invariantes de representación

1. *Conjunto* :

- No debe haber elementos duplicados en el conjunto.

2. *Diccionario* :

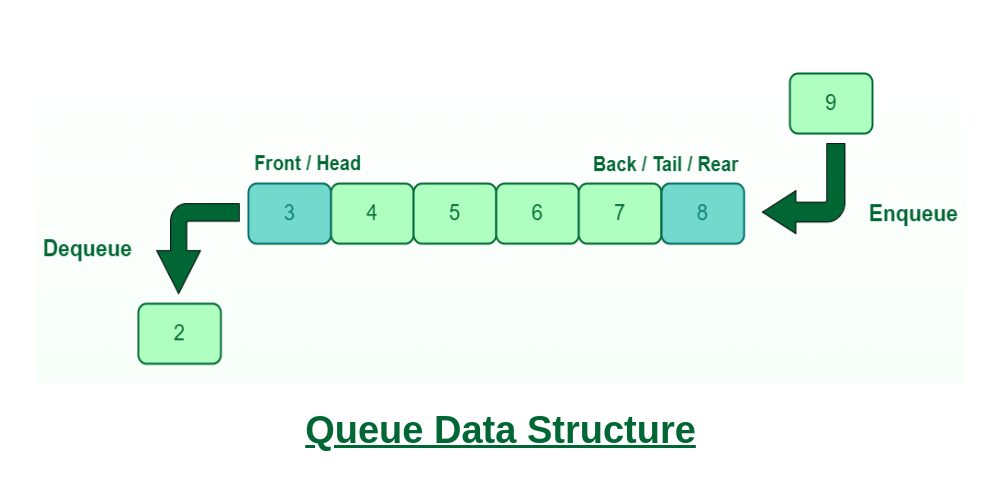
- No debe haber claves duplicadas en el diccionario.

- Cada clave debe estar asociada con un único valor.

3. *Cola* :

- Los elementos en la cola deben estar ordenados según el orden de inserción (FIFO).

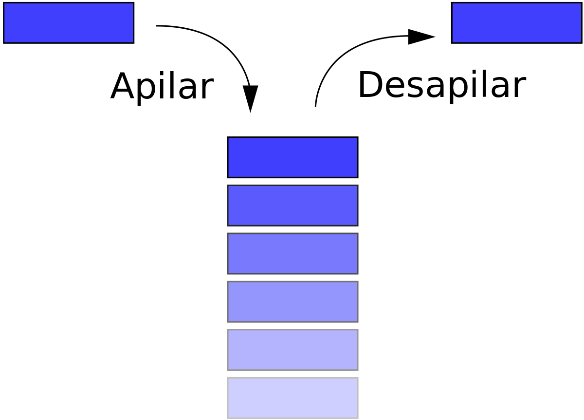
- Si la cola está vacía, no debe contener ningún elemento.



4. *Pila* :

- Los elementos en la pila deben estar ordenados según el orden de inserción y eliminación (LIFO).

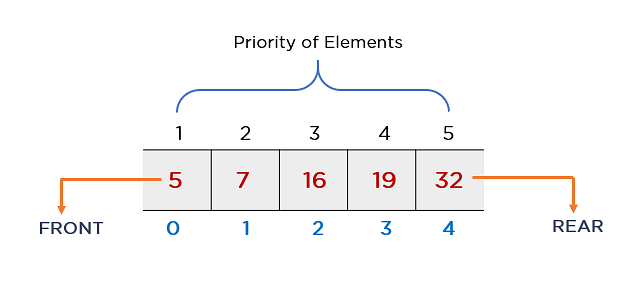
- Si la pila está vacía, no debe contener ningún elemento.



5. *Cola de Prioridad* :

- Los elementos en la cola de prioridad deben estar organizados según su prioridad.

- Si la cola de prioridad está vacía, no debe contener ningún elemento.



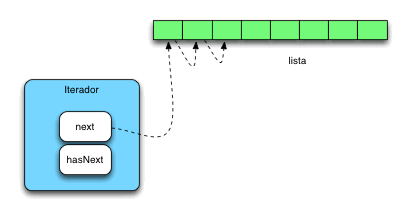
6. *Secuencia* :

- La secuencia debe permitir la inserción, eliminación y acceso eficiente a los elementos en cualquier posición.

7. *Iterador* :

- Debe haber un único iterador activo por estructura de datos a la vez.

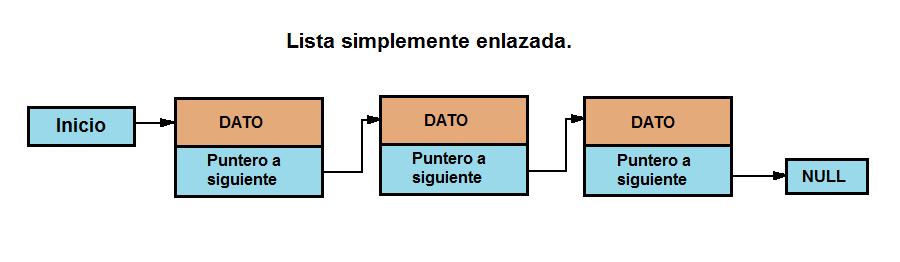
- El iterador debe mantener su posición dentro de la estructura de datos y permitir avanzar y retroceder de manera segura.



8. *Lista* *Enlazada* :

- Cada nodo de la lista enlazada debe tener un puntero válido al siguiente nodo (o a null si es el último nodo).

- No debe haber ciclos en la lista enlazada.

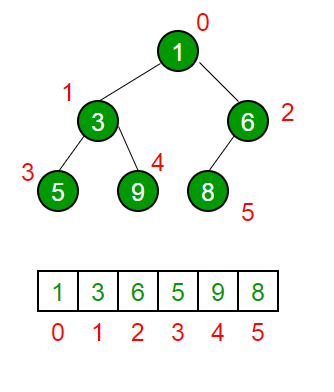


9. *Heap Binario* :

- Para un heap binario mínimo, para cada nodo padre, el valor en el nodo es menor o igual que los valores en sus hijos.

- Para un heap binario máximo, para cada nodo padre, el valor en el nodo es mayor o igual que los valores en sus hijos.

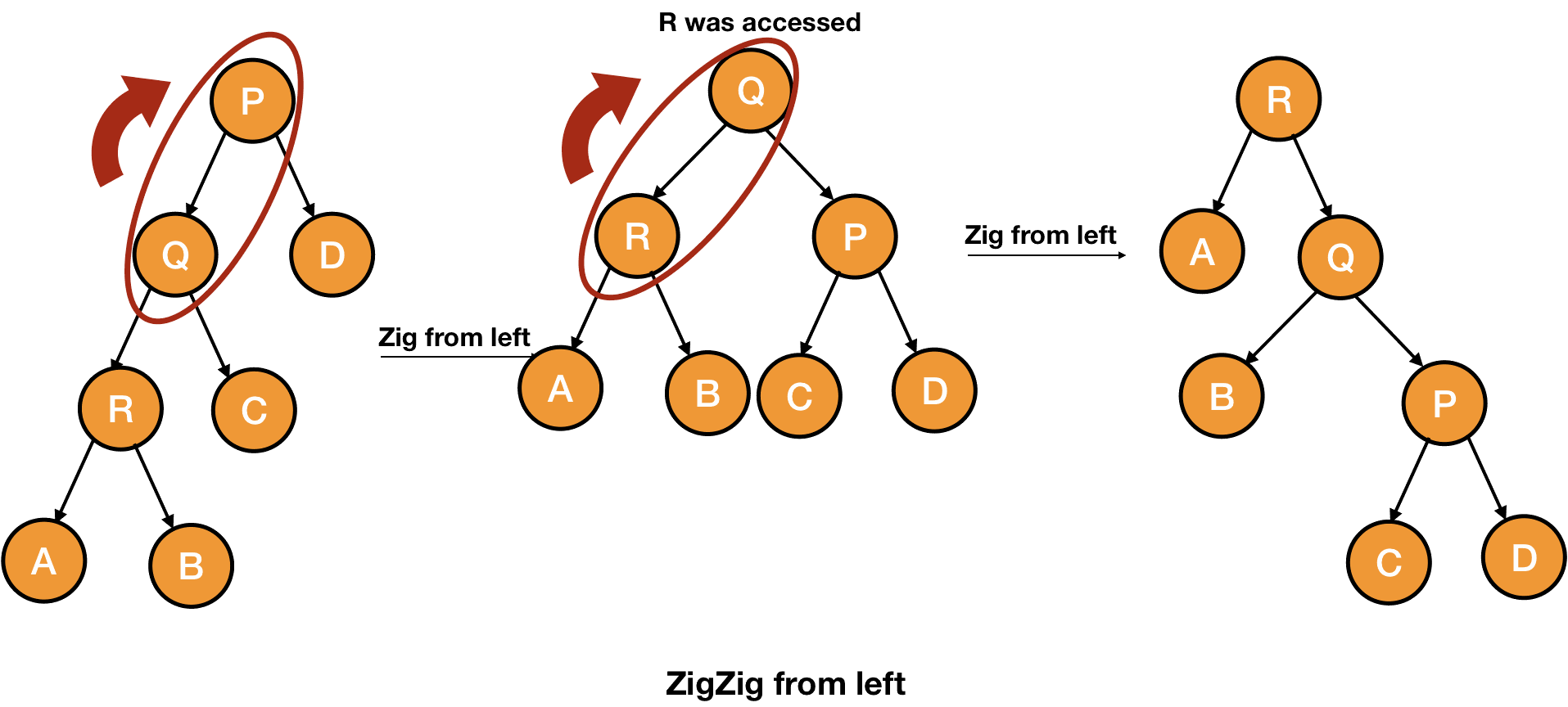
- El heap binario debe ser un árbol completo, lo que significa que todos los niveles excepto posiblemente el último están completamente llenos y, si el último nivel está incompleto, los nodos se llenan de izquierda a derecha.



10. *Splay Tree* :

- Cada nodo debe estar correctamente relacionado con sus padres y posiblemente hijos según las rotaciones realizadas.

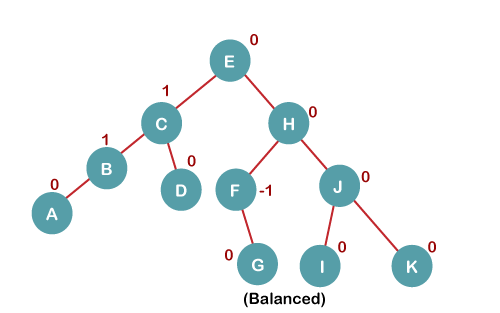
- Después de una operación de búsqueda o modificación, el nodo accedido más recientemente debe estar en la raíz del árbol.



11*. AVL (Árbol AVL)* :

- El factor de equilibrio de cada nodo (la diferencia entre las alturas de los subárboles izquierdo y derecho) debe estar en el rango {-1, 0, 1}.

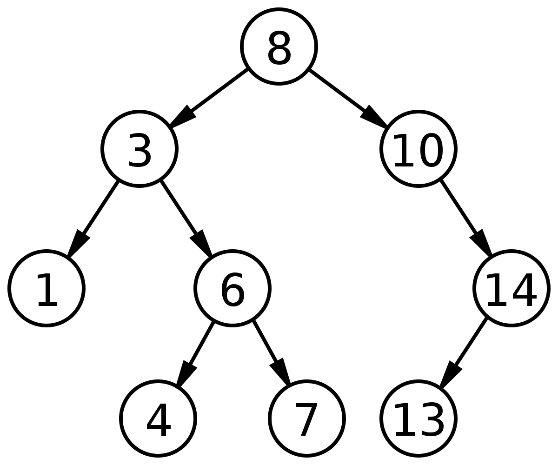
- El árbol debe ser un árbol binario de búsqueda, es decir, para cada nodo, todos los nodos en el subárbol izquierdo deben tener un valor menor, y todos los nodos en el subárbol derecho deben tener un valor mayor.



12. *ABB (Árbol Binario de Búsqueda)* :

- El árbol debe ser un árbol binario de búsqueda, es decir, para cada nodo, todos los nodos en el subárbol izquierdo deben tener un valor menor, y todos los nodos en el subárbol derecho deben tener un valor mayor.

- No debe haber nodos duplicados en el árbol.

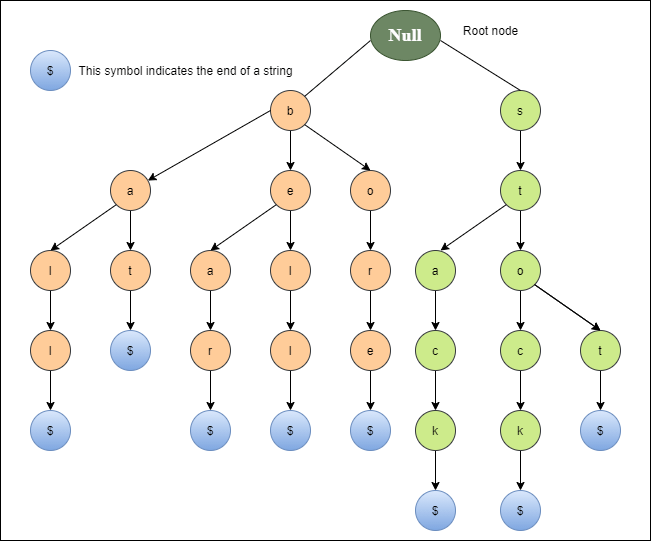


13. *Trie* :

- Cada nodo en el Trie debe representar un carácter o símbolo válido.

- Los nodos del Trie deben estar interconectados de manera que representen palabras completas.

- Los nodos intermedios en el Trie no deben representar palabras completas, solo los nodos hoja deben representar palabras completas.

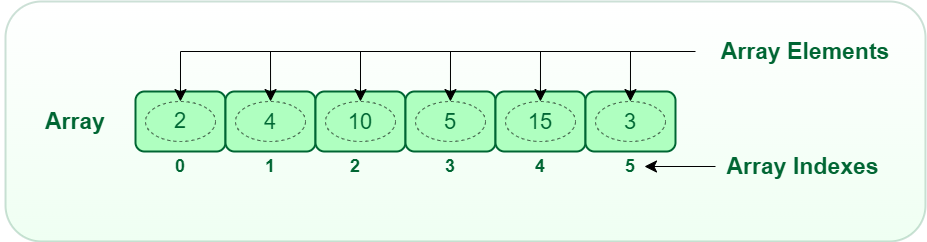


14. *Array* :

- El array debe tener un tamaño fijo o dinámico, según la implementación.

- Los elementos en el array deben estar indexados de manera consecutiva y comenzar desde 0 o desde el índice especificado.

- El array debe permitir la inserción, eliminación y acceso eficiente a los elementos en cualquier posición.



definiciones

1. *Conjunto* :

- Un conjunto es una colección de elementos únicos en la que no se permite la repetición de elementos. Es una estructura de datos que modela la idea matemática de un conjunto, donde los elementos no tienen un orden definido y se pueden agregar, eliminar y buscar elementos de manera eficiente.

2. *Diccionario* :

- Un diccionario es una estructura de datos que asocia claves con valores. Cada clave debe ser única y está vinculada a un único valor. Permite recuperar rápidamente el valor asociado con una clave dada y puede utilizarse para representar relaciones de mapeo entre diferentes conjuntos de datos.

3. Cola :

- Una cola es una estructura de datos que sigue el principio de "primero en entrar, primero en salir" (FIFO). Los elementos se agregan al final de la cola y se eliminan del principio de la cola. Es útil para modelar procesos secuenciales, como la gestión de tareas en un sistema informático.

4. Pila :

- Una pila es una estructura de datos que sigue el principio de "último en entrar, primero en salir" (LIFO). Los elementos se agregan y eliminan desde el mismo extremo de la pila. Es útil para modelar procesos de reversión, como la gestión de llamadas a funciones en la memoria de un programa.

5. Cola de Prioridad :

- Una cola de prioridad es una estructura de datos en la que cada elemento tiene asignada una prioridad y los elementos se eliminan de la cola según su prioridad relativa. Permite acceder eficientemente al elemento con la prioridad más alta en cualquier momento.

6. Secuencia :

- Una secuencia es una colección ordenada de elementos en la que los elementos pueden repetirse y se pueden acceder mediante su posición en la secuencia. Permite la inserción, eliminación y acceso eficiente a los elementos en cualquier posición.

7. Iterador :

- Un iterador es un objeto que proporciona un mecanismo para recorrer los elementos de una estructura de datos de manera secuencial. Permite acceder a los elementos de la estructura uno por uno y avanzar o retroceder según sea necesario.

8. Lista Enlazada :

- Una lista enlazada es una estructura de datos en la que cada elemento (nodo) contiene un valor y una referencia al siguiente elemento en la secuencia. Permite la inserción, eliminación y acceso eficiente a los elementos en cualquier posición.

9. Heap Binario :

- Un heap binario es un árbol binario completo en el que el valor de cada nodo es mayor o menor que los valores de sus hijos, dependiendo del tipo de heap (máximo o mínimo). Se utiliza comúnmente para implementar colas de prioridad.

10. Splay Tree :

- Un Splay Tree es un tipo de árbol binario de búsqueda en el que los nodos recién accesados se mueven a la raíz del árbol, lo que mejora el tiempo de acceso a los elementos que se acceden con mayor frecuencia.

11. AVL (Árbol AVL) :

- Un Árbol AVL es un árbol binario de búsqueda balanceado en el que la diferencia de altura entre los subárboles izquierdo y derecho de cada nodo (el factor de equilibrio) está restringida a {-1, 0, 1}, lo que garantiza un tiempo de búsqueda eficiente.

12. ABB (Árbol Binario de Búsqueda) :

- Un Árbol Binario de Búsqueda es un árbol binario en el que cada nodo tiene un valor único y los nodos en el subárbol izquierdo tienen valores menores que el nodo raíz, mientras que los nodos en el subárbol derecho tienen valores mayores.

13. Trie :

- Un Trie, también conocido como árbol de prefijos, es una estructura de datos basada en árbol utilizada para almacenar y buscar un conjunto de cadenas de caracteres. Cada nodo en el Trie representa un carácter, y los nodos están organizados de manera que cada ruta desde la raíz hasta un nodo terminal representa una cadena de caracteres válida.

14. Array :

- Un array es una estructura de datos que almacena una colección de elementos de un mismo tipo. Puede tener un tamaño fijo, donde el número de elementos se especifica en el momento de la creación y no cambia durante la vida útil del array, o un tamaño dinámico, donde el tamaño del array puede crecer o disminuir según sea necesario durante la ejecución del programa.

15. Tabla de Hash :

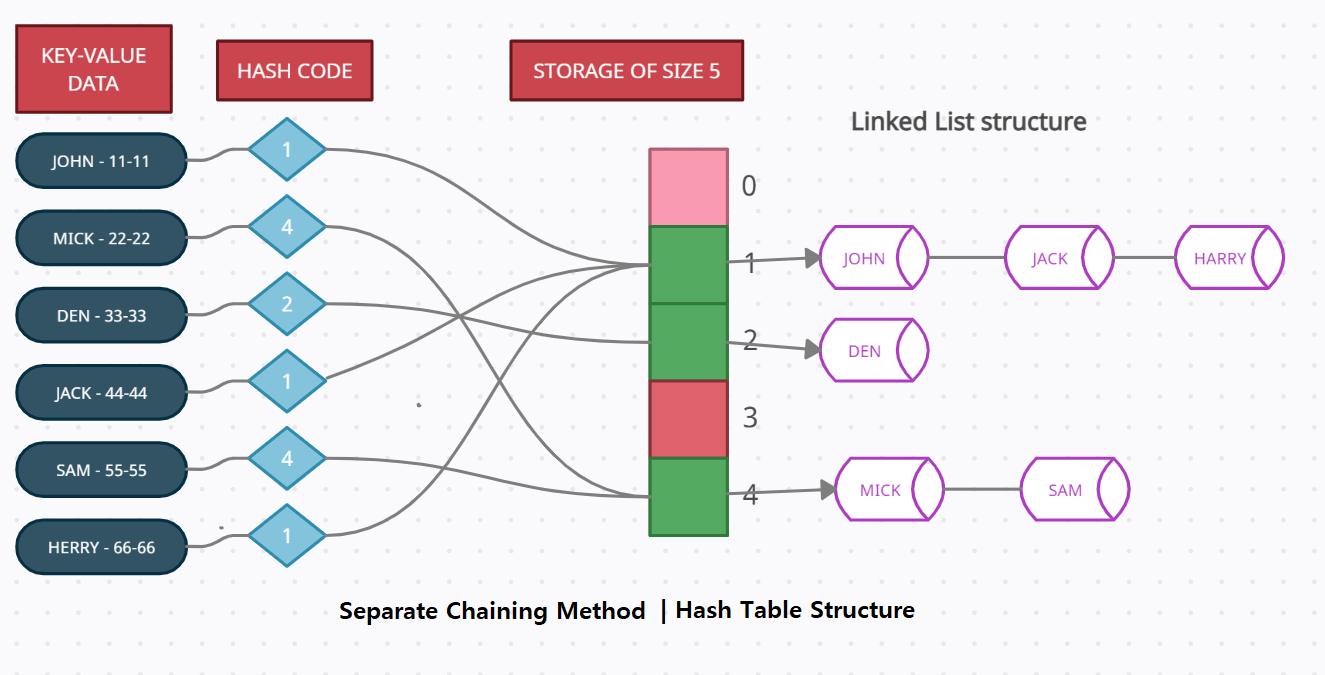
- Cada elemento almacenado en la tabla de hash debe estar asociado con una clave única.

- Los elementos deben estar distribuidos adecuadamente en los "buckets" de la tabla de hash para minimizar las colisiones.

- El tamaño de la tabla de hash debe ser lo suficientemente grande para evitar colisiones excesivas y garantizar un tiempo de acceso eficiente.

- Los elementos deben poder ser insertados, eliminados y buscados correctamente utilizando la función de dispersión y las técnicas de resolución de colisiones apropiadas.

- La función de dispersión debe mapear las claves de manera uniforme y distribuir los elementos de manera equitativa en la tabla de hash.



Direccionamiento Cerrado :

- El direccionamiento cerrado es una técnica de resolución de colisiones en las tablas de hash donde, cuando ocurre una colisión, se busca un nuevo lugar para insertar el elemento dentro de la propia tabla. Esto se hace mediante sondas sucesivas, donde se examinan las ubicaciones de la tabla en secuencia hasta encontrar un espacio vacío. Existen diversas estrategias para realizar estas sondas, como la sondas lineales, cuadráticas o de doble dispersión.

Direccionamiento Abierto :

- El direccionamiento abierto es una técnica de resolución de colisiones en las tablas de hash donde, cuando ocurre una colisión, se busca un nuevo lugar para insertar el elemento fuera de la tabla de hash. Esto se hace utilizando una función adicional de dispersión para calcular una nueva ubicación para el elemento. Al contrario del direccionamiento cerrado, aquí el espacio de almacenamiento no se restringe únicamente a la tabla de hash, sino que se pueden usar otras ubicaciones en memoria.