Name: Eden Will Sng Jin Xuan

Adm No: 201520M

Group:IT2553-02

Advanced Features:

1. Stalin Sort
2. Bogo Sort
3. Counting Sort
4. Radix Sort
5. Heap Sort

Stalin Sort - Difficulty: Easy

Stalin sort is a comparison sort. The original Stalin sort does is that for every element in the list that is not ordered properly will be eliminated immediately. In this case the element will be pop out of the list.

The time complexity of this algorithm will be O(N) which is highly efficient at the cost of records loss. This assignment uses the nondestructive version of Stalin sort, where every element which is not ordered properly in the list is sent to the index 0 (or the equivalence of being punished) and the whole sorting process is repeated. The unsorted list will be resorted until the whole list is sorted

Usefulness

It has a rather excellent best time complexity of O(N) if the array is sorted from the get-go from the randomness. The worse time complexity is O(N^2) which is the same as bubble sort and insertion sort. Therefore, this algorithm is still useful.

Time complexity:

Best – O(N)(if sorted), Worst – O(N^2)

How its applied effectively:

Stalin Sort is a sort used for comedic purposes in this assignment. It is effectively applied to sort the customer names in ascending order, customer names who are not sorted in ascending order will be placed to index [0]. Stalin sort is a good alternative to Bubble sort despite having no advantages.

Bogo Sort - Difficulty Easy

Bogo sort is an inefficient algo that uses shuffling permutation of the elements in an array. It hopes that the next shuffle it could produce will result in a sorted list.

Usefulness

Excellent illustration in why the world requires highly efficient algorithms, the inefficiencies of Bogo sort help illustrate that even with 10 records it is highly still inefficient and slow.

Time complexity

Best- O(N)(if sorted), Average case O(N\*N!), Worst case O(infinity)(due to no limits)

How its applied effectively

It is applied to sort the customer’s name in ascending order, the sorting algorithm will keep shuffling the list till it hopefully becomes sorted.

Counting Sort – Difficulty Medium

Counting Sort is a non-comparison sorting algorithm that sorts the integers through counting the number of unique occurrences of a digit. After that it is placed in a temporary array as an index before sorting. The counting sort used is an unstable sorting algorithm. Which is to say two elements of the same value may be sorted differently after the sorting is finished. Counting sort can be made into a stable algorithm

Time complexity

Best, Average and Worse is - (N + R) (where R is the range of the input)

Usefulness:

Counting sort is a fast algorithm in sorting integer values, counting sort is used in very fast sorting algorithms such as Radix Sort where counting sort is used frequently to quickly sort through large numbers.

Application:

I use counting sort to sort the package price, the counting sort uses the range of package price from lowest record to highest to store the number of unique occurrences. The records will be sorted in the temporary counting array before being outputted as the final value.

Radix Sort – Difficulty Hard

Radix sort is another non-comparison-based sorting algorithm. It uses counting sort to sort the integers by through grouping the keys by individual positions (for example 1234, it starts with 4 first as the first group as the comparison). The radix sort used here is a stable algorithm. Radix sort will pass many counting sorts in order to quickly sort the array.

Radix sort is highly efficient and good at tackling large numbers.

Radix sort exist in many forms such as Least Significant Digit which is highly efficient for integers sorting and Most significant digit, forward recursive which is great for integers with leading zeros or to sort through alphabets. Radix sort also exist hybrid approaches to the algorithm.

The time complexity is

O(d(n+b)) where d is the number of passes of counting sort, n+b is the time complexity of counting sort, where b is a constant.

Usefulness:  
radix sort is extremely useful in sorting large numbers due to its efficiency in the running time. When compared to counting sort, say we are comparing 10 records and the range is 1 million, counting sort will have to sort through 1million unique occurance of the digit while radix sort requires just 6 passes of counting sort of 10 records which is 60 due to the number of significant digits. This is more efficient than purely just using counting sort

Application:

Radix sort is used in sorting the package price of the records by ascending order. And is used for the sorting of records before users can select the rows containing the range of prices they would like to see.

Heapsort – Difficulty Hard

Heapsort is a comparison algorithm that uses the heap binary tree data structure to compare the elements in an array. Heapsort is highly efficient; it changes the array into heap binary tree data structure. The heap binary tree has few characteristics, it is a complete binary tree, the parent node must be greater than the child node(this is a max heap) or smaller which is a min heap. If the conditions are not met, it will be heapified to become a heap data structure.

The process of heapify is a recursion, however the nature of heapsort is that it is not a stable algorithm

Time complexity for heapsort is

Best, Worse, Average – O(n Log n)

Because of this time complexity heap sort is considered one of the fastest sorting algorithms, n log n is the limit for comparison sorts, it requires, o(n) to make max heap, and log n to get to the next maximum value in the heap.

Usefulnesses:

Extremely useful algorithm teaches how to better understand binary trees, recursion. Heapsort being highly efficient can be combine with other algorithms to overcome its weaknesses and flaws, for example heapsort can be used with quicksort and insertion sort to make introsort and the improved version being Pattern Destroying Quicksort. The usefulness’s of heap sort should not be underestimated.

Application:

I use heapsort to sort the customer names in the records. To create the heapsort, I had to understand heap data structure, recursion as well as explore the different types of heapsort that exists.

Shell sort – Difficulty Easy

Shell sort is a variation of insertion sort in which uses gaps to split the array into smaller sublists. Within these gaps just like insertion sort it will compare the elements in the array.

If the element is larger than the adjacent these element positions are swapped. Shell sort when optimized is faster than insertion sort, however a poorly optimized shell sort will result in it having more comparisons than insertion sort. The best combination for the gaps are 9-6-1 which leads to 26 comparisons in a list of 10 records as compared to insertion sort of 30 records compared. Shellsort only needs 3 passes in the range of 10 records due to the number of gaps

The time complexity of shellsort is o(n^2) but can be optimized to go faster depending on the list. I use shellsort in sorting the package name of the records in acending order

Sourced used in researching these algos can to all these sorting algorithms can be found in my codes.