Department of Computer Science and Engineering (Data Science)

Experiment 1

(Divide and Conquer Algorithms)

Aim: Implementations of Quick Sort and Merge Sort.

Theory:

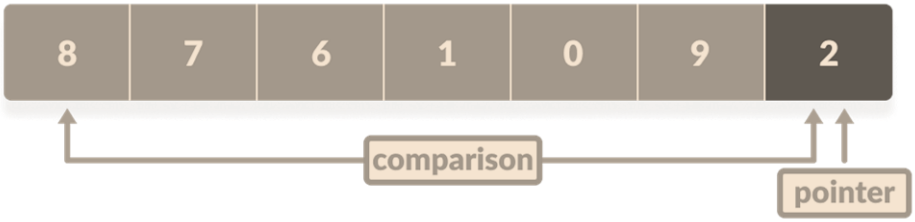
1. Quick Sort:

- Quicksort is a sorting algorithm based on the divide and conquer approach where-
  - An array is divided into subarrays by selecting a pivot element (element selected from the array).
    While dividing the array, the pivot element should be positioned in such a way that elements less than pivot are kept on the left side and elements greater than pivot are on the right side of the pivot.
  - The left and right subarrays are also divided using the same approach. This process continues until each subarray contains a single element.
  - At this point, elements are already sorted. Finally, elements are combined to form a sorted array.

Example:

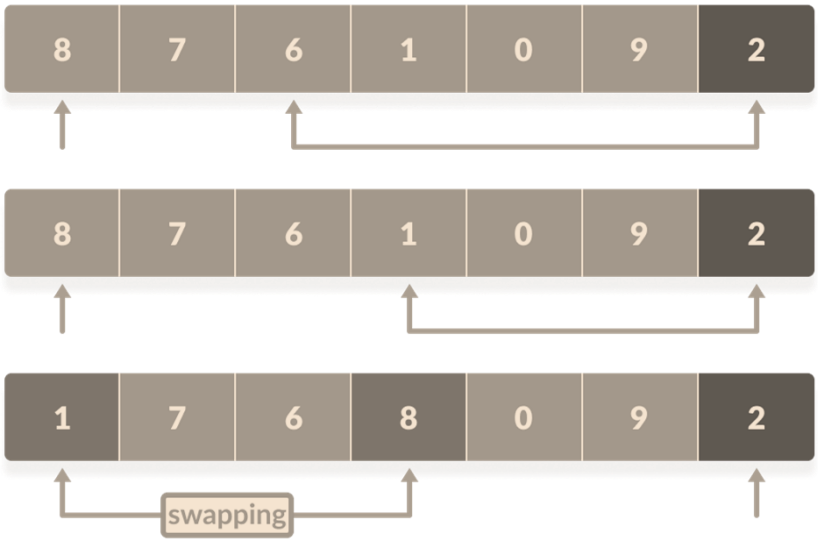
1. **Select the Pivot Element.**
   There are different variations of quicksort where the pivot element is selected from different positions. Here, we will be selecting the rightmost element of the array as the pivot element.
2. **Rearrange the Array.**
   Now the elements of the array are rearranged so that elements that are smaller than the pivot are put on the left and the elements greater than the pivot are put on the right.

Here's how we rearrange the array:
A pointer is fixed at the pivot element. The pivot element is compared with the elements beginning from the first index.

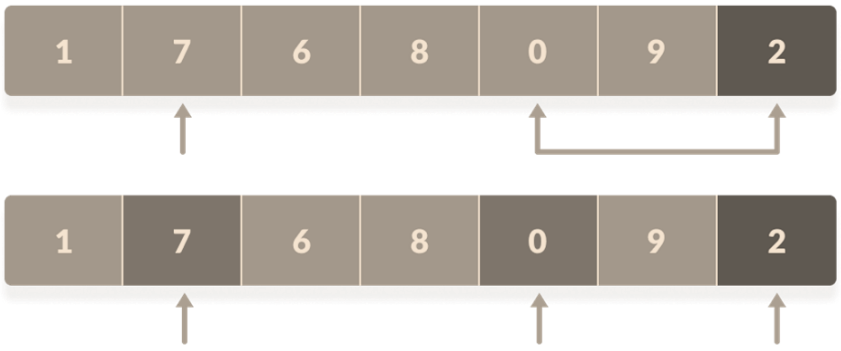


If the element is greater than the pivot element, a second pointer is set for that element.

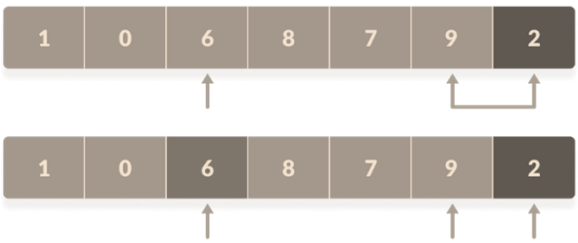
Now, pivot is compared with other elements. If an element smaller than the pivot element is reached, the smaller element is swapped with the greater element found earlier.



Again, the process is repeated to set the next greater element as the second pointer. And, swap it with another smaller element.



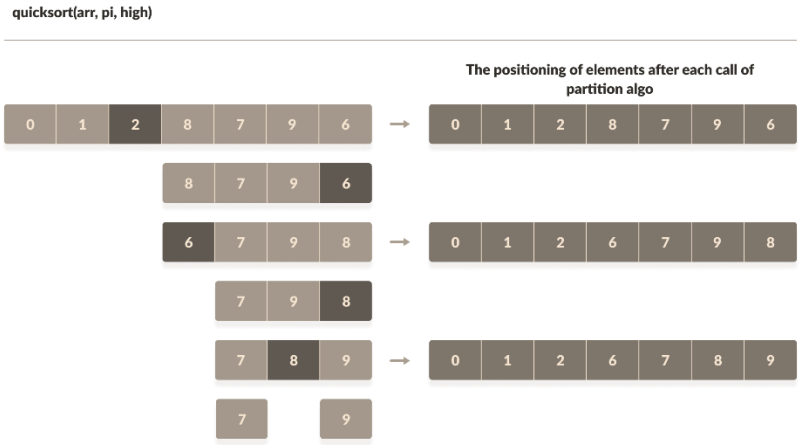
The process goes on until the second last element is reached.



Finally, the pivot element is swapped with the second pointer.

3. **Divide Subarrays.**

Pivot elements are again chosen for the left and the right sub-parts separately. And, step 2 is repeated.



The subarrays are divided until each subarray is formed of a single element. At this point, the array is already sorted.

Algorithm:

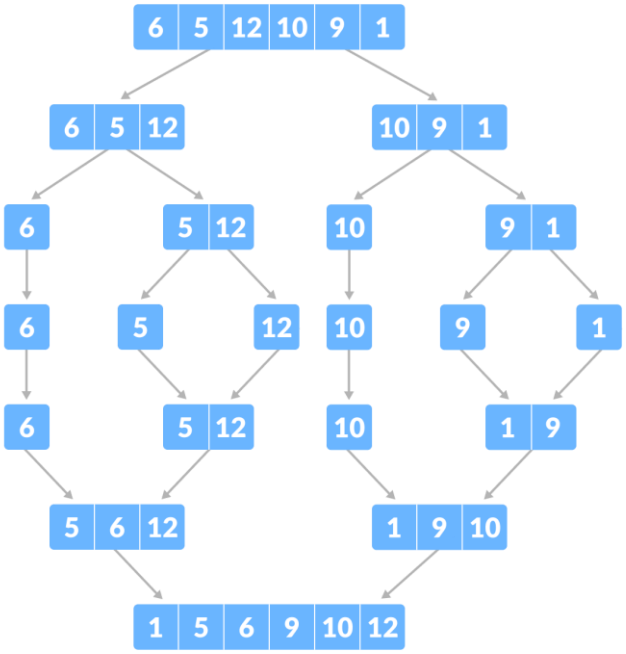
```
quickSort(arr, beg, end)
  if (beg < end)
    pivotIndex = partition(arr,beg, end)
    quickSort(arr, beg, pivotIndex)
    quickSort(arr, pivotIndex + 1, end)

partition(arr, beg, end)
  set end as pivotIndex
  pIndex = beg - 1
  for i = beg to end-1
  if arr[i] < pivot
    swap arr[i] and arr[pIndex]
    pIndex++
  swap pivot and arr[pIndex+1]
return pIndex + 1
```

2. Merge Sort:

- Merge Sort is one of the most popular sorting algorithms that is based on the principle of Divide and Conquer Algorithm.
- Here, a problem is divided into multiple sub-problems. Each sub-problem is solved individually. Finally, sub-problems are combined to form the final solution.

Department of Computer Science and Engineering (Data Science)



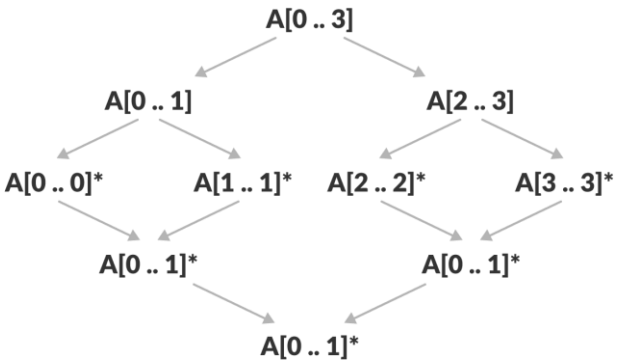
Algorithm:

- The MergeSort function repeatedly divides the array into two halves until we reach a stage where we try to perform MergeSort on a subarray of size 1 i.e. p == r, where p is array starting index and r as ending index.
- After that, the merge function comes into play and combines the sorted arrays into larger arrays until the whole array is merged.

```
MergeSort(A, p, r):
    if p > r
        return
    q = (p+r)/2
    mergeSort(A, p, q)
    mergeSort(A, q+1, r)
    merge(A, p, q, r)
```

- To sort an entire array, we need to call MergeSort(A, 0, length(A)-1).
- As shown in the image below, the merge sort algorithm recursively divides the array into halves until we reach the base case of array with 1 element. After that, the merge function picks up the sorted sub-arrays and merges them to gradually sort the entire array.

Department of Computer Science and Engineering (Data Science)



- **The merge Step of Merge Sort:**
- Every recursive algorithm is dependent on a base case and the ability to combine the results from base cases. Merge sort is no different. The most important part of the merge sort algorithm is the merge step.
- The merge step is the solution to the simple problem of merging two sorted lists(arrays) to build one large sorted list(array).
- The algorithm maintains three pointers, one for each of the two arrays and one for maintaining the current index of the final sorted array.

Have we reached the end of any of the arrays?

No:

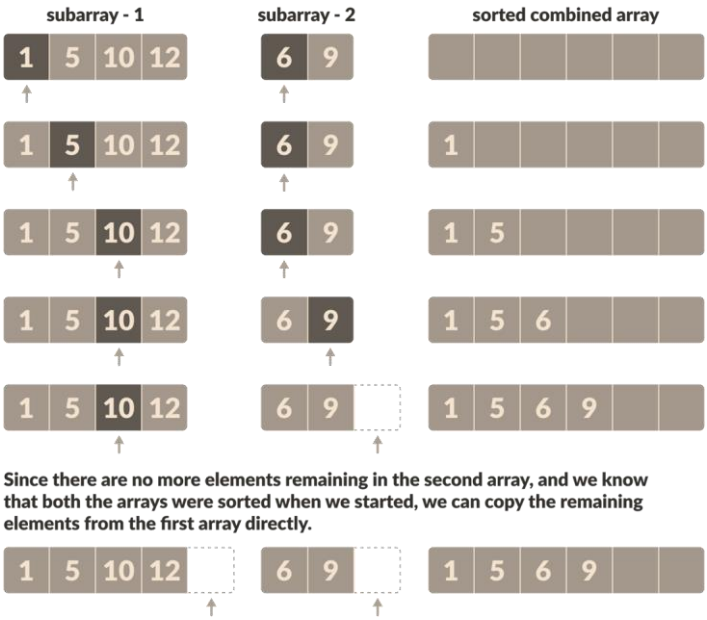
Compare current elements of both arrays

Copy smaller element into sorted array

Move pointer of element containing smaller element

Yes:

Copy all remaining elements of non-empty array



Department of Computer Science and Engineering (Data Science)

Complexity:

Quick Sort:

Time Complexity	
Best	O(n*log n)
Worst	$O(n^2)$
Average	O(n*log n)
Space Complexity	O(log n)

Merge Sort:

Time Complexity	
Best	O(n*log n)
Worst	O(n*log n)
Average	O(n*log n)
Space Complexity	O(n)