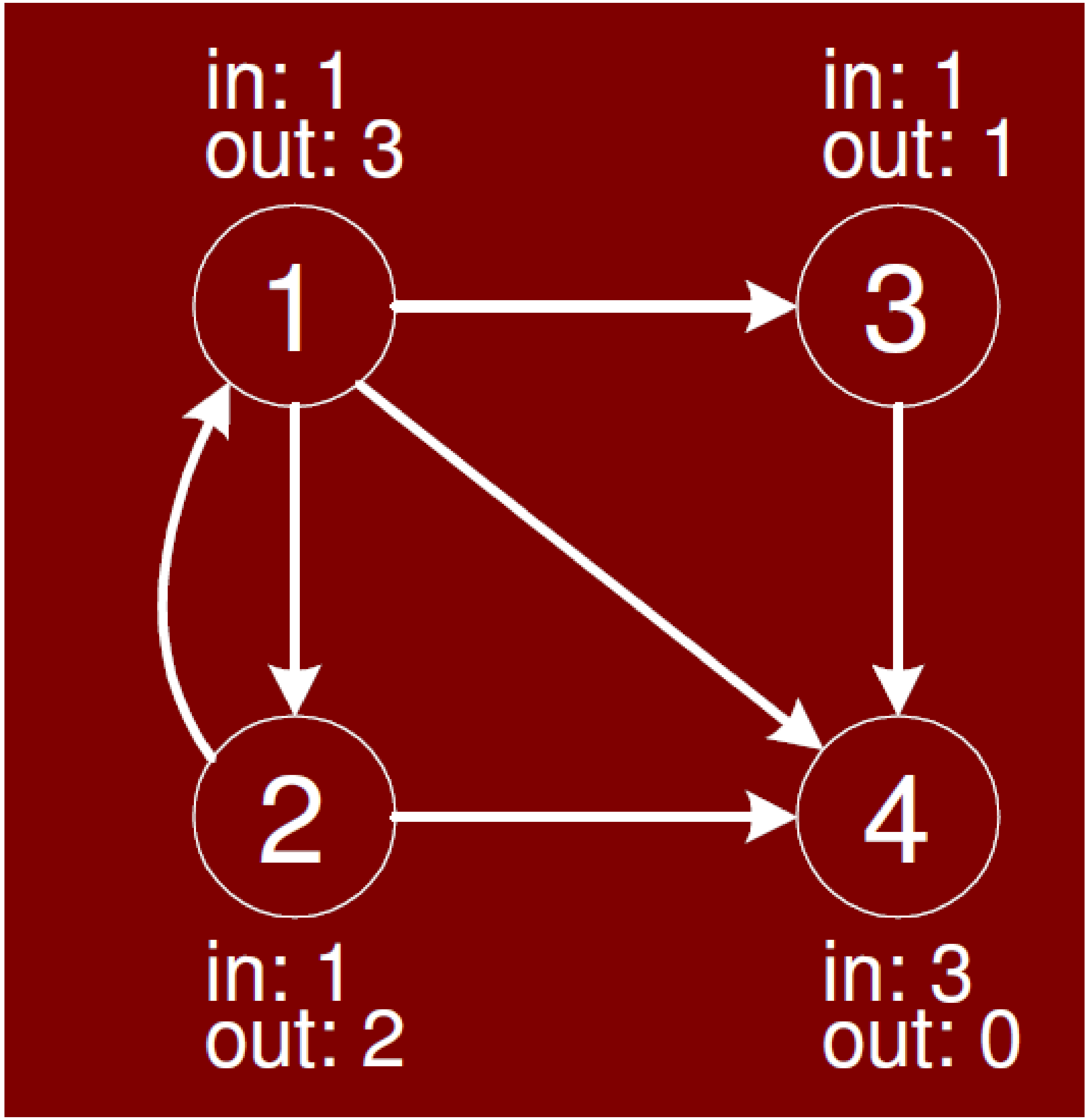
# Graphs
# Representations, Traversal

**(Class 28)**

From Book’s Page Number 547 (Chapter 20)

## Graph Representations

- In a directed graph, the number of edges coming out of a vertex is called the *out-degree* of that vertex.
- Number of edges coming in is the *in-degree*.
- In an undirected graph, we just talk of degree of a vertex.
- It is the number of edges incident on the vertex.


in: 1
out: 3
1
in: 1
out: 1
3
2
in: 1
out: 2
4
in: 3
out: 0

- For a directed graph $G = (V, E)$:

$$\sum_{v \in V} \mathit{in\text{-}degree}(v) = \sum_{v \in V} \mathit{out\text{-}degree}(v) = |\mathrm{E}|$$

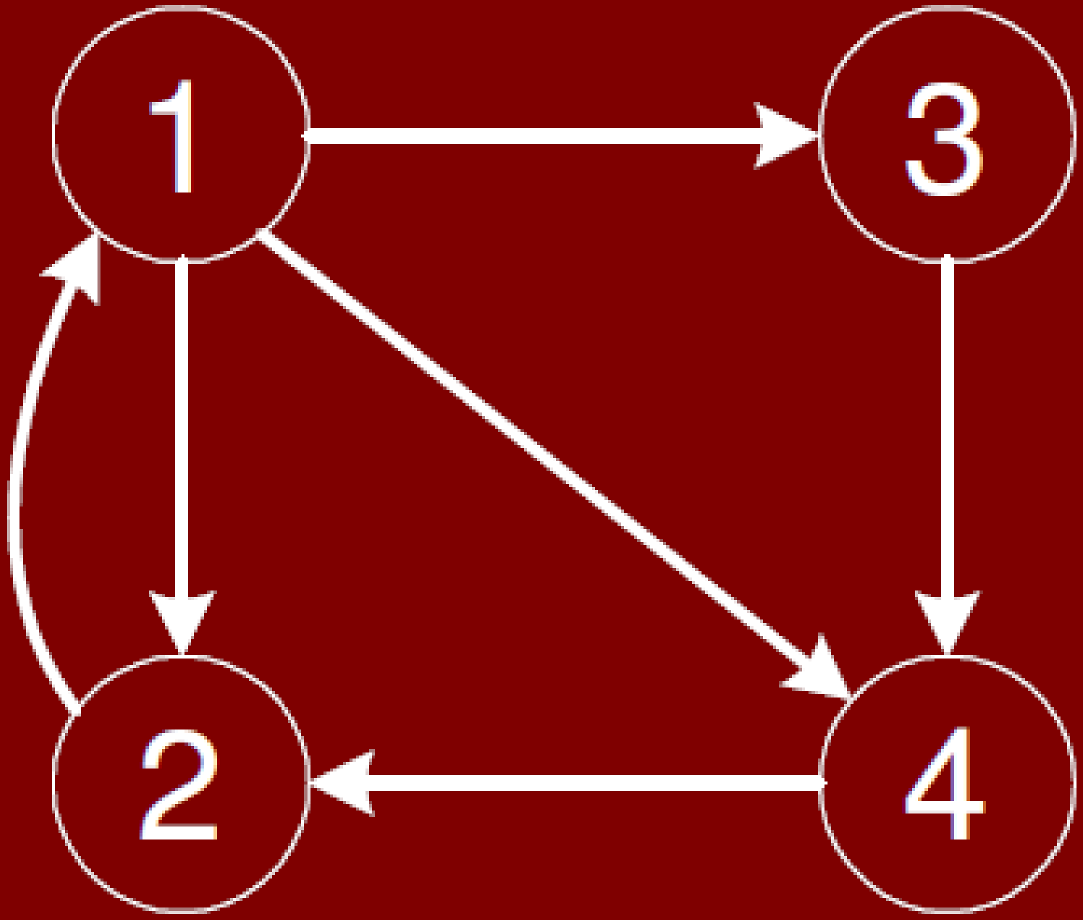
- Where $|E|$ means the cardinality of the set $E$, i.e., the number of edges.

- For an undirected graph $G = (V, E)$:

$$\sum_{v \in V} degree(v) = 2|\mathrm{E}|$$

- A *path* in a directed graphs is a sequence of vertices $(v_0, v_1, \dots\ , v_k)$ such that $(v_{i-1}, v_i)$ is an edge for $i = 1, 2, \dots, k$.
- The *length* of the paths is the number of edges, $k$.
- A vertex $w$ is *reachable* from vertex $u$ is there is a path from $u$ to $w$.
- A path is simple if all vertices (except possibly the first and last) are distinct.

- A *cycle* in a digraph is a path containing at least one edge and for which $v_0 = v_k$.
- A *Hamiltonian* cycle is a cycle that visits every vertex in a graph exactly once.
- A *Eulerian* cycle is a cycle that visits every edge of the graph exactly once.
- There are also “path” versions in which you do not need return to the starting vertex.



cycles:

1-3-4-2-1

1-4-2-1

1-2-1

- A graph is said to be *acyclic* if it contains no cycles.
- A graph is *connected* if every vertex can reach every other vertex.
- A directed graph that is acyclic is called a *directed acyclic graph* (DAG).

- There are two ways of representing graphs:
  - Adjacency Matrix
  - Adjacency List

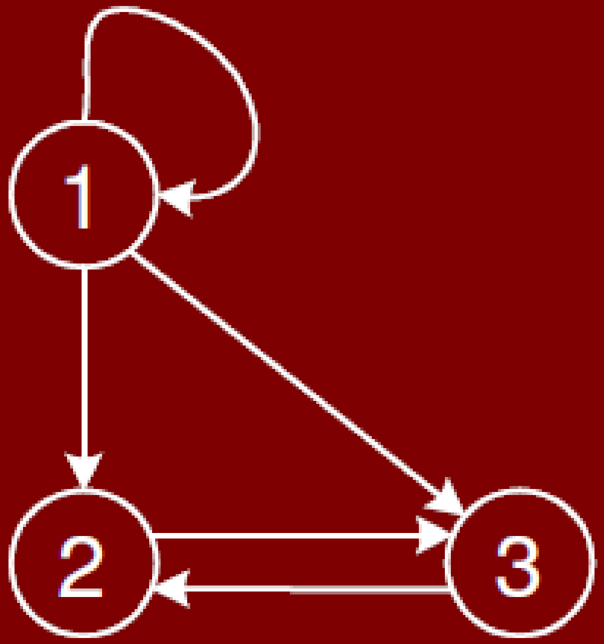
## Adjacency Matrix

- Let $G = (V, E)$ be a digraph with $n = |V|$ and let $e = |E|$.
- We will assume that the vertices of $G$ are indexed $\{1, 2, \dots, n\}$.
- An *adjacency matrix* is a $n \times n$ matrix defined for $1 \leq v, w \leq n$.

$$A[v, w] = \begin{cases} 1 & \text{if } (v, w) \in E \\ 0 & \text{otherwise} \end{cases}$$

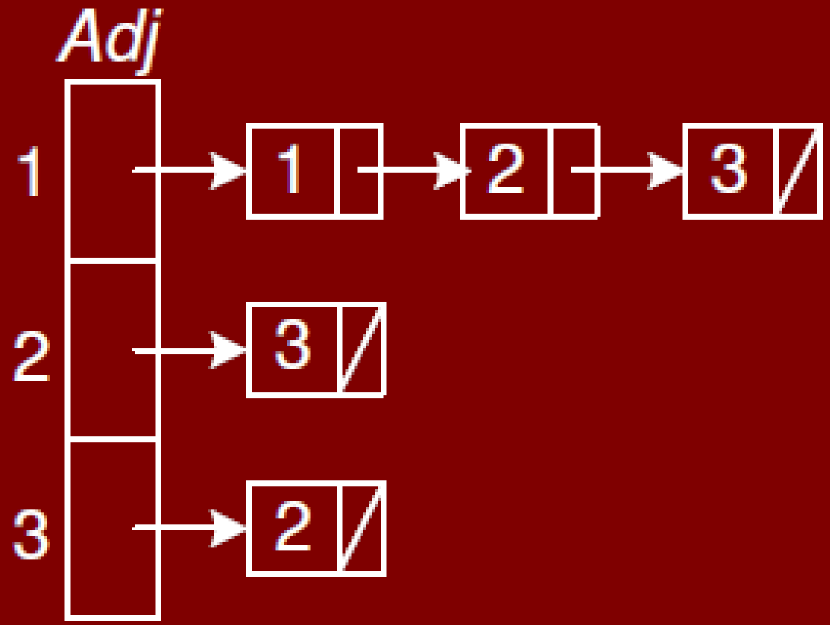
## Adjacency List

- An *adjacency list* is an array $Adj[1 \dots n]$ of pointers where for $1 \leq v \leq n$, $Adj[v]$ points to a linked list containing the vertices which are adjacent to $v$.
- Adjacency matrix requires $O(n^2)$ storage and adjacency list requires $O(n + e)$ storage.



$$\begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & 1 & 1 & 1 \\ 2 & 0 & 0 & 1 \\ 3 & 0 & 1 & 0 \end{array}$$

Adjacency Matrix



Adjacency List

# Graph Traversal: Shortest Path

- To motivate our first algorithm on graphs, consider the following problem.
- We are given an undirected graph G = (V, E) and a source vertex $s \in V$.
- The *length* of a path in a graph is the number of edges on the path.
- We would like to find the shortest path from $s$ to each other vertex in the graph.

- The final result will be represented in the following way:
  - For each vertex $v \in V$, we will store $d[v]$ which is the distance (length of the shortest path) from $s$ to $v$.
  - Note that $d[s] = 0$.
  - We will also store a predecessor (or parent) pointer $\pi[v]$ which is the first vertex along the shortest path if we walk from $v$ backwards to $s$.
  - We will set $\pi[s] = Nil$.