CS 4884: Connectivity Matrices and Node Degrees

T. M. Murali

February 5, 2019
Definition of an Undirected Graph

- **Weighted, undirected graph** $G = (V, E, w)$:
  - set $V$ of nodes.
  - set $E$ of edges.
    - Each element of $E$ is an unordered pair of nodes.
    - Exactly one edge between any pair of nodes ($G$ is not a multigraph).
    - $G$ contains no self loops, i.e., edges of the form $(u, u)$.
  - Each edge $(u, v)$ in $E$ has a weight $w(u, v) \in \mathbb{R}$
    - Weight of each edge is usually positive.
    - $G$ is *unweighted* if all edges have weight 1.
Definition of an Undirected Graph

- **Weighted, undirected graph** $G = (V, E, w)$:
  - set $V$ of nodes.
  - set $E$ of edges.
    - Each element of $E$ is an unordered pair of nodes.
    - Exactly one edge between any pair of nodes ($G$ is not a multigraph).
    - $G$ contains no self loops, i.e., edges of the form $(u, u)$.
  - Each edge $(u, v)$ in $E$ has a weight $w(u, v) \in \mathbb{R}$
    - Weight of each edge is usually positive.
    - $G$ is *unweighted* if all edges have weight 1.
**Definition of an Undirected Graph**

- **Weighted, undirected graph** \( G = (V, E, w) \):
  - set \( V \) of nodes.
  - set \( E \) of edges.
  - Each element of \( E \) is an unordered pair of nodes.
  - Exactly one edge between any pair of nodes (\( G \) is not a multigraph).
  - \( G \) contains no self loops, i.e., edges of the form \((u, u)\).
  - Each edge \((u, v)\) in \( E \) has a weight \( w(u, v) \in \mathbb{R} \)
  - Weight of each edge is usually positive.
  - \( G \) is **unweighted** if all edges have weight 1.
Definition of an Undirected Graph

- **Weighted, undirected graph** \( G = (V, E, w) \):
  - set \( V \) of nodes.
  - set \( E \) of edges.
    - Each element of \( E \) is an unordered pair of nodes.
    - Exactly one edge between any pair of nodes (\( G \) is not a multigraph).
    - \( G \) contains no self loops, i.e., edges of the form \((u, u)\).
  - Each edge \((u, v)\) in \( E \) has a weight \( w(u, v) \in \mathbb{R} \)
    - Weight of each edge is usually positive.
    - \( G \) is *unweighted* if all edges have weight 1.
Definition of a Directed Graph

- **Weighted, directed graph** $G = (V, E, w)$:
  - set $V$ of nodes.
  - set $E$ of edges.
    - Each element of $E$ is an ordered pair of nodes.
    - $e = (u, v)$: $u$ is the *tail* of the edge $e$, $v$ is its *head*; $e$ is directed from $u$ to $v$.
    - A pair of nodes $\{u, v\}$ may be connected by at most two directed edges: $(u, v)$ and $(v, u)$.
    - $G$ contains no self-loops.
  - Each edge $(u, v)$ in $E$ has a weight $w(u, v) \in \mathbb{R}$
    - Weight of each edge is usually positive.
    - $G$ is *unweighted* if all edges have weight 1.
Definition of a Directed Graph

- **Weighted, directed graph** $G = (V, E, w)$:
  - set $V$ of nodes.
  - set $E$ of edges.
    - Each element of $E$ is an ordered pair of nodes.
    - $e = (u, v)$: $u$ is the *tail* of the edge $e$, $v$ is its *head*; $e$ is directed from $u$ to $v$.
    - A pair of nodes $\{u, v\}$ may be connected by at most two directed edges: $(u, v)$ and $(v, u)$.
    - $G$ contains no self loops.
  - Each edge $(u, v)$ in $E$ has a weight $w(u, v) \in \mathbb{R}$
    - Weight of each edge is usually positive.
    - $G$ is *unweighted* if all edges have weight $1$. 

T. M. Murali  February 5, 2019  CS 4884: Computing the Brain
Definition of a Directed Graph

- **Weighted, directed graph** $G = (V, E, w)$:
  - set $V$ of nodes.
  - set $E$ of edges.
    - Each element of $E$ is an ordered pair of nodes.
    - $e = (u, v)$: $u$ is the *tail* of the edge $e$, $v$ is its *head*; $e$ is directed from $u$ to $v$.
    - A pair of nodes $\{u, v\}$ may be connected by at most two directed edges: $(u, v)$ and $(v, u)$.
    - $G$ contains no self loops.
  - Each edge $(u, v)$ in $E$ has a weight $w(u, v) \in \mathbb{R}$
    - Weight of each edge is usually positive.
    - $G$ is *unweighted* if all edges have weight 1.
Definition of a Directed Graph

- **Weighted, directed graph** $G = (V, E, w)$:
  - set $V$ of nodes.
  - set $E$ of edges.
    - Each element of $E$ is an ordered pair of nodes.
    - $e = (u, v)$: $u$ is the *tail* of the edge $e$, $v$ is its *head*; $e$ is *directed from $u$ to $v$*.
    - A pair of nodes $\{u, v\}$ may be connected by at most two directed edges: $(u, v)$ and $(v, u)$.
    - $G$ contains no self loops.
  - Each edge $(u, v)$ in $E$ has a weight $w(u, v) \in \mathbb{R}$
    - Weight of each edge is usually positive.
    - $G$ is *unweighted* if all edges have weight 1.
# Types of Brain Graphs

<table>
<thead>
<tr>
<th>Representation</th>
<th>Visualisation</th>
<th>Degrees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structural connectivity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Functional connectivity</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scale</th>
<th>Structural connectivity</th>
<th>Functional connectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microscale</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mesoscale</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Macroscale</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Types of Brain Graphs

<table>
<thead>
<tr>
<th>Representation</th>
<th>Structural connectivity</th>
<th>Functional connectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Microscale</strong></td>
<td>SEM, Tracking neurons</td>
<td></td>
</tr>
<tr>
<td><strong>Mesoscale</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Macroscale</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Semantics</th>
<th>Microscale</th>
<th>Mesoscale</th>
<th>Macroscale</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEM</td>
<td>inv</td>
<td>inv</td>
<td>inv</td>
</tr>
<tr>
<td>Tracking</td>
<td></td>
<td></td>
<td>inv</td>
</tr>
<tr>
<td>neurons</td>
<td></td>
<td>inv</td>
<td>inv</td>
</tr>
</tbody>
</table>

![Segmented neurons](image1.png) ![Layout graph](image2.png) ![Soma](image3.png) ![Axonal branch](image4.png) ![Dendritic branch](image5.png) ![Synaptic junction](image6.png) ![Connectivity graph](image7.png)
# Types of Brain Graphs

<table>
<thead>
<tr>
<th></th>
<th>Structural connectivity</th>
<th>Functional connectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Microscale</strong></td>
<td>SEM, Tracking neurons</td>
<td>Directed, weighted</td>
</tr>
<tr>
<td></td>
<td>Directed, weighted</td>
<td></td>
</tr>
<tr>
<td><strong>Mesoscale</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>** Macroscale**</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Representations:**

- **Segmented neurons**
- **Layout graph**
- **Soma:**
  - Neuron ID,
  - three-dimensional coordinates, type
- **Axonal branch:**
  - Neuron ID,
  - three-dimensional coordinates, diameter
- **Dendritic branch:**
  - Neuron ID,
  - three-dimensional coordinates, diameter
- **Synaptic junction:**
  - Pre- and postneuron ID,
  - three-dimensional coordinates, number of vesicles

**Visualisation Degrees:**

- **Microscale**
  - SEM, Tracking neurons
  - Directed, weighted

- **Mesoscale**
- **Macroscale**
  - Diffusion MRI, tractography
  - fMRI, correlations
  - Undirected, weighted
  - Weighted, can be negative
# Types of Brain Graphs

<table>
<thead>
<tr>
<th>Representation</th>
<th>Structural connectivity</th>
<th>Functional connectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Microscale</strong></td>
<td>SEM, Tracking neurons</td>
<td>Electrodes, correlations</td>
</tr>
<tr>
<td></td>
<td>Directed, weighted</td>
<td>Weighted, can be negative, can be directed</td>
</tr>
<tr>
<td><strong>Mesoscale</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Macroscale</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Segmented neurons](image1.png) ![Layout graph](image2.png) ![Soma](image3.png) ![Axonal branch](image4.png) ![Dendritic branch](image5.png) ![Synaptic junction](image6.png) ![Connectivity graph](image7.png)

T. M. Murali February 5, 2019 CS 4884: Computing the Brain
# Types of Brain Graphs

<table>
<thead>
<tr>
<th>Representation</th>
<th>Structural connectivity</th>
<th>Functional connectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microscale</td>
<td>SEM, Tracking neurons</td>
<td>Electrodes, correlations</td>
</tr>
<tr>
<td></td>
<td>Directed, weighted</td>
<td>Weighted, can be negative, can be directed</td>
</tr>
<tr>
<td>Mesoscale</td>
<td>Invasive tract tracing</td>
<td></td>
</tr>
<tr>
<td>Macroscale</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Types of Brain Graphs

<table>
<thead>
<tr>
<th></th>
<th>Structural connectivity</th>
<th>Functional connectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microscale</td>
<td>SEM, Tracking neurons Directed, weighted</td>
<td>Electrodes, correlations Weighted, can be negative, can be directed</td>
</tr>
<tr>
<td>Mesoscale</td>
<td>Invasive tract tracing Directed, weighted</td>
<td></td>
</tr>
<tr>
<td>Macroscale</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Types of Brain Graphs

<table>
<thead>
<tr>
<th>Representation</th>
<th>Structural connectivity</th>
<th>Functional connectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Microscale</strong></td>
<td>SEM, Tracking neurons</td>
<td>Electrodes, correlations</td>
</tr>
<tr>
<td></td>
<td>Directed, weighted</td>
<td>Weighted, can be negative, can be directed</td>
</tr>
<tr>
<td><strong>Mesoscale</strong></td>
<td>Invasive tract tracing</td>
<td>Did not discuss</td>
</tr>
<tr>
<td></td>
<td>Directed, weighted</td>
<td></td>
</tr>
<tr>
<td><strong>Macroscale</strong></td>
<td>Diffusion MRI, tractography</td>
<td>fMRI, correlations</td>
</tr>
<tr>
<td></td>
<td>Undirected, weighted</td>
<td>Weighted, can be negative, can be directed</td>
</tr>
</tbody>
</table>

*Did not discuss*
# Types of Brain Graphs

<table>
<thead>
<tr>
<th>Representation</th>
<th>Structural connectivity</th>
<th>Functional connectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Microscale</strong></td>
<td>SEM, Tracking neurons Directed, weighted</td>
<td>Electrodes, correlations Weighted, can be negative, can be directed</td>
</tr>
<tr>
<td><strong>Mesoscale</strong></td>
<td>Invasive tract tracing Directed, weighted</td>
<td>Did not discuss</td>
</tr>
<tr>
<td><strong>Macroscale</strong></td>
<td>Diffusion MRI, tractography</td>
<td></td>
</tr>
</tbody>
</table>

T. M. Murali  
February 5, 2019  
CS 4884: Computing the Brain
# Types of Brain Graphs

<table>
<thead>
<tr>
<th>Scale</th>
<th>Structural connectivity</th>
<th>Functional connectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microscale</td>
<td>SEM, Tracking neurons, Directed, weighted</td>
<td>Electrodes, correlations, Weighted, can be negative, can be directed</td>
</tr>
<tr>
<td>Mesoscale</td>
<td>Invasive tract tracing, Directed, weighted</td>
<td>Did not discuss</td>
</tr>
<tr>
<td>Macroscale</td>
<td>Diffusion MRI, tractography, Undirected, weighted</td>
<td></td>
</tr>
</tbody>
</table>
# Types of Brain Graphs

<table>
<thead>
<tr>
<th></th>
<th>Structural connectivity</th>
<th>Functional connectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Microscale</strong></td>
<td>SEM, Tracking neurons Directed, weighted</td>
<td>Electrodes, correlations</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Weighted, can be negative, can be directed</td>
</tr>
<tr>
<td><strong>Mesoscale</strong></td>
<td>Invasive tract tracing Directed, weighted</td>
<td>Did not discuss</td>
</tr>
<tr>
<td>** Macroscale**</td>
<td>Diffusion MRI, tractography Undirected, weighted</td>
<td>fMRI, correlations</td>
</tr>
</tbody>
</table>

![Brain Graph Diagram](image)
### Types of Brain Graphs

<table>
<thead>
<tr>
<th></th>
<th>Structural connectivity</th>
<th>Functional connectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Microscale</strong></td>
<td>SEM, Tracking neurons Directed, weighted</td>
<td>Electrodes, correlations Weighted, can be negative, can be directed</td>
</tr>
<tr>
<td><strong>Mesoscale</strong></td>
<td>Invasive tract tracing Directed, weighted</td>
<td>Did not discuss</td>
</tr>
<tr>
<td><strong>Macroscale</strong></td>
<td>Diffusion MRI, tractography Undirected, weighted</td>
<td>fMRI, correlations Weighted, can be negative can be directed</td>
</tr>
</tbody>
</table>

![Image of brain connectivity](image)
Thresholding and Binarisation

Human functional connectivity matrix from fMRI data.
Every element has a nonzero value.
Matrix after thresholding to retain only the 20% strongest weights.
Thresholding and Binarisation

Matrix after thresholding and binarisation.
Representing an Undirected Graph

Graph $G = (V, E)$ has two input parameters: $|V| = n, |E| = m$.

- We define the size of $G$ to be $m + n$. 

We can modify these ideas for directed graphs.
Representing an Undirected Graph

- Graph $G = (V, E)$ has two input parameters: $|V| = n, |E| = m$.
  - We define the size of $G$ to be $m + n$.
- Assume $V = \{1, 2, \ldots, n - 1, n\}$.
- Adjacency matrix representation: $n \times n$ Boolean matrix, where the entry in row $i$ and column $j$ is 1 iff the graph contains the edge $(i, j)$.
  - Space used is $O(n^2)$, which is optimal in the worst case.
- Check if there is an edge between node $i$ and node $j$ in $O(1)$ time.
- Iterate over all the edges incident on node $i$ in $O(n)$ time.

We can modify these ideas for directed graphs.
Representing an Undirected Graph

- Graph $G = (V, E)$ has two input parameters: $|V| = n, |E| = m$.
  - We define the size of $G$ to be $m + n$.
- Assume $V = \{1, 2, \ldots, n - 1, n\}$.
- **Adjacency matrix** representation: $n \times n$ Boolean matrix, where the entry in row $i$ and column $j$ is 1 iff the graph contains the edge $(i, j)$.
  - Space used is $O(n^2)$, which is optimal in the worst case.
  - Check if there is an edge between node $i$ and node $j$ in
Representing an Undirected Graph

- Graph $G = (V, E)$ has two input parameters: $|V| = n, |E| = m$.
  - We define the size of $G$ to be $m + n$.
- Assume $V = \{1, 2, \ldots, n-1, n\}$.
- **Adjacency matrix** representation: $n \times n$ Boolean matrix, where the entry in row $i$ and column $j$ is 1 iff the graph contains the edge $(i, j)$.
  - Space used is $O(n^2)$, which is optimal in the worst case.
  - Check if there is an edge between node $i$ and node $j$ in $O(1)$ time.
  - Iterate over all the edges incident on node $i$ in $O(d(v))$ time.
Representing an Undirected Graph

- Graph $G = (V, E)$ has two input parameters: $|V| = n, |E| = m$.
  - We define the size of $G$ to be $m + n$.
- Assume $V = \{1, 2, \ldots, n - 1, n\}$.
- **Adjacency matrix** representation: $n \times n$ Boolean matrix, where the entry in row $i$ and column $j$ is 1 iff the graph contains the edge $(i, j)$.
  - Space used is $O(n^2)$, which is optimal in the worst case.
  - Check if there is an edge between node $i$ and node $j$ in $O(1)$ time.
  - Iterate over all the edges incident on node $i$ in $O(n)$ time.

We can modify these ideas for directed graphs.
Representing an Undirected Graph

- Graph $G = (V, E)$ has two input parameters: $|V| = n, |E| = m$.
  - We define the size of $G$ to be $m + n$.
- Assume $V = \{1, 2, \ldots, n - 1, n\}$.
- *Adjacency matrix* representation: $n \times n$ Boolean matrix, where the entry in row $i$ and column $j$ is 1 iff the graph contains the edge $(i, j)$.
  - Space used is $O(n^2)$, which is optimal in the worst case.
  - Check if there is an edge between node $i$ and node $j$ in $O(1)$ time.
  - Iterate over all the edges incident on node $i$ in $O(n)$ time.
- *Adjacency list* representation: array $\text{Adj}$, where $\text{Adj}[v]$ stores the list of all nodes adjacent to $v$.
  - An edge $e = (u, v)$ appears twice: in $\text{Adj}[u]$ and $\text{Adj}[v]$. 
Representing an Undirected Graph

- Graph $G = (V, E)$ has two input parameters: $|V| = n, |E| = m$.
  - We define the size of $G$ to be $m + n$.
- Assume $V = \{1, 2, \ldots, n - 1, n\}$.
- **Adjacency matrix** representation: $n \times n$ Boolean matrix, where the entry in row $i$ and column $j$ is 1 iff the graph contains the edge $(i, j)$.
  - Space used is $O(n^2)$, which is optimal in the worst case.
  - Check if there is an edge between node $i$ and node $j$ in $O(1)$ time.
  - Iterate over all the edges incident on node $i$ in $O(n)$ time.
- **Adjacency list** representation: array $\text{Adj}$, where $\text{Adj}[v]$ stores the list of all nodes adjacent to $v$.
  - An edge $e = (u, v)$ appears twice: in $\text{Adj}[u]$ and $\text{Adj}[v]$.
  - $d(v) = \text{the number of neighbours of node } v$.
  - Space used is
Representing an Undirected Graph

- Graph $G = (V, E)$ has two input parameters: $|V| = n, |E| = m$.
  - We define the size of $G$ to be $m + n$.
- Assume $V = \{1, 2, \ldots, n - 1, n\}$.
- **Adjacency matrix** representation: $n \times n$ Boolean matrix, where the entry in row $i$ and column $j$ is 1 iff the graph contains the edge $(i, j)$.
  - Space used is $O(n^2)$, which is optimal in the worst case.
  - Check if there is an edge between node $i$ and node $j$ in $O(1)$ time.
  - Iterate over all the edges incident on node $i$ in $O(n)$ time.
- **Adjacency list** representation: array $\text{Adj}$, where $\text{Adj}[v]$ stores the list of all nodes adjacent to $v$.
  - An edge $e = (u, v)$ appears twice: in $\text{Adj}[u]$ and $\text{Adj}[v]$.
  - $d(v) =$ the number of neighbours of node $v$.
  - Space used is $O(n + \sum_{v \in G} d(v)) =$
Representing an Undirected Graph

- Graph $G = (V, E)$ has two input parameters: $|V| = n, |E| = m$.
  - We define the size of $G$ to be $m + n$.
- Assume $V = \{1, 2, \ldots, n - 1, n\}$.
- Adjacency matrix representation: $n \times n$ Boolean matrix, where the entry in row $i$ and column $j$ is 1 iff the graph contains the edge $(i, j)$.
  - Space used is $O(n^2)$, which is optimal in the worst case.
  - Check if there is an edge between node $i$ and node $j$ in $O(1)$ time.
  - Iterate over all the edges incident on node $i$ in $O(n)$ time.
- Adjacency list representation: array Adj, where Adj[$v$] stores the list of all nodes adjacent to $v$.
  - An edge $e = (u, v)$ appears twice: in Adj[$u$] and Adj[$v$].
  - $d(v) =$ the number of neighbours of node $v$.
  - Space used is $O(n + \sum_{v \in G} d(v)) = O(n + m)$, which is optimal for every graph.
  - Check if there is an edge between node $u$ and node $v$ in...
Representing an Undirected Graph

- Graph $G = (V, E)$ has two input parameters: $|V| = n, |E| = m$.
  - We define the size of $G$ to be $m + n$.

- Assume $V = \{1, 2, \ldots, n - 1, n\}$.

- **Adjacency matrix** representation: $n \times n$ Boolean matrix, where the entry in row $i$ and column $j$ is 1 iff the graph contains the edge $(i, j)$.
  - Space used is $O(n^2)$, which is optimal in the worst case.
  - Check if there is an edge between node $i$ and node $j$ in $O(1)$ time.
  - Iterate over all the edges incident on node $i$ in $O(n)$ time.

- **Adjacency list** representation: array $\text{Adj}$, where $\text{Adj}[v]$ stores the list of all nodes adjacent to $v$.
  - An edge $e = (u, v)$ appears twice: in $\text{Adj}[u]$ and $\text{Adj}[v]$.
  - $d(v) =$ the number of neighbours of node $v$.
  - Space used is $O(n + \sum_{v \in G} d(v)) = O(n + m)$, which is optimal for every graph.
  - Check if there is an edge between node $u$ and node $v$ in $O(d(u))$ time.
  - Iterate over all the edges incident on node $u$ in
Representing an Undirected Graph

- Graph $G = (V, E)$ has two input parameters: $|V| = n, |E| = m$.
  - We define the size of $G$ to be $m + n$.
- Assume $V = \{1, 2, \ldots, n - 1, n\}$.
- **Adjacency matrix** representation: $n \times n$ Boolean matrix, where the entry in row $i$ and column $j$ is 1 iff the graph contains the edge $(i, j)$.
  - Space used is $O(n^2)$, which is optimal in the worst case.
  - Check if there is an edge between node $i$ and node $j$ in $O(1)$ time.
  - Iterate over all the edges incident on node $i$ in $O(n)$ time.
- **Adjacency list** representation: array $\text{Adj}$, where $\text{Adj}[v]$ stores the list of all nodes adjacent to $v$.
  - An edge $e = (u, v)$ appears twice: in $\text{Adj}[u]$ and $\text{Adj}[v]$.
  - $d(v) = \text{the number of neighbours of node } v$.
  - Space used is $O(n + \sum_{v \in G} d(v)) = O(n + m)$, which is optimal for every graph.
  - Check if there is an edge between node $u$ and node $v$ in $O(d(u))$ time.
  - Iterate over all the edges incident on node $u$ in $O(d(u))$ time.

We can modify these ideas for directed graphs.
Representing an Undirected Graph

- Graph $G = (V, E)$ has two input parameters: $|V| = n, |E| = m$.
  - We define the size of $G$ to be $m + n$.
- Assume $V = \{1, 2, \ldots, n - 1, n\}$.
- **Adjacency matrix** representation: $n \times n$ Boolean matrix, where the entry in row $i$ and column $j$ is 1 iff the graph contains the edge $(i, j)$.
  - Space used is $O(n^2)$, which is optimal in the worst case.
  - Check if there is an edge between node $i$ and node $j$ in $O(1)$ time.
  - Iterate over all the edges incident on node $i$ in $O(n)$ time.
- **Adjacency list** representation: array $\text{Adj}$, where $\text{Adj}[v]$ stores the list of all nodes adjacent to $v$.
  - An edge $e = (u, v)$ appears twice: in $\text{Adj}[u]$ and $\text{Adj}[v]$.
  - $d(v)$ = the number of neighbours of node $v$.
  - Space used is $O(n + \sum_{v \in G} d(v)) = O(n + m)$, which is optimal for every graph.
  - Check if there is an edge between node $u$ and node $v$ in $O(d(u))$ time.
  - Iterate over all the edges incident on node $u$ in $O(d(u))$ time.
- We can modify these ideas for directed graphs.
Visualising Matrices
Visualising Matrices
Visualising Matrices
Anatomical Projection
Circular Layout
Force-Directed Layout
Node Degree

- Undirected graph $G = (V, E)$: degree $d(v)$ of a node $v$ is the number of edges in $E$ that are incident on $v$.

T. M. Murali February 5, 2019 CS 4884: Computing the Brain
Node Degree

- Undirected graph $G = (V, E)$: degree $d(v)$ of a node $v$ is the number of edges in $E$ that are incident on $v$.
  \[ d(v) = |\{u \text{ such that } (u, v) \in E\}| \]

- Directed graph $G = (V, E)$:

```text
k = 3
```

(a)
Node Degree

- Undirected graph $G = (V, E)$: degree $d(v)$ of a node $v$ is the number of edges in $E$ that are incident on $v$.
  $$d(v) = |\{u \text{ such that } (u, v) \in E\}|$$

- Directed graph $G = (V, E)$:
  - *in-degree* $d_{in}(v)$ of node $v$ is the number of edges with $v$ as the head.
  - *out-degree* $d_{out}(v)$ of node $v$ is the number of edges with $v$ as the tail.
  $$d_{in}(v) = |\{u \text{ such that } (u, v) \in E\}|$$
  $$d_{out}(v) = |\{u \text{ such that } (v, u) \in E\}|$$

- Textbook also defines *strength* of a node: total weight of edges incident on that node.
Node Degree

- Undirected graph $G = (V, E)$: degree $d(v)$ of a node $v$ is the number of edges in $E$ that are incident on $v$.
  
  
  $d(v) = |\{u \text{ such that } (u, v) \in E\}|$

- Directed graph $G = (V, E)$:
  - in-degree $d_{in}(v)$ of node $v$ is the number of edges with $v$ as the head.
  - out-degree $d_{out}(v)$ of node $v$ is the number of edges with $v$ as the tail.

  $d_{in}(v) = |\{u \text{ such that } (u, v) \in E\}|$
  
  $d_{out}(v) = |\{u \text{ such that } (v, u) \in E\}|$

- Textbook also defines strength of a node: total weight of edges incident on that node.
Degree Distribution

- A way to summarize information about a graph.
Degree Distribution

- A way to summarize information about a graph.
- *Degree distribution* of an undirected graph $G$: for every integer $k \geq 0$, the fraction $p(k)$ of nodes in $G$ whose degree is $k$. 
Degree Distribution

- A way to summarize information about a graph.
- **Degree distribution** of an undirected graph $G$: for every integer $k \geq 0$, the fraction $p(k)$ of nodes in $G$ whose degree is $k$.
- **Cumulative degree distribution** of $G$: for every integer $k \geq 0$, the fraction $P(k)$ of nodes in $G$ whose degree is at most $k$.
Degree Distribution

- A way to summarize information about a graph.
- **Degree distribution** of an undirected graph $G$: for every integer $k \geq 0$, the fraction $p(k)$ of nodes in $G$ whose degree is $k$.
- **Cumulative degree distribution** of $G$: for every integer $k \geq 0$, the fraction $P(k)$ of nodes in $G$ whose degree is at most $k$.
- Plotting the cumulative degree distribution can offer interesting insights into a graph.
Degree Distribution

- A way to summarize information about a graph.
- *Degree distribution* of an undirected graph $G$: for every integer $k \geq 0$, the fraction $p(k)$ of nodes in $G$ whose degree is $k$.
- *Cumulative degree distribution* of $G$: for every integer $k \geq 0$, the fraction $P(k)$ of nodes in $G$ whose degree is at most $k$.
- Plotting the cumulative degree distribution can offer interesting insights into a graph.
- What is the value of $\sum_k kp(k)$?
Degree Distribution

- A way to summarize information about a graph.
- *Degree distribution* of an undirected graph $G$: for every integer $k \geq 0$, the fraction $p(k)$ of nodes in $G$ whose degree is $k$.
- *Cumulative degree distribution* of $G$: for every integer $k \geq 0$, the fraction $P(k)$ of nodes in $G$ whose degree is at most $k$.
- Plotting the cumulative degree distribution can offer interesting insights into a graph.
- What is the value of $\sum_k k p(k)$?
- Define $n(k) = np(k)$, the number of nodes with degree $k$.

$$\sum_{k \geq 0} kp(k) = \frac{1}{n} \sum_{k \geq 0} kn(k)$$
Degree Distribution

- A way to summarize information about a graph.
- **Degree distribution** of an undirected graph $G$: for every integer $k \geq 0$, the fraction $p(k)$ of nodes in $G$ whose degree is $k$.
- **Cumulative degree distribution** of $G$: for every integer $k \geq 0$, the fraction $P(k)$ of nodes in $G$ whose degree is at most $k$.
- Plotting the cumulative degree distribution can offer interesting insights into a graph.
- What is the value of $\sum_k kp(k)$?
- Define $n(k) = np(k)$, the number of nodes with degree $k$.

$$\sum_{k\geq0} kp(k) = \frac{1}{n} \sum_{k\geq0} kn(k) = \frac{1}{n} \sum_{v \in V} d(v) = \frac{2m}{n}$$
Degree Distributions of Real-World Networks

- Degree distributions of many real-world networks follow a power law (Barabasi and Albert, 1999).

\[ p(k) = \Pr\{\text{degree} = k\} \sim k^{-\gamma} \]

- In most networks, \(2 < \gamma < 3\).
Degree Distributions of Real-World Networks

- Degree distributions of many real-world networks follow a power law (Barabasi and Albert, 1999).
  \[ p(k) = \Pr\{\text{degree} = k\} \sim k^{-\gamma} \]

- In most networks, \(2 < \gamma < 3\).
- Read Box 4.1 on pages 127–128 of the textbook on the ubiquity of power laws.
Degree Distributions of Real-World Networks

- Degree distributions of many real-world networks follow a power law (Barabasi and Albert, 1999).

\[ p(k) = \Pr\{\text{degree} = k\} \sim k^{-\gamma} \]

- In most networks, \(2 < \gamma < 3\).
- Read Box 4.1 on pages 127–128 of the textbook on the ubiquity of power laws.
- Broad-scale networks show power law behaviour over limited range of degree, e.g.,

\[ p(k) = \Pr\{\text{degree} = k\} \sim k^{-\gamma} e^{-k/k_c} \]
Example of Degree Distributions of Brain Networks

In/out degree distributions of the *C. elegans* neuronal network
Example of Degree Distributions of Brain Networks

Degree distribution of a 383-region macaque connectome collated from published tract-tracing studies.
Example of Degree Distributions of Brain Networks

Degree distribution of a 78-region human cortical connectome from diffusion MRI (+: data, solid: exponentially truncated power law, dashed: exponential, dotted: power law).