DSA Assignment no. 4

bscs 01 Section b

Muhammad qalb e ali

REg no. 200901105

10th Dec 2021

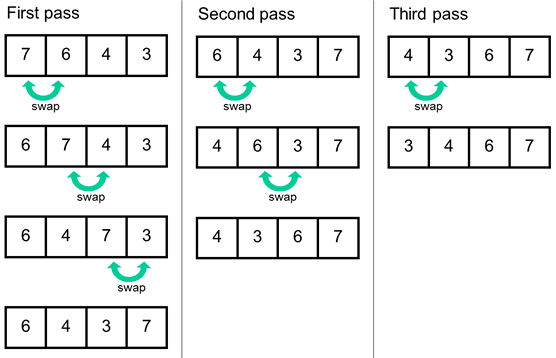
**Types Of Sorting Algorithm:**

**COMPARISON SORT:**

It is a type of sorting algorithm which compares and swaps the elements in an array as needed in order to sort those elements to make a sorted array. Following are the types of comparison sort:

**BUBBLE SORT:**

Bubble sort is a simple sorting algorithm. It works by repeatedly stepping through the list to be sorted, comparing each pair of adjacent elements and swapping 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. Because it only uses comparisons to operate on elements as it is a comparison sort.

****

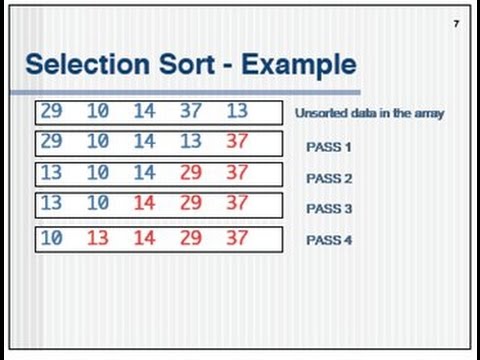
**COMPLEXITIES:**

Worst case scenario: *O*(*n*2)

Best case scenario: *O*(*n*)

Average case scenario: *O*(*n*2)

**SELECTION SORT:**

****Selection sort is a sorting algorithm that selects the smallest element from an unsorted list in each iteration and places that element at the beginning of the unsorted list.

**ALGORITHM:**

Find the minimum value in the list

Swap it with the value in the first position

Repeat the steps above for the remainder of the list (starting at the second position and advancing each time)

Effectively, we divide the list into two parts: the sublist of items already sorted and the sublist of items remaining to be sorted.

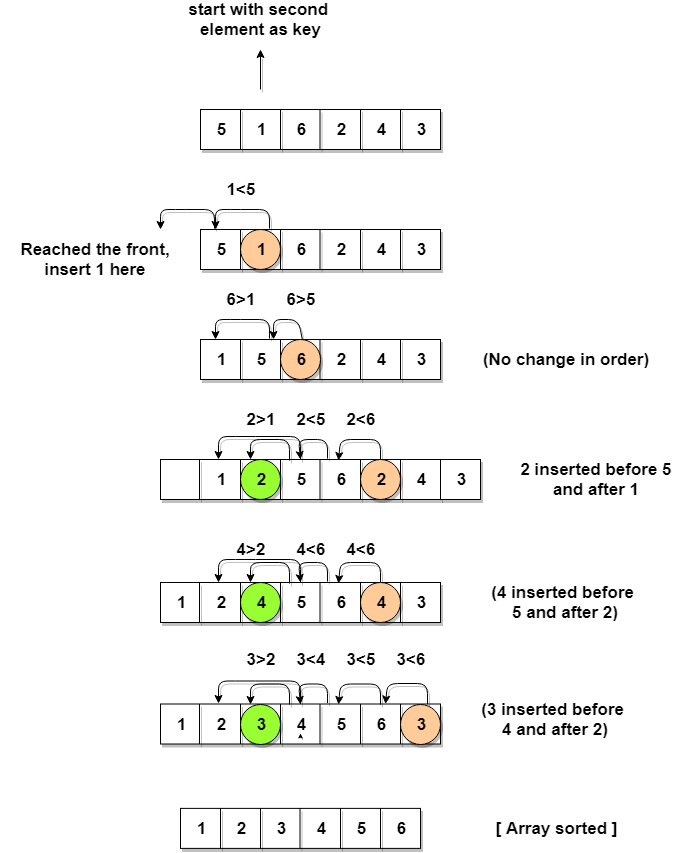
**COMPLEXITIES:**

Worst case scenario: *O*(*n*2)

Best case scenario: *O*(*n*2)

Average case scenario: *O*(*n*2)

**INSERTION SORT:**

****Insertion sort is a comparison sort in which the sorted array (or list) is built one entry at a time. This is an in-place comparison-based sorting algorithm. Here, a sub-list is maintained which is always sorted. For example, the lower part of an array is maintained to be sorted. An element which is to be inserted in this sorted sub-list, must find its appropriate place and then it has to be inserted there. Hence the name, insertion sort. The array is searched sequentially, and unsorted items are moved and inserted into the sorted sub-list (in the same array).

**COMPLEXITIES:**

Worst case performance: O(n2)

Best case performance: O(n)

Average case performance: O(n2)

**INTEGER SORT:**

Integer sorting is a subclass of the sorting problem where the elements have integer values, and the largest element is polynomially bounded in the number of elements to be sorted. It is useful for applications in which the size of the maximum value of element to be sorted is bounded. Following are two types of integer sort:

**COUNTING SORT:**

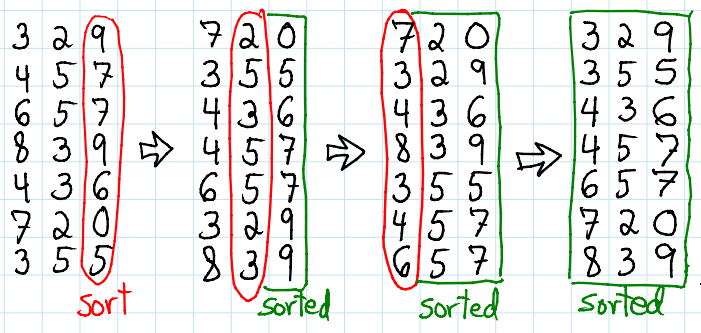
Counting sort is a sorting algorithm that sorts the elements of an array by counting the number of occurrences of each unique element in the array. The count is stored in an auxiliary array and the sorting is done by mapping the count as an index of the auxiliary array. The time complexity of counting sort algorithm is O(n+k) where n is the number of elements in the array and k is the range of the elements. Counting sort is most efficient if the range of input values is not greater than the number of values to be sorted.

Diagram

Description automatically generated with medium confidence

**RADIX SORT:**

Radix sort is an integer sorting algorithm that sorts data with integer keys by grouping the keys by individual digits that share the same significant position and value (place value). Radix sort uses counting sort as a subroutine to sort an array of numbers. Because integers can be used to represent strings, radix sort works on data types other than just integers.

****

**Why bubble sort has O(n) complexity in best case while the selection sort has O(n^2) complexity even in the best case.**

The time complexity of bubble sort in the best case scenario is O(n) because it has to traverse through all the elements once, just to realise that the array is already sorted. While selection sort always takes O(n2) operations, regardless of the characteristics of the data being sorted. Because first N-1 elements has to compare and swap necessarily with all other elements. Even if it's best case, it is not skipping the swapping operation. Swapping is not value based, it is index based. Therefore, it will perform swapping unnecessarily even in best case.