# Algorithm Complexity Analysis

## • US08:

US08 is an N-ary tree, so it has several aspects that must be analyzed.
The complexity of the insertion depends on where the new node is inserted. If it is in the root, the insertion can be done in O(1), however some cases can lead to complexity until it can take O(n).
To go through all the nodes in the tree, the complexity is O(n) since it is necessary to go through each node once.
Finally, to join subtrees into a main tree, the complexity in the worst case is O(m*n) where n is the number of nodes in one tree and m is the number of nodes in the other tree.

## • US09:

US09 aims to create a map that associates each node with identifiers of operations or materials. Thus, it is necessary to traverse the N-ary Tree, and for this it is necessary to go through each node once, so the complexity is O(n). After having the map, identifying the node by the key will only have O(1) complexity.

## • US10:

US10 initially needs to traverse all the nodes of the N-ary tree once to group them into a map, classifying them by different quantities, hence the complexity O(n) to traverse and O(1) to insert into the map.
After that, it is necessary to insert it into a Binary Search Tree. Thus, with p being the number of groups of materials, then the insertion in the BTS will be O(p * log (k)), with k being the number of elements in the tree.

## • US11:

US11 initially traverses each node of the N-aru tree once, having O(n) complexity. Then, each of the nodes is inserted into a Heap Priority Queue structure, each insertion has complexity O(log k) as they may need to traverse the entire height of the heap, which is logarithmic in relation to the number of elements in the queue at the time of insertion(k ). However, since the insertion occurs for each node, the complexity is O (n * log k).

## • US12:

US12 intends to update the quantity of a given material that belongs to the N-ary Tree.
If this material is a leaf then the complexity is O(1), but if it is not, all the children of the node of this material also need to have their complexity changed, therefore it is necessary to go through the rest of the nodes once each, and in the worst case has O(n) complexity.

## • US13:

US13 aims to calculate the quantity of each material in the N-ary tree. To do this, it is necessary to traverse each node of the tree once, thus having a complexity of O(n). Each of these quantities and materials are being associated in a map having only O(1) complexity.

- **US14:**

US14 initially traverses each node of the N-aru tree once, having O(n) complexity. Then, each of the nodes is inserted into a Heap Priority Queue structure, each insertion has complexity O(log k) as they may need to traverse the entire height of the heap, which is logarithmic in relation to the number of elements in the queue at the time of insertion(k ). However, since the insertion occurs for each node, the complexity is O (n * log k).

Subsequently, with m being the number of nodes with maximum depth, all of these will be removed, and each removal in the Heap Priority Queue has complexity O (log n), so the total complexity of the removals is O (m * log n).

Finally, each node with maximum depth will be traversed to its root, and as it is an N-ary Tree, the average complexity when traversing the maximum depth is O (log n).

- **US15:**

US15 initially traverses each node of the N-ary tree once, with O(n) complexity, and associates each operation of the respective node to a depth (integer) through a map and each of these operations has O(1) complexity.

Later it is necessary to add each group of operations with the same depth to an AVL Tree structure. Each insertion in the AVL has complexity O (logk), where k is the number of nodes in the AVL. Since the insertion will occur as many times as the number of groups of operations with the same depth (p), then the total insertion complexity is O(p * logk).