Interface Set.

Características	HashSet	LinkedHashSet	TreeSet
Permite duplicados	No	No	No
Ordena los elementos	No	Orden de inserción	Orden alfabetico
Complejidad	O(1)	O(1)	O(log(n))
Permite (null)	Si, 1 valor	Si, 1 valor	No
Estructura	Tabla Hash	Tabla Hash + Lista enlazada	Arbol Rojo - Negro

Tabla Hash

Una tabla hash es una estructura de datos que asocia claves (keys) con valores (values) mediante una función de hash. Su objetivo es permitir búsquedas, inserciones y eliminaciones rápidas en tiempo promedio O(1).

Características Principales

- Almacenamiento basado en clave-valor.
- Uso de una función hash para calcular la posición de almacenamiento.
- Eficiencia alta en búsquedas, inserciones y eliminaciones (O(1) en promedio).
- Puede manejar colisiones mediante técnicas como encadenamiento o direccionamiento abierto.

Cómo Funciona una Tabla Hash

- Función Hash: Convierte una clave (key) en un índice de la tabla.
- Inserción: El valor se almacena en la posición dada por la función hash.
- Búsqueda: Se usa la clave para calcular la posición y acceder rápidamente al valor.
- Colisiones: Cuando dos claves generan el mismo índice, se maneja con estrategias como listas enlazadas.

Manejo de Colisiones

Cuando dos claves producen el mismo índice en la tabla hash, se usa una estrategia para resolverlo.

- Encadenamiento (Chaining): Se usa una lista enlazada en cada celda para almacenar múltiples valores.
- Direccionamiento Abierto (Open Addressing): Se buscan otras posiciones libres dentro de la tabla.

Nota: En Java, HashMap usa encadenamiento para manejar colisiones.

¿Cuándo Usar una Tabla Hash?

- Cuando necesitas búsquedas rápidas por clave.
- Cuando no importa el orden de los elementos.
- Cuando el número de elementos es grande y necesitas eficiencia.

Lista Enlazada

Una lista enlazada (Linked List) es una estructura de datos lineal y dinámica donde los elementos (nodos) están conectados mediante punteros o referencias en lugar de ocupar posiciones contiguas en memoria. Cada nodo contiene:

- Un dato (valor almacenado).
- Una referencia al siguiente nodo en la lista.

Tipos de Listas Enlazadas

- Lista Enlazada Simple (SLL): Cada nodo apunta al siguiente. Se recorre en una sola dirección.
- Lista Doblemente Enlazada (DLL): Cada nodo apunta al siguiente y al anterior. Se puede recorrer en ambas direcciones.
- Lista Circular
  - En la versión simple, el último nodo apunta al primero.
  - En la versión doble, el primer y el último nodo están conectados en ambos sentidos.

Arbol Rojo-Negro

Características	BST	Arbol Rojo - Negro
Orden	I<R<D	I<R<D
Balanceo automático	No	Si
Complejidad	O(N) peor caso (Totalmente desbalanceado)	O(log(n))
Coloración nodos	No usa	Nodo, Rojo -Negro
Uso en java	TreeSet o TreeMap. No.	TreeSet o TreeMap, Si.

¿Qué es un Árbol Rojo-Negro?

Es un árbol binario de búsqueda (BST) balanceado donde cada nodo tiene un color rojo o negro y sigue estas reglas:

- Cada nodo es rojo o negro.
- La raíz siempre es negra.
- Un nodo rojo no puede tener un hijo rojo (no hay nodos rojos consecutivos).
- Cada camino desde la raíz hasta una hoja tiene el mismo número de nodos negros (propiedad de balanceo).
- Si un nodo es rojo, sus hijos deben ser negros.
- Un nodo negro puede tener hijos negros.

Estas reglas garantizan que el árbol esté siempre balanceado, manteniendo la altura en O(log N), lo que lo hace más eficiente que un BST simple.

¿Cuándo usar un Árbol Rojo-Negro?

- Cuando necesitas búsquedas, inserciones y eliminaciones rápidas (O(log N)).
- Cuando el orden es importante (como en TreeSet y TreeMap).
- Cuando quieres evitar que un BST se desbalancee.

Conclusión

Permitir que un nodo negro tenga hijos negros ayuda a:

- Mantener la propiedad de balanceo del árbol rojo-negro.
- Evitar que la altura crezca demasiado, garantizando O(log N).
- Reducir la necesidad de demasiadas rotaciones y recoloreos.

En resumen, un nodo negro puede tener hijos negros porque es una estrategia clave para el balanceo del árbol y su eficiencia.