**Graphs**

**And**

**Trees**

**Name: Mohammed Al-zubairi**

In data structures, particularly in C++, ***trees*** and ***graphs*** are both used to represent hierarchical or networked data but have distinct characteristics:

**1. Tree**:

- Structure: A tree is a hierarchical data structure composed of nodes. It starts with a root node and has child nodes, forming a parent-child relationship. Each node may have zero or more children, but it can have only one parent.

- Acyclic: Trees are acyclic by definition, meaning there is no way to start at one node and return to it by following the parent-child relationships.

Connected: Trees are always connected; there's a path between the root node and every other node in the tree.

**Common Types:**

- Binary Tree: Each node has at most two children (left and right).

- Binary Search Tree (BST): A binary tree where the left child contains values less than the parent, and the right child contains values greater than the parent.

- AVL Tree: A self-balancing binary search tree.

- Red-Black Tree: Another type of self-balancing binary search tree.

**Examples:**

File system hierarchies, organizational charts, decision trees.

**2. Graph:**

**Structure:**

A graph is a collection of nodes (called vertices) and edges that connect pairs of nodes. Unlike trees, graphs do not have to follow a hierarchical structure.Cyclic or Acyclic: Graphs can be cyclic (contain cycles) or acyclic (contain no cycles).

**Directed or Undirected:**

- Directed Graph (Digraph): The edges have a direction, going from one vertex to another.

- Undirected Graph: The edges have no direction, meaning if there's an edge between node A and node B, you can traverse it in both directions.

**Connected or Disconnected:**

Graphs can be connected (there is a path between any pair of vertices) or disconnected (some vertices may not have any paths connecting them).

**Common Types:**

- Weighted Graph: Each edge has a weight or cost associated with it.

- DAG (Directed Acyclic Graph): A directed graph with no cycles, often used in scheduling algorithms, dependency graphs, etc.

**Examples:**

Social networks, transportation networks, communication networks.

**Differences:**

- Hierarchical vs. Networked: Trees represent hierarchical relationships, while graphs are more general and can represent any type of relationship between nodes.

- Acyclic vs. Cyclic: Trees are inherently acyclic, whereas graphs can have cycles.

- Rooted vs. Unrooted: Trees have a root node, whereas graphs do not necessarily have a starting point.

- Parent-Child Relationship: In trees, there's a clear parent-child relationship. In graphs, any node can be connected to any other node, without a hierarchical constraint.

**Code Example in C++**

Tree:

cpp

struct TreeNode {

int val;

TreeNode \*left;

TreeNode \*right;

TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}

};

Graph:

cpp

struct GraphNode {

int val;

vector<GraphNode\*> neighbors;

GraphNode(int x) : val(x) {}

};