جامعة الإسكندرية
ALEXANDRIA UNIVERSITY
كلية الحاسبات وعلوم البيانات

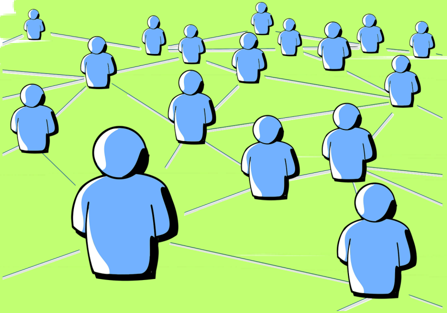
04.

Matrix Representation

Lecturer: Dr. Reem Essameldin Ebrahim

Introduction to Social Networks

Based on CS224W Analysis of Networks *Mining and Learning with Graphs: Stanford University*



In this Lecture

Topics to be covered are:

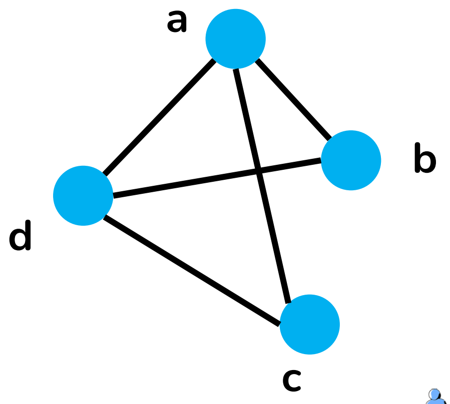
- From Graph to Matrix
- Relationship Matrix
- Adjacency Matrix
- Building Adiacency Matrix
- Representing graph: Other ways
- Matrix Operations

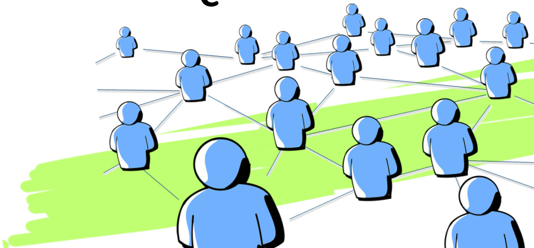
∗ Note that ∗ Adjacent vs Terminal

Two vertices u and v in an **undirected** graph G are called **adjacent** (or neighbors) in G if $\{u, v\}$ is an edge in G. The vertices u and v are called **endpoints** of the edge $\{u, v\}$.

Simple Example

Vertex	Adjacent Vertices
a	b, c, d
b	a, d
c	a, d
d	a, b, c





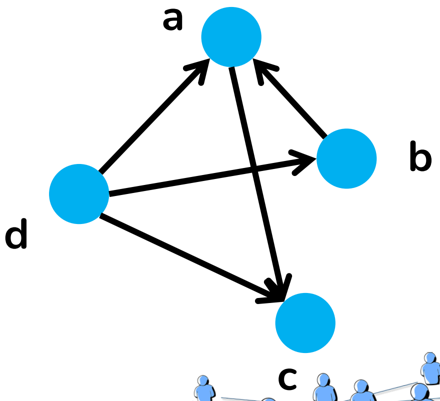
* Note that * Adjacent vs Terminal

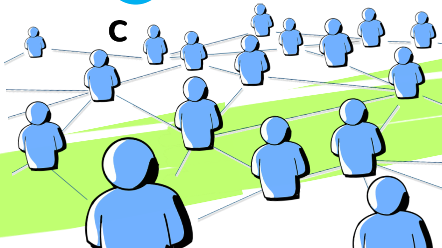
When (u, v) is an edge of the graph G with **directed** edges, u is said to be **adjacent to** v, and v is said to be **adjacent from** u.

$\Longrightarrow$ The vertex u is called the **initial** vertex of (u, v), and v is called the **terminal** vertex of (u, v). The **initial** vertex and **terminal** vertex of a **loop** are the same.

Simple Example

Initial Vertex	Terminal Vertices
a	c
b	a
c	
d	c, b, a



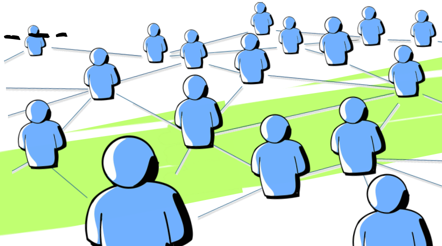


* From Graph to Matrix

Networks represented as graphs feel quite **intuitive** to us. With many tens, hundreds, thousands, or millions of nodes and edges, creating a graph with these amounts of data only results in what network analysts call a "**hairball**." Nothing can be understood through intuition.

Benefits of Matrix Representation

It provides us more **analytical** leverage when we switch to representing the network as a matrix. When we represent the network as a matrix, we are able to efficiently **calculate features** of the network that we would not be able to see with our eyes alone.



* Relationship Matrix

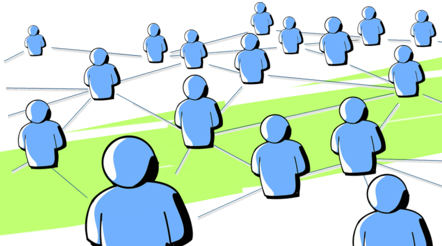
we will be focusing on a particular type of matrix called a **relationship matrix**. A relationship matrix is when, instead of asking what value of an attribute a case has, we ask about the value of describing how a case relates to another case.

Example of a Relationship Matrix

	Case 1	Case 2	Case 3
Case 1	Value 1	Value 4	Value 7
Case 2	Value 2	Value 5	Value 8
Case 3	Value 3	Value 6	Value 9

- A relationship matrix thus captures the relationship between two cases. In the above Figure, the relationship between Case 1 and Case 2 is Value 4, and the relationship between Case 2 and Case 1 is Value 2. Wait, **would that mean Value 2 and 4 are the same?** The answer is maybe. Depends on what type of relationship is being captured.

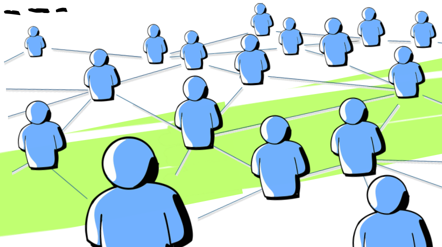
- We describe the **size** of the matrix by the number of rows by the number of columns. n is used to represent the number of rows and m is the number of columns. Thus, the above table is described as a **3x3** graph.



* Adjacency Matrix

There are many types of relationship matrices as the basic principles just stated can be varied to capture different underlying facets of relationships. The **adjacency matrix** is the most important, most commonly used way of formatting data in network analysis.

An adjacency matrix asks if two cases share a relationship or not. If two actors have a relationship, they are next to each other (**adjacent**) in the network, whereas if they do not share a relationship, they are not next to each other.



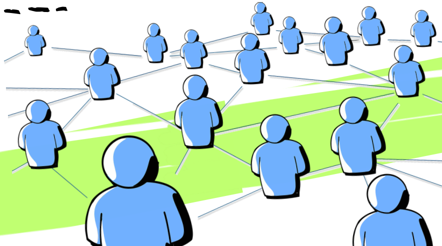
* Adjacency Matrix

There are many types of relationship matrices as the basic principles just stated can be varied to capture different underlying facets of relationships. The **adjacency matrix** is the most important, most commonly used way of formatting data in network analysis.

The Mathematical Definition

Let $G = (V, E)$ be a simple graph with $|V| = n$. Suppose that the vertices of G are listed in arbitrary order as $v_1, v_2, \ldots\ldots, v_n$. The **adjacency matrix** A (or A_G) of G, with respect to this listing of the vertices, is the n×n zero-one matrix. In other words, for an adjacency matrix $A = [a_{ij}]$, $a_{ij} = 1$ if $\{v_i, v_j\}$ is an edge of G, $a_{ij} = 0$ otherwise.

Note that: Size of matrix = number of nodes in the graph.

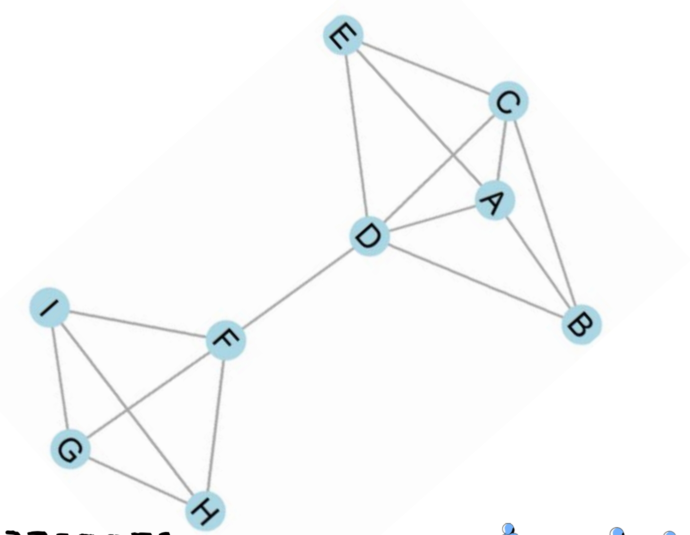


* Building Adjacency Matrix

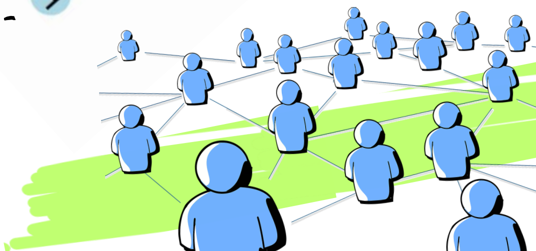
If we want to build an adjacency matrix of a network, we simply list all the actors in the rows and columns, and ask if the two share a relationship in order to fill in the values.

Illustrating Example

- The first step in building the adjacency matrix that represents the graph is to **list all the nodes** {A, B, C, D, E, F, G, H, I} as both a row and a column entry for each node.
- Next, one goes sequentially across the rows and columns, asking the question "does actor i have the relationship I am examining with actor j?" If the question asked is about the absence or presence of a relationship, 0's and 1's are used. If A has a relationship with B, the value 1 is marked. Otherwise, 0.



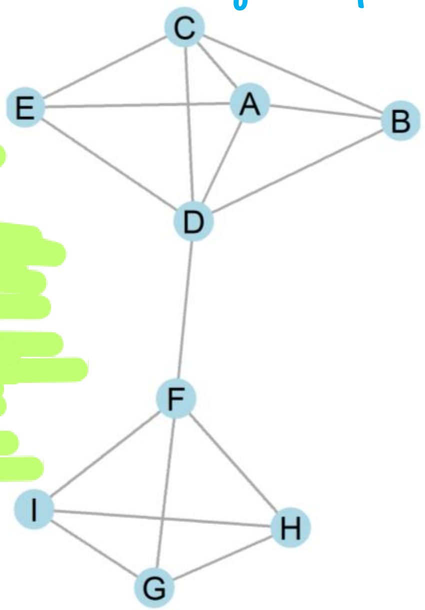
Note that: It's not sociologically meaningful for A to have a relationship with itself in most cases. For example, asking "Is A friends with A?" does not make much sense, but there are rare cases when it does, such as when A is a **group of people** and not an individual, and the relationship under examination might occur both within and between groups. These are called **reflexive-ties** or **loops**.



* Building Adjacency Matrix

If we want to build an adjacency matrix of a network, we simply list all the actors in the rows and columns, and ask if the two share a relationship in order to fill in the values.

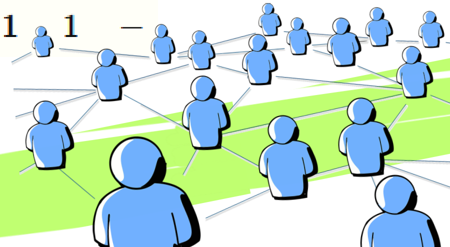
Illustrating Example



1

	A	B	C	D	E	F	G	H	I
A	–	1	1	1	1	0	0	0	0
B	1	–	1	1	0	0	0	0	0
C	1	1	–	1	1	0	0	0	0
D	1	1	1	–	1	1	0	0	0
E	1	0	1	1	–	0	0	0	0
F	0	0	0	1	0	–	1	1	1
G	0	0	0	0	0	1	–	1	1
H	0	0	0	0	0	1	1	–	1
I	0	0	0	0	0	1	1	1	–

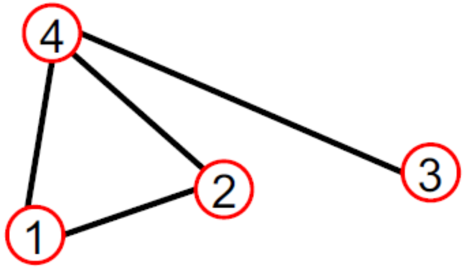
2



Building Adjacency Matrix

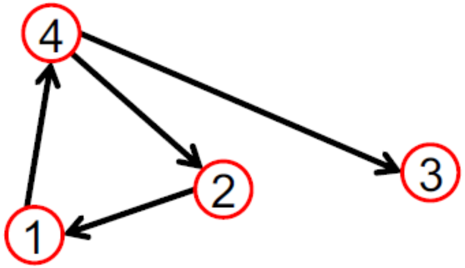
- If we want to build an adjacency matrix of a network, 1 we simply list all the actors in the rows and columns, 2 and ask if the two share a relationship in order to fill in the values.

Undirected

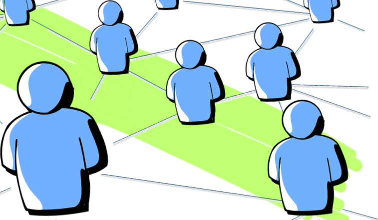


$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Directed



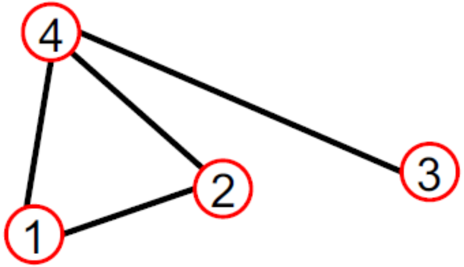
$$A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$



Building Adjacency Matrix

- If we want to build an adjacency matrix of a network, 1 we simply list all the actors in the rows and columns, 2 and ask if the two share a relationship in order to fill in the values.

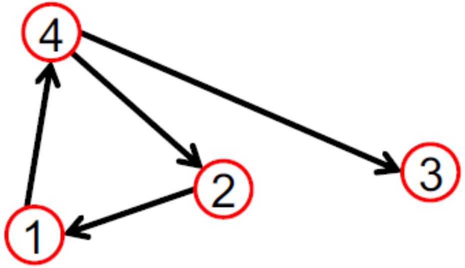
Undirected



$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

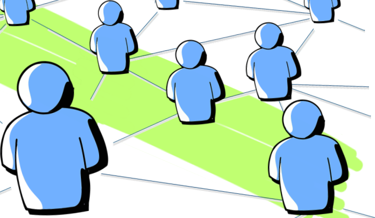
The adjacency matrix of an undirected network is **symmetric**, $A_{ij} = A_{ji}$, $A_{ii} = 0$.

Directed



$$A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

For a directed graph the matrix is **asymmetric**, $A_{ij} \neq A_{ji}$, $A_{ii} = 0$.



Building Adjacency Matrix

* If we want to build an adjacency matrix of a network, 1 we simply list all the actors in the rows and columns, 2 and ask if the two share a relationship in order to fill in the values.

Undirected

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

You can also compute **degrees** out of the adjacency matrix by basically asking how many nonzero elements are in a given row or column.

Node Degree:

$$k_i = \sum_{j=1}^{N} A_{ij}$$

$$k_j = \sum_{i=1}^{N} A_{ij}$$

Because the network is **symmetric**, the sum of a given row equals the sum of a given column.

Directed

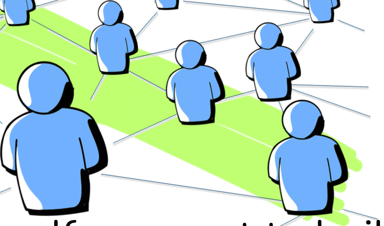
$$A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

Node Degree:

$$k_i^{out} = \sum_{j=1}^{N} A_{ij}$$

$$k_j^{in} = \sum_{i=1}^{N} A_{ij}$$

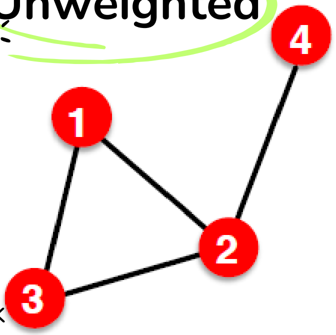
Outdegree is the sum of the numbers in the row and the indegree is the sum of the numbers in the column. E.g $k_4^{in} = 1$.



Building Adjacency Matrix

* If we want to build an adjacency matrix of a network, we simply list all the actors in the rows and columns, and ask if the two share a relationship in order to fill in the values.

Unweighted

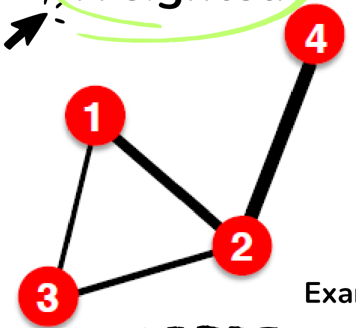


Examples: Friendship, Hyperlink

$$A_{ij} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

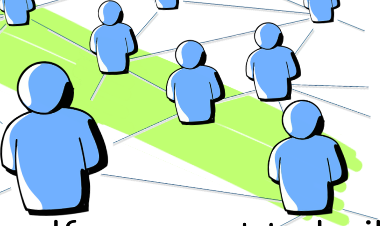
The adjacency matrix of an undirected, unweighted network is **symmetric**, $A_{ij} = A_{ji}$, A_{ii}= 0.

Weighted



Examples: Internet, Roads

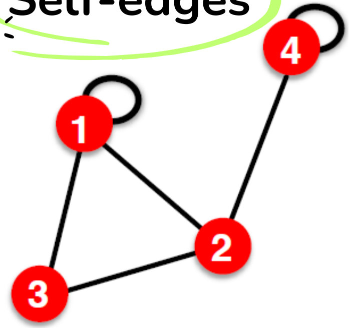
$$A_{ij} = \begin{pmatrix} 0 & 2 & 0.5 & 0 \\ 2 & 0 & 1 & 4 \\ 0.5 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 \end{pmatrix}$$



Building Adjacency Matrix

* If we want to build an adjacency matrix of a network, we simply list all the actors in the rows and columns, and ask if the two share a relationship in order to fill in the values.

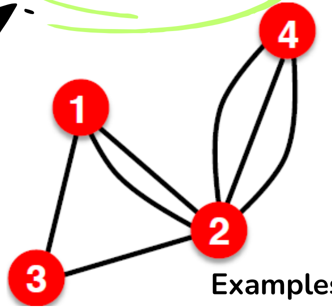
Self-edges



Examples: Proteins, Hyperlinks

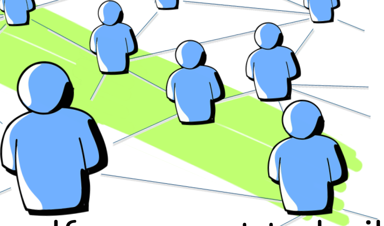
$$A_{ij} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

Multigraph



Examples: Communication, Collaboration

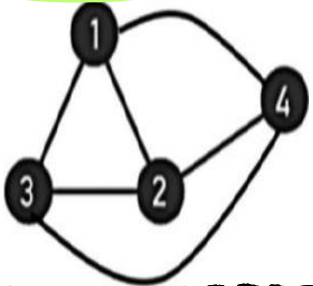
$$A_{ij} = \begin{pmatrix} 0 & 2 & 1 & 0 \\ 2 & 0 & 1 & 3 \\ 1 & 1 & 0 & 0 \\ 0 & 3 & 0 & 0 \end{pmatrix}$$



Building Adjacency Matrix

* If we want to build an adjacency matrix of a network, 1 we simply list all the actors in the rows and columns, 2 and ask if the two share a relationship in order to fill in the values.

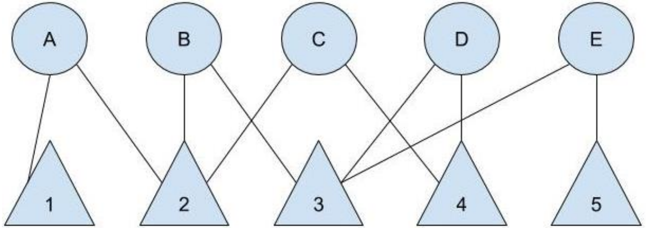
Complete



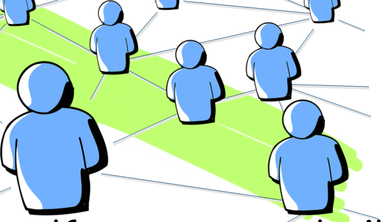
$$A_{ij} = \begin{matrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{matrix}$$

$$\boxed{A_{i\neq j} = 1;\ A_{ii} = 0}$$

Bipartite



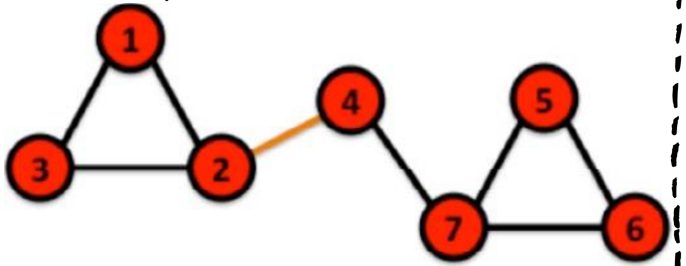
	1	2	3	4	5
A	1	1	0	0	0
B	0	1	1	0	0
C	0	1	0	1	0
D	0	0	1	1	0
E	0	0	1	0	0



Building Adjacency Matrix

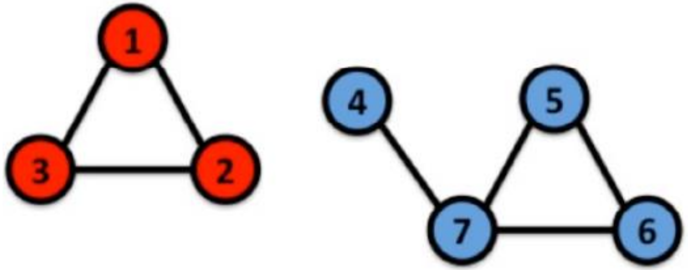
* If we want to build an adjacency matrix of a network, 1 we simply list all the actors in the rows and columns, 2 and ask if the two share a relationship in order to fill in the values.

Connected



$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Disconnected



$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

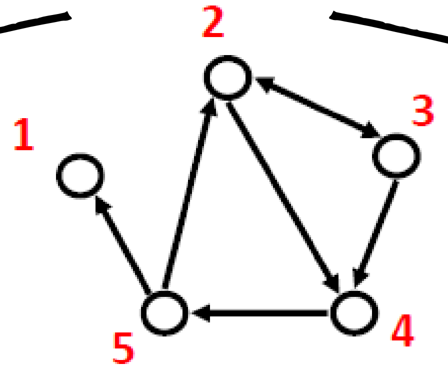
Adjacency Matrices are Sparse

If we visualize an adjacency matrix, this is a giant matrix and we can see the pattern of this matrix. This is how the network looks, basically it is empty with some dots here and there. The question is what would we learn from this type of data? We will do algorithms to tell us what is behind this shape.

A real network visualization

Representing Graph

Edge List



* In lecture 3 while discussing the special graphs, we have seen another way of representing graphs; where relationships are represented as a set of edges.

⟹ The edge list for the given graph is:

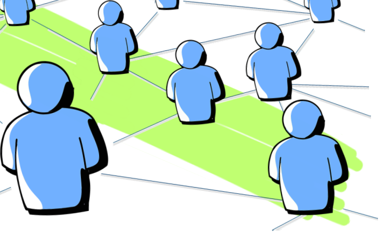
(2, 3)
(2, 4)
(3, 2)
(3, 4)
(4, 5)
(5, 1)
(5, 2)

Adjacent List

* It allows us to quickly retrieve all neighbors of a given node. It is easier to work with if network is:
  - Large
  - Sparse

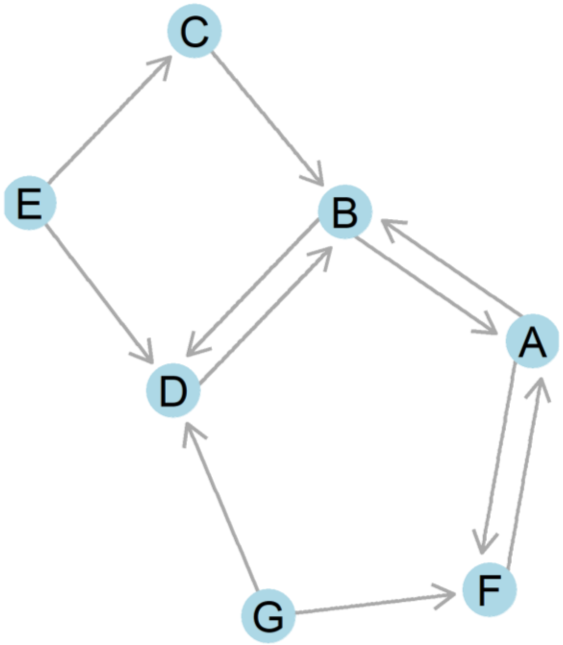
⟹ The adjacent list for the given graph is:

1:
2: 3, 4
3: 2, 4
4: 5
5: 1, 2

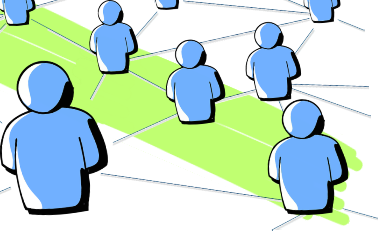


Test Yourself

* For the given **directed graph**, find the corresponding **asymmetric** adjacency matrix.

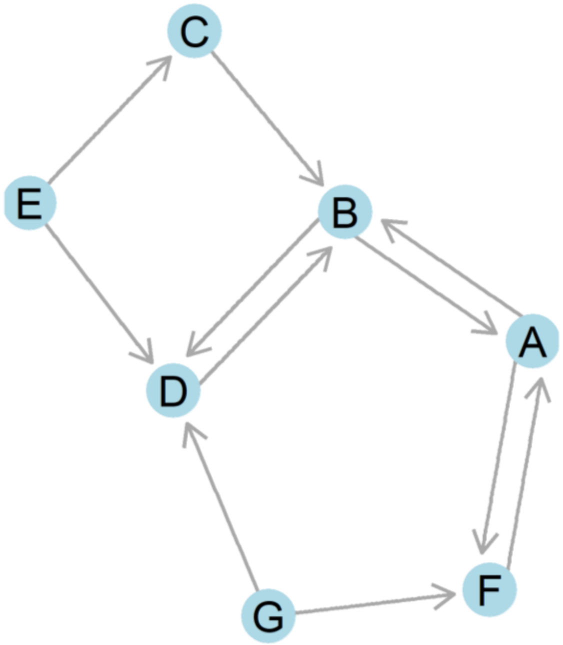


Solution:



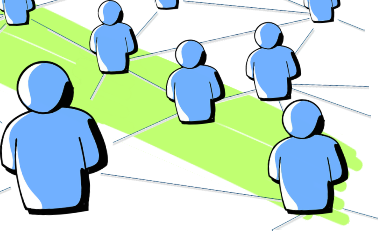
Test Yourself

* For the given **directed graph**, find the corresponding **asymmetric** adjacency matrix.



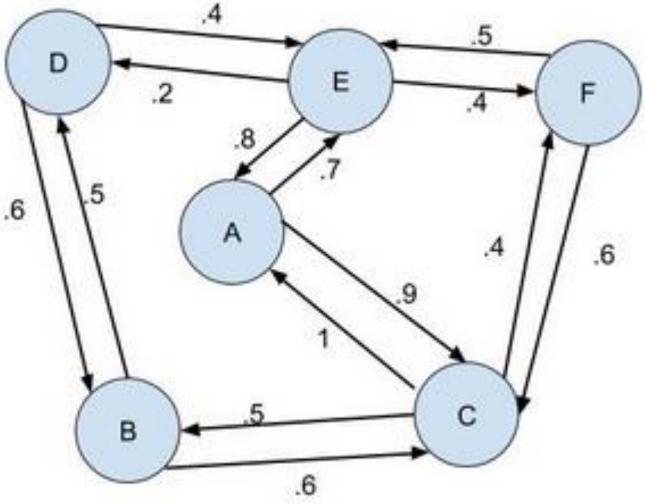
Solution:

	A	B	C	D	E	F	G
A	–	1	0	0	0	1	0
B	1	–	0	1	0	0	0
C	0	1	–	0	0	0	0
D	0	1	0	–	0	0	0
E	0	0	1	1	–	0	0
F	1	0	0	0	0	–	0
G	0	0	0	1	0	1	–

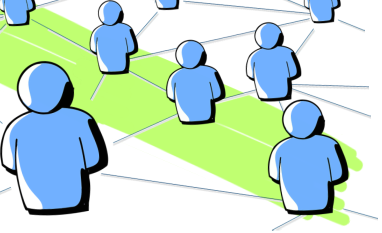


Test Yourself

* For the given **weighted graph**, find the corresponding **weighted** adjacency matrix.

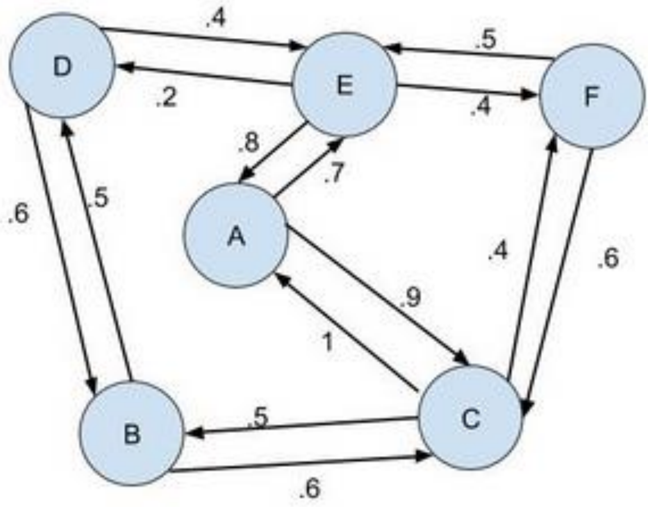


Solution:



Test Yourself

* For the given **weighted graph**, find the corresponding **weighted** adjacency matrix.



Solution:

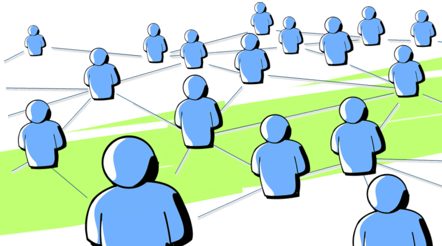
	A	B	C	D	E	F
A	—	0	.9	0	.7	0
B	0	—	.6	.5	0	0
C	1	.5	—	0	0	.4
D	0	.6	0	—	.4	0
E	.8	0	0	.2	—	.4
F	0	0	0	.6	.5	—

* Matrix Operations

Taking the matrix and performing mathematical **operations** on the matrix can actually unveil patterns or **features** about the social world that we are trying to understand through the network.

- While the matrices we've worked with so far are simple, recall that in **reality networks** can be as **complicated** as the social world they represent. e.g. Facebook's friendship networks, the connections between servers on the World Wide Web, and airline transportation networks have millions of nodes and edges.

- It's important to understand some of the **fundamentals** of how to manipulate a matrix if we were to ever look at networks beyond many of the simple examples.



Matrix Operations

Addition

- By taking two matrices of the same dimension nXm, add up each corresponding cell i,j in the two matrices, and return the result into the same cell i,j in a new matrix size nxm.

$$\begin{bmatrix} 4 & 2 & 6 \\ 3 & -1 & 3 \end{bmatrix} + \begin{bmatrix} 5 & 3 & 7 \\ 7 & 3 & 4 \end{bmatrix}$$

$$= \begin{bmatrix} 9 & 8 & 13 \\ 10 & 2 & 7 \end{bmatrix}$$

Note that: Only matrices with the same dimension nxm are capable of being summed together. Subtraction of two matrices works in the same way as addition.

Scalar Multiplication

Taking a single scalar, one can shape the original matrix into an essentially larger or smaller version of itself, the scale changing based on the size of the scalar.

$$3 \begin{bmatrix} 5 & 6 & 3 \\ 3 & 2 & 4 \end{bmatrix}$$

$$= \begin{bmatrix} 15 & 18 & 9 \\ 9 & 6 & 12 \end{bmatrix}$$

Matrix Multiplication

- When we multiply two matrices together, we are fundamentally multiplying along **rows** in the first matrix, and **columns** in the second matrix.

$$[n_1 * m_1][n_2 * m_2]$$

$$\begin{bmatrix} 5 & 6 & 3 \\ 3 & 2 & 2 \end{bmatrix} \begin{bmatrix} 5 & 6 \\ 5 & 6 \\ 5 & 6 \end{bmatrix} = \begin{bmatrix} 70 & 88 \\ 35 & 42 \end{bmatrix}$$

Note that: Matrix multiplication is not commutative in the same way normal multiplication is. Multiplying two matrices together, in different orders, will not give the same results.