Lab Lecture 3

What is Divide and Conquer?

Divide and Conquer is an algorithmic paradigm in which the problem is solved using the Divide, Conquer, and Combine strategy.

A typical Divide and Conquer algorithm solves a problem using the following three steps:

1. **Divide:** This involves dividing the problem into smaller sub-problems.
2. **Conquer:** Solve sub-problems by calling recursively until solved.
3. **Combine:** Combine the sub-problems to get the final solution of the whole problem.

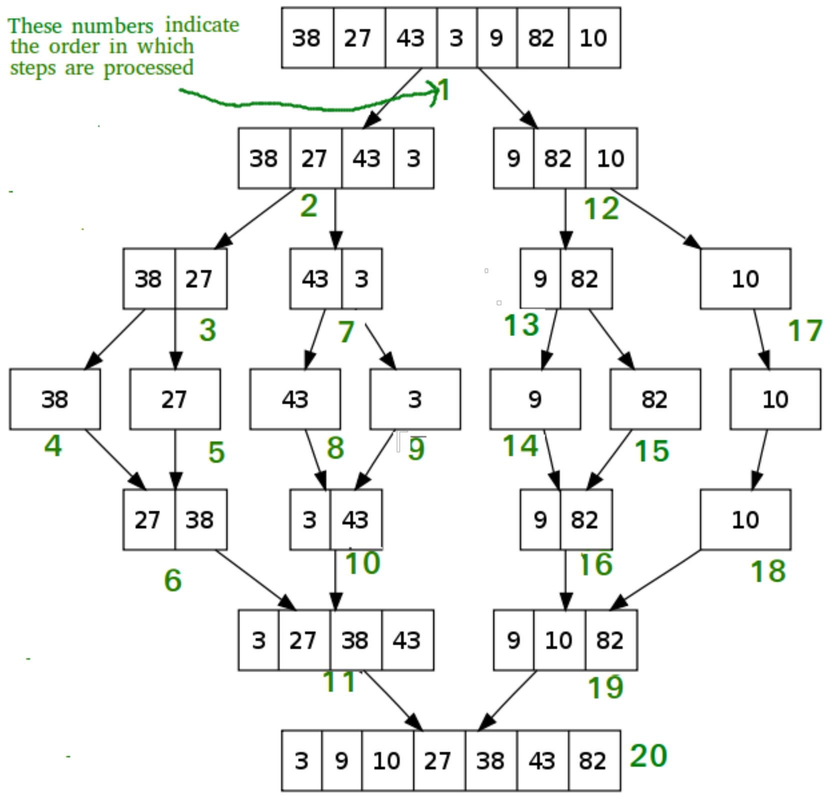
Standard algorithms that follow Divide and Conquer algorithm

The following are some standard algorithms that follow Divide and Conquer algorithm.

1. **Quicksort** is a sorting algorithm. The algorithm picks a pivot element and rearranges the array elements so that all elements smaller than the picked pivot element move to the left side of the pivot, and all greater elements move to the right side. Finally, the algorithm recursively sorts the subarrays on the left and right of the pivot element.
2. **Merge Sort** is also a sorting algorithm. The algorithm divides the array into two halves, recursively sorts them, and finally merges the two sorted halves.
3. **Closest Pair of Points** The problem is to find the closest pair of points in a set of points in the x-y plane. The problem can be solved in O(n^2) time by calculating the distances of every pair of points and comparing the distances to find the minimum. The Divide and Conquer algorithm solves the problem in O(N log N) time.

Example of Divide and Conquer algorithm

A classic example of Divide and Conquer is Merge Sort demonstrated below. In Merge Sort, we divide array into two halves, sort the two halves recursively, and then merge the sorted halves.



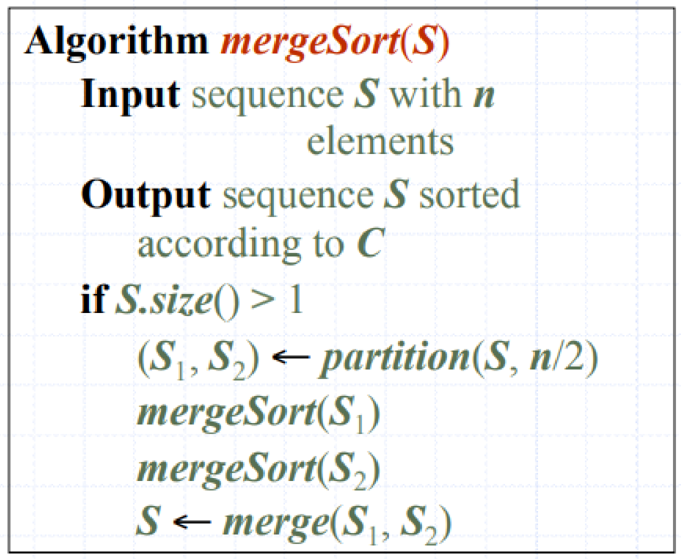
The Merge-Sort Algorithm

Merge-sort on an input sequence S with n elements consists of three steps:

Divide: partition S into two sequences S1 and S2 of about n/2 elements each

Recursion: recursively sort S1 and S2

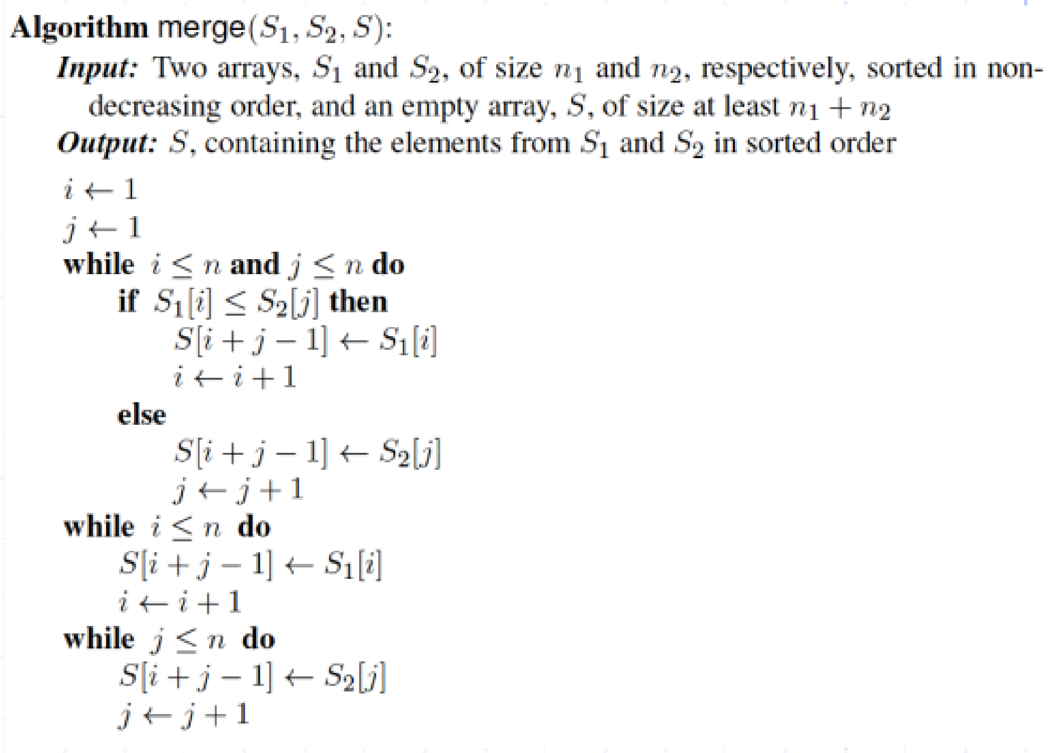
Conquer: merge S1 and S2 into a unique sorted sequence



```
Algorithm mergeSort(S)
    Input sequence S with n
                        elements
    Output sequence S sorted
        according to C
    if S.size() > 1
        $(S_1, S_2) \leftarrow partition(S, n/2)$
        $mergeSort(S_1)$
        $mergeSort(S_2)$
        $S \leftarrow merge(S_1, S_2)$
```

Merging Two Sorted Sequences

- ✓ The conquer step of merge-sort consists of merging two sorted sequences A and B into a sorted sequence S containing the union of the elements of A and B
- ✓ Merging two sorted sequences, each with n/2 elements and implemented by means of a doubly linked list, takes O(n) time



```
Algorithm merge($S_1, S_2, S$):
    Input: Two arrays, $S_1$ and $S_2$, of size $n_1$ and $n_2$, respectively, sorted in non-
      decreasing order, and an empty array, $S$, of size at least $n_1 + n_2$
    Output: $S$, containing the elements from $S_1$ and $S_2$ in sorted order
    $i \leftarrow 1$
    $j \leftarrow 1$
    while $i \leq n$ and $j \leq n$ do
        if $S_1[i] \leq S_2[j]$ then
            $S[i + j - 1] \leftarrow S_1[i]$
            $i \leftarrow i + 1$
        else
            $S[i + j - 1] \leftarrow S_2[j]$
            $j \leftarrow j + 1$
    while $i \leq n$ do
        $S[i + j - 1] \leftarrow S_1[i]$
        $i \leftarrow i + 1$
    while $j \leq n$ do
        $S[i + j - 1] \leftarrow S_2[j]$
        $j \leftarrow j + 1$
```