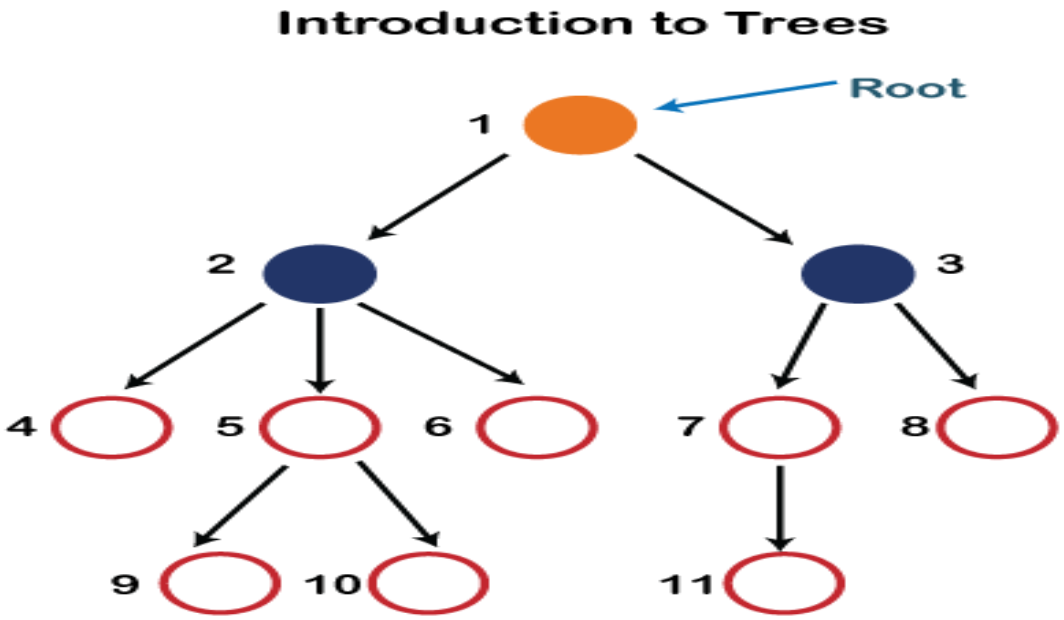
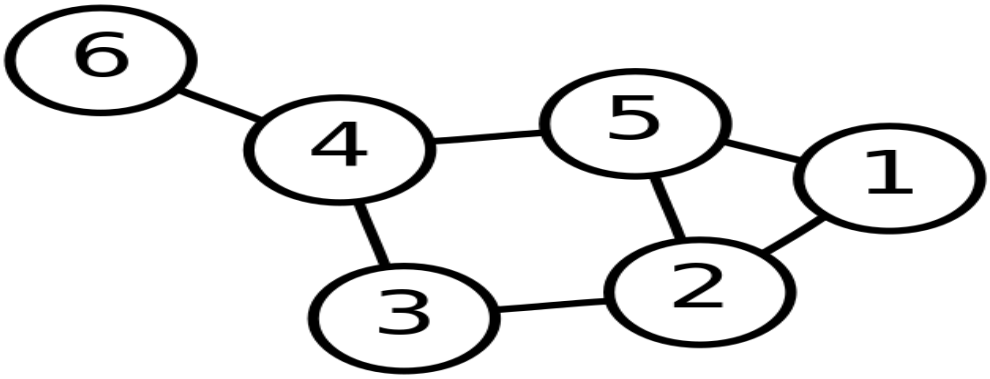
Non-Linear Data Structures

- Non-linear data structures do not follow a sequential order; each element can connect to multiple elements, forming complex relationships.
- The following data structures are considered non-linear data structures:
    a) Tree: A hierarchical structure with nodes connected by edges, commonly used for organizing hierarchical data like file systems.



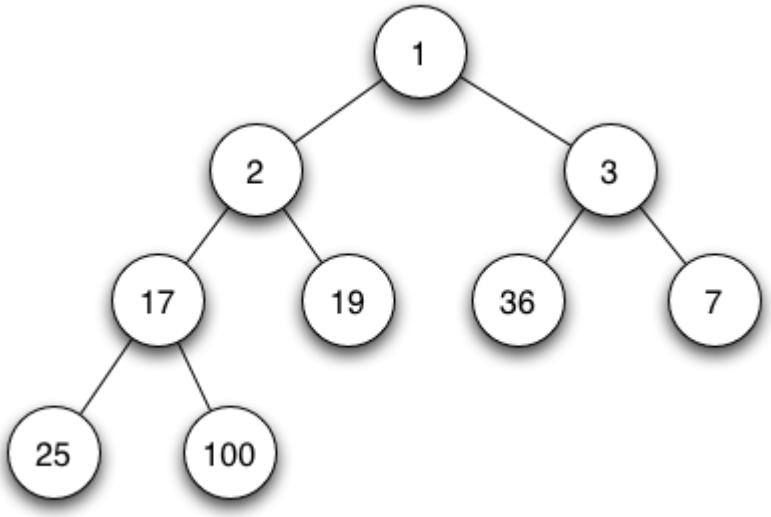
cc: Javatpoint

b) Graph: A collection of nodes connected by edges, allowing versatile representation of relationships between various entities.



cc: Wikipedia

c) Heap: A specialized tree-based structure that satisfies the heap property, often used in priority queues and memory allocation.



cc: Wikipedia

The Need for Non-Linear Data Structures

1) Complex Relationships:
    a) Non-linear structures represent intricate relationships between elements, suitable for scenarios with intricate connections.
2) Hierarchical Organization:
    a) Trees organize data hierarchically, making them ideal for structures like company organization charts.
3) Graph-based Modeling:
    a) Graphs enable modelling of various real-world networks, from social connections to computer networks.
4) Efficient Priority Management:
    a) Heaps efficiently manage priority-based operations, such as extracting the highest-priority element.

Operations on Non-Linear Data Structures

1) Tree Traversal: Navigating trees using methods like in-order, pre-order, and post-order traversal to explore hierarchical data.
2) Graph Traversal: Exploring graphs through traversal algorithms like breadth-first search (BFS) and depth-first search (DFS).
3) Heap Operations: Performing heap-specific operations like insertion, deletion, and heapifying to maintain the heap property.
4) Balancing: Ensuring balanced trees, like AVL or Red-Black trees, to maintain efficient search and insertion operations

Real-world Examples of Non-Linear Data Structures:

1) Tree:
    a) File System: Organizing files in a hierarchical structure mirrors the tree concept, with directories as nodes and files as leaves.
    b) Family Genealogy: Representing family relationships, like a family tree, illustrates the hierarchical nature of tree structures.
2) Graph:
    a) Social Networks: Social media platforms model users and their connections using graph structures to facilitate friend suggestions.
    b) Road Networks: Maps utilize graphs to represent roads and intersections, helping navigation systems find the shortest routes.
3) Heap:
    a) Priority Queue: A hospital's patient queue can be modeled using a heap, with patients ordered by priority for efficient treatment.
    b) Memory Allocation: The memory heap in programming languages allocates memory dynamically, utilizing heap data structure principles.