Paths, Shortest Paths, and Connected Components

UNIVERSITY OF
SAN FRANCISCO

James D. Wilson

ICPSR: Network Analysis I

Outline

- Paths and Shortest Paths
  - Diameter
  - Numeric Calculation
- Connected Components
- Traversing a Graph
  - Breadth-First Search (BFS)
  - Depth-First Search (DFS)
  - Dijkstra's Algorithm

Paths in Unweighted Graphs

Setting: Let $G = (V, E)$ be an unweighted (directed or undirected) graph with associated adjacency matrix $\mathbf{A}$. Let $u, v \in V$.

- A **path** between nodes u and v is a collection of edges

$$P(u, v) = \{(u, u_1), (u_1, u_2), \ldots, (u_{k-1}, u_k), (u_k, v)\} \subseteq E$$

that connects u with v.

Note:

- A path generally *does not* exist between every two vertices
- Often, *many* paths exist between two nodes

Paths in Unweighted Graphs

- The **path length** of a path $P(u, v)$:

$$\mathcal{L}(P(u, v)) = A_{u,u_1} + A_{u_k,v} + \sum_{i=1}^{k} A_{u_i,u_{i+1}}$$

Note:

- If no path exists between nodes u and v, we say $\mathcal{L}(P(u, v)) = \infty$

The Shortest Path and Diameter

- Suppose that P_1, P_2 . . ., P_k are the k paths between u and v
- The **shortest path** between u and v is the path P^*:

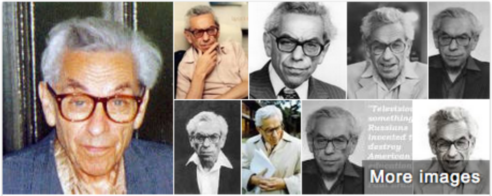
$$P^* = \text{argmin}_{j=1,\ldots,k}\{P_j\}$$

- The **diameter** of a graph G is *maximum* shortest path between any two nodes in the graph G.

The Shortest Path Length

- In many cases, we want the *length* of the shortest path between two nodes u, v.
- Typically referred to as the shortest path length, or geodesic distance between u and v
- **Example - Traveling Salesperson Problem**:
  - "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?"

Shortest paths and the Erdös - Number



Paul Erdős

Hungarian mathematician

Paul Erdős was a Hungarian mathematician. He was one of the most prolific mathematicians of the 20th century. He was known both for his social practice of mathematics and for his eccentric lifestyle. Wikipedia

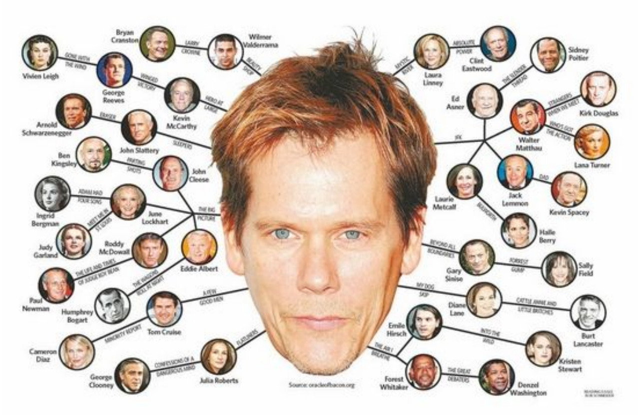
Born: March 26, 1913, Budapest, Hungary

Died: September 20, 1996, Warsaw, Poland

Education: Eötvös Loránd University (1934)

- Erdös # - the number of co-authors a writer is from publishing a paper with Erdös. (Mine is 4!)

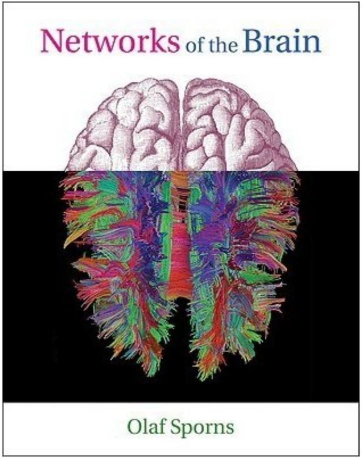
And... the Bacon Number!



Bacon # = distance (in films) an actor is from Kevin Bacon

And the... Erdös-Bacon # = Bacon # + Erdös #

Applications of Diameter - the Brain



Smaller diameter → faster communication between regions of the brain

The "Small-world" phenomenon in Social Networks

Setting: A natural demographic question: "how close are the people of an observed population?"

- In the early 1900s, this question gained a lot of interest, particularly among the people in the United States
- 1909 - Guglielmo Marconi (co-inventor of wireless telegraphy) hypothesized that people in the world were connected through acquantances by only 5 people.

The "Small-world" phenomenon in Social Networks

- Marconi's conjecture brought about the hype of the "Six Degrees of Separation" hypothesis, which social scientists wanted to test.
- 1967 - Stanley Milgram (of Harvard University) conducted the "small-world" experiment to test this hypothesis

The “Small-world” Experiment

Milgram sent cards to randomly chosen people in Omaha, Nebraska and Wichita, Kansas with instructions:

1) the name and details of a contact/target person in Boston, Massachusetts
2) if the person knew the contact (on a first-name basis), then they were asked to send the card directly to them

The "Small-world" Experiment

Milgram sent cards to randomly chosen people in Omaha, Nebraska and Wichita, Kansas with instructions:

3) if the person did not know the contact, then they were asked to send it to an acquantance whom they believed did know the contact
4) each person was asked to send their own contact information directly to Harvard

Work and results were published in "The Small World Problem" in *Psychology Today*, May 1967.

Results: “Six Degrees of Separation”

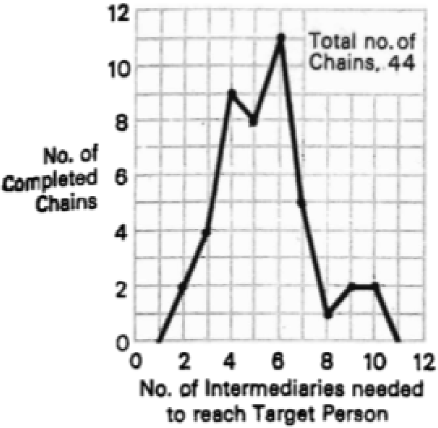
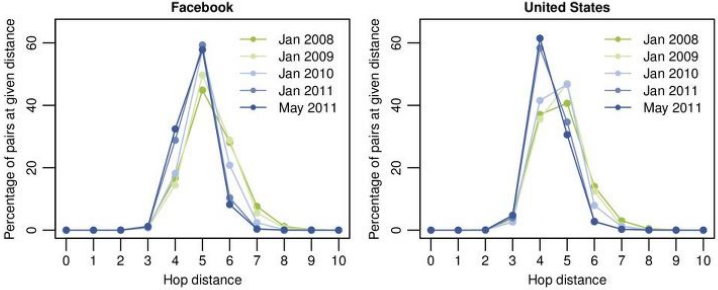


Figure: From the results in “The Small World Problem”



- Average distance separating every person on Facebook was ≈ 5 globally, and ≈ 4.5 in the U.S. in 2011
- (February 4, 2016): Users of Facebook are separated by 3.57 people, on average! (https://research.fb.com/three-and-a-half-degrees-of-separation/)

Connected Components

- Let $H = (V_H, E_H)$ be an induced subgraph of $G = (V, E)$.
- Let $I = (V_I, E_I)$ be the remainder of G: $I = G/H$.
- H is a **connected component** if
  - there is a path between all pair of vertices $u, v \in V_H$
  - there is no path between any $u \in V_H$ and $v \in V_I$

Connected Components

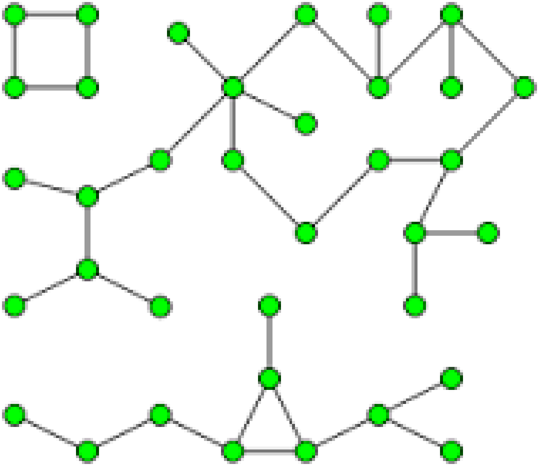


Figure: An undirected graph with 3 connected components

Connectedness

- A graph $G = (V, E)$ is said to be connected if it is possible to "travel" from one vertex to any other vertex. In other words, there is a path between each pair of vertices in V
- G is disconnected if it is not connected
- Singleton vertices that are not connected to any other vertices are called isolates
- In practice, one often analyzes connected components of a disconnected network separately

Connectedness

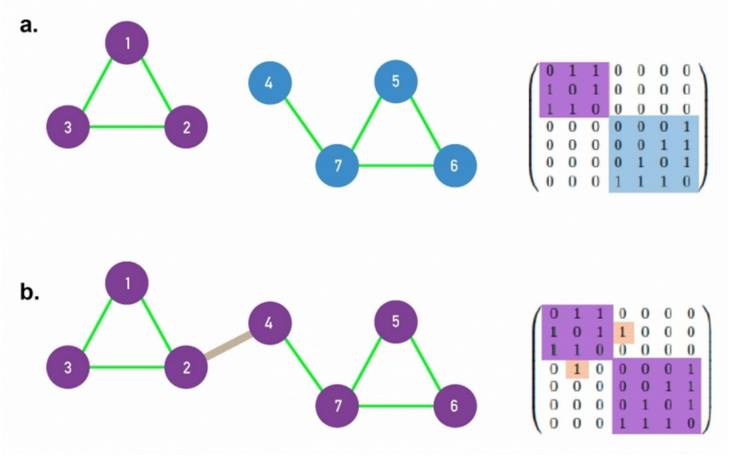


Figure: Disconnected and Connected Graphs

Computing Path Lengths and Shortest Paths

- Let $G = (V, E)$ be an unweighted graph with associated adjacency matrix **A**.
- There are three primary algorithms for computing the path length between vertices in V:
  - Matrix multiplication - $O(|V|^3)$
  - Breadth First Search - $O(|V| + |E|)$
  - Depth First Search

Path Lengths from Matrix Computation

Path Lengths in Undirected Graphs

The number of paths of length d between nodes u and v is

$$N_d(u,v) = (\mathbf{A}^d)_{uv}$$

It follows that the geodesic distance between u and v is given by the smallest d for which $N_d(u,v) > 0$.

- This is an incrediblly elegant solution to this calculation, but ...
- ... it relies on repeated matrix multiplication, which for a graph of size n requires $O(n^3)$ calculations.
- This is infeasible for large n!

Depth-First Search (DFS)

- An iterative algorithm for traversing or searching tree or graph data structures.
  - One starts at the root (selecting some arbitrary node as the root)
  - Explores as far as possible along each branch before backtracking
- First used in the 19th century by French mathematician Charles Pierre Trémaux as a strategy for solving mazes

Depth-First Search (DFS)

Pseudo-code

```
DepthFirstSearch(G, vertex, visited[ ]) {
  visited[vertex] = True         # mark vertex as visited
  for each in G[vertex]:         # for each neighbor of vertex
    if ( !visited[neighbor] ) {  # if neighbor has not been visited
      DepthFirstSearch(G, neighbor, visited[ ]) # visit neighbor
    }
}
```

Overall complexity: $O(|V| + |E|)$

Breadth-First Search (BFS)

- An iterative algorithm for traversing or searching tree or graph data structures
  - Starts at the tree root (or some arbitrary node of a graph)
  - Next explores (and records) the neighborhood nodes first, and then repeats the process at the neighborhood nodes
- First created by Konrad Zuse in his (rejected) Ph.D. thesis in 1945

Breadth-First Search (BFS)

Pseudo-code

```
BreadthFirstSearch(G, vertex, visited[ ]) {
 initialize q                      # queue
 initialize next                   # next vertex

 q.enqueue(vertex)                 # add vertex to queue
 while (q is not empty) {
   next = q.dequeue()              # remove from top of queue
   if ( !visited[next] ) {         # next has not been visited
     visited[next] = True          # mark next as visited
     for each neighbor in G[next]: # for all of next's neighbors
       q.enqueue(neighbor)         # add neighbor to queue
   }
 }
}
```

Overall complexity: $O(|V| + |E|)$

Dijkstra's Algorithm

- An iterative algorithm for finding the shortest paths between nodes in a (possibly weighted) graph
- First a single node as a source node and finds the shortest paths from the source to all other nodes in the graph, producing a shortest-path tree
- Repeats this process across all nodes and combines shortest-path trees
- Conceived by computer scientist Edsger W. Dijkstra in 1956 and published three years later

Dijkstra's Algorithm Pseudo-code

- keep a table to track distance from start node, previous node in path, and whether a node's place in path is known
- initialize distance for start node as 0 and all other distances to infinity
- initialize current node, i = start node.
- repeat until all nodes have been visited:
  - for all j adjacent to i,
    - if $\text{distance}_i + \text{weight}_{i,j} < \text{distance}_j$, update $\text{distance}_j = \text{distance}_i + \text{weight}_{i,j}$ and $\text{previous}_j = i$
  - update i = to the unvisited node that has the shortest distance to the start node

V	DIST	PREV	KNOWN
A	0		
B	∞		
C	∞		
...	...	...	...