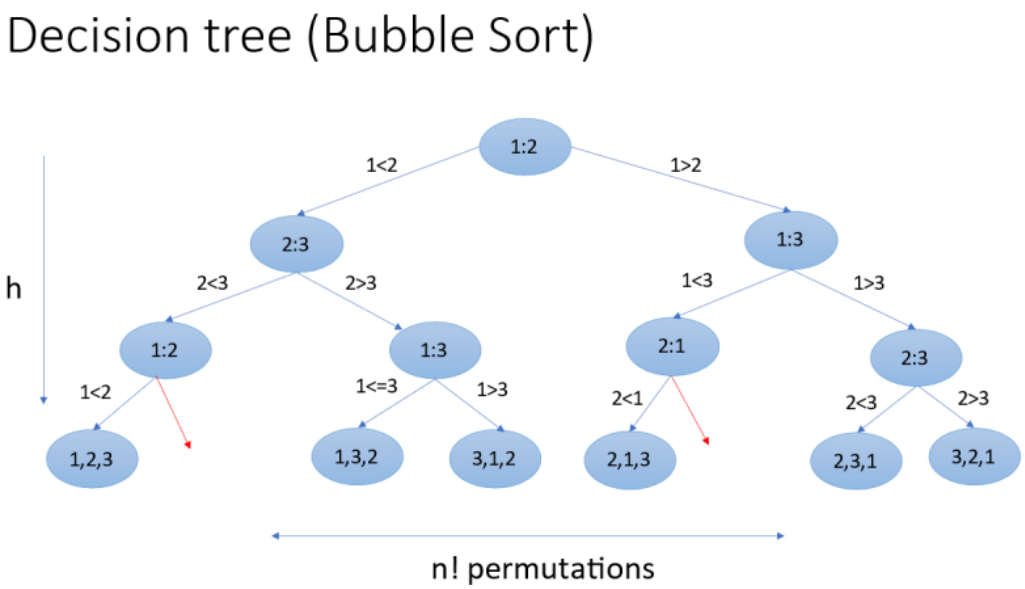
Algorithms: Design and Analysis - CS 412

Weekly Challenge 02: Sorting

Ali Muhammad Asad - aa07190

1. (1 point) A decision tree is a binary tree that represents the comparisons between elements that are performed by a particular sorting algorithm operating on an input of a given size. Control, data movement, and all other aspects of the algorithm are ignored. The leaf nodes represent all possible permutations (arrangements) of the input array. For example, the following is a decision tree of the bubble sort algorithm for $n = 3$, where n is the total number of elements in an array.



Take $n = 3$, and compare the efficiency of selection sort and insertion sort, in terms of the number of comparisons, with the help of a decision tree when the array is:

1. Fully sorted
2. Fully unsorted

Please draw the decision tree for both algorithms and give your analysis in three to four lines.

Solution: Selection sort works by repeatedly finding the minimum element from the unsorted array, and putting it at at the beginning, thus dividing the array into two groups; the sorted subarray and the unsorted subarray. The decision tree for selection sort for 3 arbitrary elements $\{a, b, c\}$ is shown below:

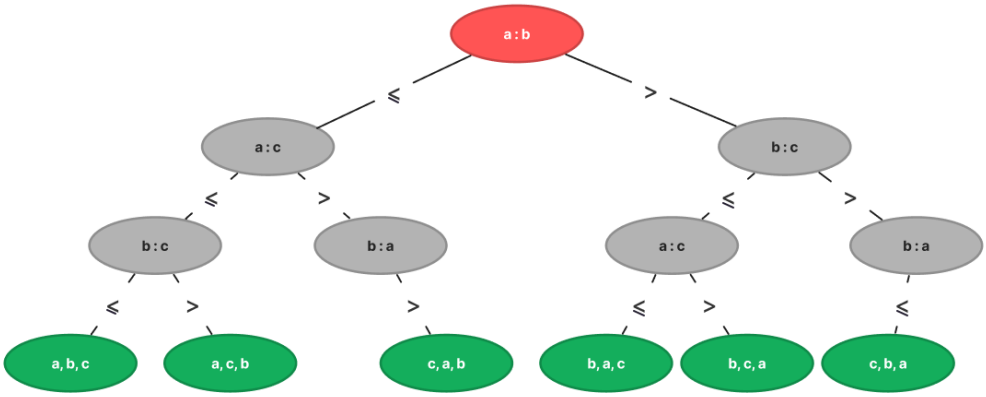


Figure 1: Decision tree for selection sort

Insertion sort works by repeatedly inserting the next element from the unsorted subarray into the sorted subarray. The decision tree for insertion sort for 3 arbitrary elements $\{a, b, c\}$ is shown below:

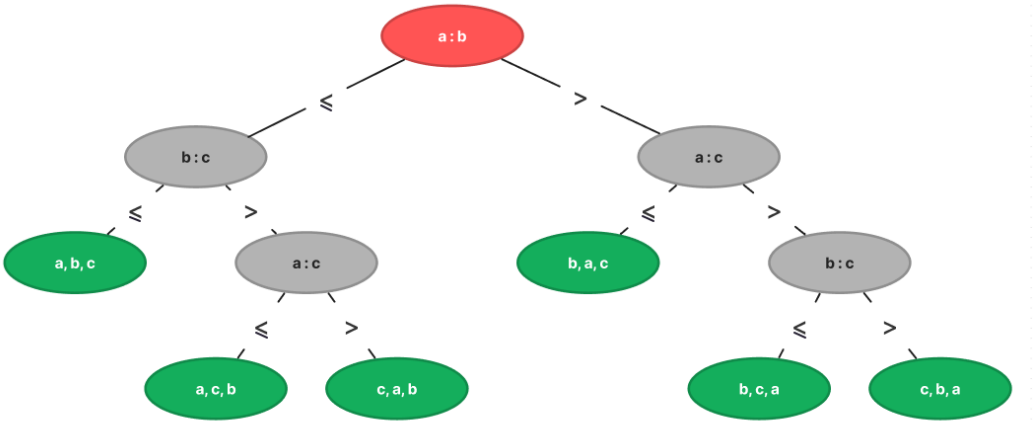


Figure 2: Decision tree for insertion sort

In both the above figures, the red node represents the root node, the green nodes represent the leaf nodes (which are also the possible permutations of our elements), while the other nodes are the intermediary nodes.

1. **Fully sorted:**
    - Selection Sort: there are 3 comparisons for the fully sorted array, since selection sort works by comparing each element to the rest of the elements in the array, and then swapping the appropriate elements.
    - Insertion Sort: there are 2 comparisons for the fully sorted array, since insertion sort compares each element first by its previous element. Therefore, when the array is fully sorted, excluding the first element, all elements are compared once.
2. **Fully unsorted:**
    - Selection Sort: there are 3 comparisons for the fully unsorted array, by the same argument as above.
    - Insertion Sort: there are 3 comparisons for the fully unsorted array, since insertion sort compares each element first by its previous element. Therefore, when the array is fully unsorted, all elements are compared.

While both algorithms have the same performance for the fully unsorted array, insertion sort is more efficient for the fully sorted array.

In general, selection sort does $\frac{n(n-1)}{2} = O(n^2)$ comparisons in the best and worst case, while insertion sort does $(n-1) = O(n)$ comparisons in the best case and $\frac{n(n-1)}{2} = O(n^2)$ comparisons in the worst case.