#Colas (Queues)

Las colas son una estructura de datos lineal que sigue el principio \*FIFO (First In, First Out)\*, es decir, el primer elemento en entrar es el primero en salir. Se usan para gestionar elementos en orden secuencial, como en colas de espera.

- \*Operaciones principales\*:

- \*Enqueue\*: Agregar un elemento al final de la cola.

- \*Dequeue\*: Remover el elemento del frente de la cola.

- \*Front/Peek\*: Ver el elemento del frente sin removerlo.

- \*IsEmpty\*: Verificar si la cola está vacía.

- \*Implementación\*: Pueden implementarse con arreglos o listas enlazadas. En arreglos, se usan índices para el frente y el final; en listas enlazadas, se agregan al final y se remueven del inicio.

- \*Complejidad de tiempo\*: Enqueue y Dequeue son O(1) en listas enlazadas; en arreglos circulares, también O(1) promedio.

- \*Ejemplos de uso\*: Colas de impresión, gestión de procesos en sistemas operativos, algoritmos como BFS (Breadth-First Search) en grafos.

#Árboles (Trees)

Los árboles son estructuras de datos jerárquicas no lineales compuestas por nodos conectados por aristas. No tienen ciclos y cada nodo (excepto la raíz) tiene exactamente un padre.

- \*Componentes clave\*:

- \*Raíz\*: Nodo superior sin padre.

- \*Hojas\*: Nodos sin hijos.

- \*Niveles\*: Profundidad desde la raíz.

- \*Altura\*: Máxima profundidad.

- \*Tipos comunes\*: Árboles binarios, AVL, B-trees, etc.

- \*Operaciones principales\*: Inserción, eliminación, búsqueda, recorridos (inorden, preorden, postorden).

- \*Implementación\*: Usualmente con nodos que contienen referencias a hijos.

- \*Complejidad de tiempo\*: Depende del tipo; en árboles no balanceados, búsque|da puede ser O(n), pero en balanceados como AVL, es O(log n).

- \*Ejemplos de uso\*: Representar jerarquías (como sistemas de archivos), algoritmos de búsqueda, expresiones matemáticas.

# Árboles Binarios (Binary Trees)

Son un tipo especial de árbol donde cada nodo tiene como máximo dos hijos: izquierdo y derecho. Son fundamentales en informática por su simplicidad y eficiencia.

- \*Tipos\*:

- \*Completo\*: Todos los niveles llenos excepto posiblemente el último, que se llena de izquierda a derecha.

- \*Lleno\*: Todos los nodos tienen cero o dos hijos.

- \*Balanceado\*: La altura de subárboles izquierdo y derecho difiere en máximo 1 (ej. AVL, Red-Black).

- \*Árbol de Búsqueda Binaria (BST)\*: Valores menores a la izquierda, mayores a la derecha; búsqueda en O(log n) promedio.

- \*Operaciones principales\*: Inserción, eliminación, búsqueda, recorridos (inorden para BST ordenado).

- \*Implementación\*: Nodos con punteros a izquierdo, derecho y valor.

- \*Complejidad de tiempo\*: En BST balanceado, operaciones son O(log n); en no balanceado, O(n).

- \*Ejemplos de uso\*: Heaps (para colas de prioridad), BST para bases de datos, expresiones aritméticas.

# Grafos (Graphs)

Los grafos son estructuras de datos que representan relaciones entre objetos. Consisten en \*vértices (nodos)\* conectados por \*aristas (edges)\*. Pueden tener ciclos y no son jerárquicos como los árboles.

- \*Tipos\*:

- \*Dirigidos\*: Aristas tienen dirección (ej. redes sociales con seguidores).

- \*No dirigidos\*: Aristas bidireccionales (ej. mapas de carreteras).

- \*Ponderados\*: Aristas con pesos (ej. distancias en mapas).

- \*Representaciones\*:

- \*Lista de adyacencia\*: Lista de vértices conectados a cada nodo (eficiente para grafos dispersos).

- \*Matriz de adyacencia\*: Matriz booleana o ponderada (eficiente para grafos densos).

- \*Operaciones principales\*: Agregar/eliminar vértices/aristas, recorridos (DFS - Depth-First Search, BFS - Breadth-First Search), algoritmos de caminos más cortos (Dijkstra, Bellman-Ford).

- \*Complejidad de tiempo\*: Depende del algoritmo; DFS/BFS son O(V + E), donde V es vértices y E aristas.

- \*Ejemplos de uso\*: Redes sociales, mapas (GPS), circuitos eléctricos, algoritmos de IA como búsqueda en grafos de estados.

Recuperacion:

