SORTING ALGORITHM

Key Takeaways from this pdf:

- Bubble Sort
- Selection Sort
- Insertion Sort
- Merge Sort
- Quick Sort
- Heap Sort

MAD Algos

MAD Algos

Sorting-Algorithms

Sorting means to arrange a following set of numbers in ascending/increasing/non decreasing or descending/decreasing/non increasing order, and we need certain algorithms in programming to implement the same.

Various Sorting Algorithms are as follows:

Bubble Sort

Bubble sort, sometimes referred to as sinking sort, is a simple sorting algorithm that repeatedly steps through the list to be sorted, compares each pair of adjacent items and swaps them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted. The algorithm, which is a comparison sort, is named for the way smaller or larger elements "bubble" to the top of the list. Although the algorithm is simple, it is too slow and impractical for most problems even when compared to insertion sort. Bubble sort can be practical if the input is in mostly sorted order with some out-of-order elements nearly in position.

Time complexity analysis:

Worst Case	Average Case	Best Case
$O(n^2)$	$\Theta(n^2)$	$\Omega(n)$

In-place?	Stable?
Yes	Yes

Selection Sort

Selection sort is a sorting algorithm, specifically an in-place comparison sort. It has O(n2) time complexity, making it inefficient on large lists, and generally performs worse than the similar insertion sort. Selection sort is noted for its simplicity, and it has performance advantages over more complicated algorithms in certain situations, particularly where auxiliary memory is limited.

Time complexity analysis:

Worst Case	Average Case	Best Case
$O(n^2)$	$\Theta(n^2)$	$\Omega(n^2)$

In-place?	Stable?
Yes	No

Insertion Sort

Insertion sort is a simple sorting algorithm that builds the final sorted array (or list) one item at a time. It is much less efficient on large lists than more advanced algorithms such as quicksort, heapsort, or merge sort. Insertion sort iterates, consuming one input element each repetition, and growing a sorted output list. At each iteration, insertion sort removes one element from the input data, finds the location it belongs within the sorted list, and inserts it there. It repeats until no input elements remain.

Time complexity analysis:

Worst Case	Average Case	Best Case
$O(n^2)$	$\Theta(n^2)$	$\Omega(n)$

In-place?	Stable?
Yes	Yes

Counting Sort

Counting sort is an algorithm for sorting a collection of objects according to keys that are small integers; that is, it is an integer sorting algorithm. It operates by counting the number of objects that have each distinct key value, and using arithmetic on those counts to determine the positions of each key value in the output sequence. Its running time is linear in the number of items and the difference between the maximum and minimum key values, so it is only suitable for direct use in situations where the variation in keys is not significantly greater than the number of items. However, it is often used as a subroutine in another sorting algorithm, radix sort, that can handle larger keys more efficiently.

Time complexity analysis:

Worst Case	Average Case	Best Case
$O(n+k)$	$\Theta(n+k>)$	$\Omega(n+k)$

In-place?	Stable?
No	Yes

Heap Sort

Heapsort is a comparison-based sorting algorithm. Heapsort can be thought of as an improved selection sort: like that algorithm, it divides its input into a sorted and an unsorted region, and it iteratively shrinks the unsorted region by extracting the largest element and moving that to the sorted region. The improvement consists of the use of a heap data structure rather than a linear-time search to find the maximum.

The heapsort algorithm involves preparing the list by first turning it into a max heap. The algorithm then repeatedly swaps the first value of the list with the last value, decreasing the range of values considered in the heap operation by one, and sifting the new first value into its position in the heap. This repeats until the range of considered values is one value in length.

Time complexity analysis:

Worst Case	Average Case	Best Case
O(n log(n))	Θ(n log(n))	Ω(n) (bottom up)

In-place?	Stable?
Yes	No

Merge Sort

Merge sort is an efficient, general-purpose, comparison-based sorting algorithm. Most implementations produce a stable sort, which means that the implementation preserves the input order of equal elements in the sorted output. Merge sort is a divide and conquer algorithm. Conceptually, a merge sort works as follows:

1. Divide the unsorted list into n sublists, each containing 1 element (a list of 1 element is considered sorted).
2. Repeatedly merge sublists to produce new sorted sublists until there is only 1 sublist remaining. This will be the sorted list.

Time complexity analysis:

Worst Case	Average Case	Best Case
O(n log(n))	Θ(n log(n))	Ω(n log(n))

In-place?	Stable?
No	Yes

Quick Sort

Quicksort (sometimes called partition-exchange sort) is an efficient sorting algorithm, serving as a systematic method for placing the elements of an array in order. When implemented well, it can be about two or three times faster than its main competitors, merge sort and heapsort.

Quicksort is a comparison sort, meaning that it can sort items of any type for which a "less-than" relation (formally, a total order) is defined. In efficient implementations it is not a stable sort, meaning that the relative order of equal sort items is not preserved. Quicksort can operate in-place on an array, requiring small additional amounts of memory to perform the sorting. It is very similar to selection sort, except that it does not always choose worst-case partition.

Time complexity analysis:

Worst Case	**Average Case**	**Best Case**
$O(n^2)$	$\Theta(n \log(n))$	$\Omega(n \log(n))$

In-place?	**Stable?**
Yes	No