# Mathematics behind Internet, MAA507

Sergei Silvestrov, Christopher Engström

January 24, 2013

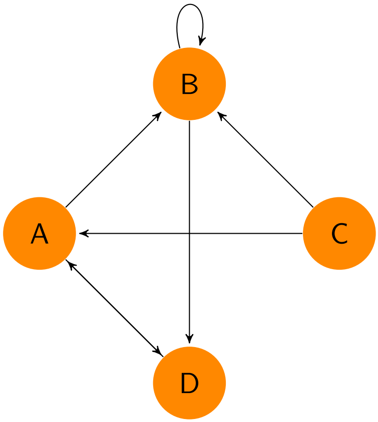
# MAA507 Mathematics behind Internet

Todays lecture:

- Introduction to graph theory, adjacency and distance matrix.
- Connectivity and Irreducibility.
- If time, shortest path and Djikstra's algorithm.

# Introduction to Graph-theory

A graph is a collection of vertices (nodes) and edges (links) such as the one below.



# Introduction to Graph-theory



There are many types of graphs those we will work with are:

- **Simple** graphs, where we only allow a single edge from one vertice to another.
- **Undirected** graphs, where edges do not have a direction so we are only interested in if there is a edge between two vertices.
- **Directed graphs**, where edges do have a direction, a edge $A \to B$ does not necessary mean there is a edge $B \to A$.
- **Weighted graph**, where we assign (positive) weights as scalars to every edge, often denoting some kind of distance.

# Adjacency matrix, other applications

A undirected or directed graph can be used to represent other things such as:

- Links between homepages or servers.
- Electrical or water flow networks.
- Road or rail networks.
- Linguistic relations between words or phrases.

# Paths and cycles in simple graphs.

## Definition

A **path** in a graph is a sequence of edges connecting a sequence of vertices.

- A **cycle** is a path in which the first and last vertice is the same.
- A **simple** path or cycle is a path in which no vertice is repeated except for the first and last vertice in a simple cycle.
- The **length** of a path is the number of edges used by the path, counting multiple edges multiple times.

- Often in literature "simple" is implied such that a path usually means a simple path.

# Paths and cycles in simple graphs.

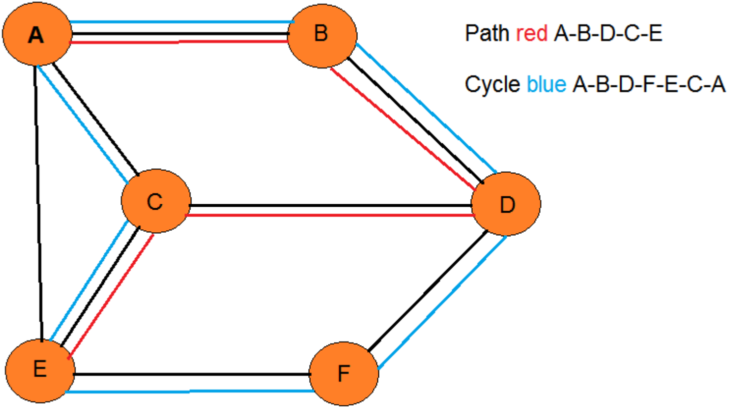



Figure: Example of a path (red) and cycle (blue) in a undirected graph

# Connectivity

- Two vertices is said to be **connected** if there exists a path containing both vertices in the undirected graph.
- A graph is said to be connected if it is **connected** for all pair of vertices in the undirected graph.
- If there is also a path from every vertice $u$ to every vertice $v$ in the directed graph we say that it's **strongly connected**.

# Connectivity: connected components

A **connected component** in a undirected graph is a maximal part of the graph where all nodes are connected with eachother.

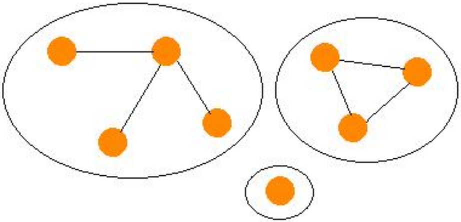


Figure: A undirected graph with 3 connected components

# Connectedness: strongly connected components

A **strongly connected component** is a part of the graph which is strongly connected.

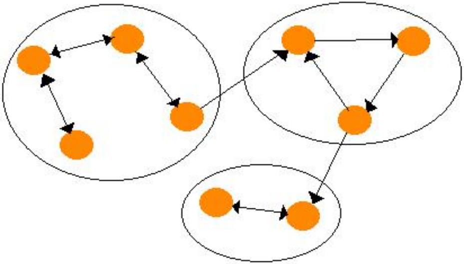


Figure: A directed graph with 3 strongly connected components

# Graphs and the adjacency matrix

How can we represent a (simple) graph using a matrix?

- Multiple ways such as the adjacency matrix, distance matrix, incidence matrix, Laplacian matrix, etc.
- Which graph type and which matrix representation you use depend on your graph and application.
- We will start today by taking a look at the adjacency and distance matrix.

# Graphs and the adjacency matrix

Finding connected components, or find if a path exist between two vertices is often needed in practice. For example in answering questions such as:

- If I start in this state, can I reach this potentially problematic other state in a Markov chain process?
- Is it possible traveling by train/car to reach my destination?

# Graphs and the adjacency matrix

## Definition

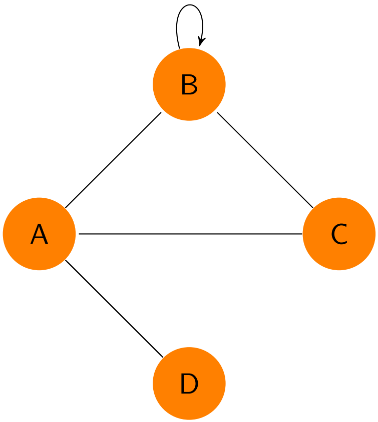
The **adjacency matrix** $A$ of a graph $G$ with $n$ vertices is a square $n \times n$ matrix with elements $a_{ij}$ such that:

$$a_{ij} = \begin{cases} 1, & \text{if there is a link from vertice } i \text{ to vertice } j \\ 0, & \text{otherwise} \end{cases}$$

If the graph is undirected we consider every edge between two vertices as linking in both directions.

# Graphs and the adjacency matrix

Example of an undirected graph and corresponding adjacency matrix (vertices ordered A,B,C,D).




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

# Graphs and the adjacency matrix



We note that the adjacency matrix is not unique in itself, we need to choose in what order we put our vertices in the matrix!

- Sometimes changing the order of the vertices (essentially re-labeling the graph) can make certain structures more obvious.
- For example we could group vertices in the same connected component together, essentially dividing the graph in smaller subgraphs.

# Graphs and the adjacency matrix

The adjacency matrix gives a clear view of which vertices have an edge between them, but it can also be used to find if there is a path between vertices.

- We take a look at the powers $A^k$ of the adjacency matrix.
- A non-zero element of $A^1$ means there is a path of length 1 between the two vertices. (length as in number of edges)

# Graphs and the adjacency matrix

We look at $A^2$ and when a single element $a_{ij}^2 > 0$.

- For $a_{ij}^2 > 0$ we need that at least one of $a_{ik}a_{kj} > 0, k = 1, 2, \ldots, n$.
- $a_{ik} = 1$ if there is a edge between $i, k$ and $a_{ki} = 1$ if there is a edge between $k, j$.
- So for $a_{ij}^2 > 0$ we need at least one path of exactly length 2 between $i, j$.
- It's easy to see using for example induction that the same is true for $A^k, \ k > 0$ as well.

# Graphs and the adjacency matrix

While we can compute all of $A, A^2, A^3, \ldots, A^k$ for some large $k$ in order to find connected components, there is a better way:

- We notice that if we let every vertice link to itself, then if there is a path from one vertice to another of length $k$, then there is a path of length $k+1$ as well simply by looping in the last vertice once.
- Additionally the longest possible shortest path is $n-1$, since we after traveling to $n-1$ new vertices we need to return to one we have already visited.
- Looking at the elements of $(I+A)^{n-1}$ we can thus see if there is at least one path between any two vertices.

# Irreducible non-negative matrices.

### Theorem

*A non-negative square $n \times n$ matrix $A$ is said to be* **Irreducible** *if any of the following equivalent properties is true:*

1. *The graph corresponding to $A$ is strongly connected.*
2. *For every pair $i, j$ there exists a natural number $m$ such that $A^m_{i,j} > 0$.*
3. *$(I + A)^{n-1}$ is a positive matrix.*

# Irreducible non-negative matrices.

Irreducible non-negative matrices have some very useful properties which are used extensively in for example:

- Perron-Frobenius theory.
- Stochastic processes.
- Markov chains and Markov chain monte carlo.

# Vertex and edge connectivity

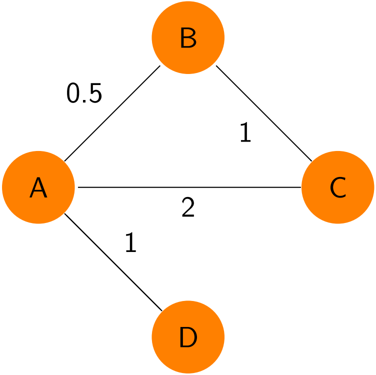
Even if the graph is irreducible it might not be very well connected or it might be interesting to see how easy it is to splitt the graph in two disjoint parts.

- The **vertex connectivity** of a graph is the minimum number of vertices whose removal makes the graph reducible.
- The **edge connectivity** of a graph is the minimum number of edges whose removal makes the graph reducible.
- Finding he edge connectivity is related to finding the max-flow min-cut problem.

# Distance matrix

While good to know whether there exist a path between two vertices, the adjacency matrix says nothing of the length of those paths.

- To do this we introduce (positive) weights on the edges representing the distance between to cities.
- This gives a weighted graph such as:



# Distance matrix

To represent this weighted graph we only need to make a short modification of the adjacency matrix:

## Definition

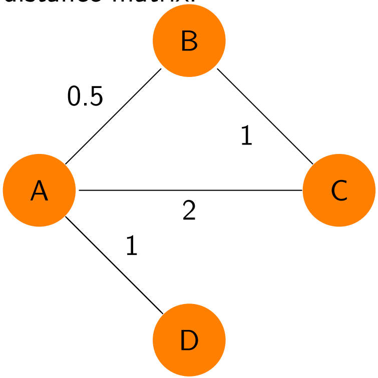
The **distance matrix** $D$ of a graph $G$ with $n$ vertices and weighted edges $E$ is a square $n \times n$ matrix with elements $d_{ij}$ such that:

$$d_{ij} = \begin{cases} E_{ij}, & \text{if there is a link from vertice } i \text{ to vertice } j \\ 0, & \text{otherwise} \end{cases}$$

Where $E_{ij}$ is the weight of the edge from vertice $i$ to vertice $j$. As with the adjacency matrix, if the graph is undirected we consider every edge between two vertices as linking in both directions.

# Distance matrix

Here is an example of the previous graph and corresponding distance matrix.




$$\begin{bmatrix} 0 & 0.5 & 2 & 1 \\ 0.5 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

# Weighted graphs and applications

A weighted graph and it's distance matrix can for example be used to represent:

- Distance between vertices physical or some other defined distance, or time needed to travel from one vertice to another.
- Electrical networks, where the weights represent the resistance over the wires.
- Water flow network with capacities represented by the weights.
- Markov processes and random walks where the weights represent a probability to move from one state to another.

# Distance matrix and applications



Using the distance matrix we can do the same analysis as we did for the adjacency matrix to for example see if the matrix is irreducible.
However the distance matrix can also be used for more advanced analysis such as:

- Finding the shortest path between two or more vertices.
- Finding stationary distributions, hitting times, hitting probabilities, and much more for Markov chains.
- Calculating the potential of electrical networks.

# Application: shortest path

- A common algorithm to find the shortest distance between one source node and one or all other nodes is Djikstra's algorithm.
- It works by assigning a distance from the initial node to every other node and then travel through the graph by always choosing the next node among the unvisited nodes that minimizes the distance.
- At every node it updates the distance to every node linked to by the current node.

# Application: Djikstra's algorithm

More exact the method works through the following steps:

1. Assing distance zero to initial node and infinity to all other nodes.
2. Mark all nodes as unvisited and set the initial node as current node.
3. For the current node, calculate the distance to all it's unvisited neighbors as the current distance plus the distance to the neighbors. Update these nodes distance if the newly found distance is lower.
4. Mark current node as visited.
5. If the smallest distance in among the unvisited nodes is infinity: stop! Otherwise set the unvisited node with the smallest distance as current node and go to step 3.

# Application: Djikstra's algorithm



Think about the following and if or how it effects the usefulness of Djikstra's algorithm.

- What could happen if there are negative weight's in the graph.
- Can you think of any application where you would like to find the shortest path between two vertices?