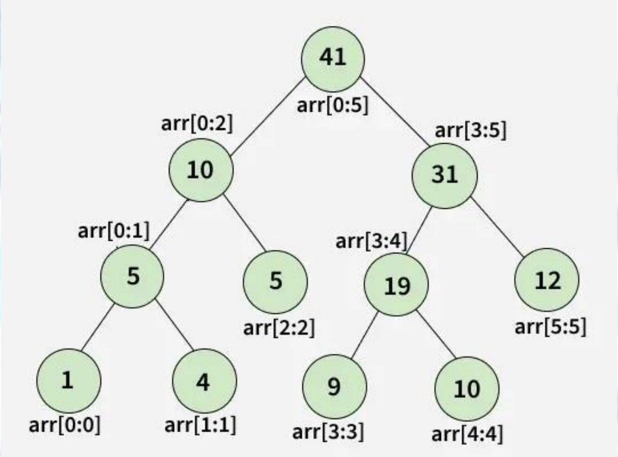
SEGMENT TREE:

- A Segment Tree is a binary tree where each node represents a segment or range of the original array.
- The root node represents the entire array, and leaf nodes represent individual elements.
- Internal nodes store aggregated information (e.g., sum, minimum, maximum) of their child segments.
- Following are the steps for constructing a segment tree:
  - Start from the leaves of the tree
  - Recursively build the parents from the merge operation

The following diagram shows a segment tree built for an array [1, 4, 5, 9, 10, 12]

of size 6



Operations:

- **Build:** Constructs the tree recursively, typically taking O(n) time.
- **Query:** Retrieves aggregated information for a given range in O(log n) time.
- **Update:** Modifies a single element in the array and updates affected nodes in the tree in O(log n) time.

Space Complexity: O(n), as it requires approximately 2n nodes for a complete binary tree.

- **Advantages:** More versatile for various range query types (e.g., sum, min, max, GCD, etc.) and can be extended to handle 2D queries.
- **Disadvantages:** Can be more complex to implement compared to BITs.

BINARY INDEXED TREE (FENWICK TREE)

- A Binary Indexed Tree is an array-based data structure that efficiently calculates prefix sums and supports point updates.
- It leverages binary representations of indices to efficiently navigate and update relevant elements.

Operations:

- **Build:** Constructs the BIT, typically taking O(n log n) time by repeatedly calling the update operation.
- **Prefix Sum Query:** Calculates the sum of elements from index 0 to a given index in O(log n) time.
- **Update:** Modifies a single element and updates the BIT in O(log n) time.

Space Complexity: O(n), as it uses an array of size n+1.

Advantages: Simpler to implement and often has a smaller constant factor in its time complexity, making it faster in practice for certain operations (especially prefix sums).

Disadvantages: Primarily designed for prefix sum queries; extending it to other range operations (like minimum or maximum) is less straightforward or requires modifications.