Erdianti Wiga Putri A

13522053 – K01

**Rangkuman Algoritma dan Struktur Data**

**Chapter 12 – Speeding Up All the Things with Binary Trees**

A tree is also a node-based data structure, but within a tree, each node can have links to multiple nodes. There are many different kinds of tree-based data structures, but in this chapter, we’ll be focusing on a particular tree known as a binary tree. A binary tree is a tree that abides by the following rules, each node has either zero, one, or two children, if a node has two children, it must have one child that has a lesser value than the parent, and one child that has a greater value than the parent. The binary tree search algorithm starts at the root node, inspecting values and recursively navigating either the left or right subtree based on whether the target value is less or greater than the current node. In general, binary tree search has a time complexity of O(log N), efficiently narrowing down the possible values at each step, comparable to binary search in ordered arrays. To insert a value into a binary tree, the algorithm searches for the appropriate node, traversing based on the value's relationship to the current node. After locating the correct node, the new value is inserted as a left or right child, requiring one additional step beyond the search. Binary trees offer efficient O(log N + 1) insertion, making them advantageous for dynamic data, but maintaining balance is crucial for optimal search efficiency, and inserting sorted data may result in an imbalanced structure. Deletion in a binary tree involves searching for the node to be removed and performing different actions based on the number of children the node has. If the node has no children or one child, it is straightforward to delete or replace it. For nodes with two children, the algorithm replaces the node with its successor node, the smallest value among the greater descendants. Special consideration is given when the successor node has a right child, involving additional steps to maintain the tree's structure.