LITERATURE REVIEW OF SORTING ALGORITHMS USING JAVASCRIPT.

0.1 Introduction.

We describe different sorting algorithms which include bubble sort, insertion sort, selection sort, quick sort, mange sort, shell sort Each algorithm is defined and explained their advantages and disadvantages are summarized.

0.2 Selection Sort

The algorithm works by finding [2] the smallest unsorted item and then swapping it with the item in the next position to be filled. The selection sort works as follows: you look through the entire array for the smallest element, once you find it you swap it (the smallest element) with the first element of the array. Then you look for the smallest element in the remaining array (an array without the first element) and swap it with the second element. Then you look for the smallest element in the remaining array (an array without first and second elements) and swap it with the third element, and so on.

0.2.1 Advantages

Better efficiency than bubble sort. Simplicity and ease to implement selection sort would recommend for certain conditions.

0.2.2 Disadvantage

Selection sort is not efficient for large arrays instead insertions sort is preferred.

0.3 Bubble sort

Bubble sort [2] is a simple and the slowest sorting algorithm which works by comparing each element in the list with its neighboring elements and swapping them if they are in undesirable order. The algorithm continues this operation until it makes a pass right through the list without swapping any elements, which shows that the list is sorted. This process makes the algorithm works slower when the size of the input n increased. Because of this reason it considered to be the most inefficient sorting algorithm with large amount of data.

0.3.1 Advantage

Simplicity and ease of implementation and the ability to identify the list is already sorted if it is efficiently implemented.

0.3.2 Disadvantage

Code inefficient, inappropriate for large volumes of data elements and repetitive problems as well.

0.4 Insertion sort

Insertion sort [1] is a simple and efficient sorting algorithm useful for small lists and mostly sorted list. It works by inserting each element into its appropriate position in the final sorted list. For each insertion it takes one element and finds the appropriate position in the sorted list by comparing with neighboring elements and inserts it in that position. This operation is repeated until the list becomes sorted in the desired order. Insertion sort is an in- place algorithm and needed only a constant amount of additional memory space. It becomes more inefficient for the greater size of input data when compared to other algorithms.

0.4.1 Advantage

Simple and very efficient for smaller arrays, hence it works twice as efficiently as the bubble sort.

0.4.2 Disadvantage

Inefficient for large arrays.

0.5 Merge sort

Merge sort uses [1] the divide and conquer approach to solve a given problem. It works by splitting the unsorted array into n sub array recursively until each sub array has 1 element. In general an array with one element is considered to be sorted. Consequently it merges each sub array to generate a

final sorted array. The divide and conquer approach works by dividing the array into two halves such as sub array and follows the same step for each sub array recursively until each sub array has 1 element. Later it combines each sub array into a sorted array until there is only 1 sub array with desired order. This can be also be done non-recursively however, most consider only recursive approaches for the reason that non recursive is not efficient. Merge sort is a stable sort meaning that it preserves the relative order of elements with equal key.

0.5.1 Advantage

Well suited for large arrays.

0.5.2 Disadvantages

It requires twice as much additional memory than other sophisticated sorting algorithms. It is not recommended for smaller arrays. It is difficult to implement the merge operation.

0.6 Quick sort

Quick sort [1] is the fastest general purpose internal sorting algorithm on the average among other sophisticated algorithms. Unlike merge sort it does not require any additional memory space for sorting an array. For the reason that it is widely used in most real time application with large data sets. Quick sort uses divide and conquer approach for solving problems. It works by selecting elements from unsorted array named as a pivot and split the array into two parts called sub arrays and reconstruct the former part with the elements smaller than the pivot and the latter with elements larger than the pivot. This operation is called as partitioning. The algorithm repeats this operation recursively for both the sub arrays. In general, the leftmost or the rightmost element is selected as a pivot. Selecting the left most and right most element as pivot was practiced in the early version of quick sort and this causes the worst case behavior, if the array is already sorted.Later it was solved by various practices such as selecting a random pivot and taking the median of first, middle and last elements. Quick sort is an in-place algorithm and it works very well, even in a virtual memory environment.

0.6.1 Advantages

Quick sort is fast and efficient for large data sets.

0.6.2 Disadvantages

It is not efficient if the array elements are already sorted and also each element in the array are equal.

It's not efficient to sort real objects as its space expensive for large data sets.

0.7 Shell sort

Shell sort [2] is mainly a variation of insertion sort. While in insertion sort elements only move one position ahead, many movements are involved. In shell sort allows exchange of far items then then progressively reduces the gap between elements to be compared.

0.7.1 Advantage

Efficient for medium-size lists.

0.7.2 Disadvantage

Complex algorithm and not as efficient as merge ,quick and heap sort.

0.8 Conclusion

Depending on the array size, sorting algorithms like bubble, insertion and selection sort are best suited for arranging arrays with bubble sort being the less efficient compared to insertion and selection sort algorithms while for large arrays shell, quick and merge sort algorithms are best suited. shell sort is less efficient than quick and merge sort while merge requires more memory than quick sort.

0.9 References

[1] A Survey, Discussion and Comparison of Sorting Algorithms (2014) Ashok Kumar Karunanithi https://pdfs.semanticscholar.org/ Accessed 15 April 2018.

[2] Sorting: Runestone interactive. http://interactivepython.org/runestone/static/pythonds/SortSearch/sorting.html, 15 April 2018.