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.
- Goal: deliver message.
- Restriction: can only forward to someone you know by first name.
- Outcome: message delivered with average of 5 intermediaries.

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.

Graph Data Type

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

<table>
<thead>
<tr>
<th>graph</th>
<th>vertices</th>
<th>edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>communication</td>
<td>telephones, computers</td>
<td>fiber optic cables</td>
</tr>
<tr>
<td>circuits</td>
<td>gates, registers, processors</td>
<td>wires</td>
</tr>
<tr>
<td>mechanical</td>
<td>joints</td>
<td>rods, beams, springs</td>
</tr>
<tr>
<td>hydraulic</td>
<td>reservoirs, pumping stations</td>
<td>pipelines</td>
</tr>
<tr>
<td>financial</td>
<td>stocks, currency</td>
<td>transactions</td>
</tr>
<tr>
<td>transportation</td>
<td>street intersections, airports</td>
<td>highways, airway routes</td>
</tr>
<tr>
<td>scheduling</td>
<td>tasks</td>
<td>precedence constraints</td>
</tr>
<tr>
<td>software systems</td>
<td>functions</td>
<td>function calls</td>
</tr>
<tr>
<td>internet</td>
<td>web pages</td>
<td>hyperlinks</td>
</tr>
<tr>
<td>games</td>
<td>board positions</td>
<td>legal moves</td>
</tr>
<tr>
<td>social relationship</td>
<td>people, actors</td>
<td>friendships, movie casts</td>
</tr>
<tr>
<td>neural networks</td>
<td>neurons</td>
<td>synapses</td>
</tr>
<tr>
<td>protein networks</td>
<td>proteins</td>
<td>protein-protein interactions</td>
</tr>
<tr>
<td>chemical compounds</td>
<td>molecules</td>
<td>bonds</td>
</tr>
</tbody>
</table>

### Kissing Network

Reference: Cosmopolitan, Nov. 2000

### One Week of Enron Emails

Finding Patterns In Corporate Chatter

Corporate analysts are warning about a tell-taleترون emails. Here is a map of week's email patterns in late 2007, when a more name suddenly appeared. Somepast found that the event's pattern differed greatly from others, suggesting different conversations and activities that might reveal untypical, that year and analysis of those messages.

### FCC Lobbying Graph

"The Evolution of FCC Lobbying Coalitions" by Pierre de Vries in IoSS Visualization Symposium 2010
Protein Interaction Network

Reference: Jeong et al, Nature Review | Genetics

ARPANET

Internet Movie Database

Input format. Movie followed by list of performers, separated by slashes.

- "more_movies.txt"
- Tin Men (1987)/DeBoy, David/Glumenfeld, Alan/... /Geppi, Cindy/Gershoy, Barbara
- Times sur le plianite (1960)/Haymann, Claude/.../Berger, Nicole (II)
- Titanic (1997)/Easton, Bill/DeCaprio, Leonardo/.../Hiselet, Kate
- Titus (1999)/Keating, Herman/Sky, Matthew/.../McDermott, Geraldine
- To All a Good Night (1980)/George, Michael (II)/.../Gentile, Linda
- To Be or Not to Be (1943)/Verhees, Ernö (II)/.../Lambard, Cécile (I)
- To Be or Not to Be (1983)/Brooks, Mel (II)/.../Bancroft, Anne
- To Catch a Thief (1955)/Sarkis, Reun (I)/.../Eck, Joanne
- To Die For (1989)/Bond, Steve (II)/Jones, Julianne (II)/.../Haddad, Julie
- To Die Standing (1990)/Sachs, Orlando/Davis, Anthony G./.../Barber, Jamie
- To End All Wars (2001)/Corine, Sasse/Killa, Greg (II)/.../Sutherland, Kiefer
- To Kill a Clown (1972)/Alda, Alan/Claverie, Eric/Lombardi, Bette/Cruzer, Kiefer
- To Live and Die in L.A. (1985)/McCrory, Pat/Williams, Donnie/.../Hayes, Willem
**Internet Movie Database**

**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)
    public Graph();  // create an empty graph
    public 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

    // imprtant txt file
    // 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
```

**Set Data Type**

**Set data type.** Unordered collection of distinct keys.

```java
public class SET<Key extends Comparable<Key>> {
    public SET();  // create a set
    boolean isEmpty();  // is the set empty?
    void add(Key key);  // add key to the set
    boolean contains(Key key);  // is key in the set?

    // 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.
Graph Implementation

```java
public class Graph {
    private ST<String, SET<String>> st;

    public Graph() {
        st = new ST<String, SET<String>>();
    }

    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>());
    }

    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]);
    }
}
```

Graph Client: Movie Finder

Perform 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);
        }
    }
}
```
Kevin Bacon Numbers

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>

Computing Bacon Numbers

Oracle of Kevin Bacon

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

Path finder API.

```java
public class PathFinder {
    PathFinder(Graph G, String s) { /* constructor */
        int distanceTo(String v) { /* length of shortest path from s to v in G */
            Iterable<String> pathTo(String v) { /* shortest path from s to v in G */
        }
    }
}
```

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"; /* create object to return shortest paths */
        PathFinder finder = new PathFinder(G, s);
        while (!StdIn.isEmpty()) /* process queries */ {
            String performer = StdIn.readLine();
            for (String v : finder.pathTo(s)) {
                StdOut.println(v);
            }
        }
    }
}
```

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.
- ...

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 Searcher: Preprocessing

```java
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: Finding the Path

To find shortest path: follow `prev[]` from vertex `v` back to source `s`.

- Consider vertices: `v`, `prev[v]`, `prev[prev[v]]`, …, `s`.
- Ex: shortest path from `C` to `A`:

```
C → G → F → B → A
```

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>all.txt</td>
<td>265,462</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>
<tr>
<td>0.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>0.21 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>G.txt</td>
<td>325,462</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 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>2,149</td>
</tr>
<tr>
<td>2</td>
<td>316,048</td>
</tr>
<tr>
<td>3</td>
<td>564,141</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>9</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>2,249</td>
</tr>
<tr>
<td>11</td>
<td>2,249</td>
</tr>
<tr>
<td>12</td>
<td>2,249</td>
</tr>
<tr>
<td>13</td>
<td>2,249</td>
</tr>
<tr>
<td>14</td>
<td>2,249</td>
</tr>
<tr>
<td>15</td>
<td>2,249</td>
</tr>
</tbody>
</table>

- 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.

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.

Erdős Numbers

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

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

Erdős Numbers

<table>
<thead>
<tr>
<th>Erdős #</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>32,380</td>
</tr>
<tr>
<td>7</td>
<td>30,431</td>
</tr>
<tr>
<td>8</td>
<td>2,214</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>&gt;15</td>
<td>4 billion+</td>
</tr>
</tbody>
</table>

Erdös Number
Erdös has a Kevin Bacon number of 4.

Erdös has a Bacon number!

Erdös has a Bacon number of 4.

% java Bacon cast.txt
Erdös, Paul
H 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.

Erdős-Bacon Numbers

Sum of your Erdős and Bacon numbers.
- For most people: infinity!
- But for some ...

Princeton Prof. of Computer Science Brian Kernighan
Erdős number 3
Brian -- Shen Lin -- Ron Graham -- Erdős
Bacon number 3!
Brian an extra in A Beautiful Mind w/Russell Crowe
Crowe in Cinderella Man w/Beau Starr
Starr in Where the Truth Lies w/Kevin Bacon

Erdős-Bacon number 6

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

Erdős number 4

Stage name: Natalie Portman
Bacon number 1
Erdős-Bacon number 6
Erdös-Bacon Numbers


Digression: Milgram’s Other Famous Experiment

Obedience to authority (Yale, 1961 - 1962).
- Role of punishment in learning.
- Experimenter: explains experiment to student.
- Student: repeat list of word pairs.
- Teacher: if student gets one wrong, administer shock in 15 volt increments.

65% of teachers punished learners to maximum of 450 volts.
- None stopped before 300 volts!

Kevin Bacon Game

Game. Given an actor or actress, find 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>Whoopi Goldberg</td>
<td>Ghost</td>
<td>Patrick Swayze</td>
</tr>
<tr>
<td>Patrick Swayze</td>
<td>Dirty Dancing</td>
<td>Jennifer Grey</td>
</tr>
<tr>
<td>Jennifer Grey</td>
<td>Ferris Bueller’s Day Off</td>
<td>Matthew Broderick</td>
</tr>
<tr>
<td>Matthew Broderick</td>
<td>The Road to Wellville</td>
<td>John Cusack</td>
</tr>
<tr>
<td>John Cusack</td>
<td>Bullets Over Broadway</td>
<td>Dianne West</td>
</tr>
<tr>
<td>Dianne West</td>
<td>Footloose</td>
<td>Kevin Bacon</td>
</tr>
<tr>
<td>Kevin Bacon</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>