A quick review

- Gene expression profiling
  - Which molecular processes/functions are involved in a certain phenotype (e.g., disease, stress response, etc.)

- The Gene Ontology (GO) Project
  - Provides shared vocabulary/annotation
  - Terms are linked in a complex structure

- Enrichment analysis:
  - Find the “most” differentially expressed genes
  - Identify **over-represented** annotations
  - Modified Fisher's exact test
Gene Set Enrichment Analysis

- Calculates a score for the enrichment of a entire set of genes
- Does not require setting a cutoff!
- Identifies the set of relevant genes!
- Provides a more robust statistical framework!

GSEA steps:

1. Calculation of an enrichment score (ES) for each functional category
2. Estimation of significance level
3. Adjustment for multiple hypotheses testing
Biological Networks Analysis

Introduction and Dijkstra’s algorithm

Genome 373
Genomic Informatics
Elhanan Borenstein
Biological networks

What is a network?

What networks are used in biology?

Why do we need networks (and network theory)?

How do we find the shortest path between two nodes?
A long history of network/graph theory!!

<table>
<thead>
<tr>
<th>Network theory</th>
<th>Graph theory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Social sciences</td>
<td>Computer science</td>
</tr>
<tr>
<td>(and Biological sciences)</td>
<td>Since 18th century!!!</td>
</tr>
<tr>
<td>Mostly 20th century</td>
<td>Modeling abstract systems</td>
</tr>
<tr>
<td>Modeling real-life systems</td>
<td>Solving “graph-related” questions</td>
</tr>
<tr>
<td>Measuring structure &amp; topology</td>
<td></td>
</tr>
</tbody>
</table>
What is a network?

- A map of interactions or relationships
- A collection of **nodes** and **links** (**edges**)
What is a network?

- A map of interactions or relationships
- A collection of **nodes** and **links** (edges)
Networks as Tools

- **The Seven Bridges of Königsberg**
  - Published by **Leonhard Euler, 1736**
  - Considered the first paper in graph theory
Edge properties and special topologies

- **Edges:**
  - Directed/undirected
  - Weighted/non-weighted
  - Simple-edges/Hyperedges

- **Special topologies:**
  - Trees
  - Directed Acyclic Graphs (DAG)
  - Bipartite networks
Transcriptional regulatory networks

- Reflect the cell’s genetic regulatory circuitry
  - **Nodes**: transcription factors and genes;
  - **Edges**: from TF to the genes it regulates
  - Directed; weighted?; “almost” bipartite

- Derived through:
  - Chromatin IP
  - Microarrays
  - Computationally
Metabolic networks

- Reflect the set of biochemical reactions in a cell
  - **Nodes**: metabolites
  - **Edges**: biochemical reactions
  - Directed; weighted?; hyperedges?

- Derived through:
  - Knowledge of biochemistry
  - Metabolic flux measurements
  - Homology

**S. Cerevisiae**
1062 metabolites
1149 reactions
Protein-protein interaction (PPI) networks

- Reflect the cell’s molecular interactions and signaling pathways (interactome)
  - **Nodes**: proteins
  - **Edges**: interactions (?)
  - Undirected

- High-throughput experiments:
  - Protein Complex-IP (Co-IP)
  - Yeast two-hybrid
  - Computationally

*S. Cerevisiae*
4389 proteins
14319 interactions
Other networks in biology/medicine
Non-biological networks

- **Computer related networks:**
  - WWW; Internet backbone
  - Communications and IP

- **Social networks:**
  - Friendship (facebook; clubs)
  - Citations / information flow
  - Co-authorship (papers)
  - Co-occurrence (movies; Jazz)

- **Transportation:**
  - Highway systems; Airline routes

- **Electronic/Logic circuits**

  Many many more...
The Kevin Bacon Number Game

Tropic Thunder (2008)

Tom Cruise
Robert Downey Jr.
Gwyneth Paltrow
Hope Davis
Kevin Bacon

Tropic Thunder
Iron Man
Proof
Flatliners
Kevin Bacon

Tropic Thunder
Iron Man
Frost/Nixon
Kevin Bacon

Frank Langella
The Paul Erdos Number Game

[Diagram showing interconnected network of mathematicians, including Paul Erdos, Vincent Moulton, Ronald L. Graham, and others]
The shortest path problem

- Find the minimal number of “links” connecting node A to node B in an undirected network
  - How many friends between you and someone on FB (6 degrees of separation, Erdös number, Kevin Bacon number)
  - How far apart are two genes in an interaction network
  - What is the shortest (and likely) infection path

- Find the shortest (cheapest) path between two nodes in a weighted directed graph
  - GPS; Google map
Dijkstra’s Algorithm

"Computer Science is no more about computers than astronomy is about telescopes."

Edsger Wybe Dijkstra 1930 – 2002
Dijkstra’s algorithm

- **Solves the single-source shortest path problem:**
  - Find the shortest path from a single source to **ALL** nodes in the network
  - Works on both **directed** and **undirected** networks
  - Works on both **weighted** and **non-weighted** networks

- **Approach:**
  - Iterative
  - Maintain shortest path to each intermediate node

- **Greedy algorithm**
  - ... but still guaranteed to provide optimal solution !!!
Dijkstra’s algorithm

1. Initialize:
   i. Assign a distance value, D, to each node. Set D to zero for start node and to infinity for all others.
   ii. Mark all nodes as unvisited.
   iii. Set start node as current node.

2. For each of the current node’s unvisited neighbors:
   i. Calculate tentative distance, $D^t$, through current node.
   ii. If $D^t$ smaller than D (previously recorded distance): $D \leftarrow D^t$
   iii. Mark current node as visited (note: shortest dist. found).

3. Set the unvisited node with the smallest distance as the next "current node" and continue from step 2.

4. Once all nodes are marked as visited, finish.
Dijkstra’s algorithm

- A simple synthetic network

1. Initialize:
   i. Assign a distance value, D, to each node.
      Set D to zero for start node and to infinity for all others.
   ii. Mark all nodes as unvisited.
   iii. Set start node as current node.

2. For each of the current node’s unvisited neighbors:
   i. Calculate tentative distance, $D^t$, through current node.
   ii. If $D^t$ smaller than D (previously recorded distance): $D \leftarrow D^t$
   iii. Mark current node as visited (note: shortest dist. found).

3. Set the unvisited node with the smallest distance as the next "current node" and continue from step 2.

4. Once all nodes are marked as visited, finish.
Dijkstra’s algorithm

- Initialization
- Mark A (start) as current node
Dijkstra’s algorithm

- Check unvisited neighbors of A

0+3 vs. ∞

0+9 vs. ∞
Dijkstra’s algorithm

- Update D
- Record path

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>0</td>
<td>9</td>
<td>3</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
</tbody>
</table>
Dijkstra’s algorithm

- Mark A as visited ...

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>0</td>
<td>9</td>
<td>3</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
</tbody>
</table>
Dijkstra’s algorithm

- Mark C as current (unvisited node with smallest D)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>∞</td>
<td>9</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>0</td>
<td>9</td>
<td>∞</td>
<td>3</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
</tbody>
</table>
Dijkstra’s algorithm

- Check unvisited neighbors of C

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>9</td>
<td>3</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>0</td>
<td>9</td>
<td>3</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
</tbody>
</table>

3+4 vs. 9
3+3 vs. ∞
3+2 vs. ∞
Dijkstra’s algorithm

- Update distance
- Record path

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>0</td>
<td>9</td>
<td>3</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>∞</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Dijkstra’s algorithm

- Mark C as visited
- Note: Distance to C is final!!

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>0</td>
<td>9</td>
<td>3</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>∞</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Dijkstra’s algorithm

- Mark E as current node
- Check unvisited neighbors of E
Dijkstra’s algorithm

- Update D
- Record path

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>0</td>
<td>9</td>
<td>3</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>∞</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>17</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Dijkstra’s algorithm

- Mark E as visited

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>0</td>
<td>9</td>
<td>3</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>∞</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>17</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Dijkstra’s algorithm

- Mark D as current node
- Check unvisited neighbors of D

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>0</td>
<td>9</td>
<td>3</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>∞</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>17</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Dijkstra’s algorithm

- Update D
- Record path (note: path has changed)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>0</td>
<td>9</td>
<td>3</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>17</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
</tbody>
</table>
Dijkstra’s algorithm

- Mark D as visited

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>0</td>
<td>9</td>
<td>3</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>17</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Dijkstra’s algorithm

- Mark B as current node
- Check neighbors

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>0</td>
<td>9</td>
<td>3</td>
<td>9</td>
<td>5</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>17</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>11</td>
</tr>
</tbody>
</table>

Diagram:

- Node A to B: 9
- Node B to C: 3
- Node B to D: 2
- Node B to E: 7
- Node B to F: 9
- Node A to C: 1
- Node C to D: 4
- Node C to E: 3
- Node D to E: 2
- Node D to F: 5
- Node E to F: 12
Dijkstra’s algorithm

- No updates..
- Mark B as visited

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>0</td>
<td>9</td>
<td>3</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>17</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>6</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
</tbody>
</table>

![Graph diagram](image)
Dijkstra’s algorithm

- Mark F as current
Dijkstra’s algorithm

- Mark F as visited

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>0</td>
<td>9</td>
<td>3</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>∞</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>11</td>
<td></td>
<td></td>
<td>11</td>
</tr>
</tbody>
</table>

```plaintext
D: 0
D: ∞,3
D: ∞,5
D: ∞,9,7
D: ∞,6
D: ∞,17,11
```
We are done!

- We now have:
  - Shortest path from A to each node (both length and path)
  - Minimum spanning tree

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>0</td>
<td>9</td>
<td>3</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>∞</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Will we always get a tree?
Can you prove it?
Computational Representation of Networks

List of edges: (ordered) pairs of nodes

\[ (A,C), (C,B), (D,B), (D,C) \]

Connectivity Matrix

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Object Oriented

- Name:A
  - ngr:
- Name:B
  - ngr:
  - p1
- Name:C
  - ngr:
  - p1
- Name:D
  - ngr:
  - p1
  - p2

Which is the most useful representation?