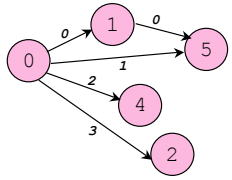
Is there a Science of Networks?

- 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?

From subsets to graphs with bits

- We'll consider SequenceSync APT
  - What is a "vertex" in the graph? Where are arcs?

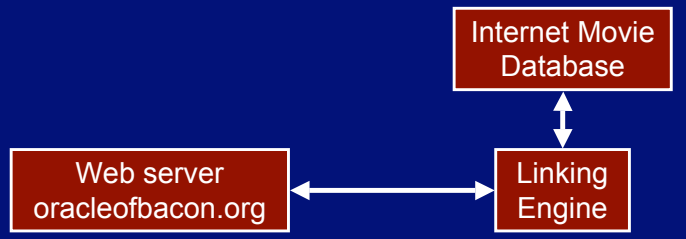


  - For state-0, we have {1,5,4,2} for transitions
- We'll consider a graph in which vertices are sets of states
  - Start with every possible state in our initial vertex

Six Degrees of Kevin Bacon

- Background
  - Stanley Milgram's Six Degrees of Separation?
  - Craig Fass, Mike Ginelli, and Brian Turtle invented it as a drinking game at Albright College
  - Brett Tjaden and Glenn Wasson put it on the web at UVA in 1995
  - Patrick Reynolds (Duke CS PhD) ran it since 1999
- Link people to Kevin Bacon based on movies they have been in together
  - Morgan Freeman was in Se7en with Brad Pitt
  - Brad Pitt was in Sleepers with Kevin Bacon
- Morgan Freeman has a Bacon number of 2
- Who has a Bacon number of 4 or more?

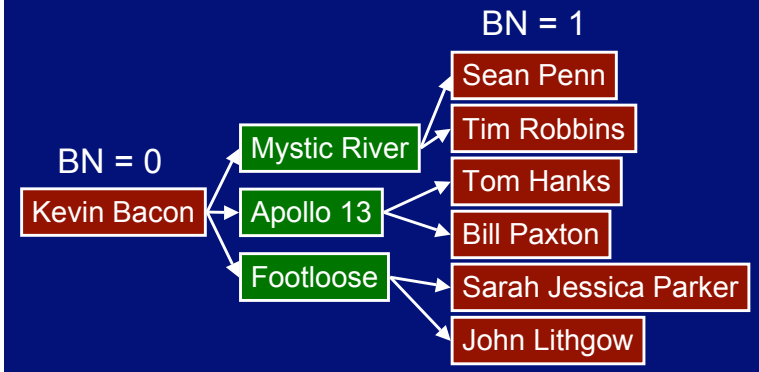
The Oracle of Bacon



- Linking Engine:
  - Is written in C
  - Stores all movie data in memory
  - Answers link, number, and center queries
  - Doesn't actually use Oracle™

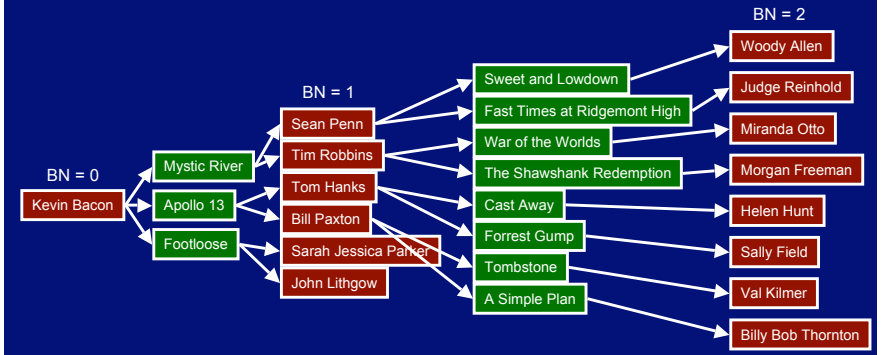
Inside the Linking Engine

- All three types of queries use breadth-first search



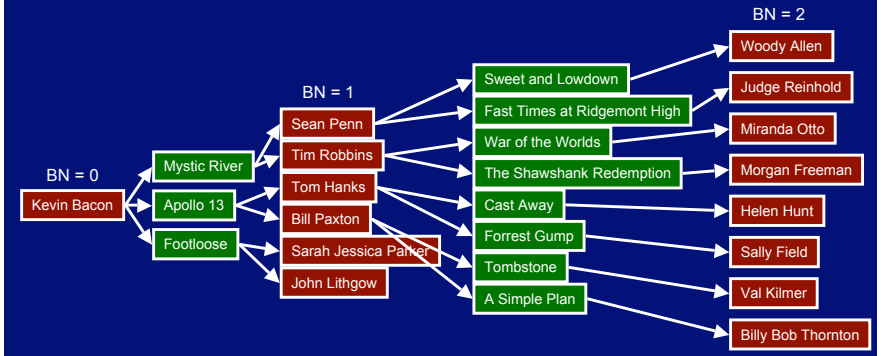
Inside the Linking Engine

- All three types of queries use breadth-first search



Inside the Linking Engine

- Link Val Kilmer to Kevin Bacon



Inside the Linking Engine

- How good a center is Kevin Bacon?
  - Average everyone's Bacon number
  - Currently 2.963, ranked 1222nd
- Who is exactly 2 hops away from Kevin Bacon?
  - This whole column
  - Currently 163,002 people

Caching

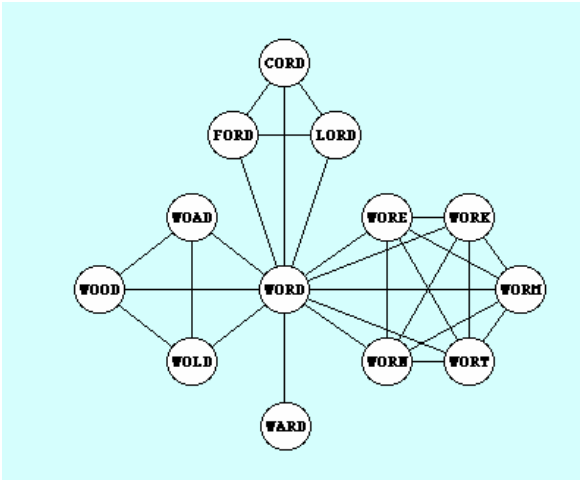
- Keep whole BFS trees for common queries
  - Takes about 4MB of memory each
- Perform 75-85% fewer BFS calculations
  - Faster website response time

actor	added	used	using	years	genres	direction	count
Bacon, Kevin	108253	11	Movies	1850-2050	010c1080	0	10482
Schwarzenegger, Arnold	31728	2341	Movies	1850-2050	010c1080	0	235
Bacon, Kevin	10	10	Movies	0-3000	00000000	0	1
Freeman, Kathleen (I)	22	22	Movies	1850-2050	010c1080	0	1
Menzel, Idina	61	45	Movies	1850-2050	010c1080	0	2
Grudzinski, Marie-Christine	117	63	Movies	1850-2050	010c1080	0	2
Rapp, Anthony	83	83	Movies	1850-2050	010c1080	0	1
Belmondo, Jean-Paul	84	84	Movies	1850-2050	010c1080	0	1
Thoms, Tracie	99	99	Movies	1850-2050	010c1080	0	1
Villechaize, Herve	105	105	Movies	1850-2050	010c1080	0	1
hits=13469 misses=4386 rate=75.44%							

Center of the Hollywood Universe

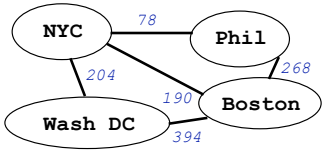
- 739,979 people can be connected to Bacon
- Is he the center of the Hollywood Universe?
  - Who is?
  - Who are other good centers?
  - What makes them good centers?
- Centrality
  - Closeness: the inverse average distance of a node to all other nodes
    - Geodesic: shortest path between two vertices
    - Closeness centrality: number of other vertices divided by the sum of all distances between the vertex and all others.
  - Degree: the degree of a node
  - Betweenness: a measure of how much a vertex is between other nodes

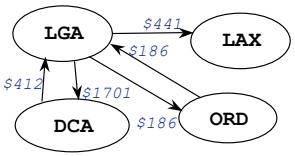
Word ladder



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 $v_0, v_1, \ldots, v_{n-1}$ 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

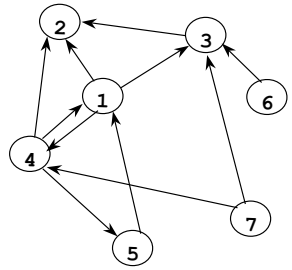




Graph questions/algorithms

- What vertices are reachable from a given vertex?
  - Two standard traversals: depth-first, breadth-first
  - Find *connected components*, groups of connected vertices
- Shortest path between any two vertices (weighted graphs?)
  - Breadth first search is storage expensive
  - Dijkstra's algorithm is efficient, uses a priority queue too!
- Longest path in a graph
  - No known efficient algorithm
- Visit all vertices without repeating? Visit all edges?
  - With minimal cost? Hard!

Vocabulary/Traversals



- Connected?
  - Connected components?
    - Weakly connected (directionless)
  - Degree: # edges incident a vertex
    - indegree (enter), outdegree (exit)
- Starting at 7 where can we get?
  - *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)
  - Rooms are stacked up, backtracking is really recursion
  - One alternative uses a queue: *breadth-first* search

Breadth first search

- In an unweighted graph this finds the shortest path between a start vertex and every vertex
  - Visit every node one away from start
  - Visit every node two away from start
    - This is every node one away from a node one away
  - Visit every node three away from start, …
- Put vertex on queue to start (initially just one)
  - Repeat: take vertex off queue, put all adjacent vertices on
  - Don't put a vertex on that's already been visited (why?)
  - When are 1-away vertices enqueued? 2-away? 3-away?
  - How many vertices on queue?

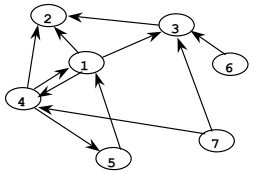
General graph traversal

```
COLLECTION_OF_VERTICES fringe;

fringe = INITIAL_COLLECTION;
while (!fringe.isEmpty()) {
  Vertex v = fringe.removeItem(QUEUE_FN);

  if (! MARKED(v)) {
    MARK(v);
    VISIT(v);
    for each edge (v,w) {
      if (NEEDS_PROCESSING(w))
        Add w to fringe according to QUEUE_FN;
    }
  }
}
```

Breadth-first search

- Visit each vertex reachable from some source in *breadth-first order*
- Like level-order traversal

```
Queue fringe;
fringe = {v};
while (!fringe.isEmpty()) {
  Vertex v = fringe.dequeue();
  if (! getMark(v)) {
    setMark(v);
    VISIT(v);
    for each edge (v,w) {
      if (MARKED(w))
        fringe.enqueue(w);
    }
  }
}
```



- How do we change to make depth-first search?
  - How does the order visited change?

How do we search in SequenceSync?

- Given a vertex (collection of states) how do we determine what vertex it's connected to?
  - Consider each transition from each state in our vertex (remember this is a set of states)
  - This yields a new set of states/vertex 1-away from vertex
- What does the code look like for bfs? When do we stop?

```
while (q.size() != 0){
    TreeSet<Integer> current = q.remove();
    for(int k=0; k < 4; k++){
        TreeSet<Integer> next = new TreeSet<Integer>();
        for(int val : current){
            next.add(matrix[val][k]);
        }
        q.add(next); // if not already seen
    }
}
```


Problems with approach?

- Creating sets and looking them up in map takes time
  - This solution times out, how to improve it?
  -
- Don't represent set of states explicitly, use sequence of bits
  - Similar to CodeBloat, advantages? Disadvantages?
  - How do we determine when we're done?
  - How to store distances (how is array like a map?)
- Rewrite solution to be efficient using int for set
  - Initial set is all ones, how to make this?

Greedy Algorithms

- A greedy algorithm makes a locally optimal decision that leads to a globally optimal solution
  - Huffman: choose two nodes with minimal weight, combine
    - Leads to optimal coding, optimal Huffman tree
  - Making change with American coins: choose largest coin possible as many times as possible
    - Change for $0.63, change for $0.32
    - What if we're out of nickels, change for $0.32?
- Greedy doesn't always work, but it does sometimes
- Weighted shortest path algorithm is *Dijkstra's* algorithm, greedy and uses priority queue

Dijkstra's Shortest Path Algorithm

- Similar to breadth first search, but uses a priority queue instead of a queue. Code below is for breadth first search

```
q.dequeue(vertex w)
foreach (vertex v adjacent to w)
  if (distance[v] == INT_MAX)        // not visited
  {
       distance[v] = distance[w] + 1;
       q.enqueue(v);
  }
```

- Dijkstra: Find minimal unvisited node, recalculate costs through node

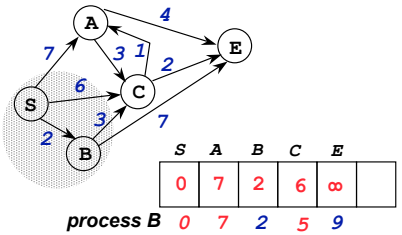
```
q.deletemin(vertex w)
foreach (vertex v adjacent to w)
  if (distance[w] + weight(w,v) < distance[v])
  {
       distance[v] = distance[w] + weight(w,v);
       q.insert(vertex(v, distance[v]));
  }
```


Shortest paths, more details

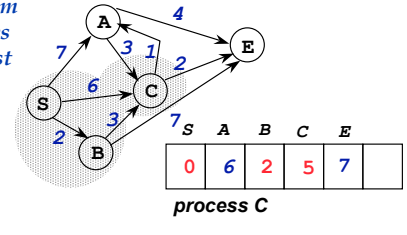
- Single-source shortest path
  - Start at some vertex S
  - Find shortest path to every reachable vertex from S
- A set of vertices is *processed*
  - Initially just S is processed
  - Each pass processes a vertex

After each pass, shortest path from S to any vertex using just vertices from processed set (except for last vertex) is always known

- Next processed vertex is closest to S still needing processing



	S	A	B	C	E
	0	7	2	6	∞
process B	0	7	2	5	9



	S	A	B	C	E
	0	6	2	5	7

process C

A Rose by any other name…C or Java?

- Why do we use Java in our courses (royal we?)
  - Object oriented
  - Large collection of libraries
  - Safe for advanced programming and beginners
  - Harder to shoot ourselves in the foot
- Why don't we use C++ (or C)?
  - Standard libraries weak or non-existant (comparatively)
  - Easy to make mistakes when beginning
  - No GUIs, complicated compilation model

Why do we learn other languages?

- Perl, Python, PHP, mySQL, C, C++, Java, Scheme, ML, …
  - Can we do something different in one language?
    - Depends on what different means.
    - In theory: no; in practice: yes
  - What languages do you know? All of them.
  - In what languages are you fluent? None of them
- In later courses why do we use C or C++?
  - Closer to the machine, we want to understand the machine at many levels, from the abstract to the ridiculous
    - Or at all levels of hardware and software
  - Some problems are better suited to one language
    - What about writing an operating system? Linux?

C++ on two slides

- Classes are similar to Java, compilation model is different
  - Classes have public and private *sections/areas*
  - Typically declaration in .h file and implementation in .cpp
    - Separate interface from actual implementation
    - Good in theory, hard to get right in practice
  - One .cpp file compiles to one .o file
    - To create an executable, we link .o files with libraries
    - Hopefully someone else takes care of the details
- We #include rather than import, this is a preprocessing step
  - Literally sucks in an entire header file, can take a while for standard libraries like iostream, string, etc.
  - No abbreviation similar to java.util.*;

C++ on a second slide

- We don't have to call new to create objects, they can be created "on the stack"
  - Using new creates memory "on the heap"
  - In C++ we need to do our own garbage collection, or avoid and run out of memory (is this an issue?)
- Vectors are similar to ArrayLists, pointers are similar to arrays
  - Unfortunately, C/C++ equate array with memory allocation
  - To access via a pointer, we don't use . we use ->
- Streams are used for IO, iterators are used to access begin/end of collection
  - ifstream, cout correspond to Readers and System.out

How do we read a file in C++ and Java?

```
Scanner s = new Scanner(new File("data.txt"));
TreeSet<String> set = new TreeSet<String>();
while (s.hasNext()){
   String str = s.next();
   set.add(str);
}
myWordsAsList = new ArrayList<String>(set);
```

```
string word;
set<string> unique;
ifstream input("data.txt");
while (input >> word){
   unique.insert(word);
}
myWordsAsVector = vector<string>(unique.begin(), unique.end());
```

- What are similarities? Differences?

How do we read a file in C?

```
FILE * file = fopen("/u/ola/data/poe.txt","r");
char buf[1024];
char ** words = (char **) malloc(5000*sizeof(char **));
int count = 0;
int k;
while (fscanf(file,"%s",buf) != EOF){
  int found = 0;   // look for word just read
  for(k=0; k < count; k++){
    if (strcmp(buf,words[k]) == 0){
      found = 1;
      break;
    }
  }
  if (!found){     // not found, add to list
    words[count] = (char *) malloc(strlen(buf)+1);
    strcpy(words[count],buf);
    count++;
  }
}
```

- What if more than 5000 words? What if string length > 1024? What if?
  - What is complexity of this code?