Sorting algorithms

Document prepared by M.Mirthula

Sorting algorithms :

- Selection sort
- Insertion sort
- Bubble sort
- Quick sort
- Merge sort
- Shell sort
- Count sort
- Radix sort
- Bucket sort
- Heap Sort

Selection Sort :

➢ Two parts available : Sorted and unsorted sub array . Sorts an array by repeatedly finding the minimum element from unsorted array and puts it in the beginning , the sorted array
➢ Efficient when list is small
➢ Not a stable algorithm
➢ Memory space is limited so it makes a minimum number of swaps during sort.

Insertion Sort :

➢ Values from unsorted subarray are placed in sorted subarray.
➢ Efficient when the list or array is already sorted.
➢ Stable algorithm
➢ Very efficient for smaller datasets.

Bubble Sort :

- ➢ Compare the adjacent element and then swapping takes place to sort the array. Total N elements , repeat the process for N-1 times.
- ➢ Two slow for large data sets
- ➢ Stable algorithm

Merge Sort :

- ➢ Merge sort works based on Divide and conquer method.
- ➢ It divides the input array into two halves and recursively calls the two two halves for sorting and then merges the sorted halves .
- ➢ Works well for Linked lists as it supports sequential access.
- ➢ Stable algorithm.

Quick Sort :

- ➢ Pick an element as pivot , sort and partition the elements before and after pivot.
- ➢ Works well for data sets that fit into memory.
- ➢ Not stable
- ➢ Efficient for large datasets compared to merge sort
- ➢ Divide and conquer method.

Count Sort :

- ➢ Sorts an element by counting the number of occurrences of each unique element in an array.
- ➢ Stable algorithm
- ➢ Efficient when range of input data is sufficiently not greater than the object to be sorted.
- ➢ Works for negative inputs too.

Shell sort :

- ➢ Generalization for insertion sort , to overcome the drawback of insertion sort by comparing elements separated by several gap positions.

- ➢ Insertion sort - Move elements one step ahead ; Shell sort - Move elements far ahead , many movements are involved (Exchange items placed in far distance)
- ➢ Not stable.

Which Sorting algorithms are stable ?

- Count sort
- Bubble sort
- Insertion sort
- Merge sort
- Radix sort - Requires another sort for sorting if that sort is stable then radix sort is stable
- Bucket sort - If underlying sort is stable

Mention the Non stable sorting algorithms:

- Quick sort
- Selection sort
- Heap sort
- Shell sort

Mention the Comparison Sort :

- Selection sort
- Bubble sort
- Insertion sort
- Merge sort
- Quick sort
- Heap sort

Mention the Non comparison Sort:

- Bucket sort
- Radix sort
- Count sort

Mention the inplace - sorting and outplace-sorting algorithms :

Inplace sorting algorithms - Algorithms that do not require any extra space for storage .

- Quick sort
- Heap sort

Outplace sorting algorithms - That requires auxiliary space for sorting

- Merge sort
- Count sort - Counting array is used as an auxiliary space
- Bucket sort - Uses hash table
- Radix sort

Mention the online and offline algorithms :

Online algorithms : Algorithms that accept new elements as an input during the on-going procedure.

- Insertion sort

Time and Space complexity :

Algorithm	Time			Space
Sorting	**Best**	**Average**	**Worst**	**Worst**
Selection sort	Ω(n^2)	θ(n^2)	Ω(n^2)	O(1)
Bubble sort	Ω(n)	θ(n^2)	O(n^2)	O(1)
Insertion sort	Ω(n)	θ(n^2)	O(n^2)	O(1)
Heap sort	Ω(nlog(n))	θ(nlog(n))	O(nlog(n))	O(1)
Quick Sort	Ω(nlog(n))	θ(nlog(n))	O(n^2)	O(logn)
Merge Sort	Ω(nlog(n))	θ(nlog(n))	O(nlog(n))	O(n)
Bucket Sort	Ω(n+k)	θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	θ(nk)	O(nk)	O(n+k)
Count Sort	Ω(n+k)	θ(n+k)	O(n+k)	O(k)
Shell Sort	Ω(nlog(n))	θ(nlog(n)^2)	O(nlog(n)^2)	O(1)