4.5 Small World Phenomenon
Small World Phenomenon

Small world phenomenon. Six handshakes away from anyone.

An experiment to quantify effect. [Stanley Milgram, 1960s]
• You are given personal info of another person. 
  e.g., occupation and age
• Goal: deliver message.
• Restriction: can only forward to someone you know by first name.
• Outcome: message delivered with average of 5 intermediaries.

Stanley Milgram
Kevin Bacon
Applications of Small World Phenomenon

Sociology applications.
- Looking for a job.
- Marketing products or ideas.
- Formation and spread of fame and fads.
- Train of thought followed in a conversation.
- Defining representative-ness of political bodies.
- Kevin Bacon game (movies, rock groups, facebook, etc.).

Other applications.
- Electronic circuits.
- Synchronization of neurons.
- Analysis of World Wide Web.
- Design of electrical power grids.
- Modeling of protein interaction networks.
- Phase transitions in coupled Kuramoto oscillators.
- Spread of infectious diseases and computer viruses.
- Evolution of cooperation in multi-player iterated Prisoner's Dilemma.

Application demands a new data type.
- **Graph** = data type that represents pairwise connections.
- **Vertex** = element.
- **Edge** = connection between two vertices.
Graph Applications

- **Graph**
  - communication
  - circuits
  - mechanical
  - hydraulic
  - financial
  - transportation
  - scheduling
  - software systems
  - internet
  - games
  - social relationship
  - neural networks
  - protein networks
  - chemical compounds

- **Vertices**
  - telephones, computers
  - gates, registers, processors
  - joints
  - reservoirs, pumping stations
  - stocks, currency
  - street intersections, airports
  - tasks
  - functions
  - web pages
  - board positions
  - people, actors
  - neurons
  - proteins
  - molecules

- **Edges**
  - fiber optic cables
  - wires
  - rods, beams, springs
  - pipelines
  - transactions
  - highways, airway routes
  - precedence constraints
  - function calls
  - hyperlinks
  - legal moves
  - friendships, movie casts
  - synapses
  - protein-protein interactions
  - bonds
Protein Interaction Network

Reference: Jeong et al, Nature Review | Genetics
The Internet as mapped by The Opte Project
http://www.opte.org
Input format. Movie followed by list of performers, separated by slashes.

% more movies.txt
...
Tin Men (1987)/DeBoy, David/Blumenfeld, Alan/.../Geppi, Cindy/Hershey, Barbara
Tirez sur le pianiste (1960)/Heymann, Claude/.../Berger, Nicole (I)
Titanic (1997)/Paxton, Bill/DiCaprio, Leonardo/.../Winslet, Kate
Titus (1999)/Weisskopf, Hermann/Rhys, Matthew/.../McEwan, Geraldine
To All a Good Night (1980)/George, Michael (II)/.../Gentile, Linda
To Be or Not to Be (1942)/Verebes, Ernö (I)/.../Lombard, Carole (I)
To Be or Not to Be (1983)/Brooks, Mel (I)/.../Bancroft, Anne
To Catch a Thief (1955)/París, Manuel/Grant, Cary/.../Kelly, Grace
To Die For (1989)/Bond, Steve (I)/Jones, Duane (I)/.../Maddalena, Julie
To Die For (1995)/Smith, Kurtwood/Kidman, Nicole/.../Tucci, Maria
To Die Standing (1990)/Sacha, Orlando/Anthony, Gerald/.../Rose, Jamie
To End All Wars (2001)/Kimura, Sakae/Ellis, Greg (II)/.../Sutherland, Kiefer
To Kill a Clown (1972)/Alda, Alan/Clavering, Eric/Lamberts, Heath/Danner, Blythe
To Live and Die in L.A. (1985)/McGroarty, Pat/Williams, Donnie/.../Dafoe, Willem
...

http://www.imdb.com/interfaces
Q. How to represent the movie-performer relationships?
A. Use a graph.

- **Vertex:** performer or movie.
- **Edge:** connect performer to movie.
### Graph API

**Graph data type.**

```java
public class Graph (graph with String vertices)

    Graph()                      // create an empty graph
    Graph(In in)                 // read graph from input stream
    void addEdge(String v, String w)  // add edge v-w
    Iterable<String> adjacentTo(String v)  // neighbors of v

    to support use with foreach
```

### Example Graph

![Graph Diagram](image)
Graph representation: use a symbol table.

- Key = name of vertex.
- Value = set of neighbors.
Set Data Type

Set data type. Unordered collection of distinct keys.

public class SET<Key extends Comparable<Key>>

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SET()</td>
<td>create a set</td>
</tr>
<tr>
<td>boolean isEmpty()</td>
<td>is the set empty?</td>
</tr>
<tr>
<td>void add(Key key)</td>
<td>add key to the set</td>
</tr>
<tr>
<td>boolean contains(Key key)</td>
<td>is key in the set?</td>
</tr>
</tbody>
</table>

Note: Implementations should also implement the Iterable<Key> interface to enable clients to access keys in sorted order with foreach loops

Q. How to implement?
A. Identical to symbol table, but ignore values.
```java
public class Graph {
    private ST<String, SET<String>> st;

    public Graph() {
        st = new ST<String, SET<String>>() {
            @Override
            public boolean contains(String v) {
                return st.get(v).size() > 0;
            }
        };
    }

    public void addEdge(String v, String w) {
        if (!st.contains(v)) addVertex(v);
        if (!st.contains(w)) addVertex(w);
        st.get(v).add(w);  // add w to v's set of neighbors
        st.get(w).add(v);  // add v to w's set of neighbors
    }

    private void addVertex(String v) {
        st.put(v, new SET<String>() {  // add new vertex v with no neighbors
            @Override
            public boolean contains(String w) {
                return false;
            }
        });
    }

    public Iterable<String> adjacentTo(String v) {
        return st.get(v);
    }
}
```
Second constructor. To read graph from input stream.

```java
public Graph(In in) {
    st = new ST<String, SET<String>>(());
    while (!in.isEmpty()) {
        String line = in.readLine();
        String[] names = line.split("/");
        for (int i = 1; i < names.length; i++)
            addEdge(names[0], names[i]);
    }
}
```

In in = new In("tiny.txt");
Graph G = new Graph(G, in);

% more tiny.txt
A/B/I
B/A/F
C/D/G/H
D/C
E/F/I
F/B/E/G/I
G/C/F/H
H/C/G
I/A/E/F
Performer and movie queries.
- Given a performer, find all movies in which they appeared.
- Given a movie, find all performers.

```java
public class MovieFinder {
    public static void main(String[] args) {
        In in = new In(args[0]);
        Graph G = new Graph(in);

        while (!StdIn.isEmpty()) {
            String v = StdIn.readLine();
            for (String w : G.adjacentTo(v))
                StdOut.println(w);
        }
    }
}
```
Graph Client: Movie Finder

% java MovieFinder action.txt
Bacon, Kevin
Death Sentence (2007)
Tremors (1990)

Roberts, Julia
I Love Trouble (1994)
Mexican, The (2001)
Ocean's Eleven (2001)

Eisgruber, Christopher

% java MovieFinder mpaa.txt
Bacon, Kevin
Air I Breathe, The (2007)
Air Up There, The (1994)
Animal House (1978)
Apollo 13 (1995)
Balto (1995)
Beauty Shop (2005)
Big Picture, The (1989)

... Sleepers (1996)
Starting Over (1979)
Stir of Echoes (1999)
Telling Lies in America (1997)
Trapped (2002)
Tremors (1990)
We Married Margo (2000)
Where the Truth Lies (2005)
White Water Summer (1987)
Wild Things (1998)
X-Men: First Class (2011)
Kevin Bacon Numbers

Bill Nighy was in “Pirate Radio” with Philip Seymour Hoffman

Philip Seymour Hoffman was in “Hunger Games” with Jennifer Lawrence

Jennifer Lawrence was in “X-Men: First Class” with Kevin Bacon

Bill Nighy was in “Total Recall” with John Cho

John Cho was in “Total Recall” with Kevin Bacon

Kevin Bacon was in “Total Recall” with John Cho
Oracle of Kevin Bacon

The Oracle of Bacon

Kevin Bacon to Buzz Mauro

- Buzz Mauro
  - Sweet Dreams (2005)
    - with Tatiana Ramirez
      - Interior de un silencio, El (2005)
        - with Andres Suarez
          - Carlita's Secret (2004)
            - with Paula Lemes (1)
              - Frost/Nixon (2008)
                - with Kevin Bacon
### Kevin Bacon Game

**Game.** Given an actor or actress, find **shortest** chain of movies connecting them to Kevin Bacon.

<table>
<thead>
<tr>
<th>Actor</th>
<th>Was in</th>
<th>With</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matthew MacFadyen</td>
<td>Pride and Prejudice</td>
<td>Keira Knightley</td>
</tr>
<tr>
<td>Keira Knightley</td>
<td>Love Actually</td>
<td>Bill Nighy</td>
</tr>
<tr>
<td>Bill Nighy</td>
<td>Pirate Radio</td>
<td>Philip S. Hoffman</td>
</tr>
<tr>
<td>Philip S. Hoffman</td>
<td>Hunger Games</td>
<td>Jennifer Lawrence</td>
</tr>
<tr>
<td>Jennifer Lawrence</td>
<td>X-Men: First Class</td>
<td>James McAvoy</td>
</tr>
<tr>
<td>James McAvoy</td>
<td>Last King of Scotland</td>
<td>Forest Whitaker</td>
</tr>
<tr>
<td>Forest Whitaker</td>
<td>The Air I Breathe</td>
<td>Kevin Bacon</td>
</tr>
<tr>
<td>Kevin Bacon</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Computing Bacon Numbers

How to compute. Find shortest path in performer-movie graph.
PathFinder API

Design principles.
• Decouple graph algorithm from graph data type.
• Avoid feature creep: don’t encrust Graph with search features; instead make a new datatype.
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";  // create object to return shortest paths
        PathFinder finder = new PathFinder(G, s);

        while (!StdIn.isEmpty()) {  // process queries
            String actor = StdIn.readLine();
            for (String v : finder.pathTo(actor))
                StdOut.println(v);
        }
    }
}

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

% java Bacon top-grossing.txt
Goldberg, Whoopi
Sister Act (1992)
Grodénchik, Max
Apollo 13 (1995)
Bacon, Kevin

eisgruber, Christopher
Computing Shortest Paths

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.
• ...

A

E

I

B

F

G

C

D

H
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.
Breadth First Search

- Breadth First Search
  - FIFO Queue
  - front
  - FIFO Queue
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);
                }
            }
        }
    }

    // other PathFinder methods go here
}
To find shortest path: follow \texttt{prev[]} from vertex \texttt{v} back to source \texttt{s}.

- Consider vertices: \texttt{v}, \texttt{prev[v]}, \texttt{prev[prev[v]]}, ..., \texttt{s}.
- Ex: shortest path from \texttt{C} to \texttt{A}: \texttt{C} – \texttt{G} – \texttt{F} – \texttt{B} – \texttt{A}
- Use stack to reverse order

```java
public Iterable<String> pathTo(String v) {
    Stack<String> path = new Stack<String>();
    while (dist.contains(v)) {
        path.push(v);
        v = prev.get(v);
    }
    return path;
}
```
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>pathTo</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>

*data as of April 9, 2007*

60MB data as of April 9, 2007.
**Exercise.** Compute histogram of Kevin Bacon numbers.

**Input.** ~2.6 million movies, ~5.7 million 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>2,769</td>
</tr>
<tr>
<td>2</td>
<td>305,215</td>
</tr>
<tr>
<td>3</td>
<td>1,021,901</td>
</tr>
<tr>
<td>4</td>
<td>253,177</td>
</tr>
<tr>
<td>5</td>
<td>20,060</td>
</tr>
<tr>
<td>6</td>
<td>2,033</td>
</tr>
<tr>
<td>7</td>
<td>297</td>
</tr>
<tr>
<td>8</td>
<td>25</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>∞</td>
<td>&gt;32,000</td>
</tr>
</tbody>
</table>

These are *very* hard to find!

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

*Data as of April 28, 2013*
Applications of Breadth First Search

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

Extensions. Google maps.
Erdös Numbers
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 …

<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 Graph
Conclusions

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.