# Binary Trees

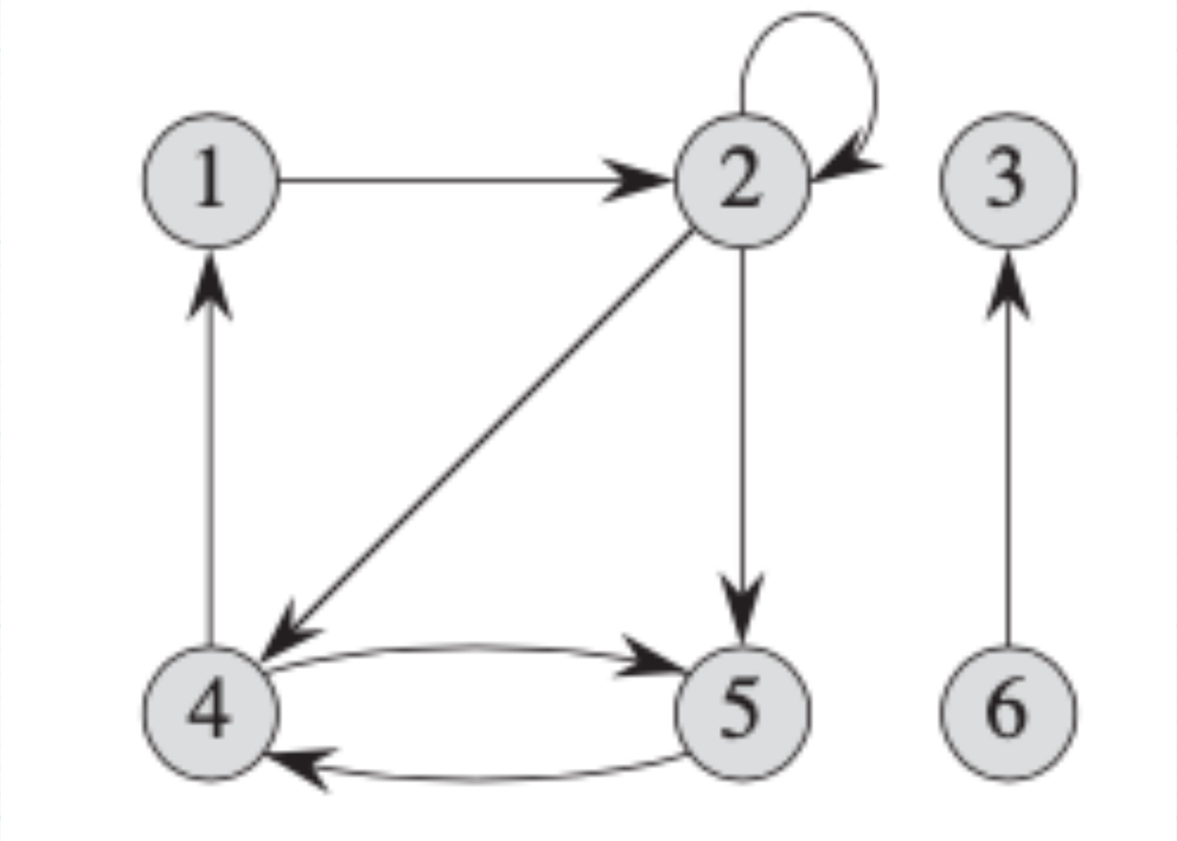
Introduction

# Free Trees

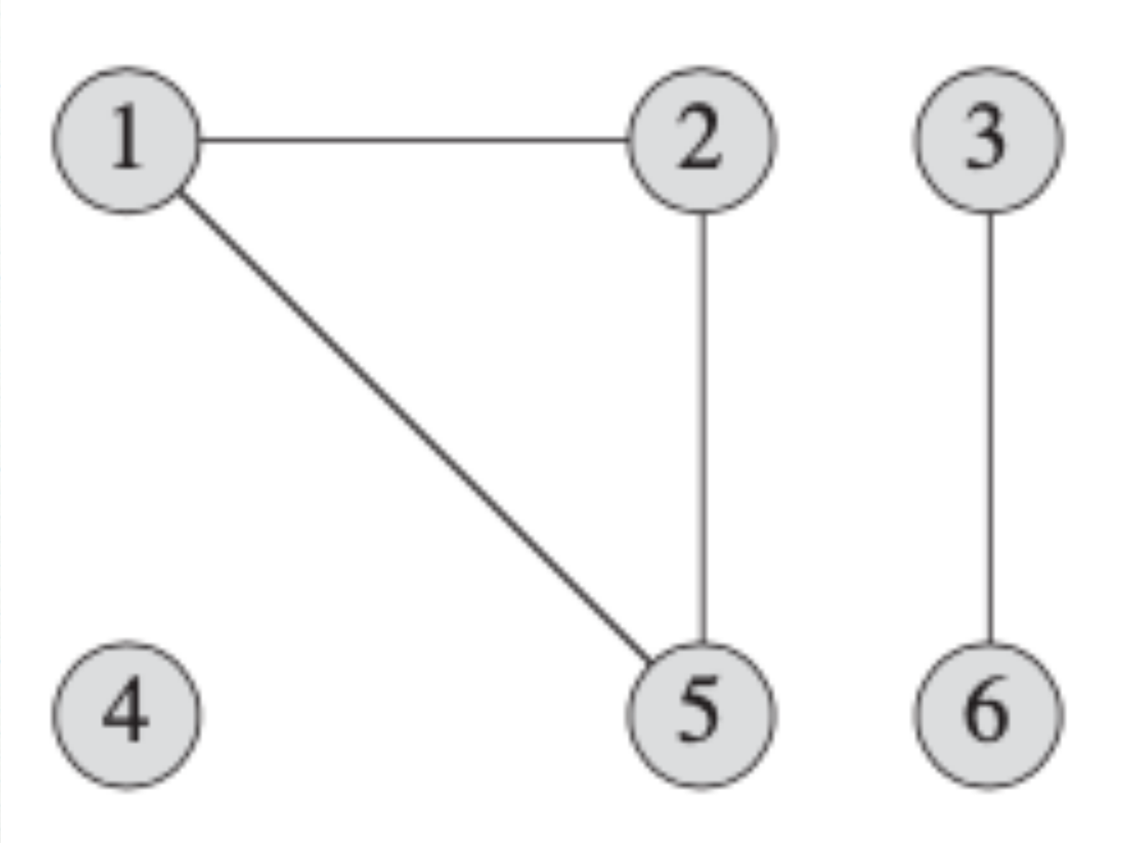
* A Free tree is a connected, acyclic, undirected graph

# Graphs

* Two kinds of Graphs
* Directed Graph




* Undirected Graph



# Directed Graph or Digraph

A **directed graph** (or digraph) G is a pair (V, E) where V is a finite set and E is a binary relation on V .

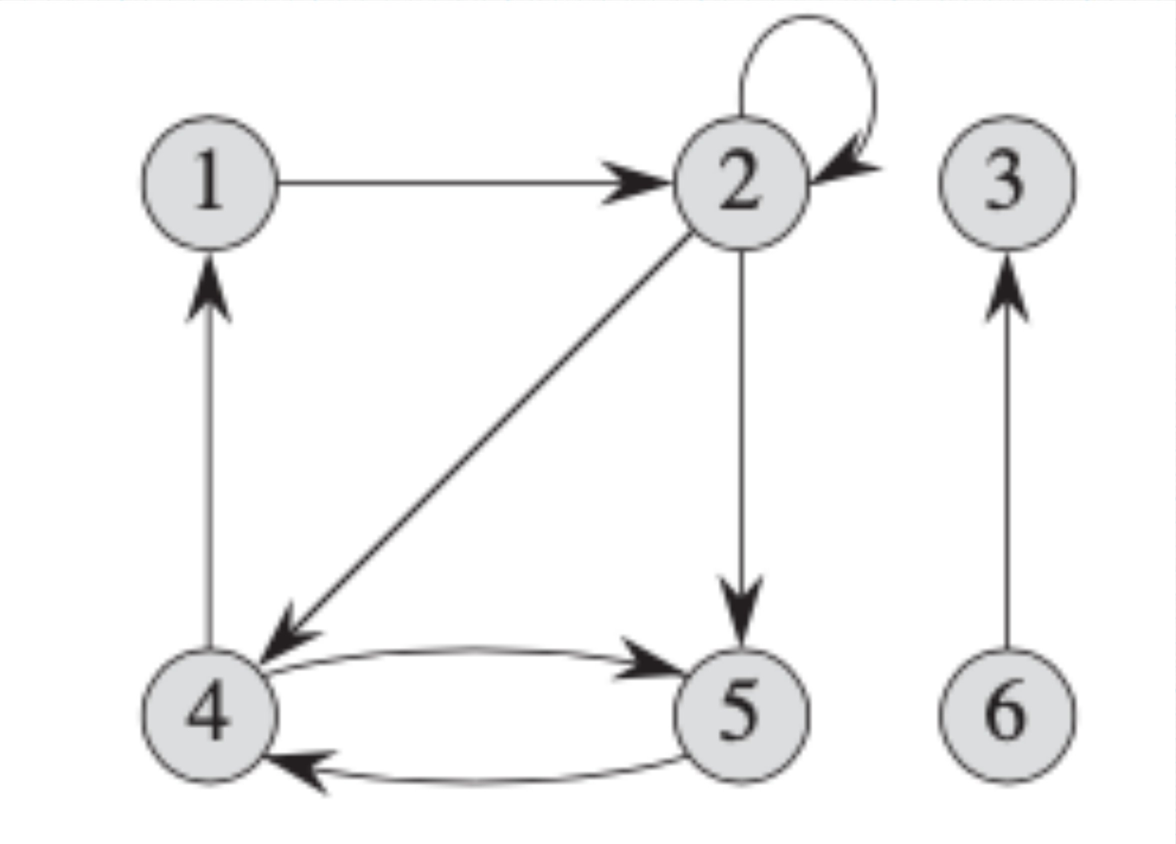
The set V - **vertex set** of G, elements - **vertices**

The set E - **edge set** of G, elements - **edges**.

Pictorial representation of a directed graph on the vertex set (1, 2, 3, 4, 5, 6).

Vertices - circles, edges - arrows.

**Self-loops:** edges from a vertex to itself.



# Undirected Graph

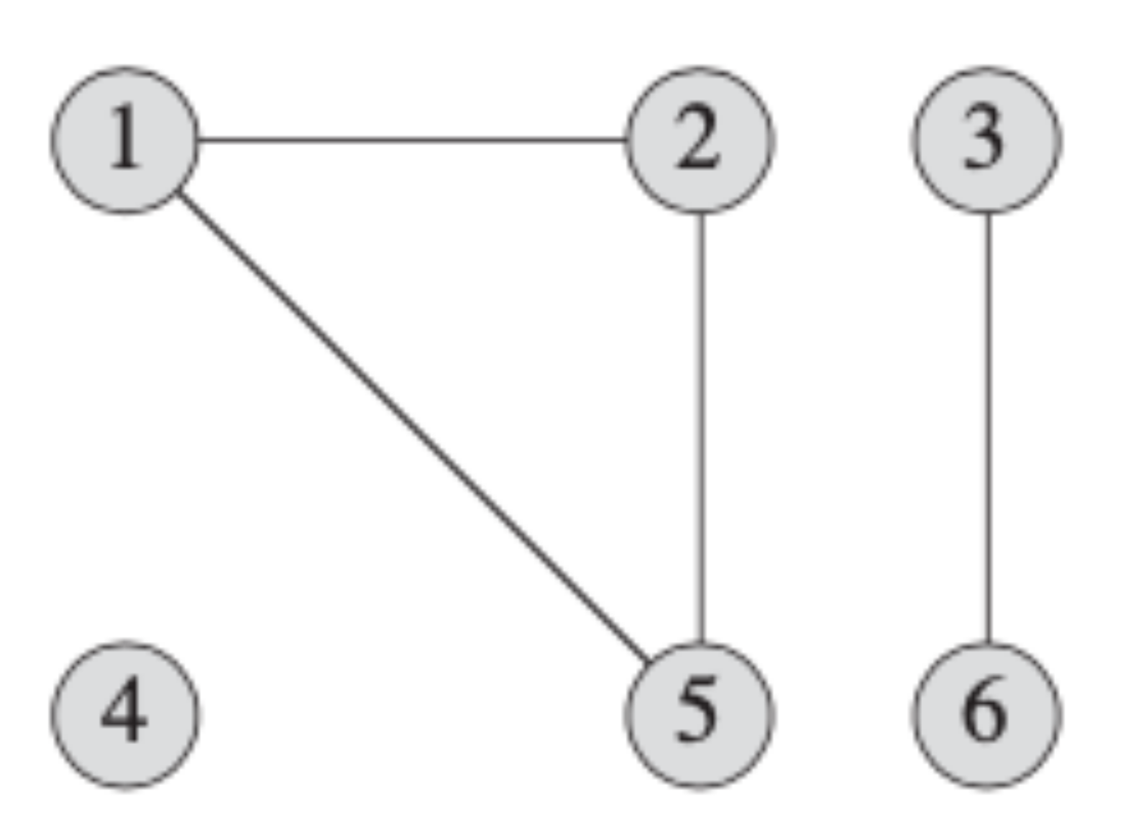
- In an **undirected graph** $G = (V, E)$, the edge set E consists of $\{u, v\}$ **unordered pairs of vertices,** rather than ordered pairs.

- An edge is a set $\{u, v\}$, where $u, v \in V$ and $u \neq v$

- Use the notation $(u, v)$ for an edge, rather than the set notation $\{u, v\}$ and we consider $(u, v)$ and $(v, u)$ to be the same edge.

- **Self-loops are forbidden** - every edge consists of two distinct vertices

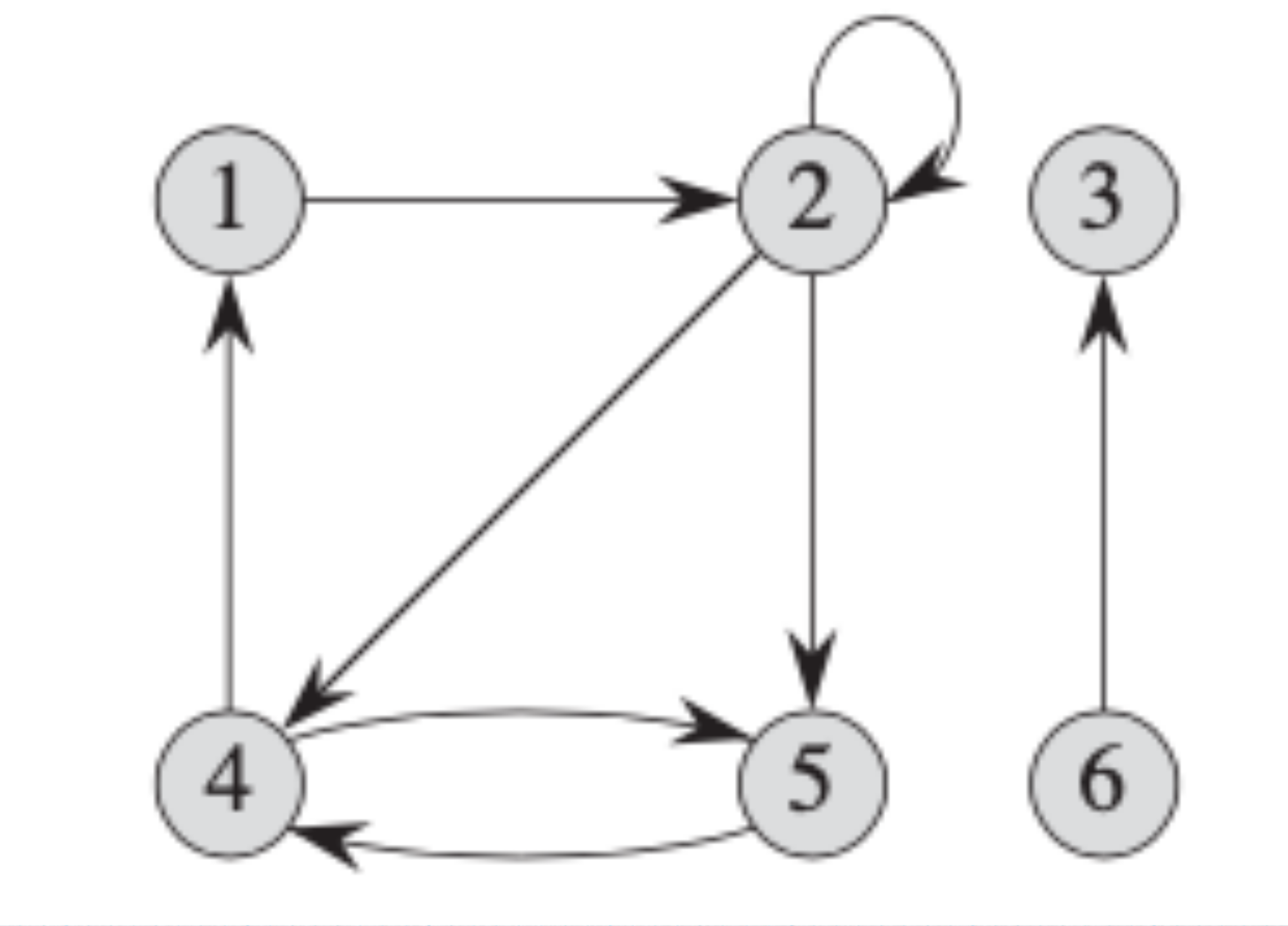
- Pictorial representation of an undirected graph on the vertex set $\{1,2,3,4,5,6\}$




*

# Path

- A **path of length** $k$ from a vertex $u$ to a vertex $u'$ in a graph $G = (V, E)$ is a sequence $< v_0, v_1, v_2, \ldots, v_k >$ of vertices such that $u = v_0$ and $u' = v_k$ and $(v_{i-1}, v_i) \in E$ for $i = 1,2,\ldots,k$

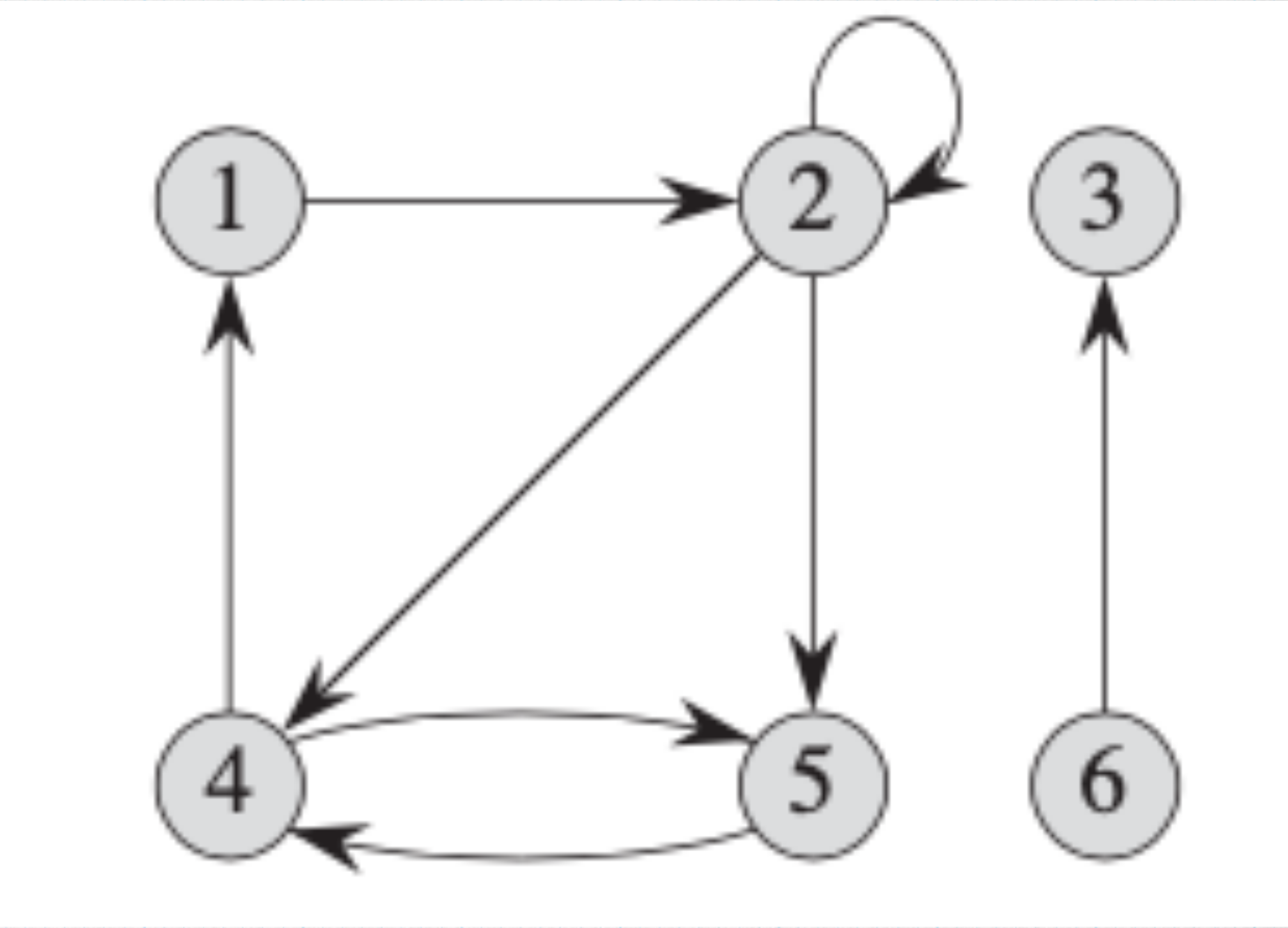



- **Length of the path :** number of edges in the path.

- The path contains the vertices $v_0, v_1, v_2, \ldots, v_k$ and the edges $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$

- There is always a **0-length path** from u to u.

# Path



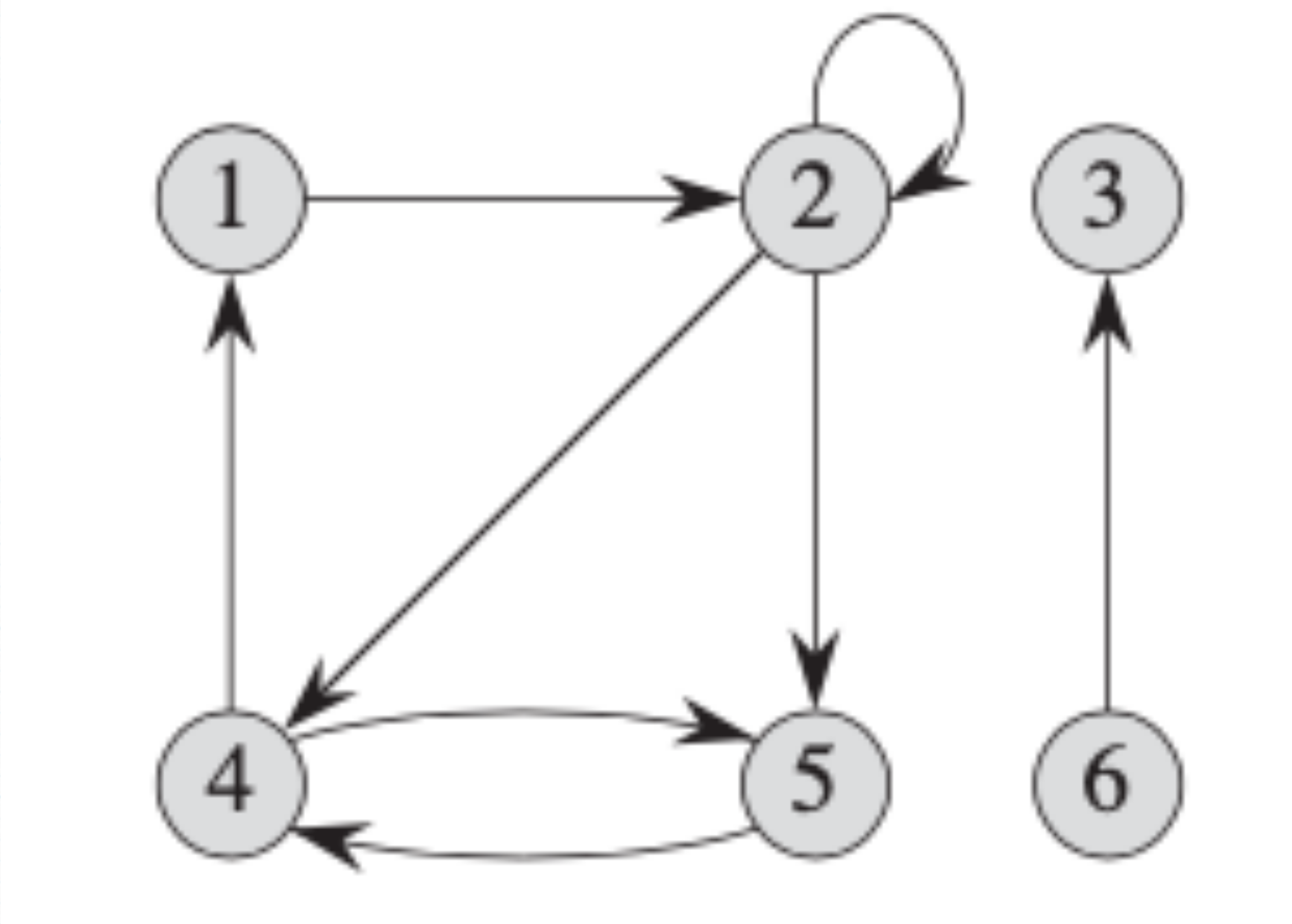

- If there is a path p from $u$ to $u'$, we say that $u'$ is **reachable** from $u$ via p

- A **path is simple** if all vertices in the path are **distinct.**

- In this Figure, the path <1,2,5,4> is a **simple path of length 3.**

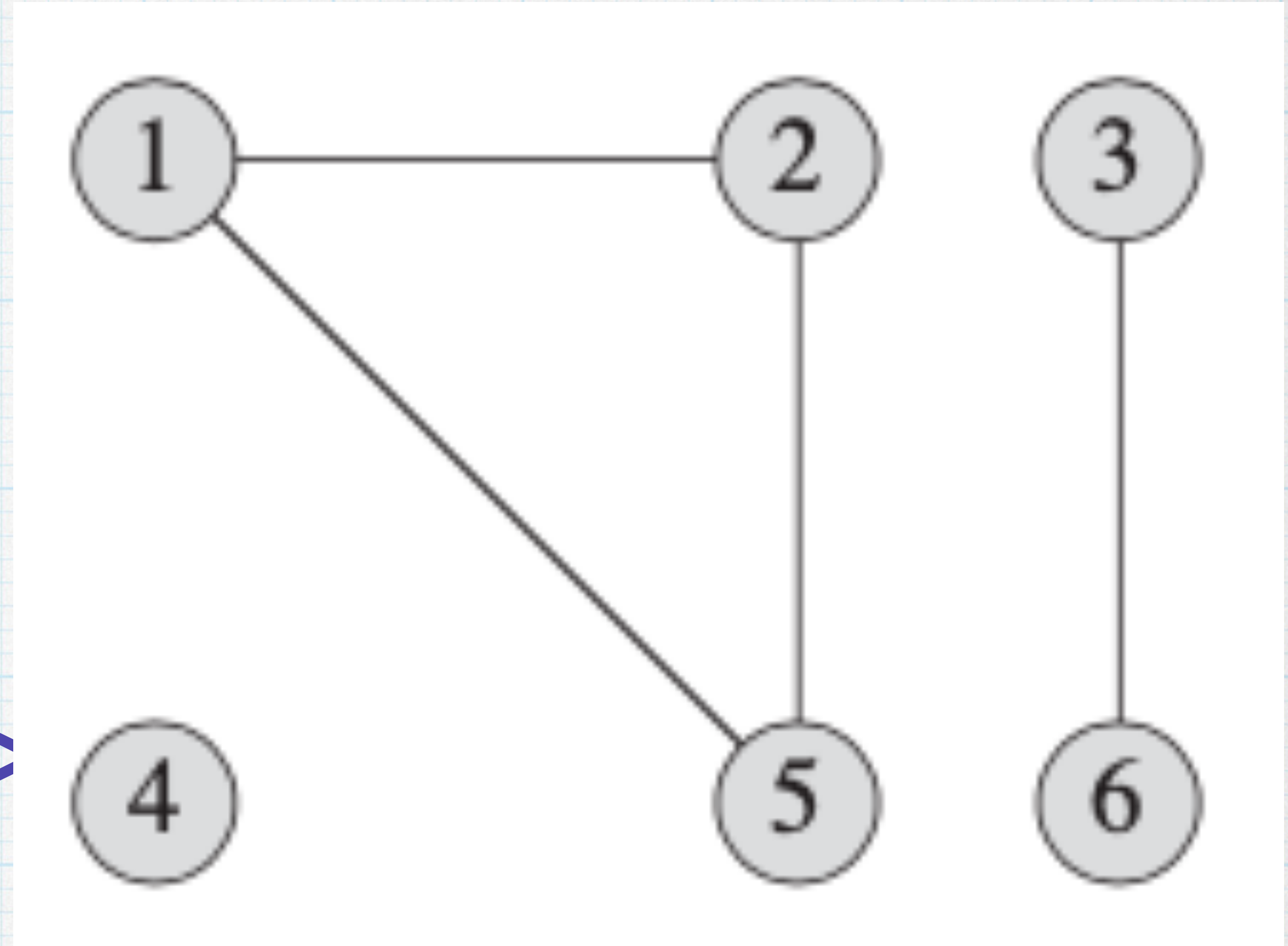
- The path <2,5,4,5> is **not simple.**

# Cycle (Directed Graph)




- **Directed graph**, a path $< v_0, v_1, v_2, \ldots, v_k >$ forms a **cycle** if $v_0 = v_k$ and the path contains at least one edge.
- **Simple cycle:** vertices are distinct except $v_0, v_k$
- A **self-loop** is a cycle of length 1.
- **Simple Cycle**: the path <1,2,4,1> forms the same cycle as the paths <2,4,1,2> and <4,1,2,4>.
- Cycle <1,2,4,5,4,1> is **not a simple cycle**.
- The cycle <2,2> formed by the edge (2,2) is a self-loop.
- A directed graph with **no self-loops** is simple graph

# Cycle (Undirected Graph)



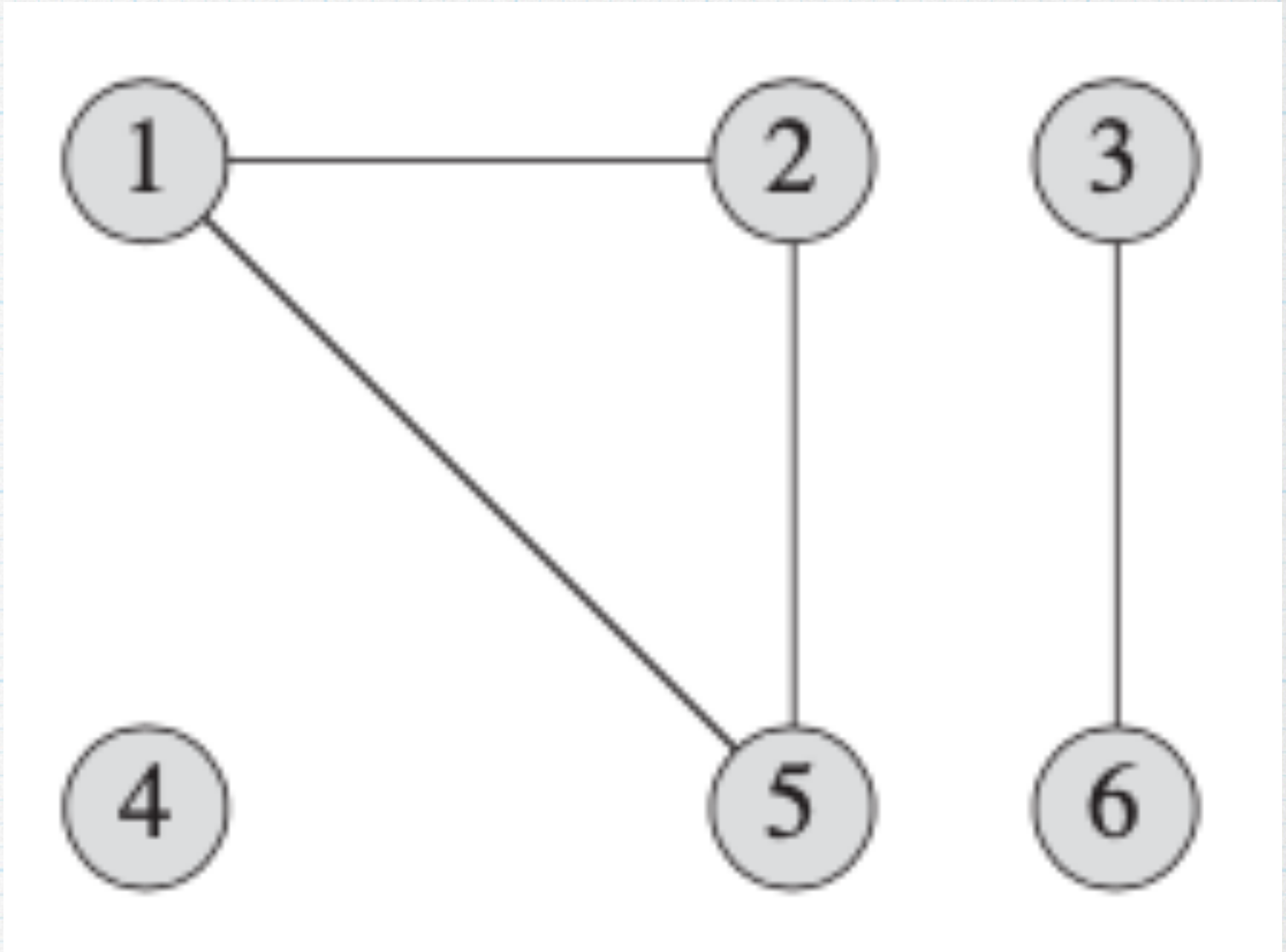

- In an undirected graph, a path $< v_0, v_1, v_2, \ldots, v_k >$ forms a **cycle** if k $\geq$ 3 and $v_0 = v_k$

- Cycle is **simple** if $v_0, v_1, v_2, \ldots, v_k$ are distinct.

- Figure: the path <1,2,5,1> is a **simple cycle**.

- A graph with no cycles is **acyclic.**

# Connected Graph




- An undirected graph is **connected** if every vertex is **reachable** from all other vertices.
- The ***connected components*** of a graph are the equivalence classes of vertices under the “is reachable from” relation.
- **Example:** Three components
- Every vertex  in {1,2,5} is reachable from every other vertex in {1,2,5}
- An undirected graph is **connected** if it has exactly one connected component.
- A ***complete graph*** is an undirected graph in which every pair of vertices is adjacent.

# Free Tree

* A **Free tree** is a **connected, acyclic, undirected graph**
* Omit the adjective ‘free’, when we say that **a graph is a tree**
* Disconnected **acyclic** undirected graph - ***forest***.

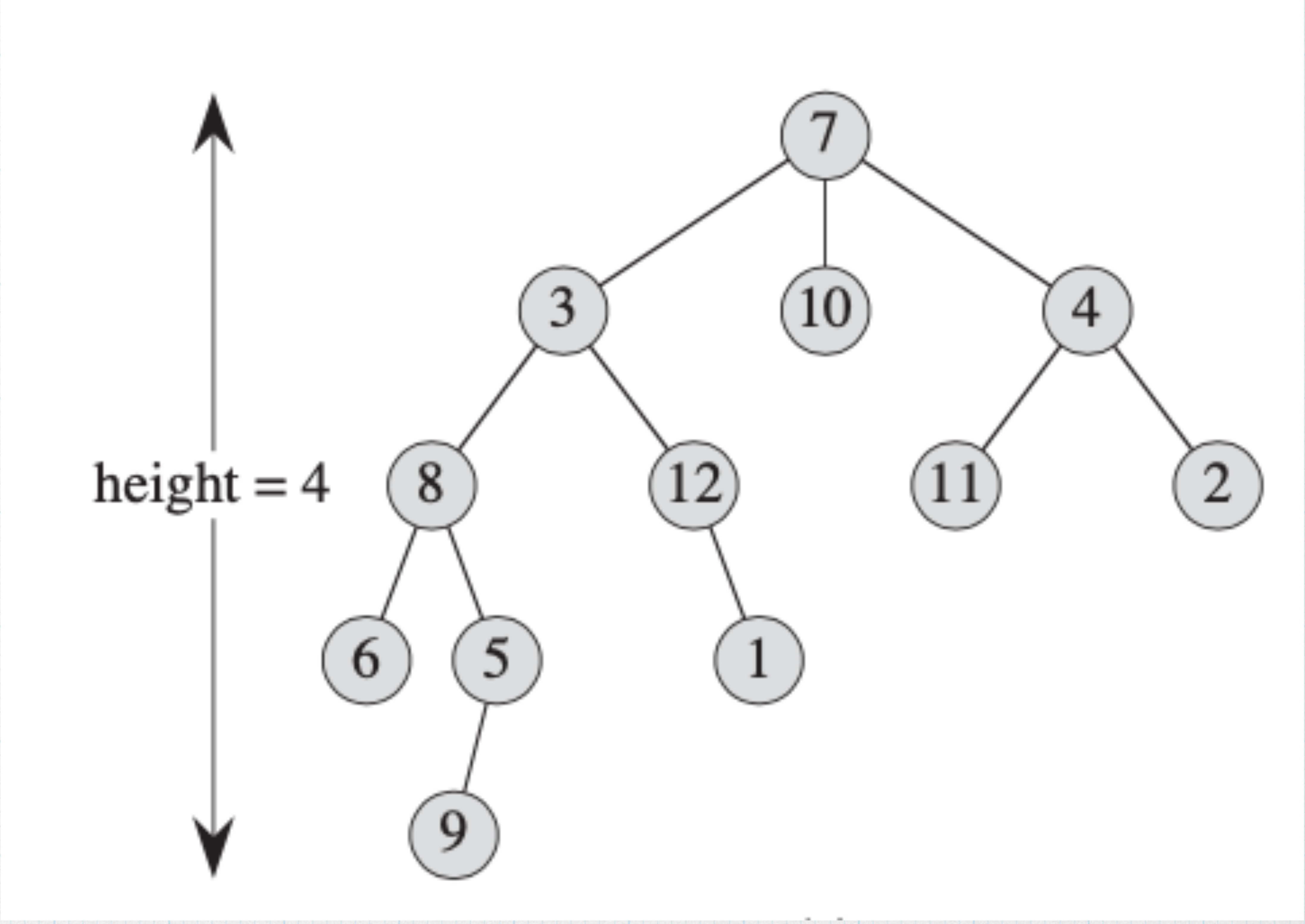
# Properties of Trees

Let $G = (V, E)$ be an undirected graph. The following statements are equivalent.

1. $G$ is a free tree.
2. Any two vertices in $G$ are connected by a unique simple path.
3. $G$ is connected, but if any edge is removed from $E$, the resulting graph is disconnected.
4. $G$ is connected, and $|E| = |V| - 1$.
5. $G$ is acyclic, and $|E| = |V| - 1$.
6. $G$ is acyclic, but if any edge is added to $E$, the resulting graph contains a cycle.

# Rooted Trees

- A **rooted tree** is a free tree in which one of the vertices is distinguished from the others.




- Distinguished vertex the **root** of the tree.

- **Vertex** of a rooted tree as a **node of the tree**.

- Figure shows a rooted tree on a **set of 12 nodes with root 7.**

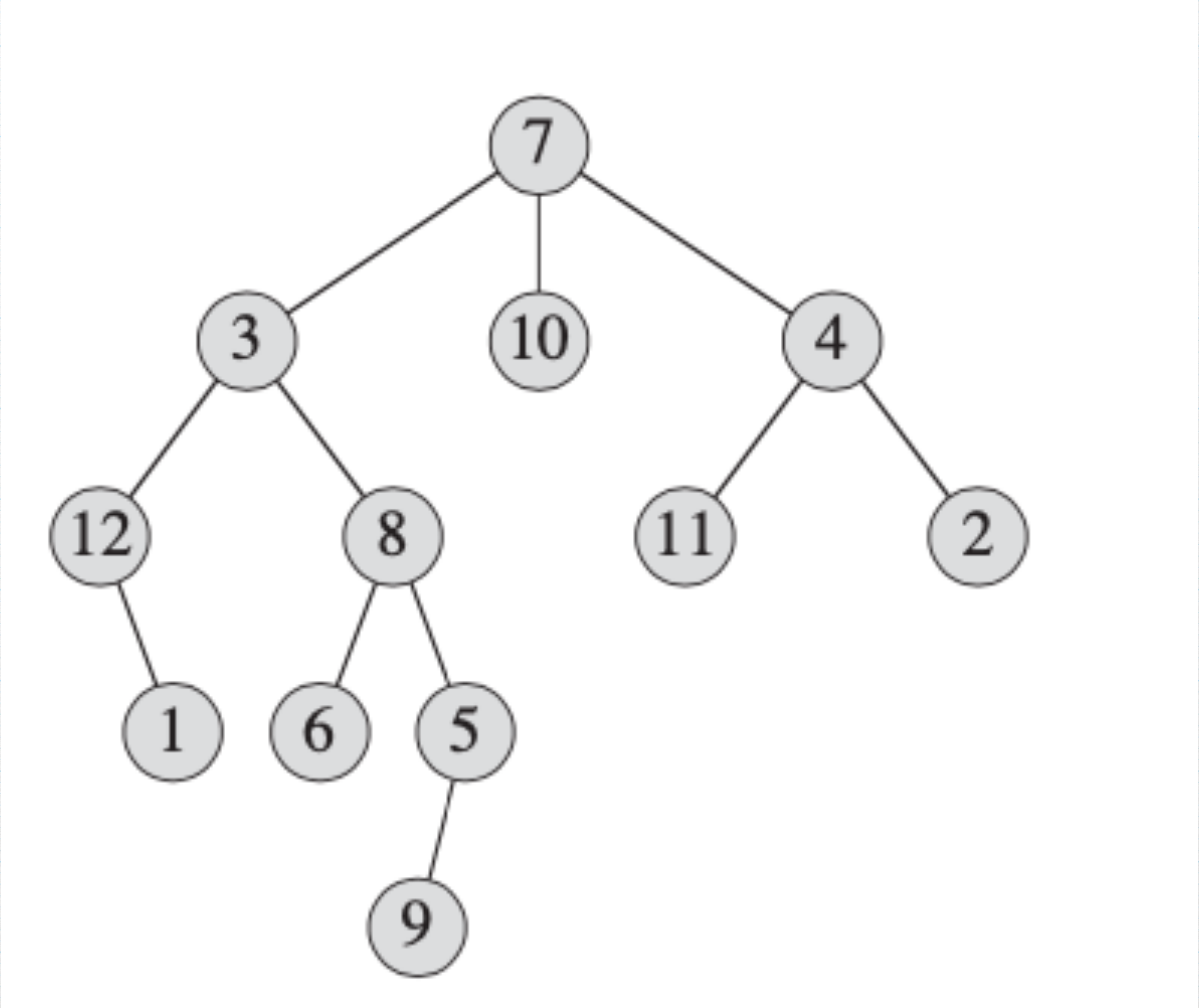
# Key terms - Rooted Trees

- **Ancestor**
- **Descendant**

**<u>Note:</u> Every node is both an ancestor and a descendant itself**

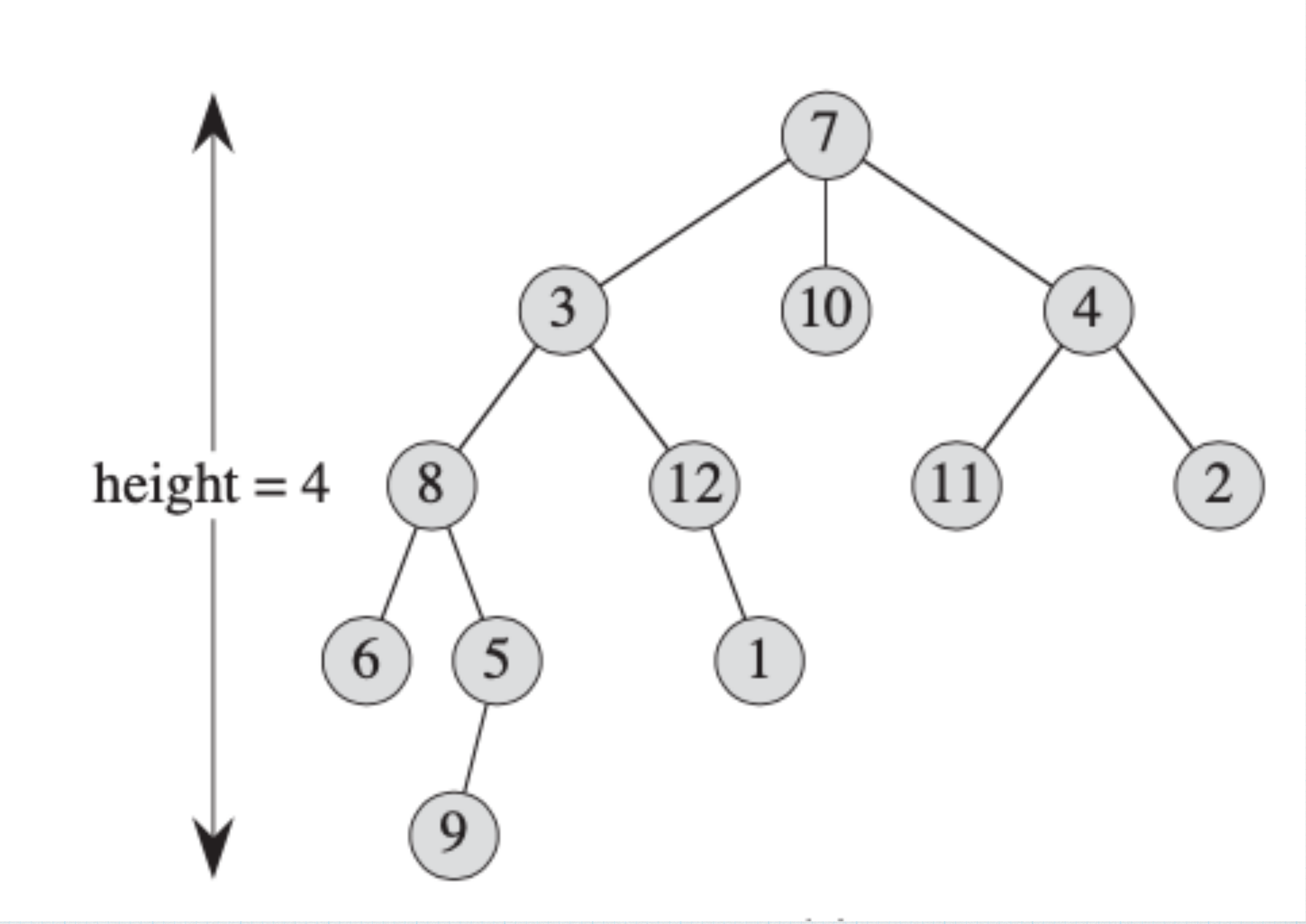
- **Proper ancestor**
- **Proper descendant**
- **Subtree rooted at x**
- **Eg: Subtree rooted at 8?**

  **Subtree rooted at 4?**

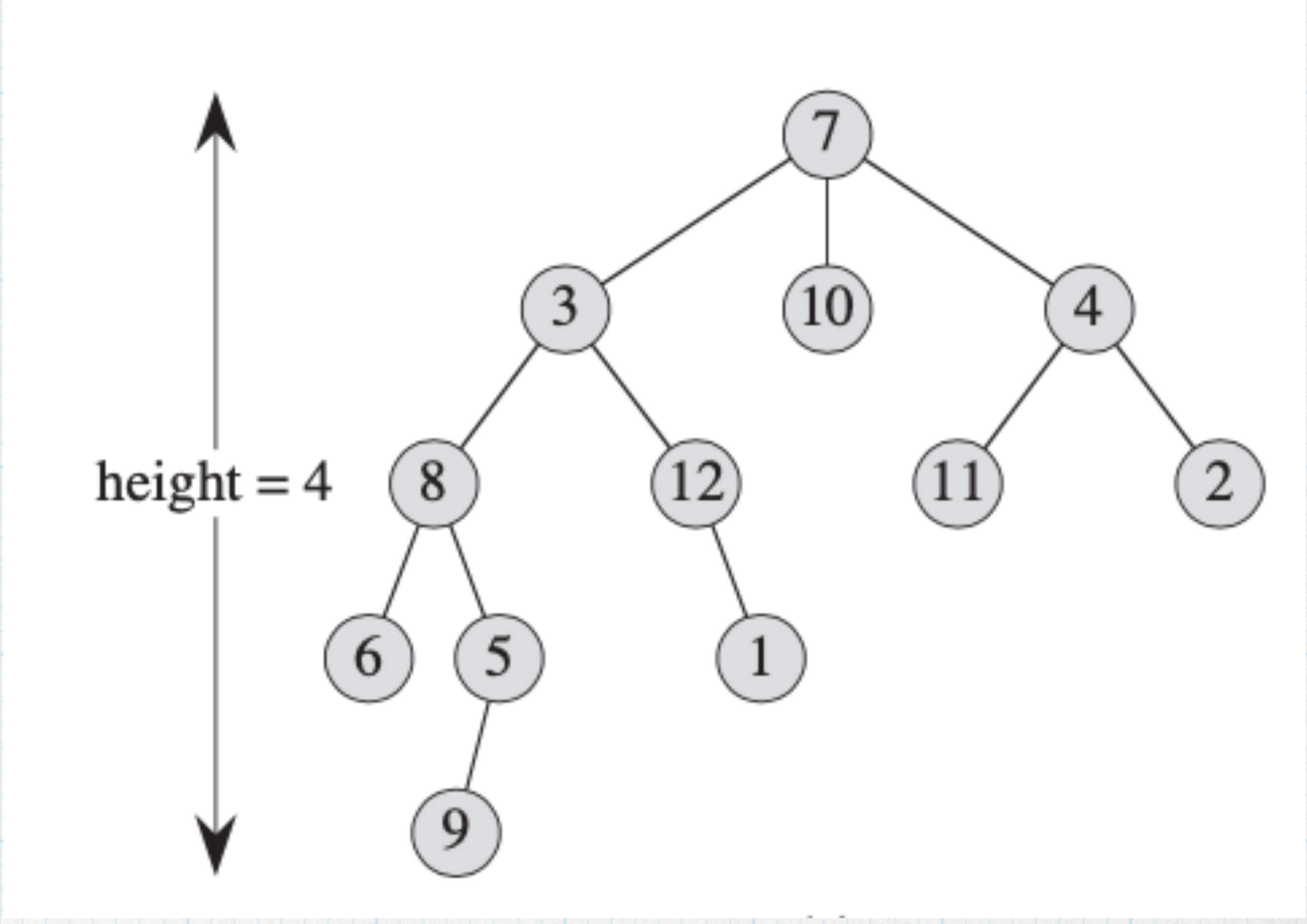


# Key terms - Rooted Trees

- **Parent**
- **Child**
- **Root**
- **Leaf node / External node**
- **Internal node**
- **Siblings**
- **<u>Note:</u> Node with no parents ?**
- **Node with no children ?**



# Degree of a node



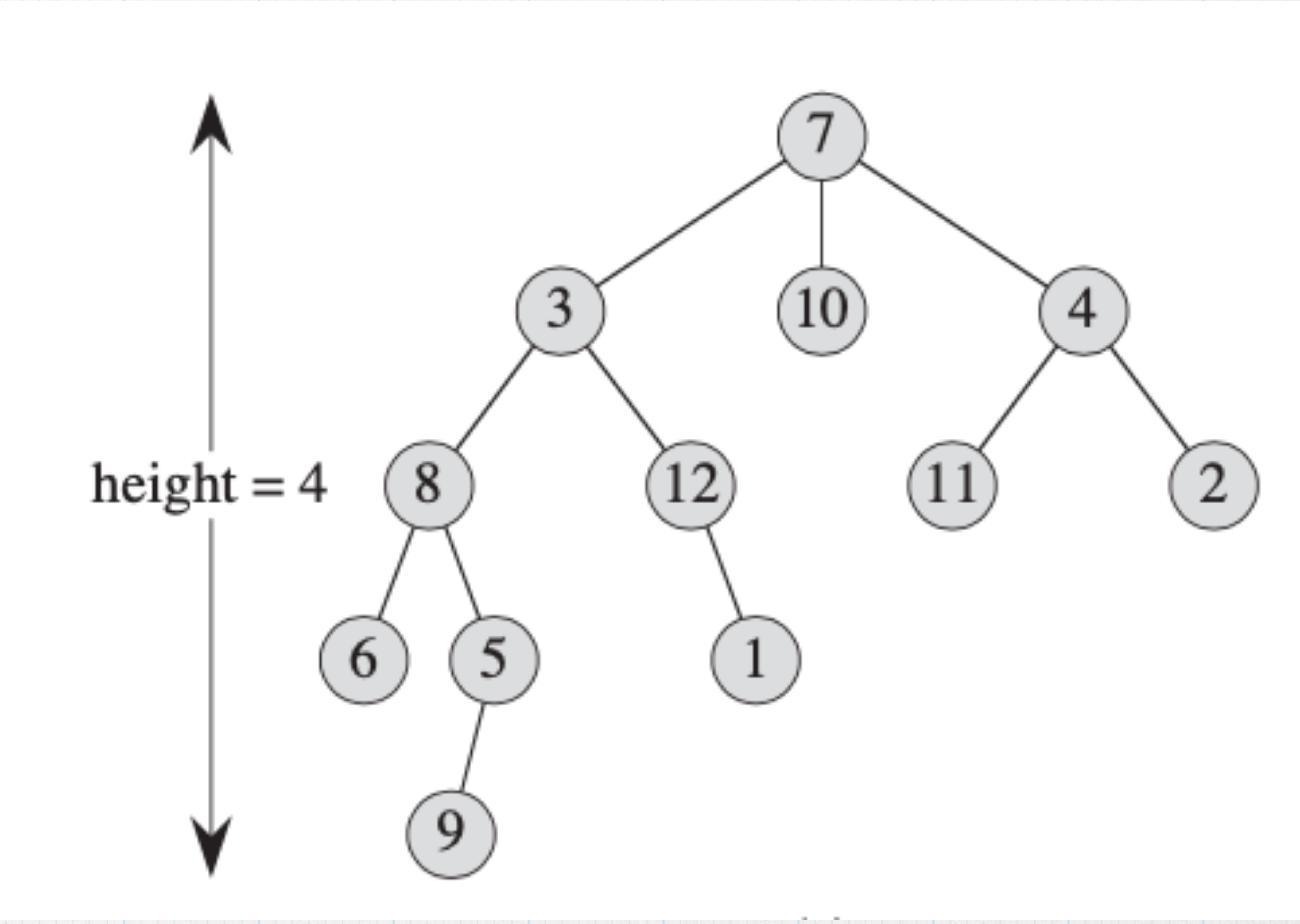

- The number of children of a node x in a rooted tree T equals the ***degree*** of x.

- Note that the degree of a node depends on whether we consider T to be a **rooted tree or a free tree.**

- The **degree** of a vertex in a **free tree** is, as in any **undirected graph**, the number of adjacent vertices.

- In a **rooted tree -** the degree is the number of children
  - the parent of a node does not count toward its degree.

# Depth, Height and Level

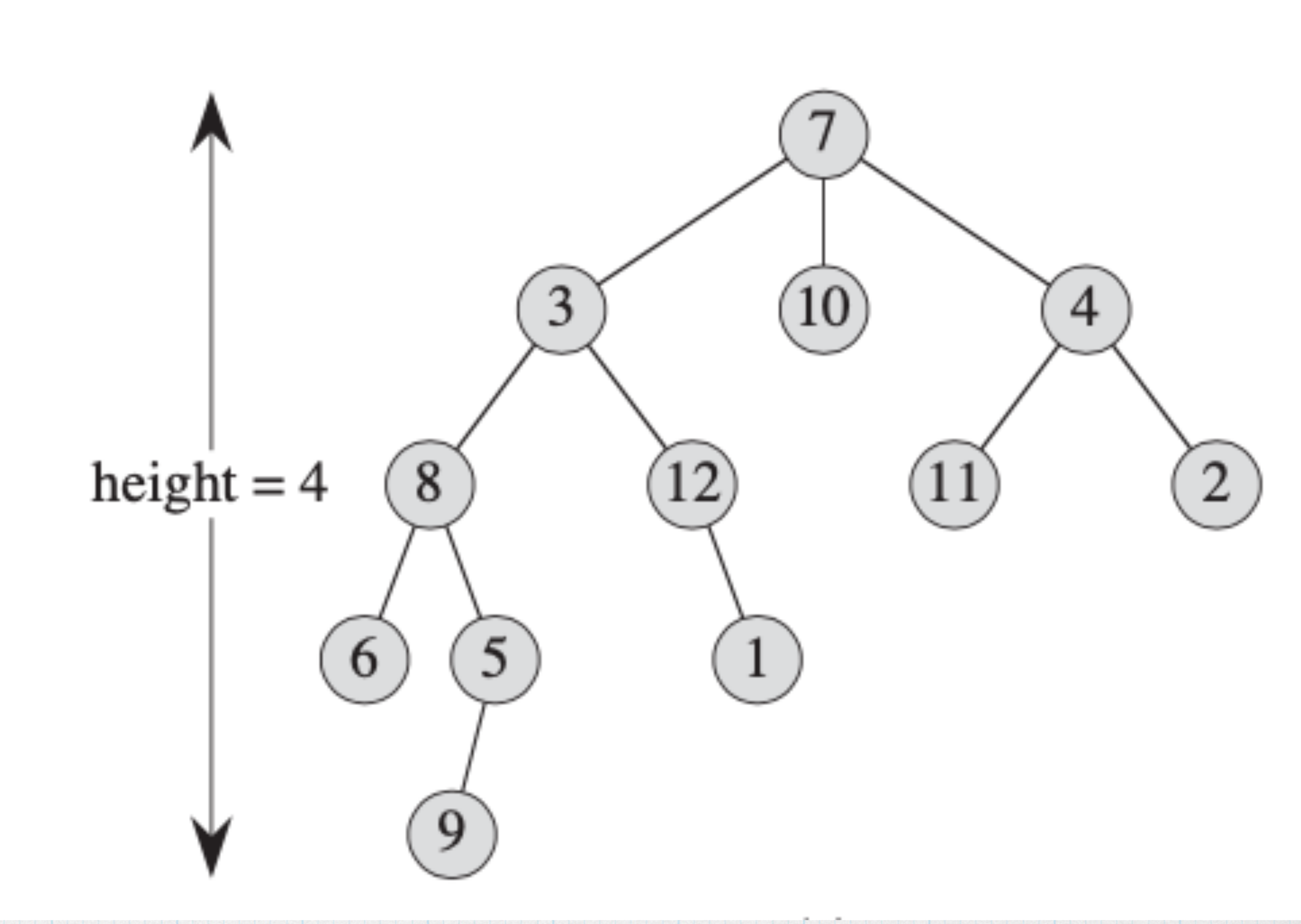



- The length of the simple path from the root r to a node x is the **depth** of x in T .
- A **level** of a tree consists of all nodes at the same depth.
- The **height** of a node in a tree is the **number of edges** on the longest simple downward path from **the node to a leaf,** and the height of a tree is the height of its root.
- The **height of a tree** is also equal to the l**argest depth of any node in the tree.**

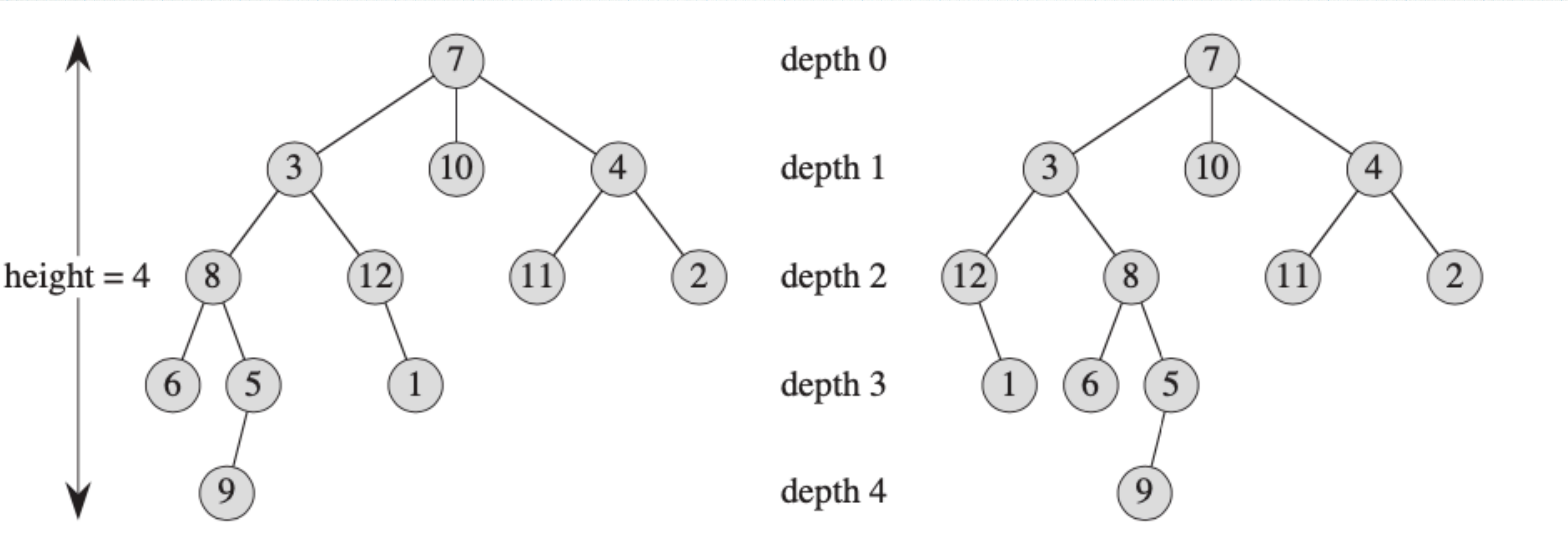
# Ordered Trees

- An ***ordered tree*** is a rooted tree in which the children of each node are ordered.

- That is, if a node has **k children**, then there is a **first child, a second child, . . . , and a kth child.**



# Ordered and Unordered trees

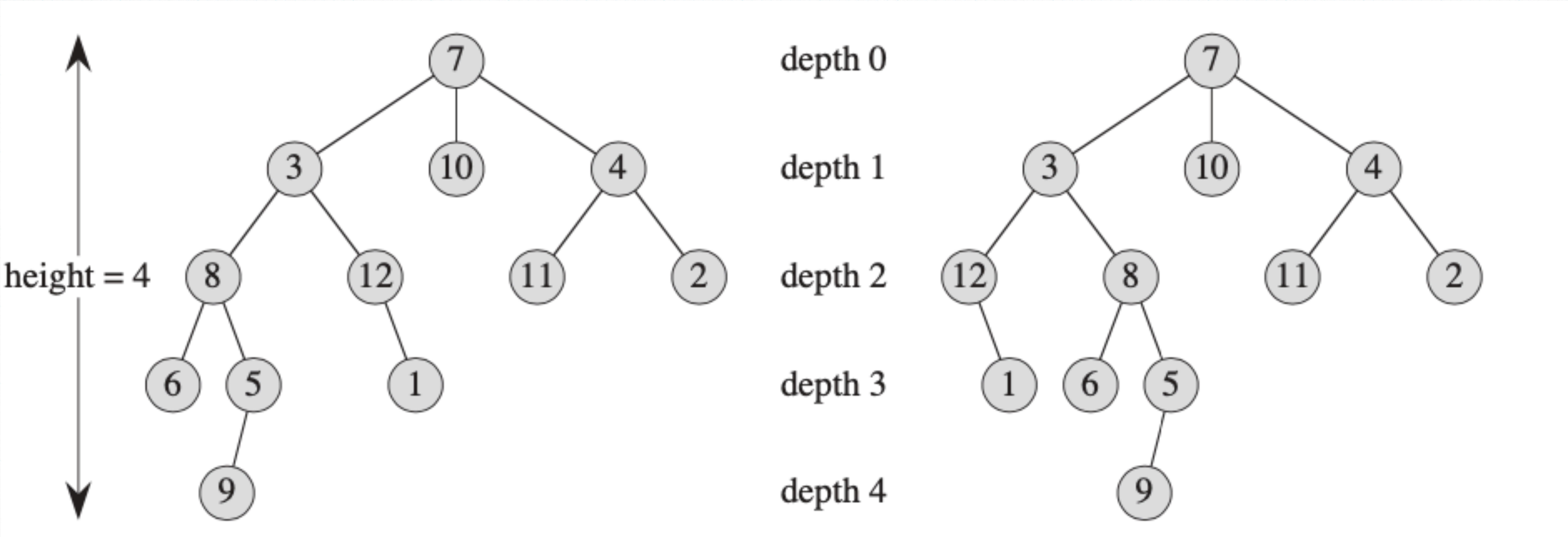



(a) A **rooted tree with height 4.** If the tree is ordered, the **relative left-to-right order of the children** of a node matters; otherwise it doesn't.

(b) As a rooted tree, it is identical to the tree in (a), but as an ordered tree it is different, since the **children of node 3 appear in a different order.**

# Ordered and Unordered trees

- The two trees in the following example are **different** when considered to be **ordered trees**, but the **same** when considered to be just **rooted trees**.



# Binary Trees

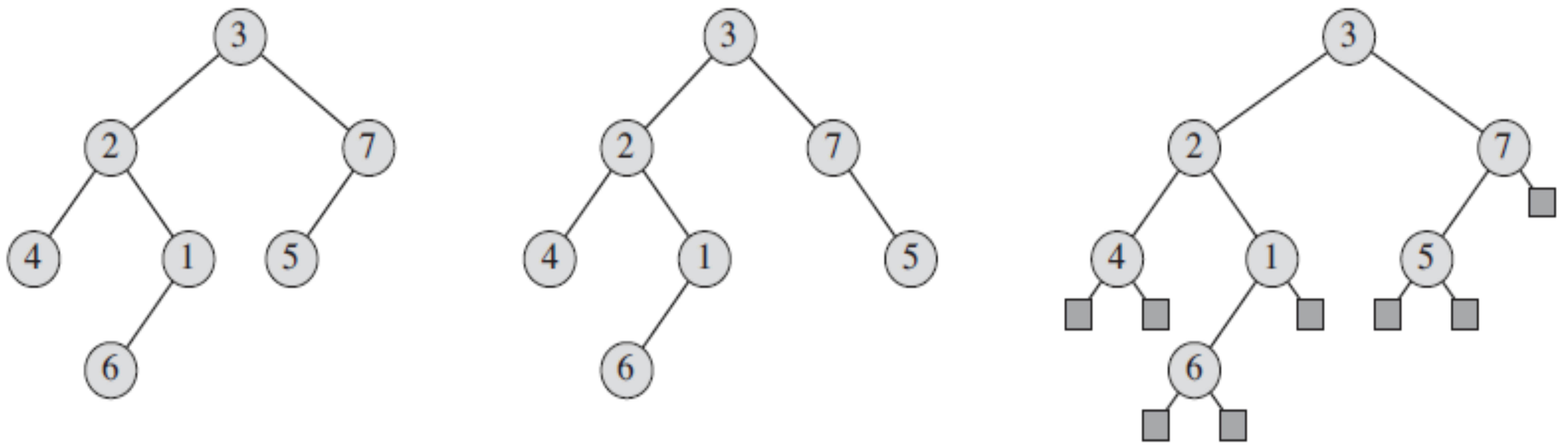
- <u>Binary tree</u>:  A binary tree is defined recursively.

- A binary tree T is a structure defined on finite set of nodes that either
  - Contains no nodes (the *empty tree* or *null tree)*  denoted NIL or
  - Composed of three disjoint set of nodes:
    - a root node
    - a binary tree called its left subtree
    - a binary tree called its right subtree
- *<u>Note:</u>* Left child, Right child, absent or missing child

# Binary Tree Example

- Binary tree - **Not simply an ordered tree** in which each node has **degree at most 2.**
- Example: If a node has just one child, **the position of the child**, whether it is the **left child or the right** child matters.
- **Ordered tree** - *No distinguishing a sole child as being either left or right.*
- Fig (a) binary tree differs from Fig (b) - due to position of one node.
- As ordered trees,

two trees are identical

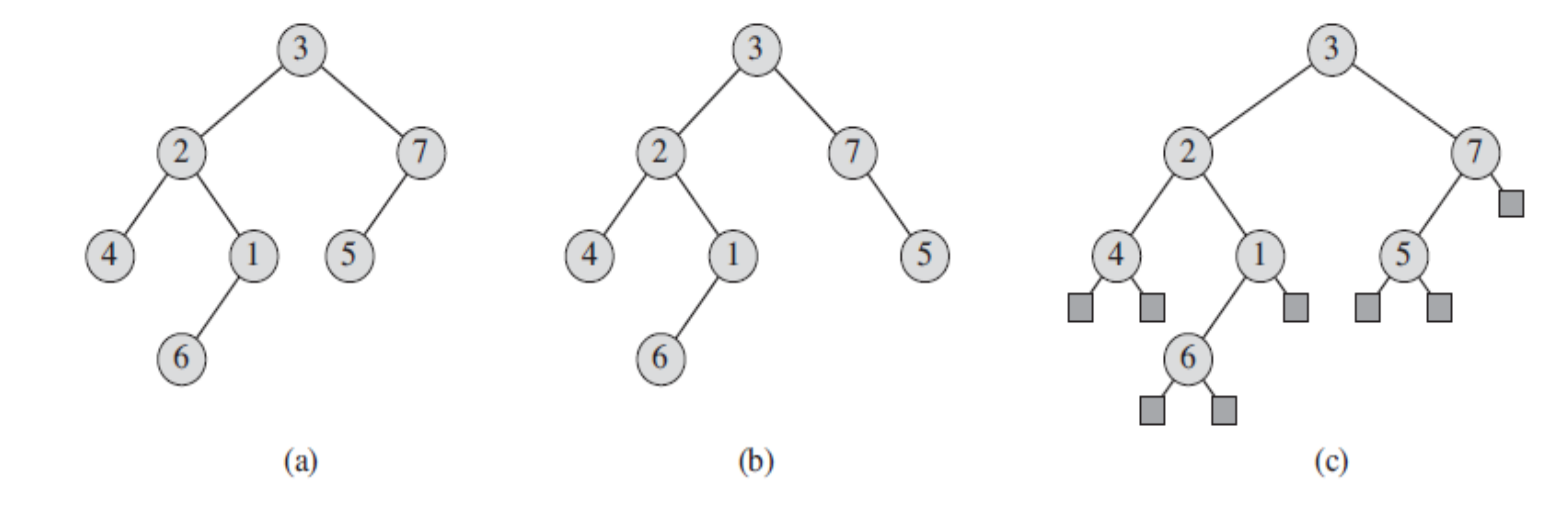



(a) (b) (c)

# Full Binary Tree

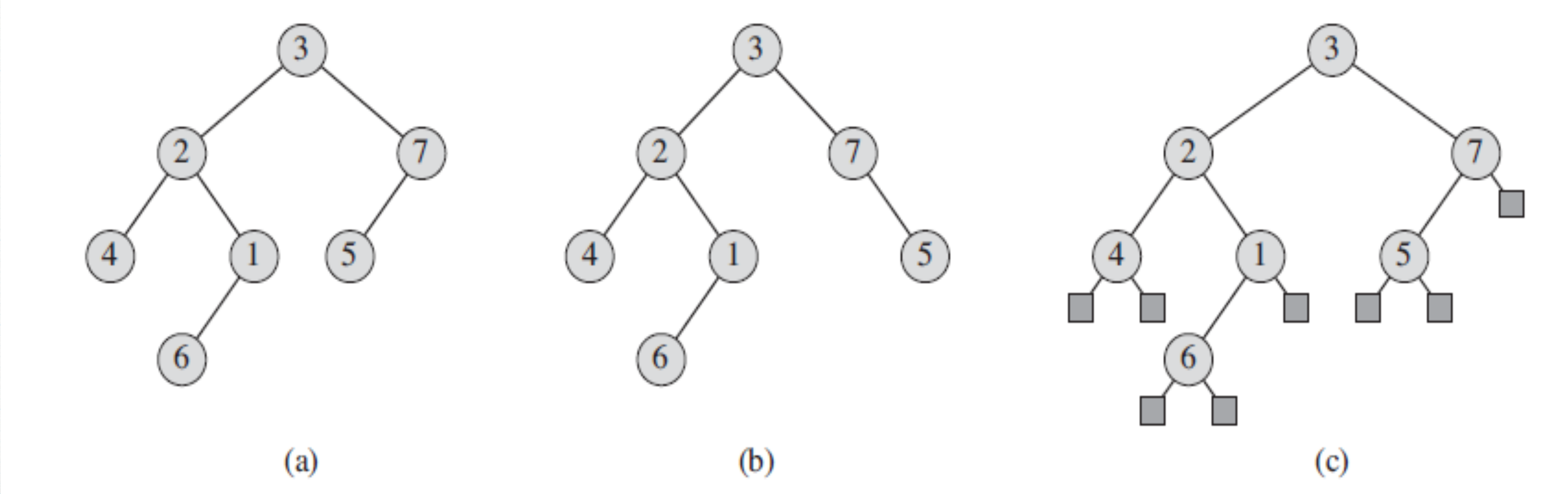
- **Replace each missing child** in the binary tree with a **node having no children**.
- These **leaf nodes** are drawn as **squares** in the figure.
- F***ull binary tree -*** each node is either a leaf or has degree exactly 2.
- **No degree-1 nodes**. the order of the children of a node preserves the

position information.



# Binary tree - Examples

*In the following example,*
*root node:*
*Node 3*
*left subtree:*
*Nodes 2,4,1 and 6 together form Left subtree*
*right subtree:*
*Nodes 7 and 5 together form Right subtree*



*Few terms….*

- The **number of children of a node** x in a rooted tree T equals the **degree** of x

Example:

- Degree of node 3:

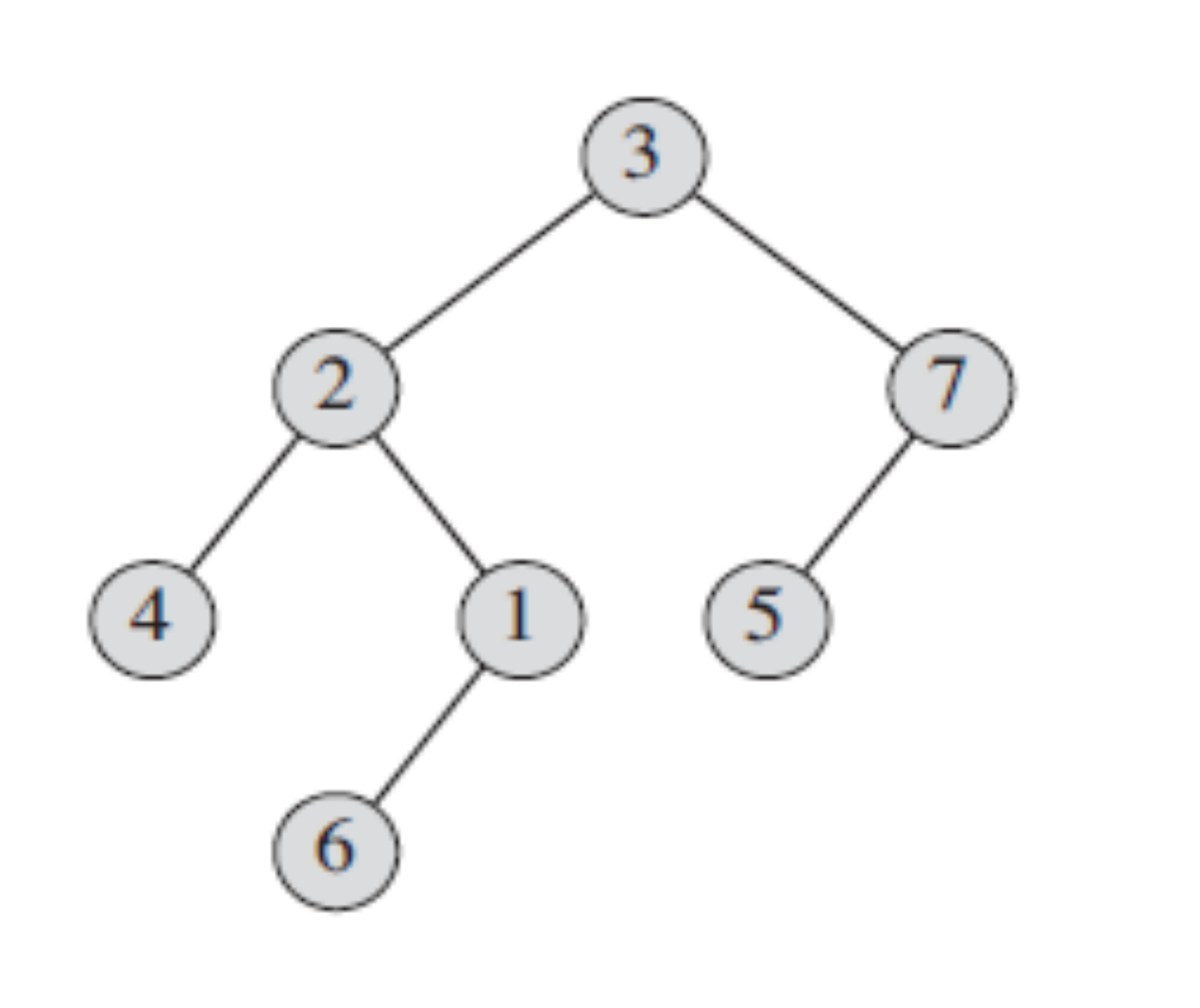
2

- Degree of node 7:

1

- Degree of node 6:

0



* The **length of the simple path** from the root r to a node x is the ***depth* of x** in T .
  * Eg: Depth of node 6:

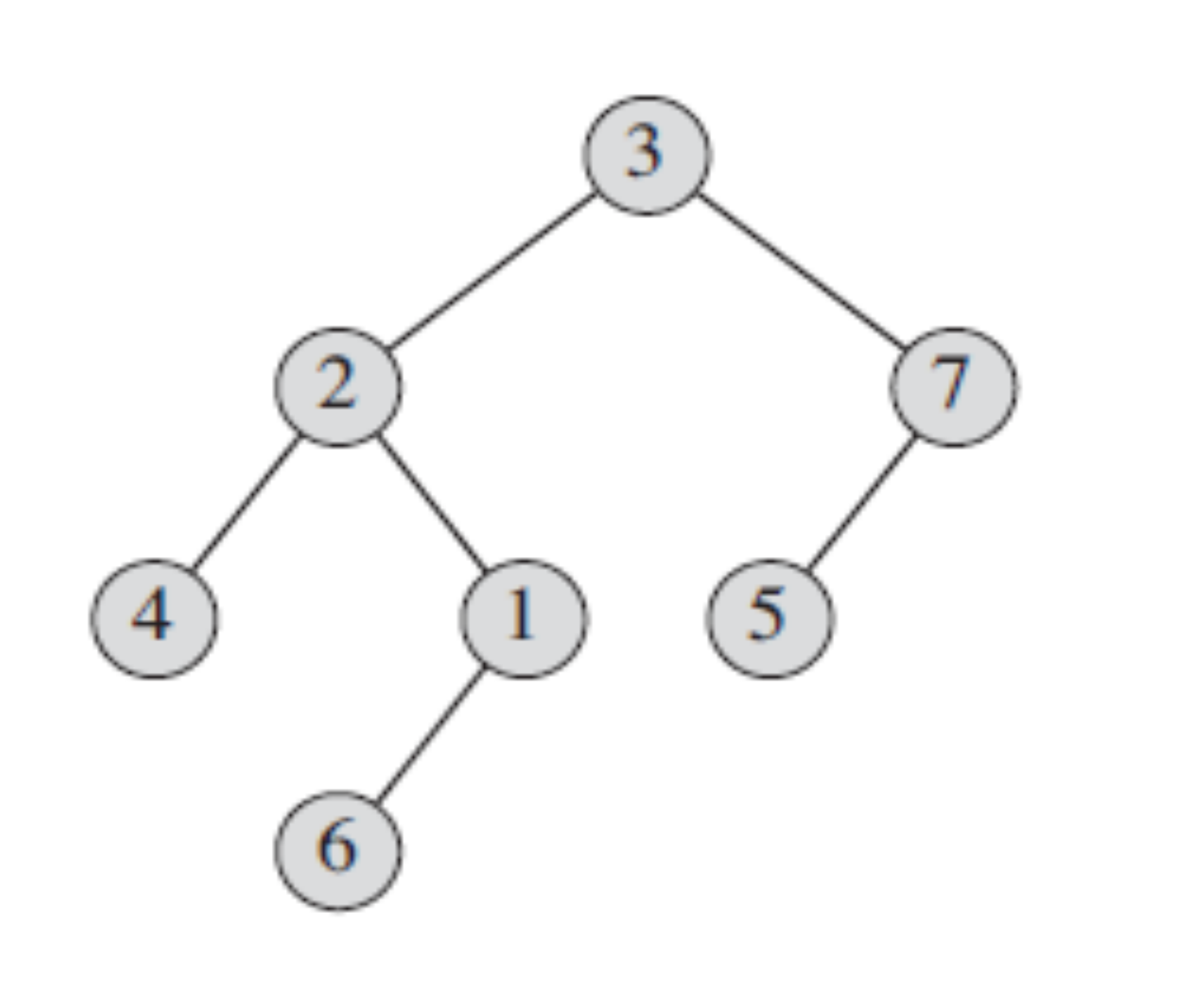
  **3**
  * Depth of nodes 1, 4, 5:

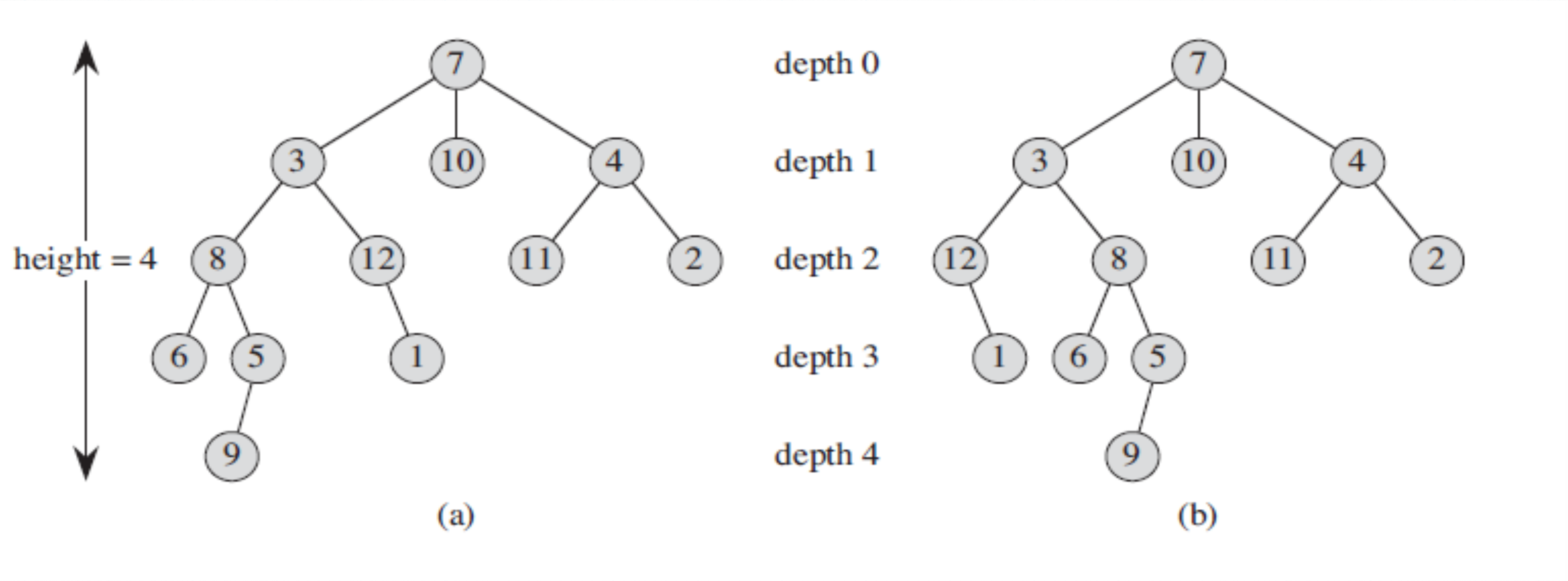
  **2**
  * Depth of nodes 2 and 7:

  **1**
  * Depth of node 3:

  **0**
* A ***level* of a tree** consists of all nodes at the same depth.
  * **Nodes at level 2:**

    4, 1 and 5
* **Ex:** What are the nodes at level 0,1 and 3 in the above tree?






## Few terms....

- A node with no children is a **leaf or external node**. A non-leaf node is an **internal node.**
- The **height of a node** in a tree is the number of edges on the longest simple downward path from the **node to a leaf**
- Height of a tree is the **height of its root.**

# Positional Trees

- Extend the positioning information that distinguishes binary trees from ordered trees to **trees with more than 2 children per node**.
- ***Positional tree***, the children of a node are labeled with **distinct positive integers**. The ith child of a node is ***absent*** if no child is labeled with integer i.
- A **k-*ary* tree** is a positional tree in which for every node, all children with **labels greater than k are missing.**
- Binary tree is a **k-ary** tree with **k = 2.**

*

# ***complete*** k***-ary tree***

- A ***complete*** k***-ary tree*** is a k-ary tree in which all leaves have the same depth and all internal nodes have **degree k.**
- The root has k children at depth 1, each of which has k children at depth 2, etc.
- How many leaves does a complete k-ary tree of **height h** have?
- Number of leaves at depth h is $\mathbf{k^h}$.
- Number of internal nodes of a complete k-ary tree of height h ?

# Complete binary tree

* A **complete binary tree** is a binary tree in which all **leaves have the same depth** and **all internal nodes have degree 2.**
* Number of **internal nodes** in a complete binary tree ?
* Number of **Leaf nodes** ?
* **Total number of nodes (n)** in a complete binary tree of height h, $2^{h+1} - 1$
* What is the **height of a complete binary tree** with n number of nodes?

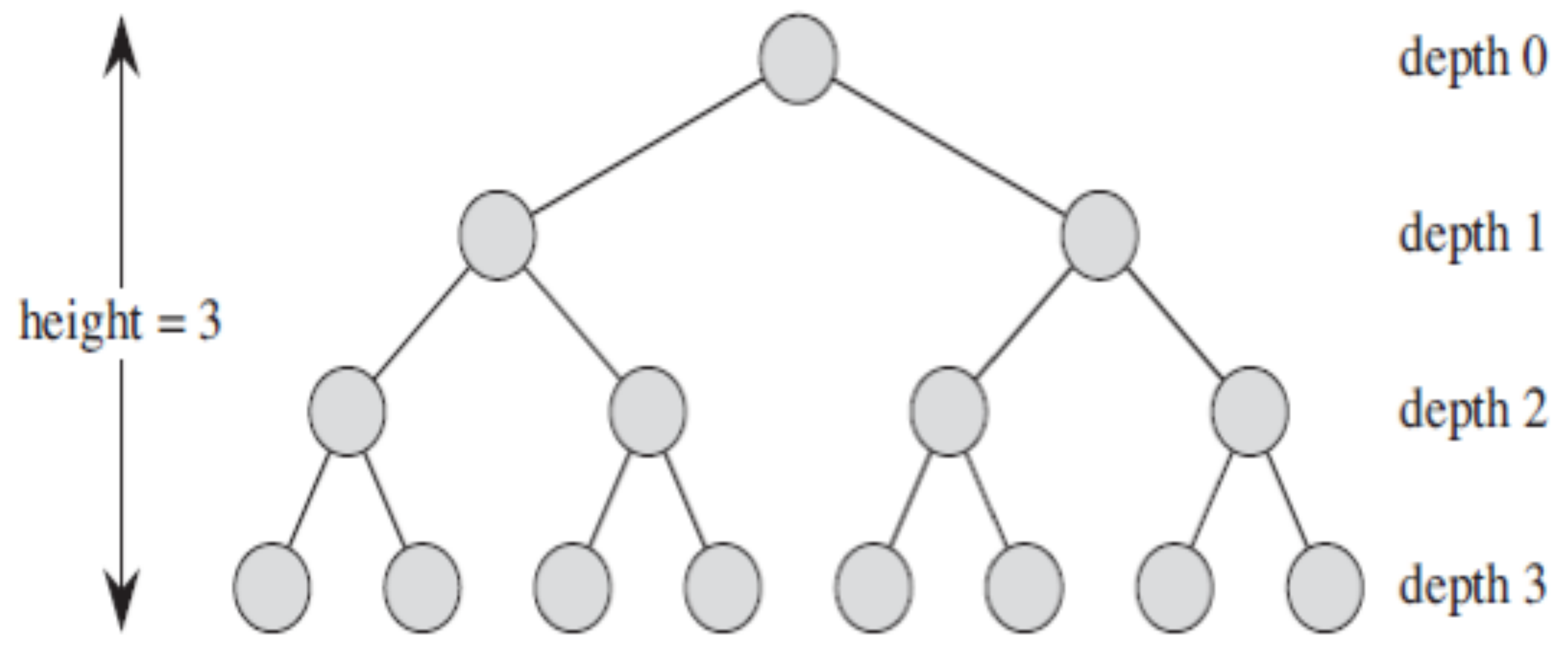



**Figure B.8** A complete binary tree of height 3 with 8 leaves and 7 internal nodes.

# Reference

* CLRS Book