Manual técnico

- Árbol B de orden 5

Un Árbol B de orden t (en este caso, $t = 5$) es un árbol de búsqueda en el que cada nodo interno puede contener hasta $2t-1$ claves y tiene hasta $2t$ hijos. Las propiedades específicas de un Árbol B de orden 5 son:

1. Cada nodo tiene como máximo 9 claves y 10 hijos.
2. Cada nodo tiene como mínimo 2 claves (excepto la raíz).
3. La raíz tiene al menos 1 clave.
4. Todas las hojas están al mismo nivel.

Propiedades Clave

1. Balanceo Automático: El Árbol B se mantiene balanceado, asegurando que las operaciones de búsqueda, inserción y eliminación sean eficientes.
2. Altura Baja: La altura de un Árbol B es $O(Log_t(n))$, donde n es el número de claves en el árbol.
3. Altura Uniforme: Todas las hojas están a la misma profundidad, garantizando tiempos de acceso consistentes.

Operaciones en un Árbol B de Orden 5

1. Búsqueda: Similar a la búsqueda en un árbol binario, pero se realiza en un nodo que puede tener múltiples claves.
2. Inserción:
   - Se inserta la nueva clave en el nodo hoja correspondiente.
   - Si el nodo se llena (más de 9 claves), se divide en dos nodos, y la clave del medio se mueve al nodo padre.
   - Este proceso puede propagarse hacia arriba, hasta la raíz si es necesario.
3. Eliminación:
   - Se eliminan las claves siguiendo reglas específicas para mantener las propiedades del árbol.
   - Si un nodo tiene menos de 2 claves después de una eliminación, se redistribuyen las claves de los nodos hermanos o se fusionan nodos si es necesario.

- Árbol binario de búsqueda

Un BST es un tipo especial de árbol binario que satisface las siguientes propiedades:

1. Cada nodo tiene como máximo dos hijos.
2. El hijo izquierdo de un nodo contiene un valor menor que el valor del nodo.
3. El hijo derecho de un nodo contiene un valor mayor que el valor del nodo.

Propiedades Clave

1. Ordenación: Los valores en el árbol están organizados de tal manera que permite búsquedas eficientes.
2. Operaciones en Tiempo Logarítmico: En un BST balanceado, las operaciones de búsqueda, inserción y eliminación se pueden realizar en tiempo $O(\log n)$, donde n es el número de nodos en el árbol.
3. Recorridos Eficientes: Permite diversos tipos de recorridos (inorden, preorden, postorden) que facilitan la visita sistemática de los nodos.

Operaciones en un Árbol Binario de Búsqueda

1. Búsqueda:
- Se comienza en la raíz y se compara el valor buscado con el valor del nodo actual.
- Si el valor es menor, se sigue hacia el hijo izquierdo; si es mayor, hacia el hijo derecho.
- El proceso se repite hasta encontrar el valor o llegar a un nodo nulo (indicando que el valor no está en el árbol).

2. Inserción:
- Se busca la posición correcta en el árbol siguiendo las reglas de comparación.
- Una vez encontrada la posición correcta (un nodo nulo), se inserta el nuevo nodo en esa posición.

3. Eliminación:
- Si el nodo a eliminar no tiene hijos, simplemente se elimina.
- Si tiene un solo hijo, se elimina el nodo y se conecta su hijo directamente con el padre del nodo eliminado.
- Si tiene dos hijos, se encuentra el nodo con el valor más pequeño en el subárbol derecho (sucesor inorden) o el nodo con el valor más grande en el subárbol izquierdo (predecesor inorden) para reemplazar el valor del nodo a eliminar, y luego se elimina ese nodo sucesor o predecesor.

- Tabla Hash

Una tabla hash es una estructura de datos que se organiza como un arreglo de tamaño fijo m, donde cada entrada puede contener un par clave-valor. Las operaciones de inserción, búsqueda y eliminación se realizan utilizando una función hash que mapea las claves a índices en el arreglo.

Propiedades Clave

1. Función Hash: Una función que convierte una clave en un índice dentro del rango de la tabla. Idealmente, distribuye las claves uniformemente entre las casillas de la tabla para minimizar las colisiones.
2. Colisiones: Ocurren cuando dos claves distintas producen el mismo índice. Se deben manejar para asegurar la integridad de los datos.
3. Eficiencia: Ofrecen una eficiencia promedio $O(1)$ para operaciones de búsqueda, inserción y eliminación, siempre y cuando la función hash distribuya bien las claves.

Operaciones en una Tabla Hash

1. Inserción:
- Calcular el índice de la clave usando la función hash.
- Colocar el par clave-valor en la posición calculada.

- Manejar colisiones si es necesario.

2. Búsqueda:

- Calcular el índice de la clave usando la función hash.
- Acceder a la casilla correspondiente.
- Si hay colisiones, buscar en la lista enlazada o utilizar la técnica de manejo de colisiones apropiada.

3. Eliminación:

- Calcular el índice de la clave usando la función hash.
- Eliminar el par clave-valor de la casilla correspondiente.
- Ajustar la estructura para manejar colisiones, si es necesario.

- Matriz de adyacencia

Una matriz de adyacencia es una forma de representar grafos en la teoría de grafos y ciencias de la computación. Se utiliza para modelar relaciones entre pares de nodos en un grafo, proporcionando una estructura compacta y eficiente para realizar operaciones sobre grafos.

Definición de Matriz de Adyacencia

Una matriz de adyacencia es una matriz cuadrada de tamaño $n \times n$, donde n es el número de nodos en el grafo. Cada entrada en la matriz indica si existe una arista entre dos nodos específicos.

Representación

Para un grafo con n nodos etiquetados como $0, 1, 2, \ldots, n-1$:

- Grafos No Dirigidos: La matriz es simétrica. Si hay una arista entre los nodos i y j, entonces $A[i][j] = 1$ y $A[j][i] = 1$.
- Grafos Dirigidos: La matriz no necesita ser simétrica. Si hay una arista del nodo i al nodo j, entonces $A[i][j] = 1$.
- Grafos Ponderados: En lugar de 1, la entrada $A[i][j]$ puede contener el peso de la arista que conecta los nodos i y j. Si no hay arista, $A[i][j] = 0$ o un valor que indique la ausencia de arista (e.g., ∞ para grafos ponderados).

Operaciones en una Matriz de Adyacencia

1. Verificación de Existencia de Arista:

- Para verificar si existe una arista entre los nodos i y j, se revisa el valor de $A[i][j]$.

2. Inserción de Arista:

- Para añadir una arista entre los nodos i y j, se asigna $A[i][j] = 1$ (o el peso de la arista en grafos ponderados).

3. Eliminación de Arista:

- Para eliminar una arista entre los nodos i y j, se asigna $A[i][j] = 0$ (o un valor que indique la ausencia de arista).

4. Recorridos:

- Se pueden realizar recorridos en el grafo (como DFS o BFS) utilizando la matriz de adyacencia para verificar conexiones entre nodos.

- Matriz de adyacencia

Una matriz dispersa es una matriz grande en la que un alto porcentaje de sus elementos son cero. Por ejemplo, una matriz de 1000 x1000 puede tener solo unos pocos cientos de elementos no cero.

Operaciones en Matrices Dispersas

1. Acceso a Elementos:
   - Verificar la existencia de un elemento en una posición dada puede requerir la búsqueda en las estructuras de datos utilizadas (e.g., búsqueda binaria en CSR/CSC).
2. Inserción de Elementos:
   - Añadir elementos no cero implica actualizar las estructuras de datos, lo que puede ser costoso dependiendo del formato utilizado.
3. Suma y Multiplicación:
   - Algoritmos especializados permiten realizar estas operaciones de manera eficiente, aprovechando la dispersión de los elementos.
4. Transposición:
   - La transposición de una matriz dispersa se puede realizar eficientemente intercambiando las estructuras de filas y columnas en CSR/CSC.