**1. Árboles B**

Un **Árbol B** es una estructura de datos autoequilibrada que mantiene datos ordenados y permite búsquedas, inserciones, eliminaciones y accesos secuenciales en tiempo logarítmico. Es especialmente útil para sistemas de archivos y bases de datos.

* **Propiedades de los Árboles B:**
  + Cada nodo tiene al menos m/2 hijos (excepto la raíz, que puede tener menos) y como máximo m hijos, donde m es el orden del árbol.
  + Un nodo interno con n hijos contiene n-1 claves.
  + Las claves de cada nodo están ordenadas.
  + Todas las hojas están al mismo nivel, lo que asegura el equilibrio del árbol.
* **Operaciones:**
  + **Inserción:** Se realiza en una hoja. Si la hoja ya tiene el número máximo de claves, se divide en dos hojas y se ajustan los padres de manera recursiva.
  + **Eliminación:** Si se elimina una clave de una hoja que contiene el número mínimo de claves, se realiza una combinación o redistribución de claves para mantener las propiedades del árbol.

**2. Árboles B+**

Un **Árbol B+** es una variación del Árbol B donde todas las claves están presentes solo en las hojas, mientras que los nodos internos solo actúan como índices para guiar la búsqueda.

* **Características del Árbol B+:**
  + Todas las hojas están conectadas mediante punteros, lo que facilita el acceso secuencial a los datos.
  + Al igual que los Árboles B, los Árboles B+ también son autoequilibrados.
* **Ventajas del Árbol B+:**
  + Búsquedas más rápidas y eficientes ya que las claves están más compactamente almacenadas en los nodos internos.
  + Es ideal para sistemas de bases de datos y sistemas de archivos donde se requiere una búsqueda secuencial y de rango eficiente.

**3. Grafos**

Un **Grafo** es una estructura matemática compuesta por un conjunto de vértices (o nodos) y un conjunto de aristas (o arcos) que conectan pares de vértices.

* **Tipos de Grafos:**
  + **Dirigidos (o Digrafos):** Las aristas tienen una dirección, de un vértice a otro.
  + **No Dirigidos:** Las aristas no tienen una dirección.
  + **Ponderados:** Las aristas tienen un peso asociado.
  + **No Ponderados:** Las aristas no tienen un peso asociado.
* **Aplicaciones:** Los grafos se utilizan para modelar redes (como internet o redes sociales), sistemas de transporte, circuitos electrónicos, etc.

**4. Metodologías de Solución Algorítmica**

Las **Metodologías de Solución Algorítmica** son enfoques o estrategias para diseñar algoritmos que resuelvan problemas computacionales.

* **Búsqueda Exhaustiva (Fuerza Bruta):** Examina todas las posibles soluciones para encontrar la mejor. Es simple pero a menudo ineficiente.
* **Divide y Vencerás:** Divide un problema en subproblemas más pequeños, resuelve cada subproblema recursivamente y combina sus soluciones.
* **Algoritmos Voraces:** En cada paso, toma la decisión que parece la mejor en ese momento, sin considerar decisiones futuras.

**5. Búsqueda Exhaustiva (Fuerza Bruta)**

**Búsqueda Exhaustiva** o **Fuerza Bruta** es una técnica donde se exploran todas las posibles soluciones para encontrar la mejor o una válida.

* **Búsqueda Lineal:** Consiste en recorrer una lista de elementos de principio a fin para encontrar un elemento específico. Es simple pero tiene una complejidad de tiempo de O(n).

**6. Algoritmos Voraces**

Un **Algoritmo Voraz** toma decisiones basadas en la mejor opción disponible en el momento, sin reconsiderar decisiones previas.

* **Características:**
  + No siempre garantiza la solución óptima, pero puede ser eficiente y simple de implementar.
  + Ejemplos de problemas donde se aplica: el problema del cambio de monedas, el problema del viajante de comercio (versión aproximada), y el problema de la mochila fraccional.

**7. Divide y Vencerás**

La estrategia **Divide y Vencerás** consiste en dividir un problema en subproblemas más pequeños, resolver cada uno de ellos recursivamente, y luego combinar sus soluciones para resolver el problema original.

* **Ejemplos de Algoritmos:**
  + **Merge Sort:** Divide la lista en dos mitades, ordena cada mitad recursivamente y luego combina las dos mitades ordenadas.
  + **Quick Sort:** Selecciona un pivote y particiona los elementos en dos sublistas, una con elementos menores que el pivote y otra con elementos mayores, y luego ordena las sublistas recursivamente.
  + **Búsqueda Binaria:** Encuentra un elemento en una lista ordenada dividiendo repetidamente el rango de búsqueda a la mitad.

**8. Dijkstra**

El **Algoritmo de Dijkstra** encuentra el camino más corto desde un nodo origen a todos los demás nodos en un grafo ponderado no dirigido o dirigido con pesos no negativos.

* **Funcionamiento:**
  + Inicializa la distancia a la fuente como 0 y a todos los demás vértices como infinito.
  + Usa una cola de prioridad para explorar los vértices de menor distancia acumulada, actualizando las distancias a los vértices vecinos.
  + Repite el proceso hasta que todos los vértices hayan sido visitados.

**9. Kruskal**

El **Algoritmo de Kruskal** encuentra el árbol de expansión mínima (MST) de un grafo no dirigido ponderado.

* **Funcionamiento:**
  + Ordena todas las aristas del grafo según sus pesos.
  + Agrega las aristas al MST de una en una, comenzando con la más corta, siempre y cuando no formen un ciclo.
  + Utiliza una estructura de datos de conjunto disjunto para detectar ciclos.

**10. Floyd-Warshall**

El **Algoritmo de Floyd-Warshall** encuentra los caminos más cortos entre todos los pares de nodos en un grafo ponderado dirigido.

* **Funcionamiento:**
  + Utiliza una técnica de programación dinámica para construir una matriz de distancias.
  + Iterativamente mejora la estimación de la distancia entre cada par de vértices considerando todos los posibles vértices intermedios.
  + Complejidad de tiempo es O(n^3), donde n es el número de vértices.