**Exploring Data Structures – Proof of Concept**

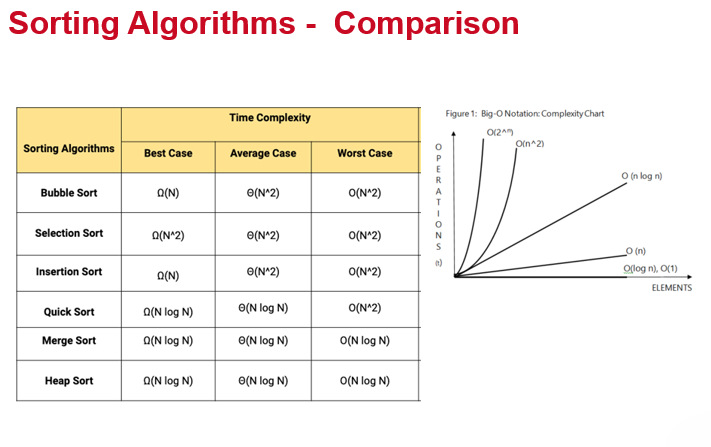
Data Structure Efficiency – Stack, Queue, Linked List

Text

Description automatically generated

The Linked List was the most efficient data structure, adding the Product objects to a list in 3 milliseconds

Sorting Algorithms Efficiency – Insertion Sort, merge Sort, Bubble Sort, Selection Sort



The table above describes the best, average, and worse case complexity of a number of Sorting algorithms.

The Bubble Sort is a simple sorting algorithm that repeatedly steps through an array, element by element. It compares the current element with the element after it, swapping their values if needed. It performs poorly – its efficiency decreases dramatically on lists of more than a small amount of elements, and is therefore rarely used in practice.   
(Source: <https://en.wikipedia.org/wiki/Bubble_sort>)

The Selection Sort is an algorithm that divides an array into two parts; a sorted sublist (built from left to right), and a sublist of the remaining unsorted items from the array. Depending on the sorting order, the smallest or largest element in the array is found, and exchanged with the leftmost unsorted element (putting it in sorting order) and moving the sublist boundaries one element to the right. The Selection Sort is inefficient on large lists, and generally performs worse than the similar insertion sort.  
(Source: <https://en.wikipedia.org/wiki/Selection_sort>)

The Insertion Sort is a simple algorithm that builds the final sorted array one item at a time by comparisons. As the current item works from left to right, everything left of the item is known to be sorted, and everything to the right is unsorted.  
It performs well when sorting small and nearly sorted data sets, but is not efficient at sorting large, unsorted data sets.  
(Sources: <https://en.wikipedia.org/wiki/Insertion_sort>, Topic 13 - B-Trees,Searching and Sorting.pptx, TAFESA)

The Merge Sort an efficient, general-purpose, comparison-based sorting algorithm. The array being sorted is recursively split in half, and when the array in in groups of one (1), it is reconstructed in sort order, and each reconstructed array is merged with the other half.  
While not appropriate for large pathologically sorted data sets, it is appropriate for large data sets, and performs efficiently on small and nearly sorted data sets.  
(Sources: <https://en.wikipedia.org/wiki/Merge_sort>, Topic 13 - B-Trees,Searching and Sorting.pptx, TAFESA)

Given our use of large data sets, the most efficient choice for the Supermarket Admin program would therefore be the Merge Sort.

Searching Algorithms Efficiency – Sequential Search, Binary Search

Text

Description automatically generated

This shows that the Binary Search is faster than the Sequential Search, provided the insertion sort is used to sort the array first.

Bibliography

Topic 13 - B-Trees, Searching and Sorting.pptx, TAFESA 2022

<https://en.wikipedia.org/wiki/Bubble_sort>

<https://en.wikipedia.org/wiki/Selection_sort>

<https://en.wikipedia.org/wiki/Insertion_sort>

<https://en.wikipedia.org/wiki/Merge_sort>