Algorithm Visualization Automation

Michael Galliers

Eastern Kentucky University - Department of Computer Science

Introduction

Computer algorithms are the basis for how computers how to solve problems. Algorithms describe the series of steps a program must take to complete a certain task. This includes checking conditions and performing actions based on those conditions. Having the ability to visualize how algorithms operate while designing them can be extremely valuable. Visualization can explain an algorithm's behavior when given a specific input and show the algorithmâĂŹs efficiency. Having the ability to automate the visualization process can be extremely useful in algorithm development by showing designers ways in which algorithms could be changed to increase efficiency.

Sorting Algorithms

One of the most studied problems in computer science is sorting. There are many different types of sorting algorithms that have been developed and each has its advantages and disadvantages. Bubble Sort is one of the most basic. It works by comparing the first 2 elements in a list and swapping them if the first is greater than the second. This repeats with the second and third elements and continues until the greatest element is "bubbled" to the last position. The entire process is repeated to "bubble" the second greatest element to be behind the greatest, and so on until the list is sorted.

Decision Trees

In algorithm analysis, a pruned decision tree is a tree-like structure that describes all possible execution paths the program could take, depending on the input, with any contradictory paths "pruned" (removed). This decision tree is valid if there is a path from the root node to a leaf node that sorts any permutation of an some n length list [1]. Once generated, a pruned-valid decision tree can be interpreted as the different execution paths a program could take and the efficiency of each path. The fewer the nodes in a path from the root to a leaf node, the less comparisons performed, and the more efficient

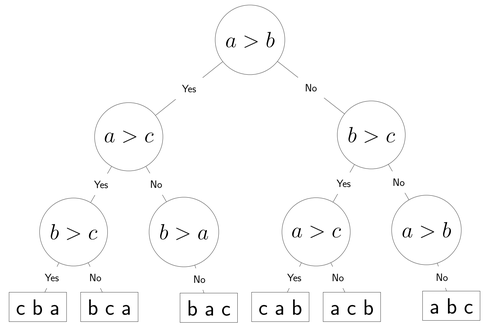

Figure 1: Bubble Sort on [a, b, c] - an inefficient sorting algorithm

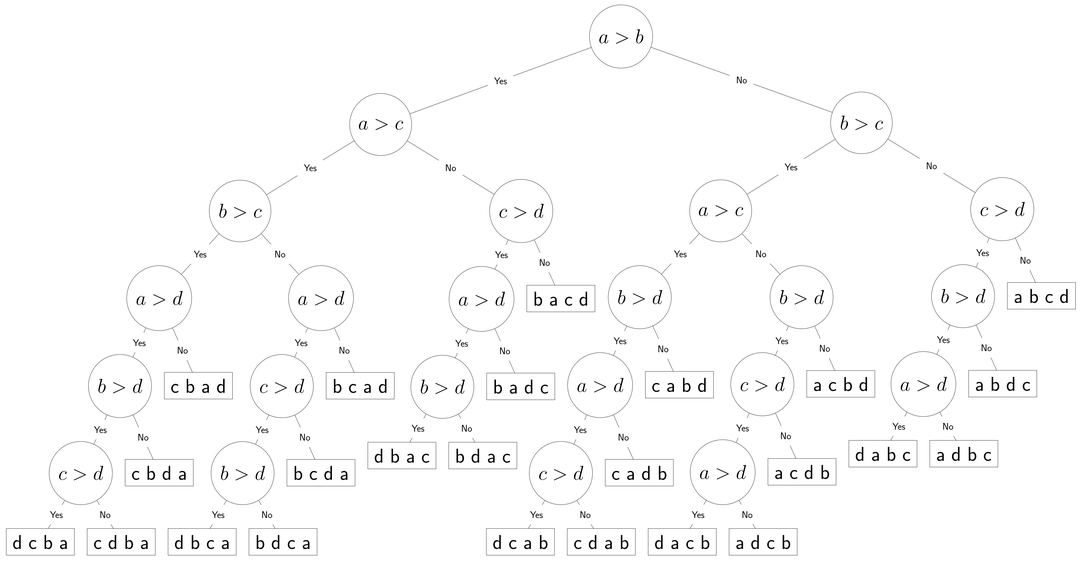

Figure 2: Insertion Sort on [a, b, c, d] - a more efficient sorting algorithm

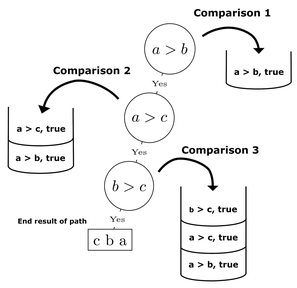

Figure 3: Analyzer Step 1 - Initial Traversal

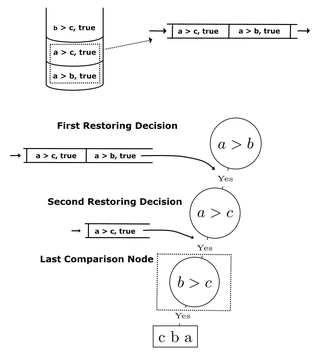

Figure 4: Analyzer Step 2 - Restoration Process

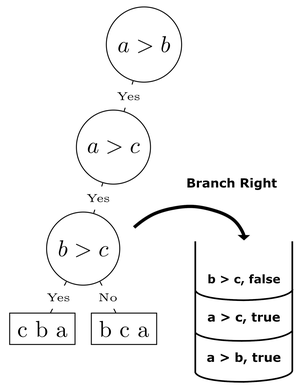

Figure 5: Analyzer Step 3 - Branching Right

Decision Tree Generator

To solve these problems, I designed an automatic analysis algorithm, the *Decision Tree Generator*. This program takes a modified version of any sorting algorithm and generates a pruned- valid decision tree for some arbitrary input variables. To build the tree, the generator must run the sorting algorithm through various situations and track how the algorithm operates.

Generator Functionality

Below is a description of one key functionality of the generator: state restoration.

1. For each comparison of records, the generator takes the comparison and decision made and pushes them to the state stack. In this case, the generator always chooses the decision to be false. Figure 3
2. Once the algorithm stops running, the generator must restore its execution state to the last *true* decision made. It does this by pulling the prior decisions out of the state stack and places them in a restoration queue. Every time a comparison is made, the next queue entry is dequeued and used as the next decision until empty. Figure 4
3. The generator then makes the *false* decision at that node, branching right. The process repeats until the whole tree is traversed. Figure 5

Conclusion

Visual analysis is key to designing efficient algorithms. The *Decision Tree Generator* is just the beginning of visual algorithm analysis programs that could be implemented to analyze other types of algorithms in many different ways.

References

[1] Richard E. Neapolitan.
Foundations of Algorithms.
Jones & Bartlett Learning, 5th edition, 2015.