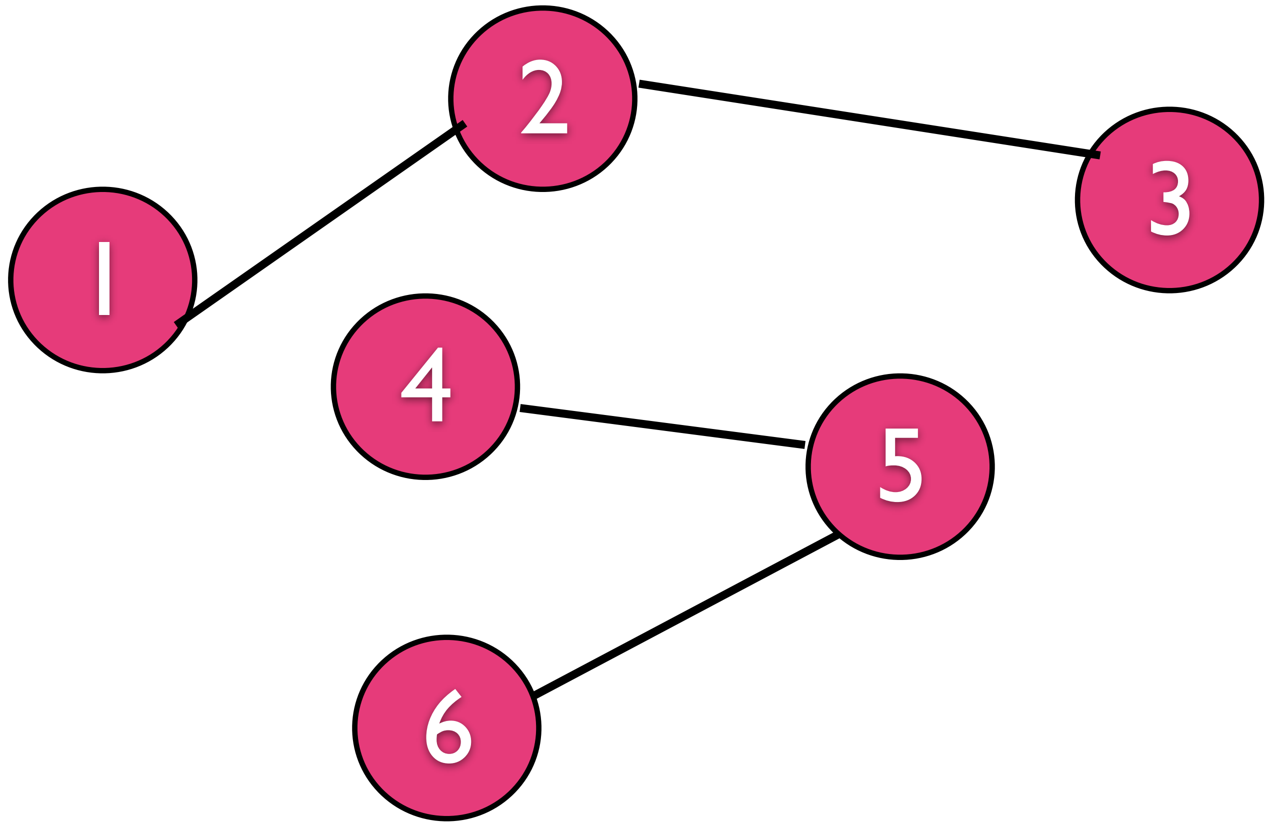
Graph Mining

Danushka Bollegala

UNIVERSITY OF LIVERPOOL

Graphs

- A Graph G can be defined as a set of vertices (nodes) V connected by a set of edges (links) E
- A graph G(V,E) is fully defined by specifying the two sets V and E

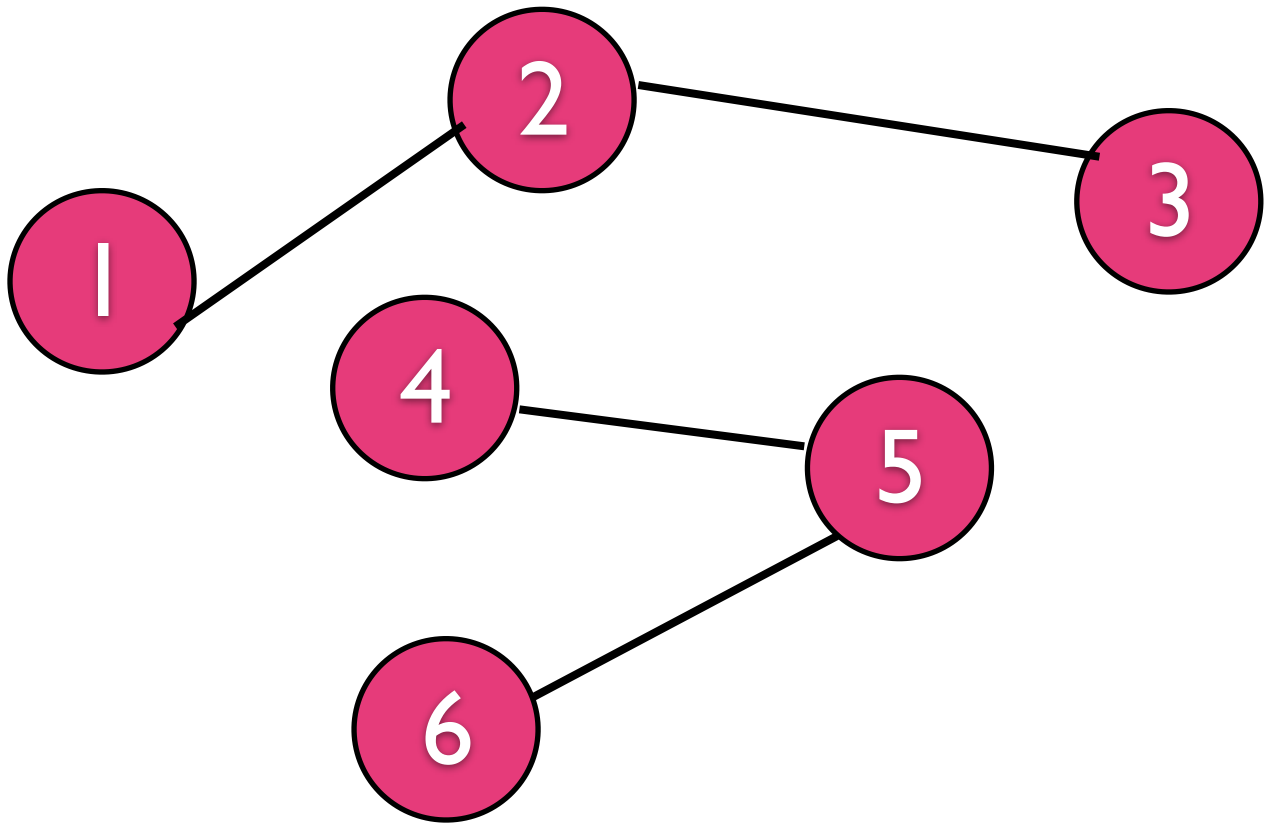


Types of Graphs

- Undirected Graph
  - There are no directional edges in the graph
- Directed Graph
  - There are directional edges in the graph
- Labeled/Coloured Graph
  - Vertex-Labeled Graph
    - Vertices are labeled (coloured)
  - Edge-Labeled Graph
    - Edges are labeled (coloured)
- Weighted Graph
  - Edges have weights associated with them
- Unweighted Graph
  - Edges have no weights associated with them. All edges have an equal weight.

Adjacency Matrix

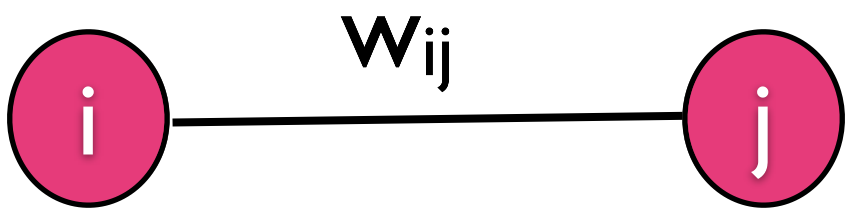
- If two vertices v_i and v_j are connected by an edge in an graph G, then the element a_{ij} in the incidence matrix will be set to 1, otherwise it will be set to 0.



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

Weight Matrix

- The weight matrix W of a weighted graph G denotes the weight of the edge between vertices v_i and v_j by the element w_{ij}
- Notes
  - A negative weight does not indicate a reverse link always (however, some abuse of notation is possible, if defined in advance)



For undirected graphs, $w_{ij} = w_{ji}$
(W is becomes a symmetric matrix)

State Transitions

- At a given time t=T, the probability of being at each vertex can be represented by a |V| dimensional vector **x**, where |V| is the total number of vertices in the graph.
- Question
  - What is the probability of being at each vertex at t =(T+1)
- Answer
  - B**x**
    - **B** is the state transition matrix
    - The probability of being at vertex V_j at t=T+1, when we are at vertex V_i at t = T is given by B_{ij}
- What about t=(T+2) then
  - B(B**x**) = B^2**x**
- What about t = (T+n) then
  - B^n**x**

Random Walk in a Graph

- Assume that you are walking in a graph
- You start with some vertex and randomly move to a vertex that is connected to the current vertex
- All connected vertices have an equal probability of getting selected for the next move
- After you have moved infinite amount of time in this graph according to the previously described mechanism, what is the probability of you ending up in some vertex v_i in the graph?

Random Walk

- If the state transition has reached a stable state, then we have the situation
  - $Ax = \lambda x$
- This means that x is the eigenvector of A corresponding to the eigenvalue λ, which is a scalar.
- Instead of moving around the graph for infinite time we can simply perform eigenvalue decomposition of A to find the final state (if it exists!)
- Moreover, final state (if exists) does not depend on the initial state!

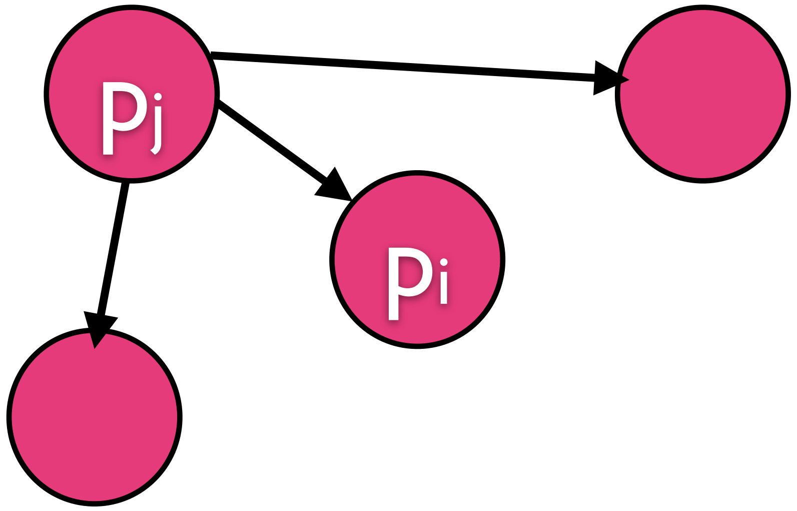
What can we learn from a Random Walk?

- Connectivity of the graph
  - If there are *islands* in the graph (ie. subgraphs that are not connected), then no matter how much we perform this random walk, we will not be able to reach those islands.
- Importance of the vertices
  - If there is a close connection between two vertices v_i and v_j, then the probability of ending up in v_j, when we start from v_i will be higher
  - But, it does not matter from where we start
    - which means that the probability of ending up at a particular vertex is an indicator of how *important* that vertex (measured by its connectivity to other vertices in the graph) in the graph
  - Highly connected people are more important/influential?

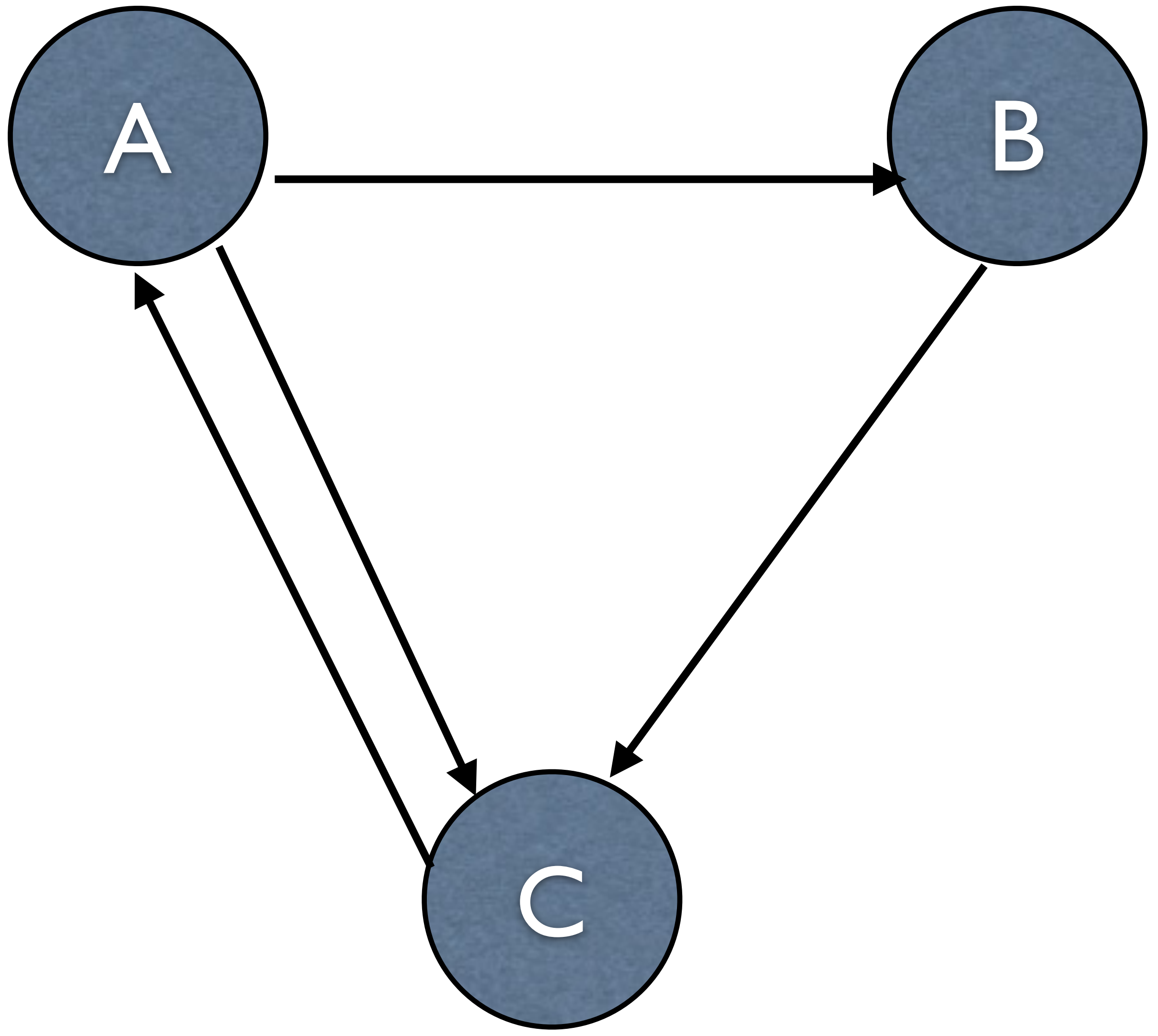
PageRank Algorithm

- One of many algorithms that are based on the idea of random walks in a graph
- Proposed by Larry Page
- Original objective
  - Compute the rank of web pages
  - vertices = web pages
  - edges = hyperlinks
- Can be applied to any graph, not limiting to web graph, to induce a ranking for the vertices.
- PR(p_i): page rank of page p_i
- M(p_i): set of nodes connected to p_i via an inbound link
- L(p_j): number of outbound links on p_j

$$PR(p_i) = \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$



Quiz: Compute the PageRanks for the following graph.



$$p_A = p_C$$

$$p_B = \frac{p_A}{2}$$

$$p_C = p_B + \frac{p_A}{2}$$

Issues with simple PageRank

- If the random walker gets trapped/struck inside a particular node, then the simple PageRank algorithm we discussed previously will fail.
- This is called "a leak" of PageRank
- To overcome this problem we use *teleportation*
    - At each node p we will select a node from the set of nodes connected via in-bound links to p, M(p), with a probability *d-1*.
    - Or, we randomly jump (teleport) to any of the remaining (N-1) nodes with probability *d*.
- This gives rise to the *damped* version of PageRank discussed in the next slide.

Damping Factor

- It is possible that a random surfer (walker) might not surf (walk) over the graph eternally (until infinite number of iterations) but will stop after a while (tired/damping).
- The following version of the PageRank algorithm takes this into consideration
  - d is the damping factor and is set to 0.85 in most practical cases
  - N is the total number of vertices (pages)

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in \mathcal{M}(p_i)} \frac{PR(p_j)}{L(p_j)}$$