**Week 5 - Assignment: Evaluate Sorting Algorithms**

Patrick Swafford

TIM-6110: Programming Languages and Algorithms

Jeevan D’Souza

29 January 2023

Table of Contents

Integer Bubblesort .......................................................................................................................... 3

Integer Quicksort ............................................................................................................................ 4

String Bubblesort ............................................................................................................................ 5

String Quicksort .............................................................................................................................. 6

Works Cited ……………………………………………………………………………………….7

Integer Bubblesort

The Big O notation for this bubblesort algorithm is O(n2). This is because the algorithm has two nested “for” loops. The outer loop iterates through the length of the array, and the inner loop compares and potentially swaps the adjacent elements. The outer loop runs (n – 1) times and the inner loop runs (n – i – 1) times. Which when multiplied out results in n2 – n + 1 iterations, resulting in O(n2) in terms of time complexity.

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generated

Integer Quicksort

The Big O notation for this quicksort algorithm is O(n log n).

This is because the quicksort algorithm uses a divide-and-conquer strategy, where it partitions the array into two sub-arrays and recursively sorts each sub-array. The partition function of the quicksort algorithm has a time complexity of O(n) since it iterates through the array once, and the quicksort function itself also has a time complexity of O(n log n) as it recursively sorts the sub-arrays.

The quicksort algorithm is generally considered more efficient than bubblesort as it has a time complexity of O(n log n) which is faster for large arrays. In comparison, the bubblesort algorithm has a time complexity of O(n^2) which can be quite slow for large arrays. It's important to notice that the complexity of quicksort is O(n log n) on average, but in the worst case it can be O(n^2) if the pivot is not chosen optimally.

Please note that this implementation was implemented with the help of pseudocode from (Cormen, Leiserson, & Rivest, 1990).

Text

Description automatically generated

Text, email

Description automatically generated

String Bubblesort

The Big O notation for this bubblesort algorithm, which sorts a String array instead of an integer array, is also O(n2). This is because the algorithm has the same structure as the previous bubblesort algorithm. The main difference is that instead of comparing integers, this implementation uses the Compare method of the String class, which compares two strings lexicographically. Generally speaking, this method has a time complexity of O(min(m,n)), where m and n are the lengths of the strings being compared. However since the number of comparisons is still n2, the overall time complexity of the algorithm is still O(n2)

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generated

String Quicksort

As with the Integer Quicksort algorithm, the Big O notation of this String Quicksort algorithm is also O(n log n).

This quicksort algorithm for strings is similar to the quicksort algorithm for integers in terms of structure and implementation. It uses the same divide-and-conquer strategy where it partitions the array into two sub-arrays and recursively sorts each sub-array. The partition function has a time complexity of O(n) since it iterates through the array once, and the quicksort function itself also has a time complexity of O(n log n) as it recursively sorts the sub-arrays.

The main difference between this algorithm and the previous quicksort algorithm is the comparison operation. Instead of comparing integers, it uses the Compare method of the String class, which compares two strings lexicographically. The implementation of this method may vary, but in most cases, it has a time complexity of O(min(m,n)) where m and n are the lengths of the strings being compared. However, since the number of comparisons is still n log n, the overall time complexity of the algorithm is still O(n log n).

Text

Description automatically generated

Text

Description automatically generated

# Works Cited

Cormen, T. H., Leiserson, C. E., & Rivest, R. L. (1990). *Introduction To Algorithms.* Cambridge, MA, USA: The MIT Press.