Arrays

Definition:

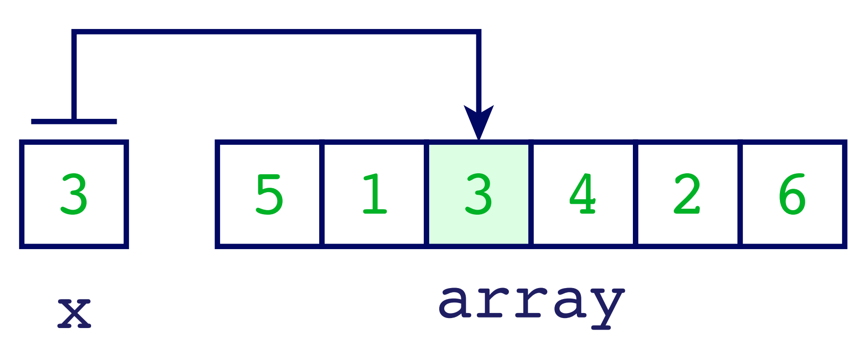
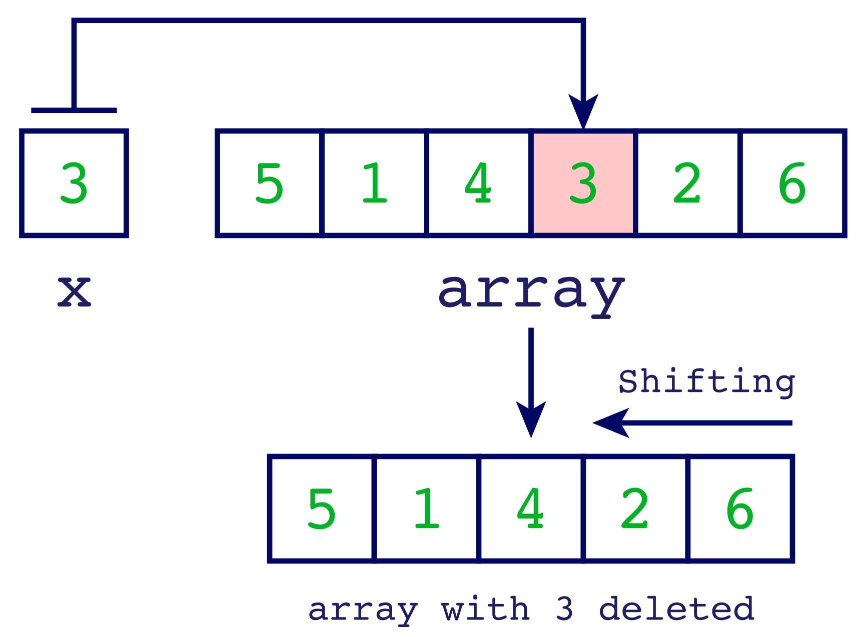
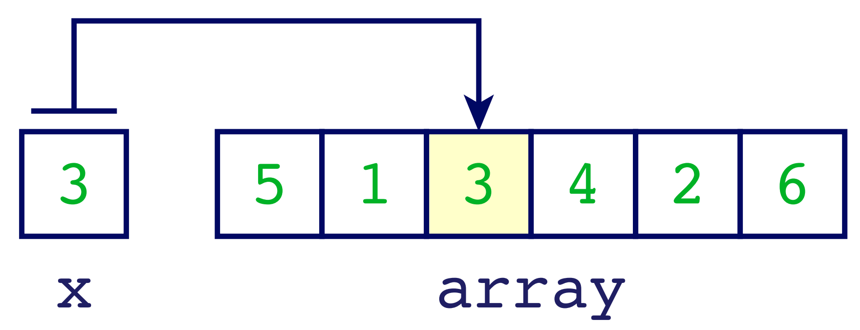
It is a data structure consisting of a collection of elements at contiguous memory locations, each identified by an index.

Elements	5	1	4	2	6
Indexes	0	1	2	3	4

Key concepts:

- **Indexing:** Elements can be accessed using their index
- **Memory allocation:** Contiguous block of memory

Common operations:

Operation	Example	Time Complexity (worst-case)
Insertion		O(n)
Deletion		O(n)
Search		O(n)

Application:

- Dynamic Programming—Store intermediate results to avoid redundant calculations, improving efficiency

Pros:

- **Quick access:** O(1) time complexity for accessing elements by index
- **Predictable memory use:** Fixed size makes memory management easy

Cons:

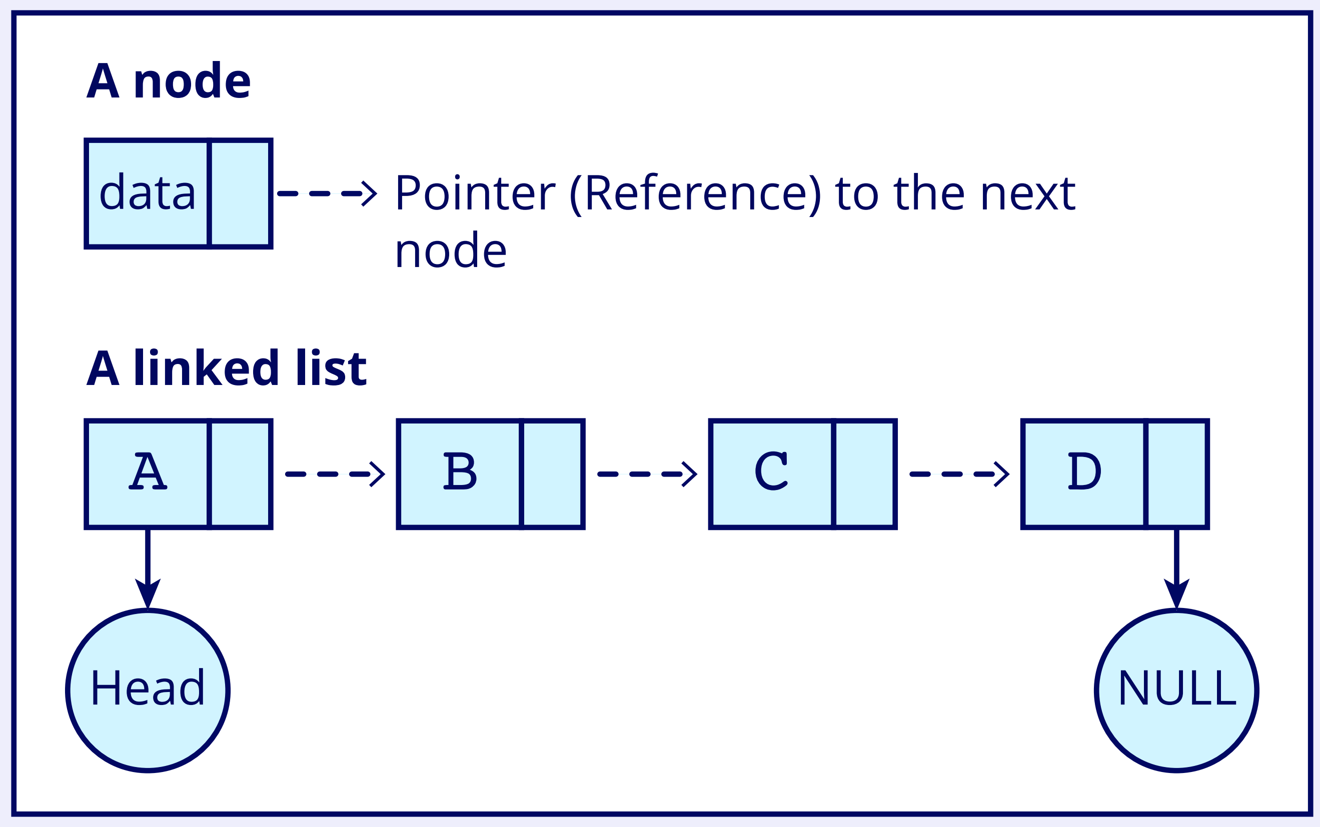
- **Fixed size:** Specifying size at the time of creation leads to potentially wasted or insufficient space
- **Expensive insertion/deletion:** Inserting or deleting elements (except at the end) requires shifting elements, resulting in O(n) time complexity

Linked List (Singly, Doubly)

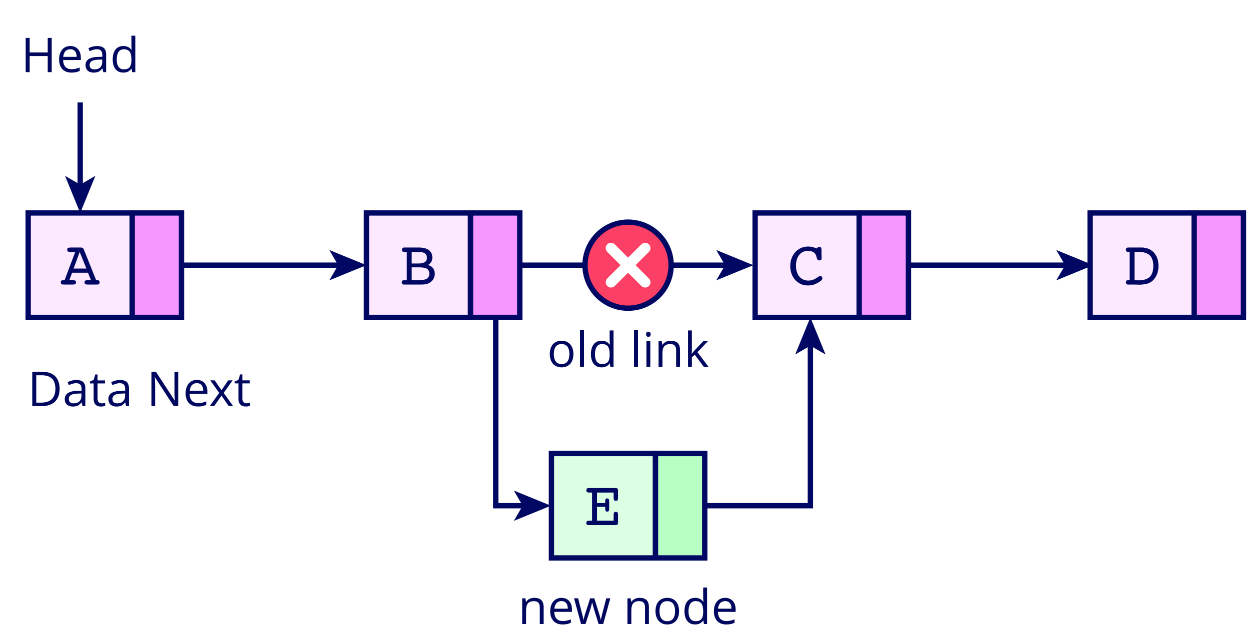
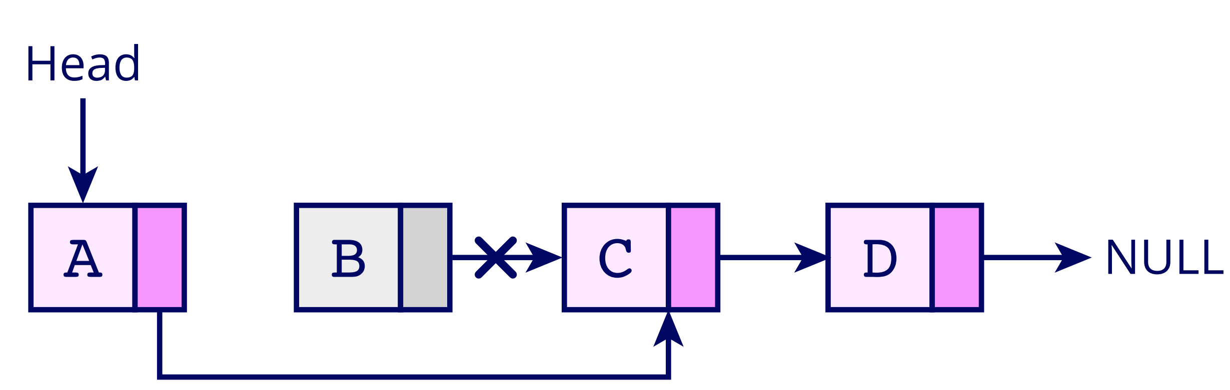
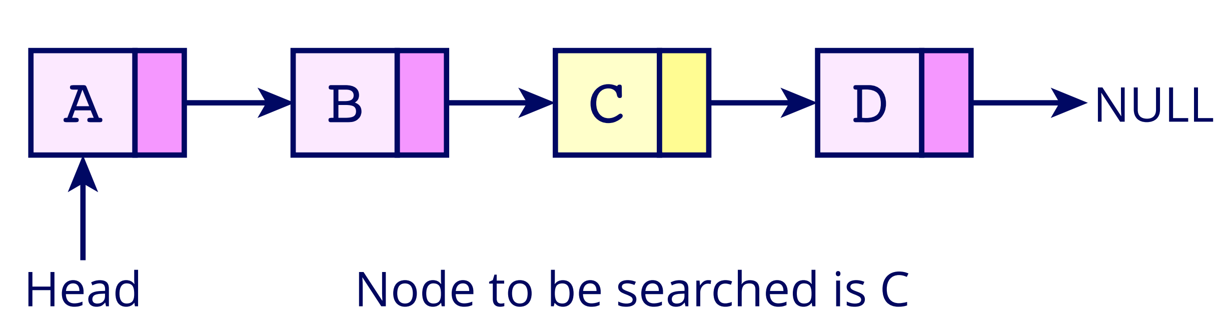
Singly Linked List

Definition:

It's a data structure consisting of nodes containing data and a reference (or link) to the immediate next node.



Common operations:

Operation	Example	Time Complexity (worst-case)
Insertion		O(n)
Deletion		O(n)
Search		O(n)

Traversal:

- **Forward:** Only in the forward direction

Application:

- Implementing stacks and queues

Pros:

- **Dynamic size:** Grow or shrink as needed
- **No contiguous memory requirement:** Does not require a large block of contiguous memory

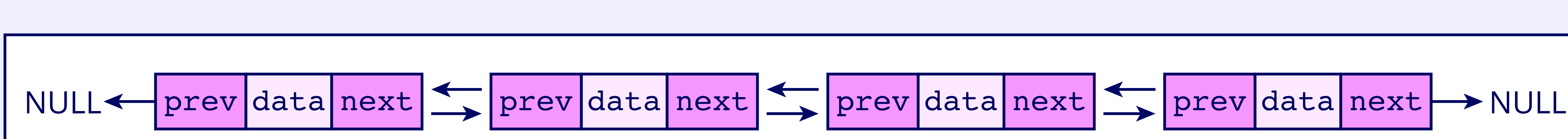
Cons:

- **Extra memory overhead:** Requires additional memory for storing pointers
- **Sequential access:** Traverse nodes sequentially to access an element

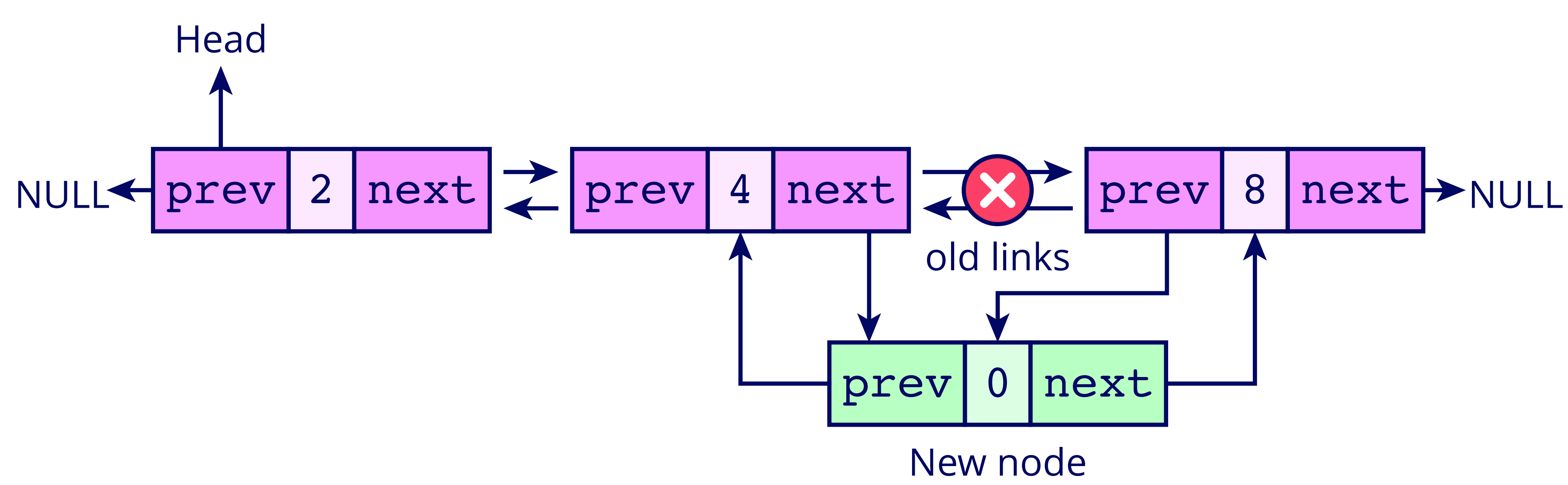
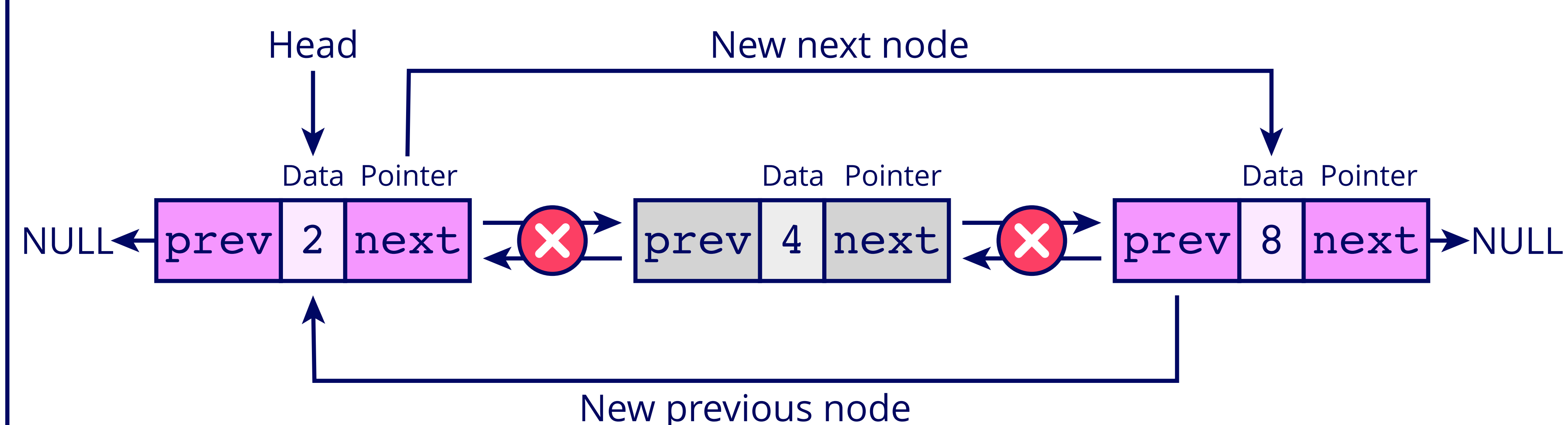
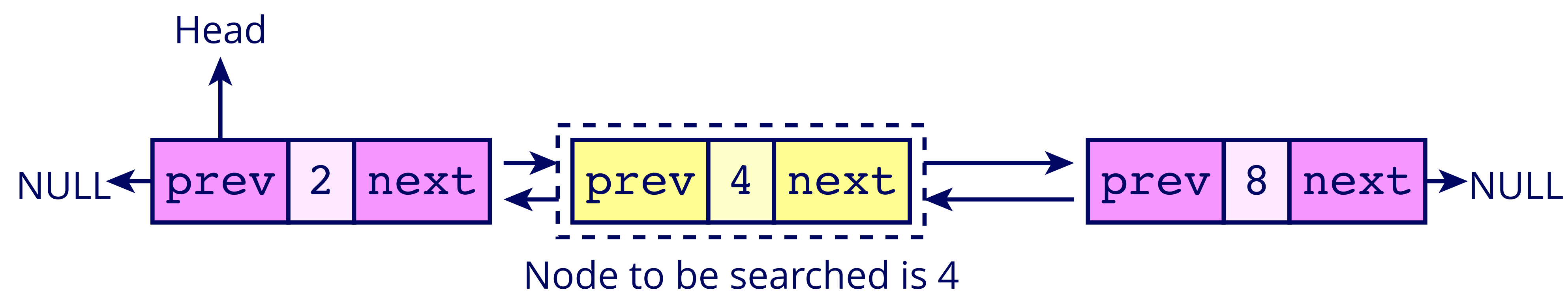
Doubly Linked List

Definition:

It's a data structure consisting of nodes containing data and references (or links) to both the immediate next and previous nodes.



Common operations:

Operation	Example	Time Complexity (worst-case)
Insertion		O(n)
Deletion		O(n)
Search		O(n)

Traversal:

- **Forward and backward:** Traversal is in both directions (forward and backward)

Application:

- Implementing complex data structures like Fibonacci heaps

Pros:

- **Bidirectional traversal:** Traverse the list in both directions
- **Dynamic Size:** Grow or shrink as needed

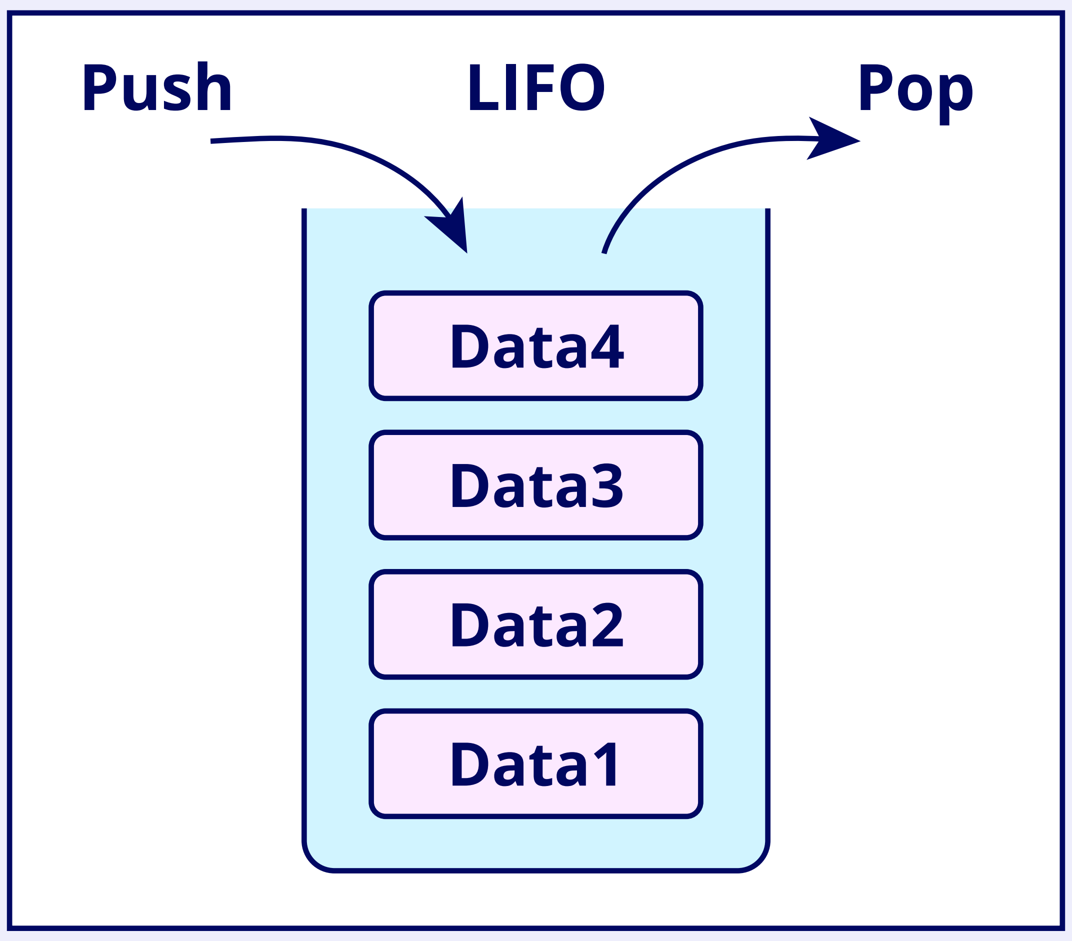
Cons:

- **Extra memory overhead:** Requires additional memory to store two pointers per node
- **Complex implementation:** More complex than singly linked lists

Stacks

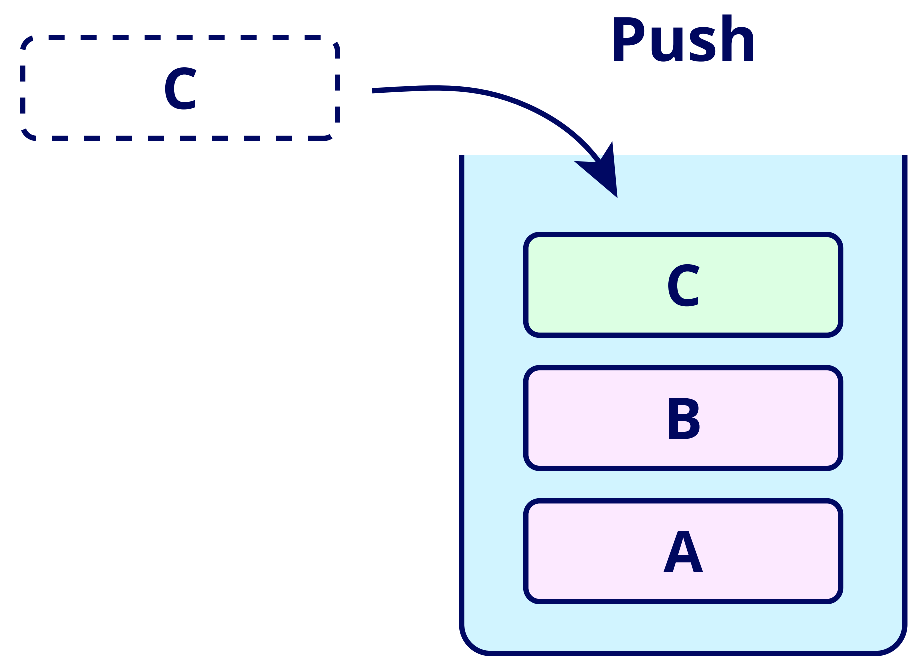
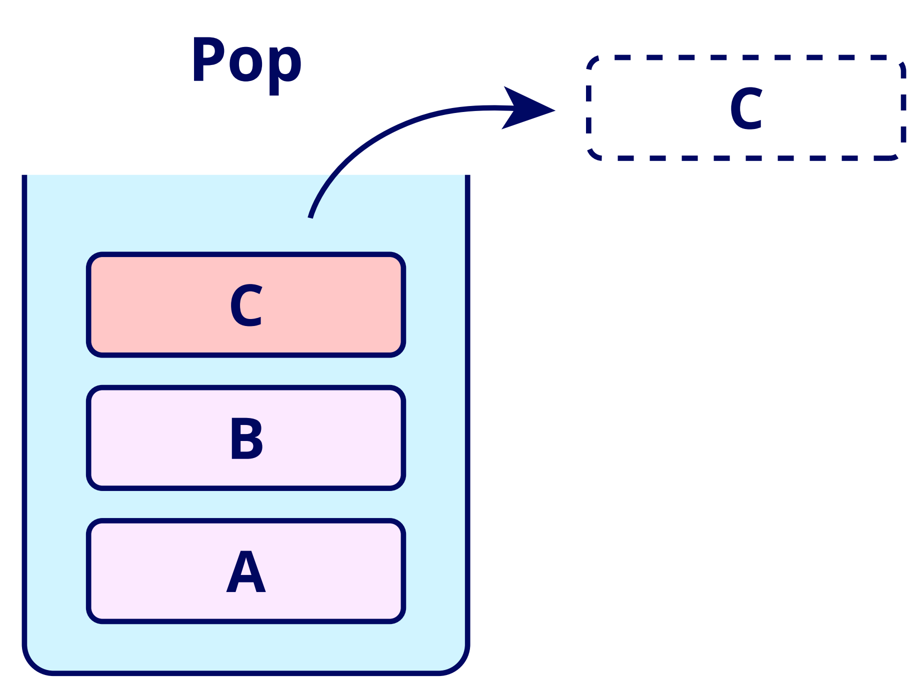
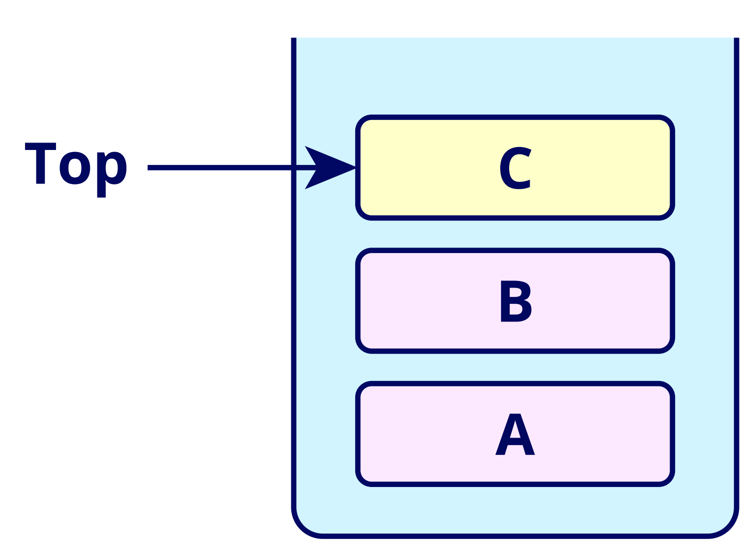
Definition:

It's a collection of elements following the Last In, First Out (LIFO) principle, where the most recently added element is the first to be removed.



Common operations:

- **Push:** O(1) — Adds an element to the top of the stack

Operation	Example	Time Complexity (worst-case)
Push		O(1)
Pop		O(1)
Top		O(1)

Application:

- Expression evaluation and syntax parsing (e.g., converting infix expressions to postfix or evaluating postfix expressions)

Pros:

- **Efficient operations:** O(1) time complexity for push and pop operations

Cons:

- **Sequential access:** Elements are accessed in a LIFO order

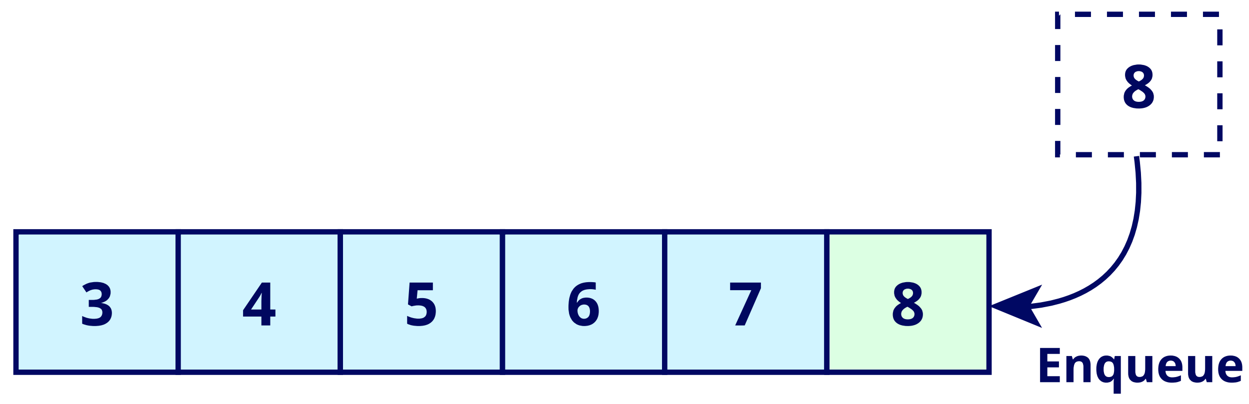
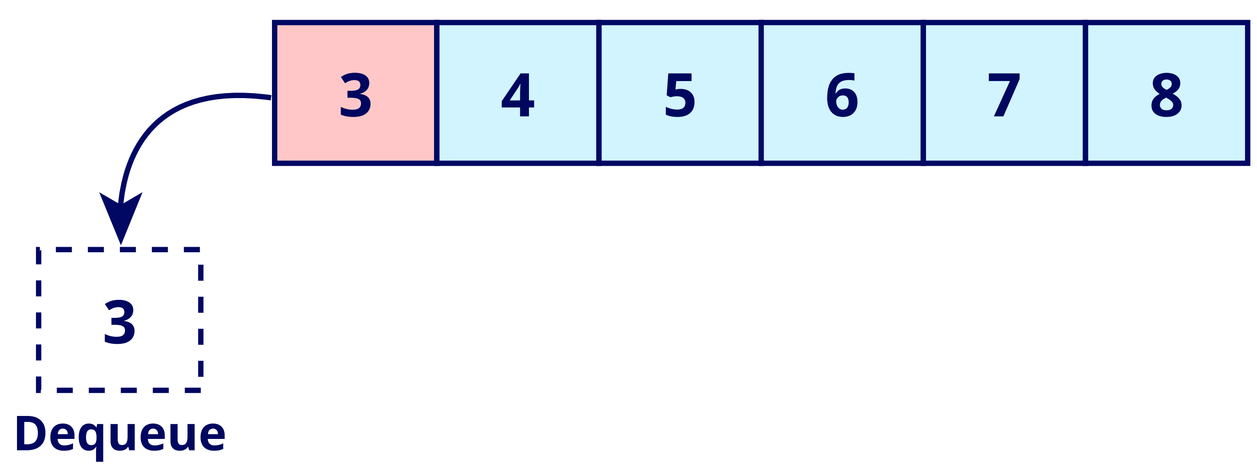
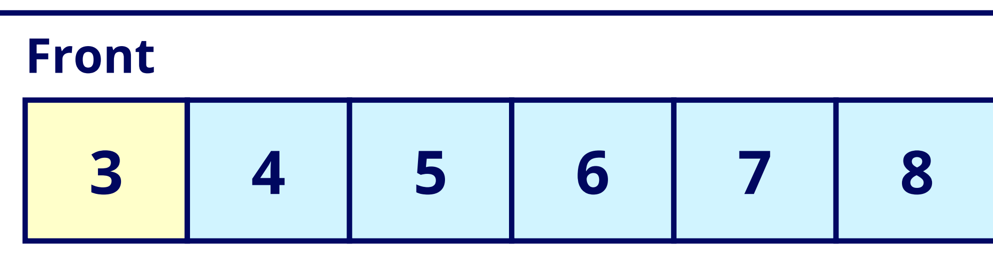
***Note:** It can be implemented using arrays or linked lists.

Queues

Definition:

It's a collection of elements following the First In, First Out (FIFO) principle, where the first element added is the first to be removed.

Common operations:

Operation	Example	Time Complexity (worst-case)
Enqueue		O(1)
Dequeue		O(1)
Front		O(1)

Application:

- Task management (e.g., printer queue)

Pros:

- **Efficient operations:** O(1) time complexity for enqueue and dequeue operations

Cons:

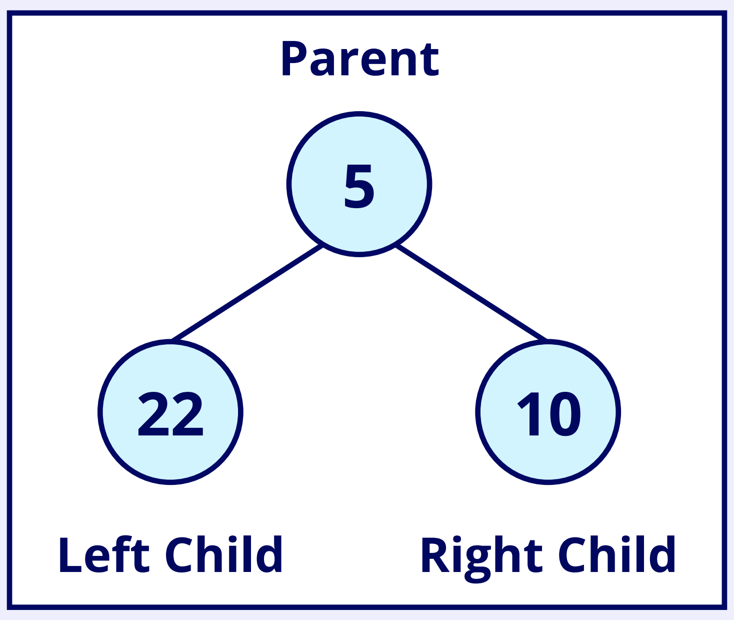
- **Sequential access:** Elements are accessed in a FIFO order

***Note:** It can be implemented using arrays or linked lists.

Binary Trees

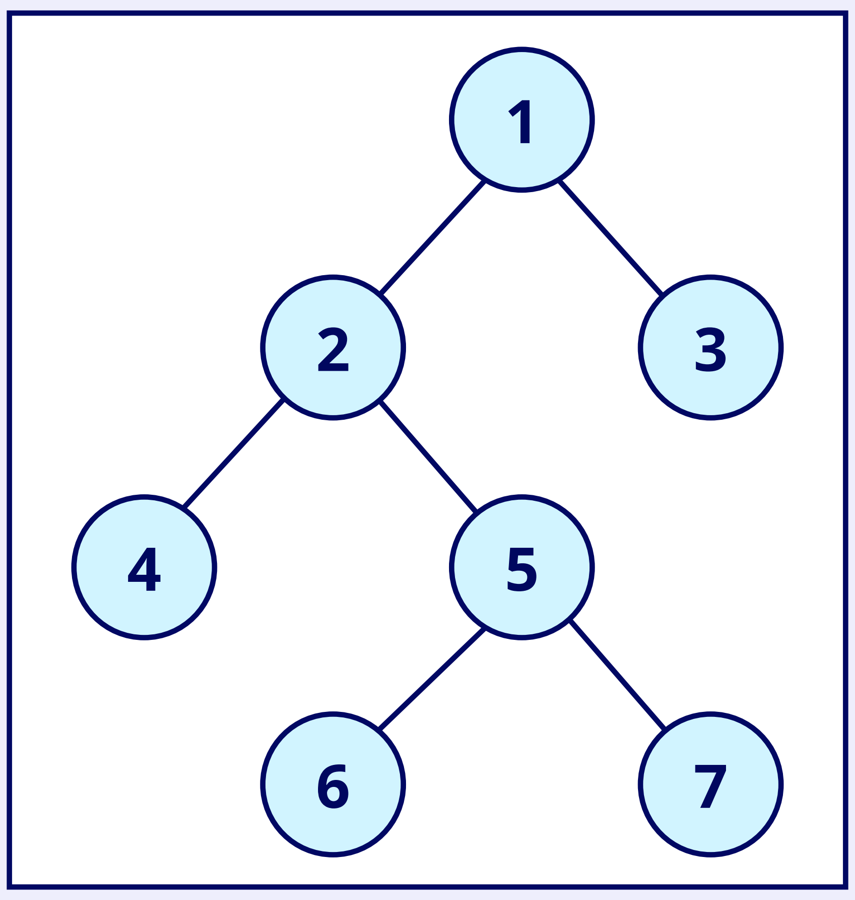
Definition:

It's a data structure in which each node has at most two children, referred to as the left and right children.

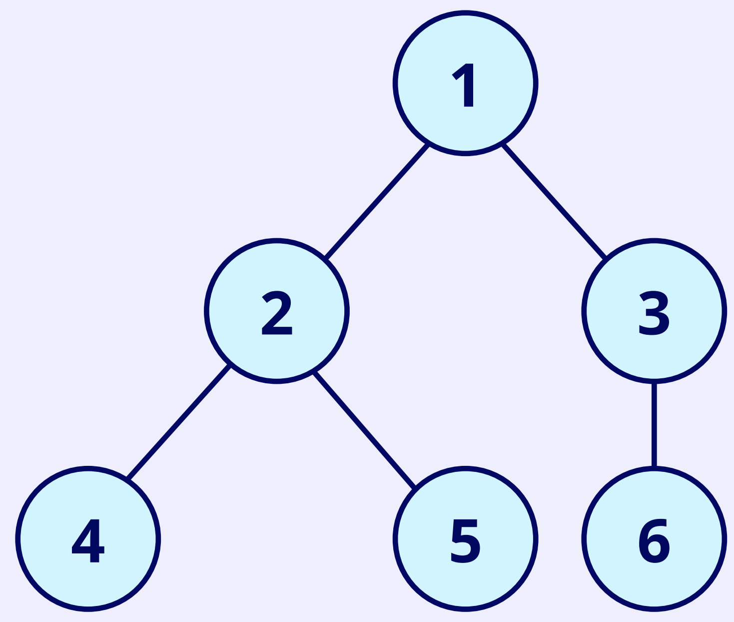


Types:

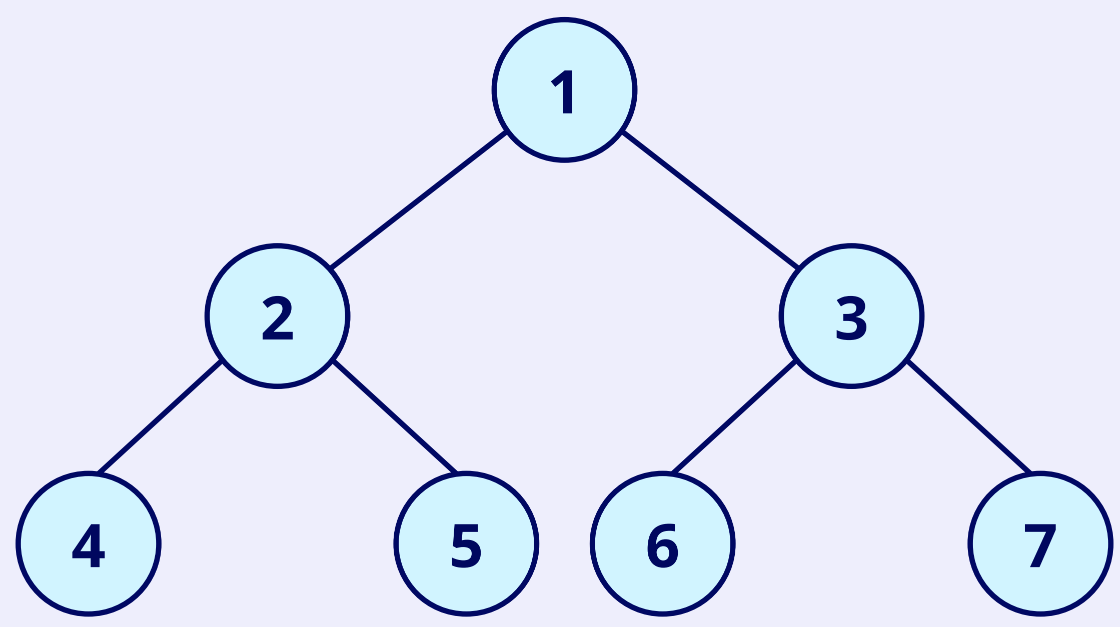
1. **Full binary tree:** Every node has 0 or 2 children.



2. Complete binary tree: All levels are completely filled except possibly the last level, filled from left to right.

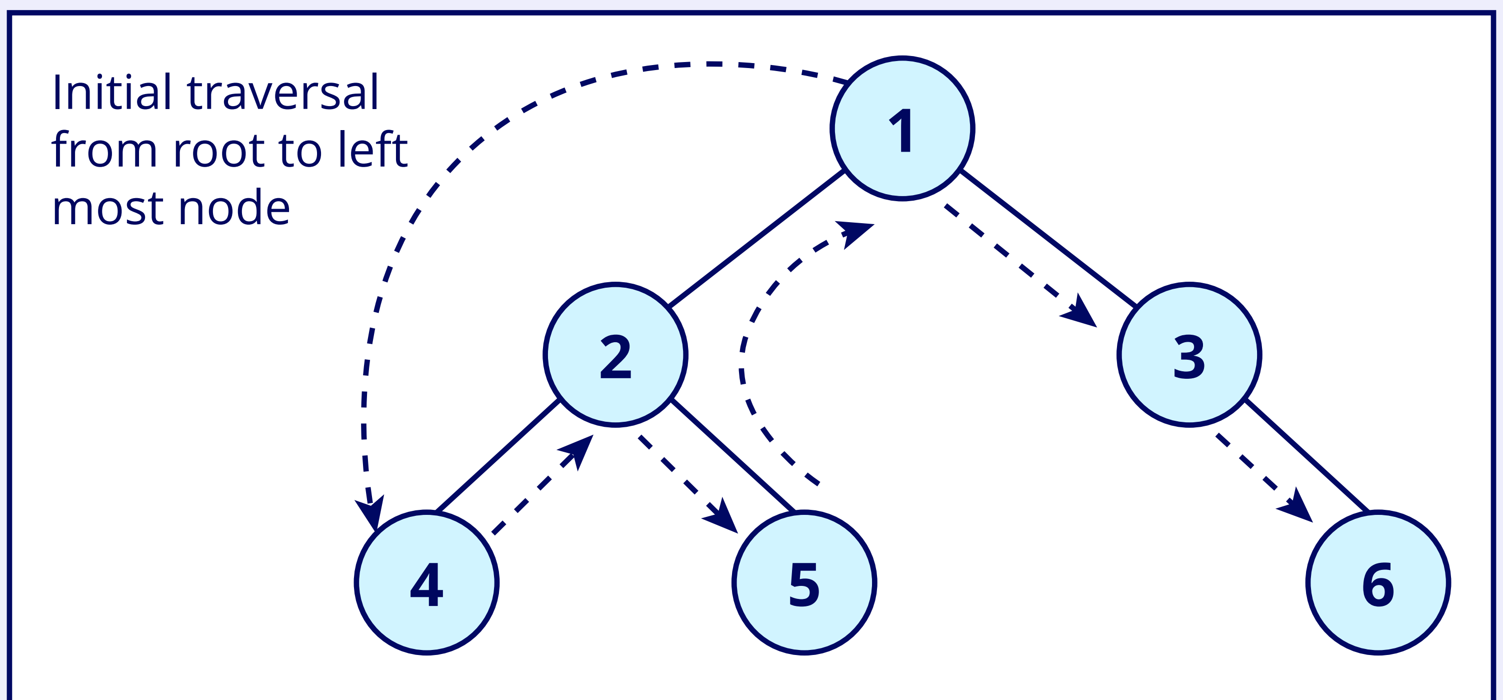


3. Perfect binary tree: All internal nodes have two children, and all leaves are at the same level.



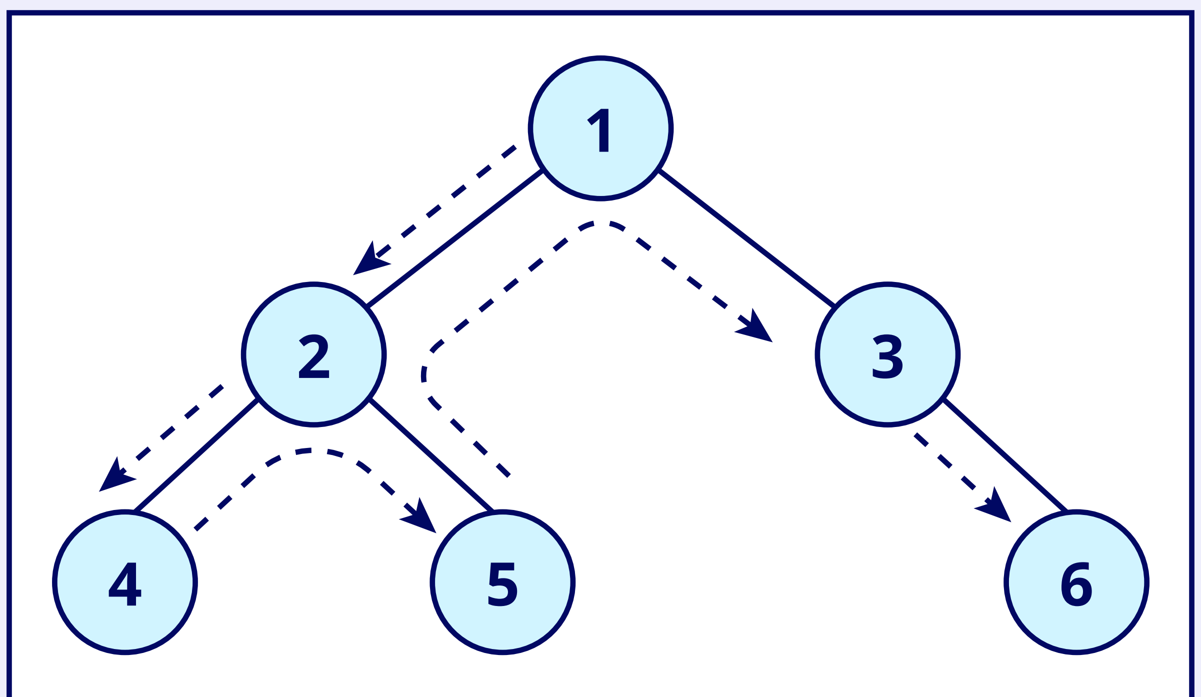
3. Traversals:

- **In-order (LNR):** Visit the left subtree, node, and the right subtree.



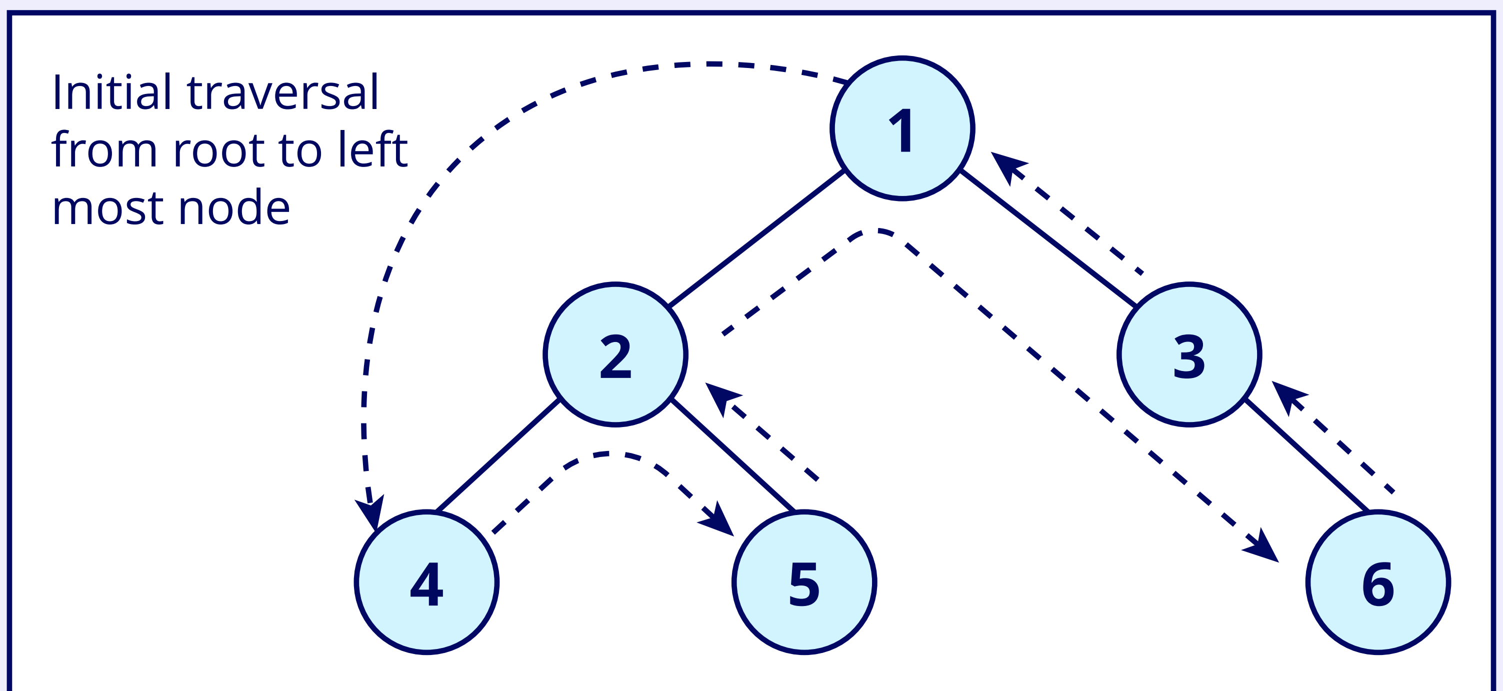
In-order traversal: 4→2→5→1→3→6

- **Pre-order (NLR):** Visit the node, left subtree, and right subtree.



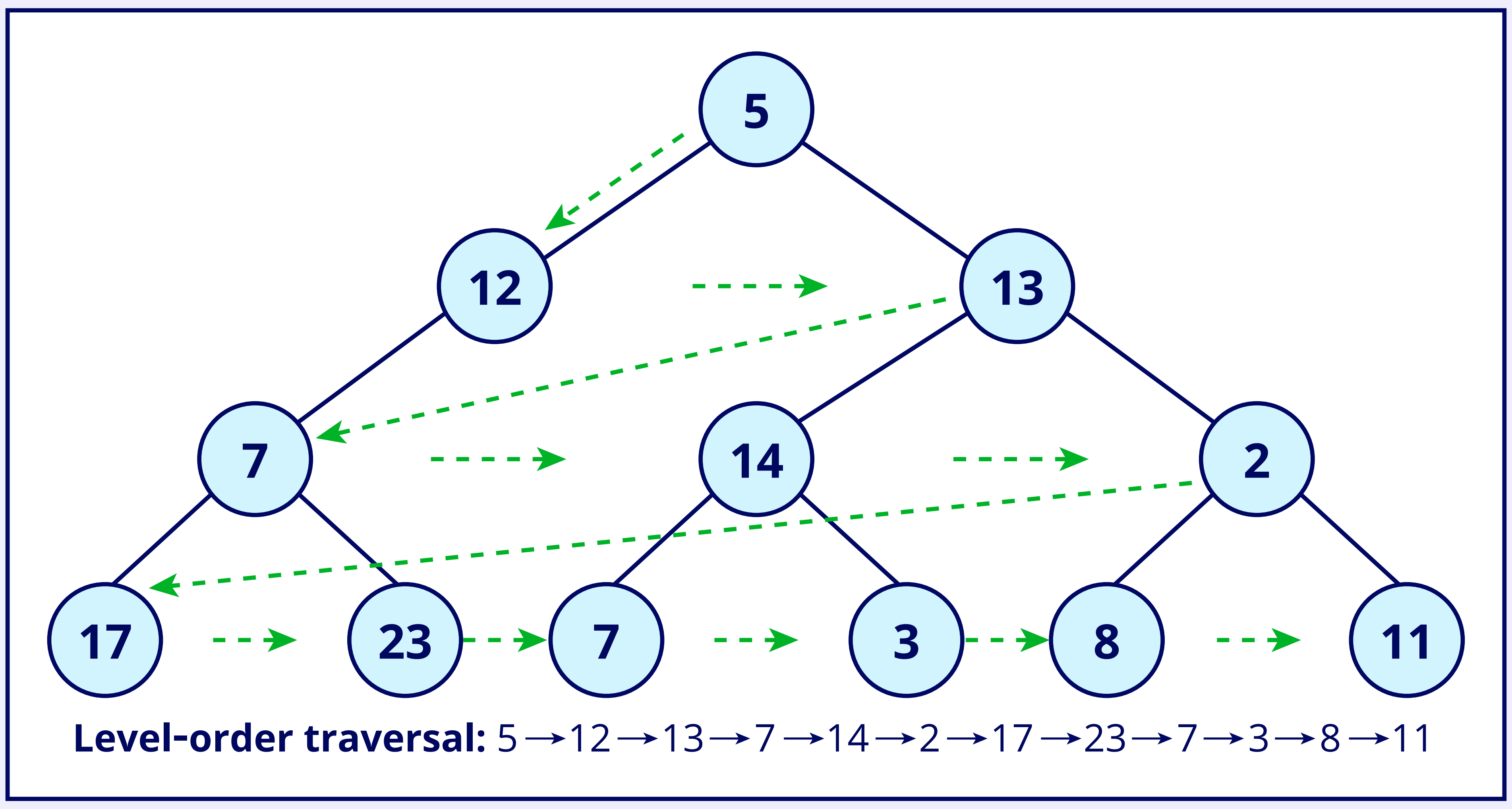
Pre-order traversal: 1→2→4→5→3→6

- **Post-order (LRN):** Visit the left subtree, right subtree, and node.



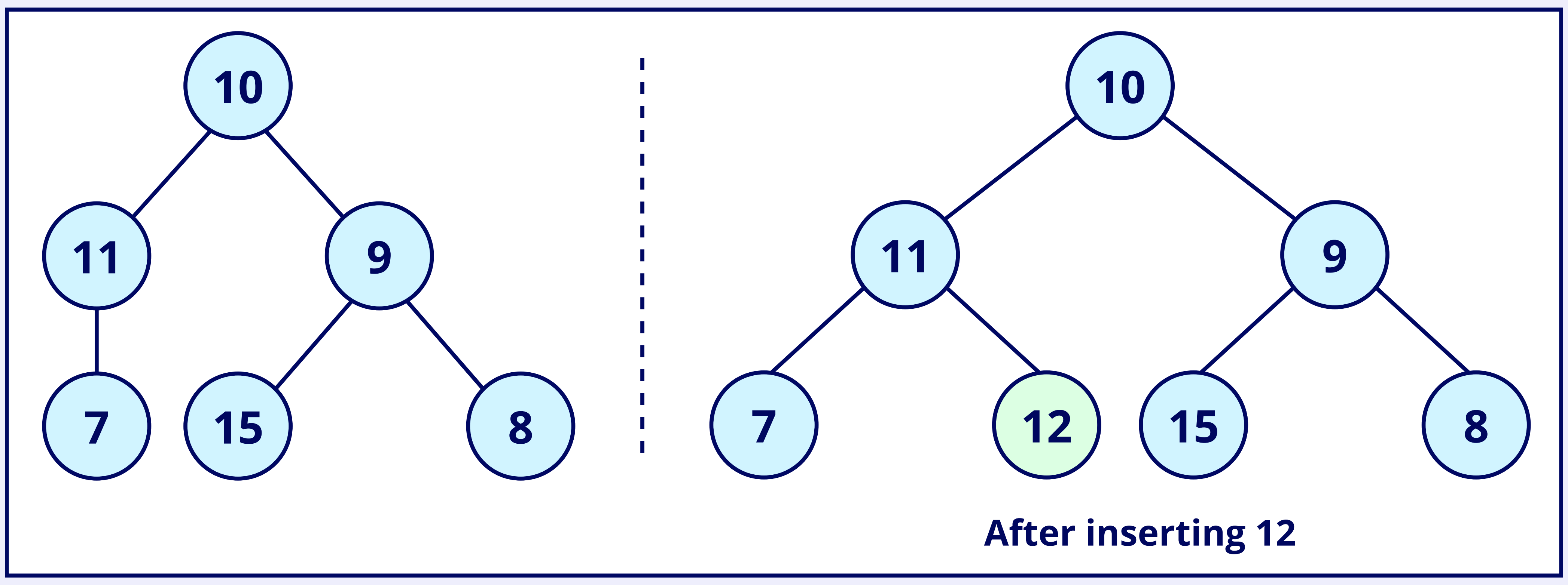
Post-order traversal: 4→5→2→6→3→1

- **Level-order:** Visit nodes level by level.

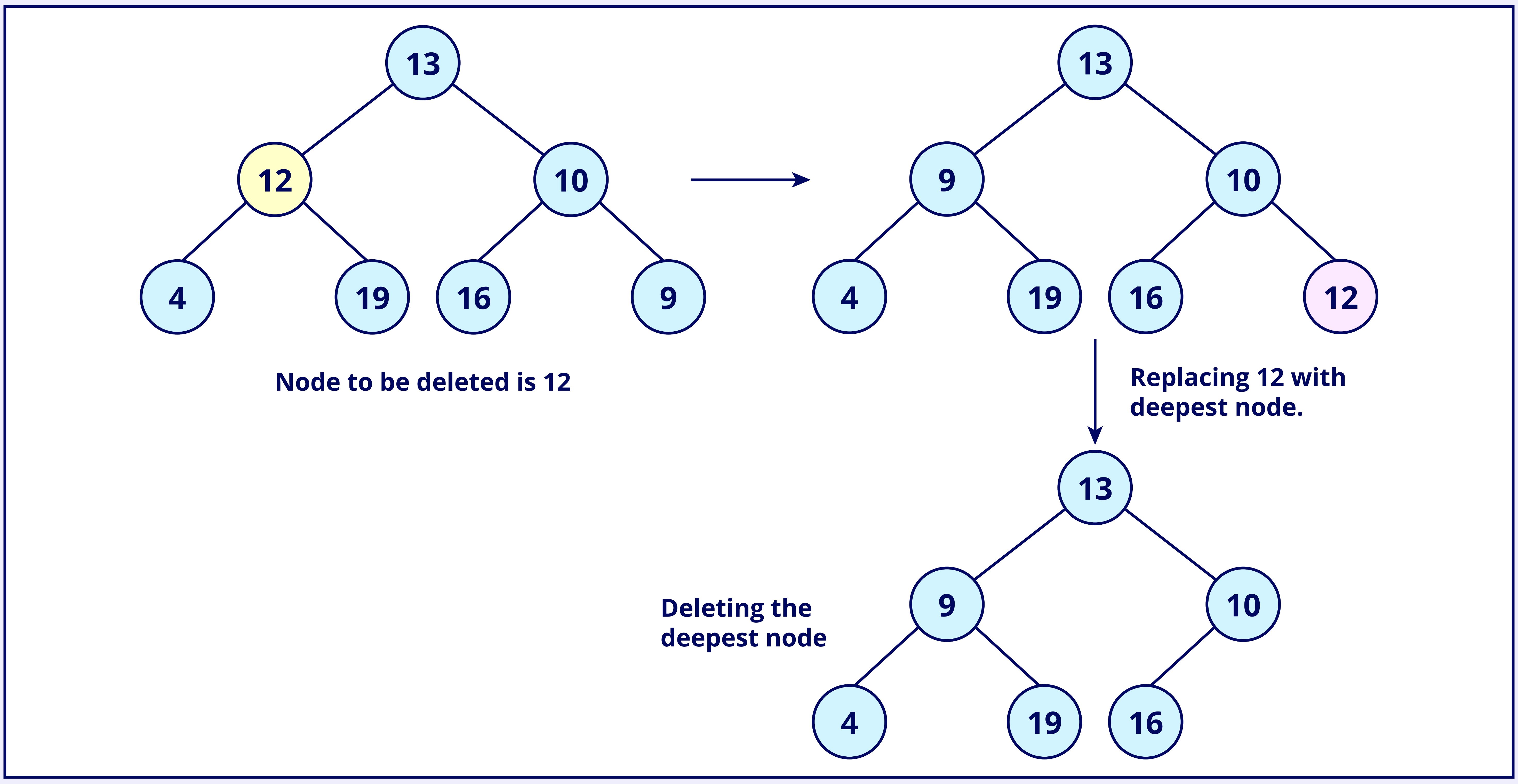


4. Common operations:

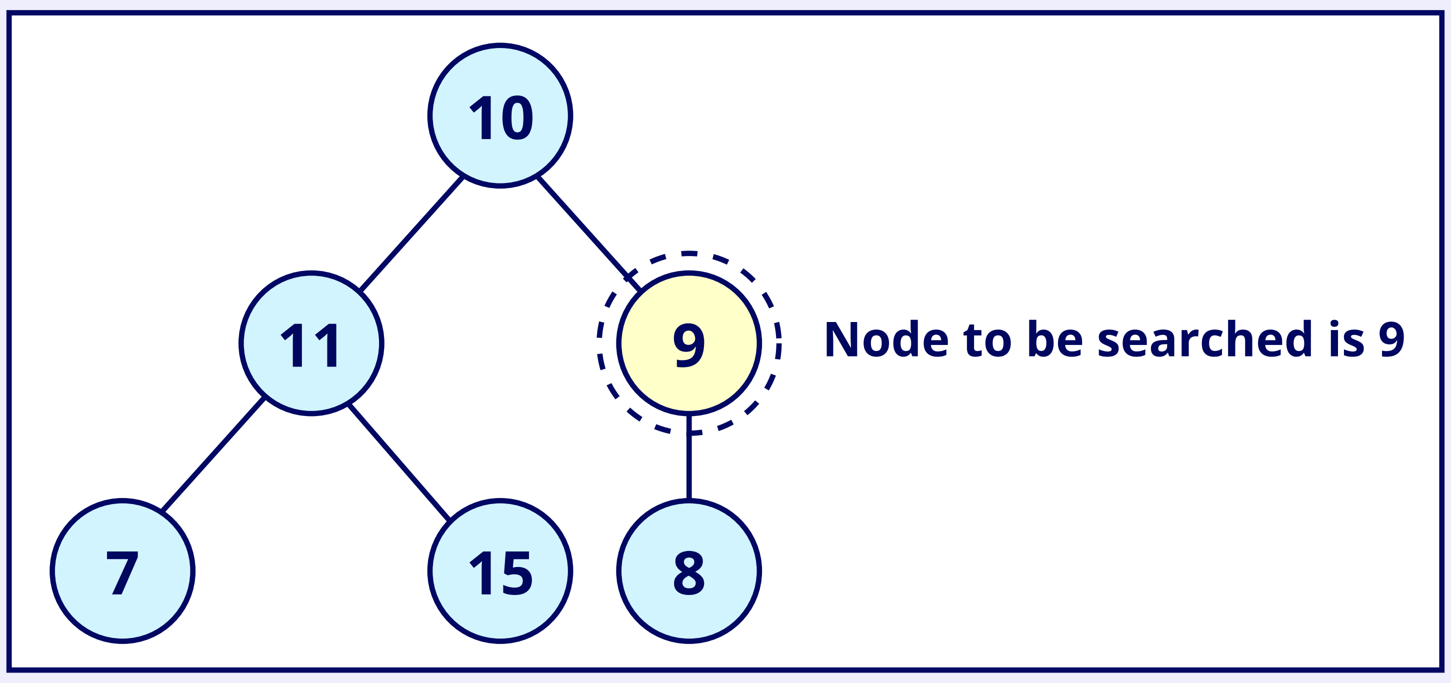
- **Insertion:** O(n)



- **Deletion:** O(n)



- **Search:** O(n)



Application:

- Hierarchical data representation (e.g., file systems, databases)

Pros:

- **Hierarchical structure:** Ideal for representing hierarchical data
- **Multiple traversals:** Provides various traversal methods for different use cases

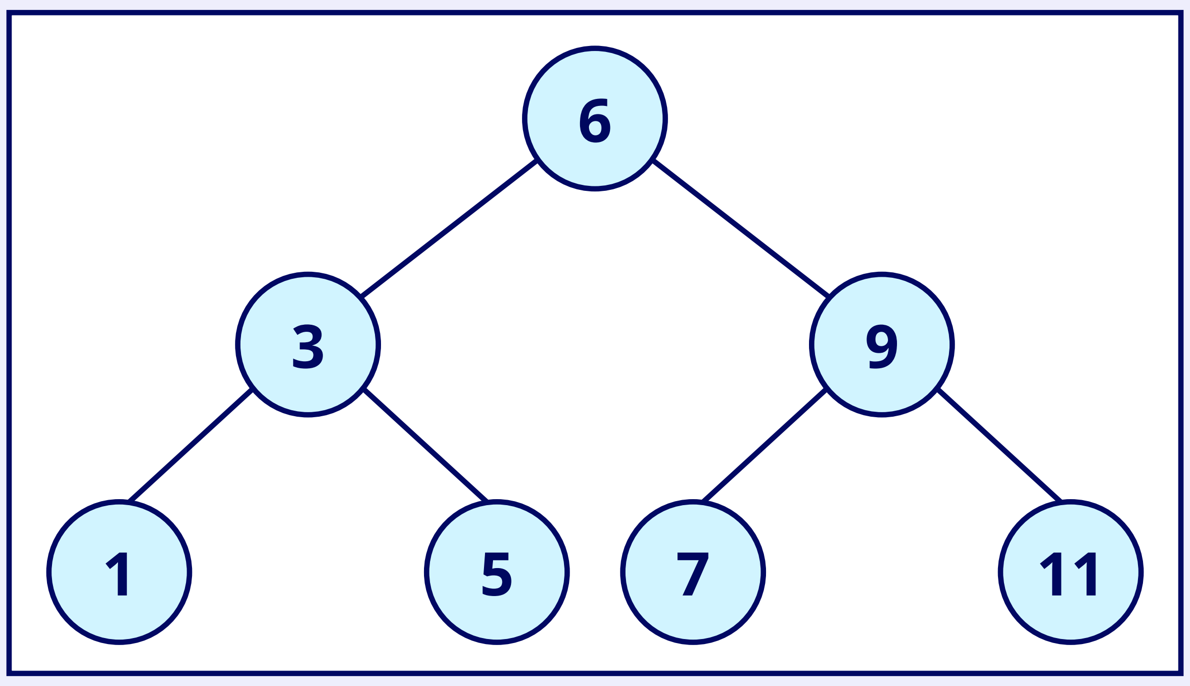
Cons:

- **Inefficient operations:** O(n) time complexity for insertion, deletion, and searching
- **Memory overhead:** Requires memory for storing pointers for each node

Binary Search Trees (BST):

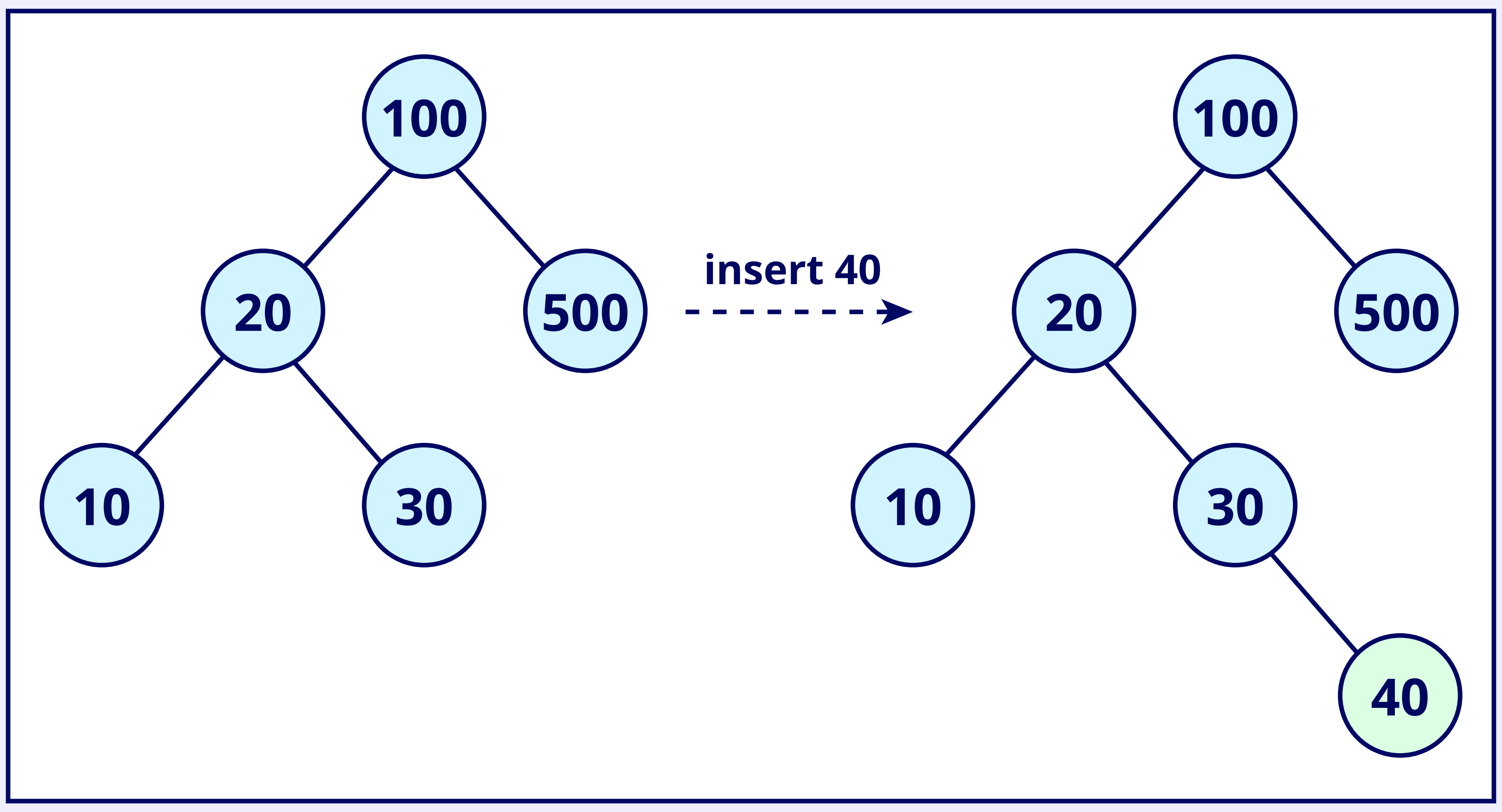
Definition:

It's a binary tree where the left child has a value less than its parent node, and the right child has a value greater than its parent node.

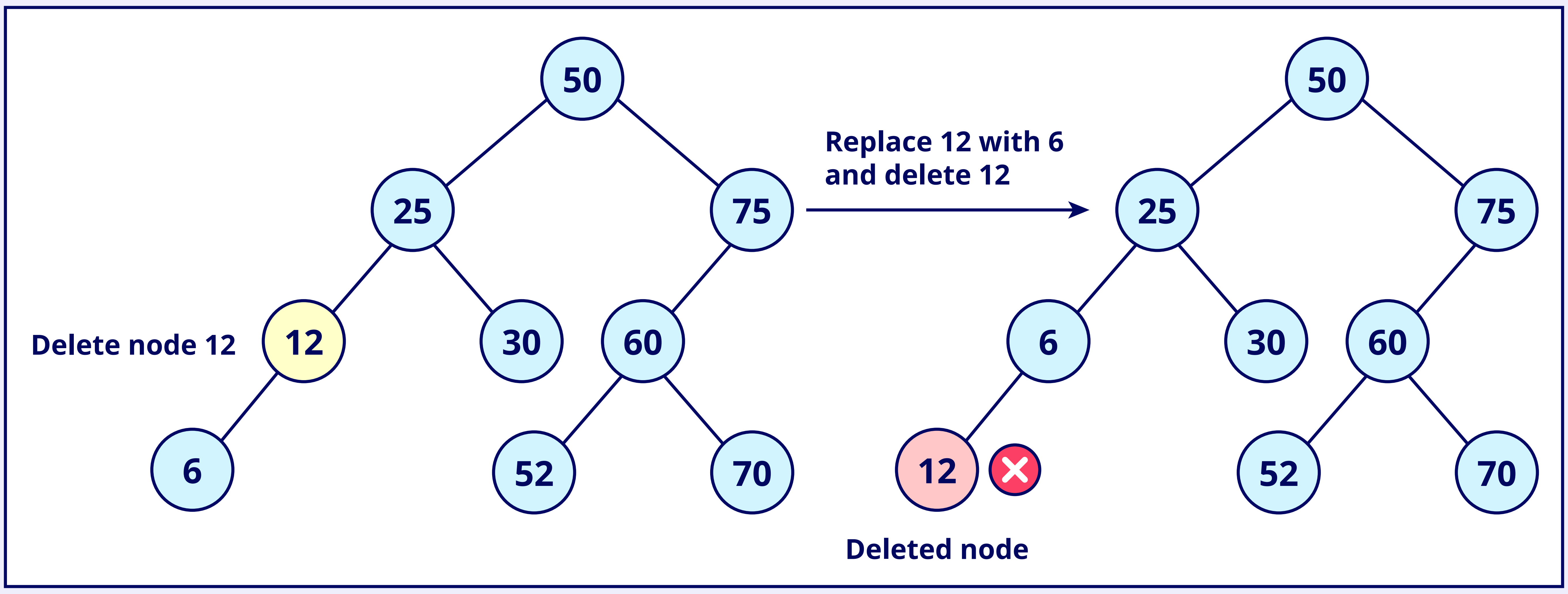


Common operations:

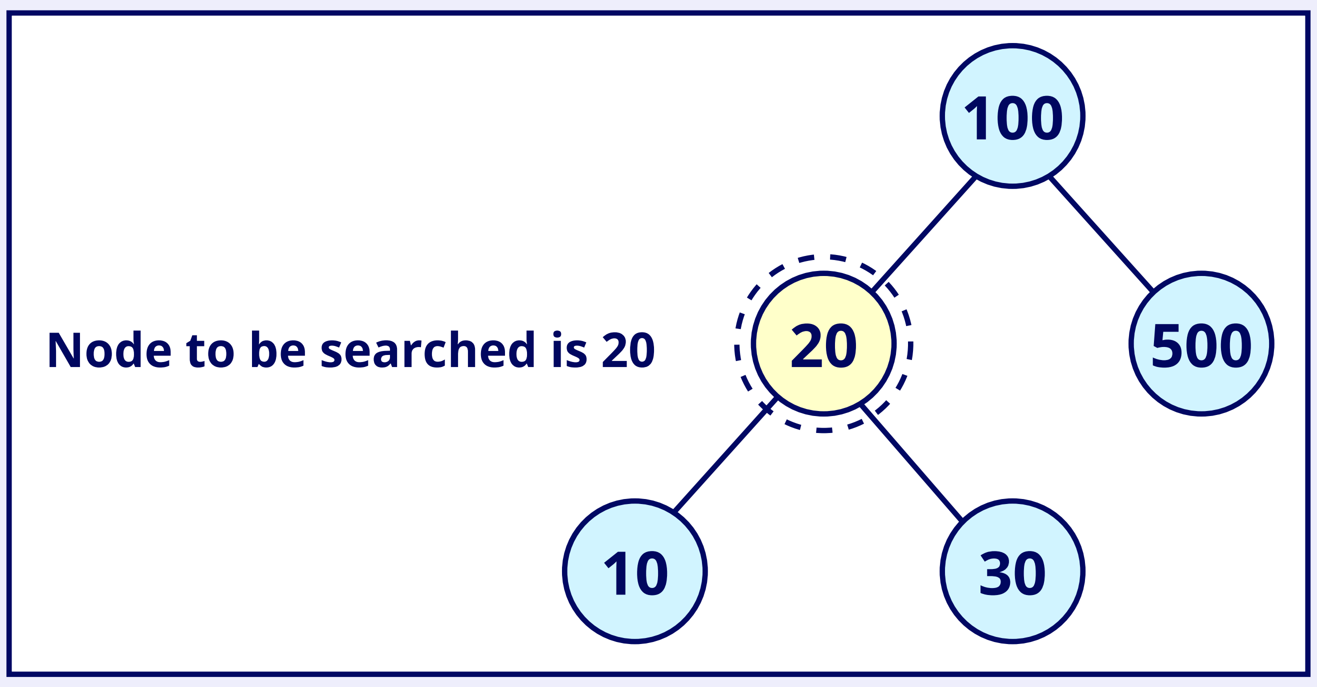
- **Insertion:** O(h) (h = height)



- **Deletion:** O(h) (h = height)



- **Search:** O(h) (h = height)



Application:

- Implementing associative arrays and priority queues

Pros:

- **Efficient searching/sorting:** O(h) time complexity for searching, insertion, and deletion
- **Maintains order:** Keeps elements in a sorted order

Cons:

- **Height-dependent:** Performance depends on the height of the tree; in the worst case (skewed tree), it can degrade to O(n)
- **Duplicate handling:** Typically, duplicates are discarded, which may not be suitable for all use cases