4</td>
</tr>
<tr>
<td>D</td>
<td>C</td>
<td>5</td>
</tr>
<tr>
<td>E</td>
<td>I</td>
<td>2</td>
</tr>
<tr>
<td>F</td>
<td>B</td>
<td>2</td>
</tr>
<tr>
<td>G</td>
<td>F</td>
<td>3</td>
</tr>
<tr>
<td>H</td>
<td>G</td>
<td>4</td>
</tr>
<tr>
<td>I</td>
<td>A</td>
<td>1</td>
</tr>
</tbody>
</table>
Running Time Analysis

**Analysis.** BFS **scales** to solve huge problems.

<table>
<thead>
<tr>
<th>data File</th>
<th>movies</th>
<th>performers</th>
<th>edges</th>
<th>read input</th>
<th>build graph</th>
<th>BFS</th>
<th>show</th>
</tr>
</thead>
<tbody>
<tr>
<td>G.txt</td>
<td>1,288</td>
<td>21,177</td>
<td>28K</td>
<td>0.26 sec</td>
<td>0.52 sec</td>
<td>0.32 sec</td>
<td>0 sec</td>
</tr>
<tr>
<td>PG13.txt</td>
<td>2,538</td>
<td>70,325</td>
<td>100K</td>
<td>0.31 sec</td>
<td>0.99 sec</td>
<td>0.72 sec</td>
<td>0 sec</td>
</tr>
<tr>
<td>action.txt</td>
<td>14,938</td>
<td>139,861</td>
<td>270K</td>
<td>0.72 sec</td>
<td>2.8 sec</td>
<td>2.0 sec</td>
<td>0 sec</td>
</tr>
<tr>
<td>mpaa.txt</td>
<td>21,861</td>
<td>280,624</td>
<td>610K</td>
<td>2.1 sec</td>
<td>7.5 sec</td>
<td>5.5 sec</td>
<td>0 sec</td>
</tr>
<tr>
<td>all.txt</td>
<td>285,462</td>
<td>933,864</td>
<td>3.3M</td>
<td>15 sec</td>
<td>56 sec</td>
<td>39 sec</td>
<td>0 sec</td>
</tr>
</tbody>
</table>

60MB

*data as of April 9, 2007*
Data Analysis

**Exercise.** Compute histogram of Kevin Bacon numbers.

**Input.** 285,462 movies, 933,864 actors.

<table>
<thead>
<tr>
<th>Bacon #</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2,249</td>
</tr>
<tr>
<td>2</td>
<td>218,088</td>
</tr>
<tr>
<td>3</td>
<td>561,161</td>
</tr>
<tr>
<td>4</td>
<td>111,149</td>
</tr>
<tr>
<td>5</td>
<td>7,905</td>
</tr>
<tr>
<td>6</td>
<td>903</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
</tr>
<tr>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td>( \infty )</td>
<td>32,294</td>
</tr>
</tbody>
</table>

*Buzz Mauro, Jessica Drizd, Pablo Capussi
Argentine short film *Sweet Dreams* (2005)*

*Fred Ott, solo actor in
*Fred Ott Holding a Bird* (1894)*

*data as of April 9, 2007*
Applications of Breadth First Search

More BFS applications.
- Particle tracking.
- Image processing.
- Crawling the Web.
- Routing Internet packets.
- ...

Extensions. Google maps.
Linked list. Ordering of elements.
Binary tree. Hierarchical structure of elements.
Graph. Pairwise connections between elements.

Data structures.
- Queue: linked list.
- Set: binary tree.
- Symbol table: binary tree.
- Graph: symbol table of sets.
- Breadth first searcher: graph + queue + symbol table.

Importance of data structures.
- Enables us to build and debug large programs.
- Enables us to solve large problems efficiently.
Erdös Number
Erdös Numbers

Paul Erdös. Legendary, brilliant, prolific mathematician who wrote over 1500 papers!

What's your Erdös number?
- Co-authors of a paper with Erdös: 1.
- Co-authors of those co-authors: 2.
- And so on ...

Paul Erdös (1913-1996)

<table>
<thead>
<tr>
<th>Erdös #</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>502</td>
</tr>
<tr>
<td>2</td>
<td>5,713</td>
</tr>
<tr>
<td>3</td>
<td>26,422</td>
</tr>
<tr>
<td>4</td>
<td>62,136</td>
</tr>
<tr>
<td>5</td>
<td>66,157</td>
</tr>
<tr>
<td>6</td>
<td>32,280</td>
</tr>
<tr>
<td>7</td>
<td>10,431</td>
</tr>
<tr>
<td>8</td>
<td>3,214</td>
</tr>
<tr>
<td>9</td>
<td>953</td>
</tr>
<tr>
<td>10</td>
<td>262</td>
</tr>
<tr>
<td>11</td>
<td>94</td>
</tr>
<tr>
<td>12</td>
<td>23</td>
</tr>
<tr>
<td>13</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>7</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>∞</td>
<td>4 billion +</td>
</tr>
</tbody>
</table>
Erdös has a Bacon number!

Erdös has a Kevin Bacon number of 4.

% java Bacon cast.txt
Erdös, Paul
N Is a Number (1993)
Patterson, Gene
Box of Moon Light (1996)
Turturro, John
Cradle Will Rock (1999)
Tim Robbins
Mystic River (2003)
Bacon, Kevin

... but so far, Kevin Bacon doesn’t have an Erdös number.
Erdös-Bacon Numbers

Sum of your Erdös and Bacon numbers.

- For most people: *infinity!*
- But for some ...

Prof. of Computer Science Bernard Chazelle

Erdös number 2:
Bernard–Janos Pach--Erdös

Bacon number 3!
Bernard in *Guy and Madeline on a Park Bench* w/Jerry Quinn
Quinn in *Mr North* w/Mary Stuart Masterson
Masterson in *Digging to China* w/Kevin Bacon

Erdös-Bacon number 5
Erdös-Bacon Numbers

Abigail A. Baird, Jerome Kagan, Thomas Gaudette, Kathryn A. Walz, Natalie Hershlag and David A. Boas

Stage name: Natalie Portman
Bacon number 2
Erdös-Bacon number 7

Erdös number 4
Erdös-Bacon Numbers

Figure 1. Examples of the five major biological networks

Xiaowei Zhu et al. Genes Dev. 2007; 21: 1010-1024
<table>
<thead>
<tr>
<th>Type of network</th>
<th>Species</th>
<th>Number of nodes</th>
<th>Number of interactions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transcription factor-binding</td>
<td>S. cerevisiae</td>
<td>3528</td>
<td>7419</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3207</td>
<td>11231</td>
</tr>
<tr>
<td>Protein–protein interaction</td>
<td>C. elegans</td>
<td>2788</td>
<td>4441</td>
</tr>
<tr>
<td></td>
<td>D. melanogaster</td>
<td>7546</td>
<td>25403</td>
</tr>
<tr>
<td></td>
<td>Homo sapiens</td>
<td>7509</td>
<td>20979</td>
</tr>
<tr>
<td></td>
<td>Mus musculus</td>
<td>209</td>
<td>393</td>
</tr>
<tr>
<td></td>
<td>S. cerevisiae</td>
<td>5325</td>
<td>51773</td>
</tr>
<tr>
<td>Phosphorylation network</td>
<td>S. cerevisiae</td>
<td>1325</td>
<td>4200</td>
</tr>
<tr>
<td>Metabolic network</td>
<td>E. coli</td>
<td>473</td>
<td>574</td>
</tr>
<tr>
<td>Genetic network</td>
<td>S. cerevisiae</td>
<td>3258</td>
<td>13963</td>
</tr>
</tbody>
</table>

*a*Transcriptional factor-binding data collected at rich-media condition.

*b*Transcriptional factor-binding data collected at a variety of growth conditions.

*c*Synthetic lethal interactions among nonessential genes.
• How do the networks specify organisms?

• What “functions” are computed by the networks?

• What are the principles that govern the structure and function of networks (or subnetworks)?

• How are networks organized?

• How are the specific functions of the cell accomplished?

• What behaviors emerge from these networks?

• How did the networks evolve?
Modules

Biological organization of networks based on functional modules

- Composed of many different types of molecules
- Functions of modules are (more or less) discrete entities and arise as a result of interactions among its components

Hartwell et al 1999
Uncovering modules: clustering
Network clustering

Distance-based hierarchical clustering
- Similarity between two proteins estimated as
  - Function of shared neighbors
  - Shortest-path profiles etc.

Dense subgraph finding
- Monte-carlo methods
- Greedy expansion
- Building from seed set

Divisive hierarchical approaches
Stochastic-flow...
Modules via Network Alignment

Ideker et al.
Network Motifs

Patterns of interconnections that recur in many different parts of a network at frequencies much higher than those found in randomized networks [Shen-Orr et al 2002]

First applied to E. coli transcriptional network
Network Motifs

Shen-Orr et al.
“Functional” properties of motifs

(a) Input X affects Y, which in turn affects Z, the output.

(b) X affects Z1, Z2, and Z3, with different time courses for each.
Theory -- Experiment