Name: Royston Loo Module No: IT2553

Admin No: 214242Q

Module Grp No: 2

I have used 6 Advanced Sorting Algorithms in my assignment: 5 Useful Advanced Sorts, 1 Nonsensical Sort. I will first be talking about my 5 useful Advanced sorting methods, starting with Pancake sort.

**Pancake Sort. Difficulty: Medium**

This is a reversal-based sorting algorithm. It is based on the real-life problem of resembling pancakes on a plate with the help of a spatula. It gets its name from the flip operation used in the algorithm analogous to flipping pancakes. Unlike most sorting algorithms that try to minimize the number of comparisons required to perform the sort, it tries to sort the array in minimum reversals. Just like selection sort, it also places the maximum element at the end.

The algorithm starts by flipping the top pancakes from the largest one to the top, so that the largest pancake goes to the top. Then, sorting the whole pancake stack we bring the largest pancake to the bottom. The subsequent move brings the second largest to the second position, and so on, recursively.

The time complexity is O(n\*2).

Next, I will be talking about Heap Sort.

**Heap Sort Difficulty: Medium**

This is a sorting algorithm that is based on Binary Heap data structure. It first finds the minimum element and place the minimum element at the beginning. It then repeats the same process for the remaining elements.

A Binary Heap is a complete Binary Tree where items are stored in a special order such that the value in a parent node is greater (or smaller) than the values in its two children nodes. The former is called max heap and the latter is called min-heap. The heap can be represented by a binary tree or array.

As a Binary Heap is a complete binary tree, it can be easily represented as an array and the arrayed based representation is space efficient.

When heapsort builds a max heap, the largest element is at index 0, and it swaps with the end of the array and reduce the heap size. This will happen until the first element is left when building a heap

The time complexity is O (log n), which is faster than the basic sorting algorithms at O(n\*2).

The advantage of heapsort is that it is efficient as the time required to perform Heap sort increases logarithmically while other algorithms may grow exponentially slower as the number of items to sort increases.

Memory usage is also minimal because apart from what is necessary to hold the initial list of items to be sorted, it needs no additional memory space to work.

Finally, it is simpler to understand than other equally efficient sorting algorithms because it does not use advanced computer science concepts such as recursion.

Next, I will be talking about Shell Sort.

**Shell Sort Difficulty: Medium**

This is a sorting algorithm that is a variation from Insertion Sort. This algorithm avoids large shifts, as in insertion sort, where the smaller value is on the far right and must be moved to the far left. Shell Sort reduces its time complexity by utilising the fact that using Insertion Sort on a partially sorted array results in fewer moves.

This sort has the time complexity of O (n log n). The main advantage that Shell sort has over the basic sorting algorithms such as bubble sort, is that it is capable of swapping indexes that are far apart, whereas something like bubble sort only swaps items that are adjacent. This is very useful to help me sort my data faster.

Next, I will be talking about Radix Sort.

**Radix Sort Difficulty: Medium**

This is a sorting algorithm that sorts the elements by first grouping the individual digits of the same place value. Then, it will sort the elements according to their increasing or decreasing order. It will first find the largest element in the array. Next, it goes through each significant place one by one by using any stable sorting techniques to sort the digits at each significant place. Afterwards, it sorts the elements based on the unit place digits, then digits at tens place, and then the digits at hundreds place.

The time complexity is O (n + k), which is comparably faster than the other sorting algorithms that are at O(n\*2), such as bubble sort, which is why I used this algorithm to sort my data faster.

Lastly, I will be talking about Cocktail Sort.

**Cocktail Sort Difficulty: Easy**

This is a sorting algorithm that is a variation of Bubble sort. Instead of traversing elements from left and move elements in an array accordingly, it traverses through the array in both directions alternatively. The first stage of the sort will work just like bubble sort, sorting from left to right after comparing values. The second stage of the sort will loop the array in the opposite direction, starting from the item that was just before the most recently sorted item, and moving back to the start of the array.

Time complexities are the same at O(n\*2), but it performs better than bubble sort. This is because Cocktail sort requires less amounts of traversals in an array in comparison to bubble sort in general. Hence, this helps to speed up the sorting process for my data.

Now, I will be moving on to my nonsensical sort used: Bogo Sort

**Bogo Sort Difficulty: Easy**

This is a sorting algorithm that randomly shuffles the objects in an array and checks if it is sorted in order correctly. If it is not, it will keep randomly shuffle until it sorts. Because this sort can make repeated shuffles of the same order, it is arguable that this sorting algorithm has the time complexity of O(inf), which is infinity. This sort here is to show what bad sorting algorithms look like in order to know what to not use in the future in comparison to the 5 advanced sorting algorithms I have explained earlier.

Overall, I have learnt a lot about many different types of sorting algorithm, be it good or bad ones. Different algorithms have different time complexities, which will affect how fast it takes to sort an unsorted array.