Divide and Conquer

Divide and Conquer

- It refers to a class of algorithmic techniques in which one breaks the input into several parts, solves the problem in each part recursively, and then combines the solutions to these subproblems into an overall solution.
- In many cases, it can be a simple and powerful method.

Divide and Conquer

- Analyzing the running time of a divide and conquer algorithm involves solving a recurrence relation that bounds the running time recursively in terms of the running time on smaller instances.

Divide and Conquer

- Applications to a number of different domains:
  - computing a distance function on different rankings of a set of objects;
  - finding the closest pair of points in the plane;
  - multiplying two integers;
  - smoothing a noisy signal.
- It is a method that often works well when combined with other algorithm design techniques.
- For example, it combined with dynamic programming to produce a space-efficient solution to a fundamental sequence comparison problem, and it is combined with randomization to yield a simple and efficient algorithm for computing the median of a set of numbers.

Divide and Conquer

- The natural brute-force algorithm may already be polynomial time, and the divide and conquer strategy is serving to reduce the running time to a lower polynomial.
- Ex: for most of the problems, brute force was exponential and the goal in designing a more sophisticated algorithm was to achieve any kind of polynomial running time.

Divide and Conquer

- The natural brute-force algorithm for finding the closest pair among n points in the plane would simply measure all $O(n^2)$ distances, for a (polynomial) running time of $O(n^2)$.
- Using divide and conquer, we will improve the running time to $O(n \log n)$.

Mergesort Algorithm

- A First Recurrence: The Merge sort Algorithm
- Unrolling the Merge sort Recurrence

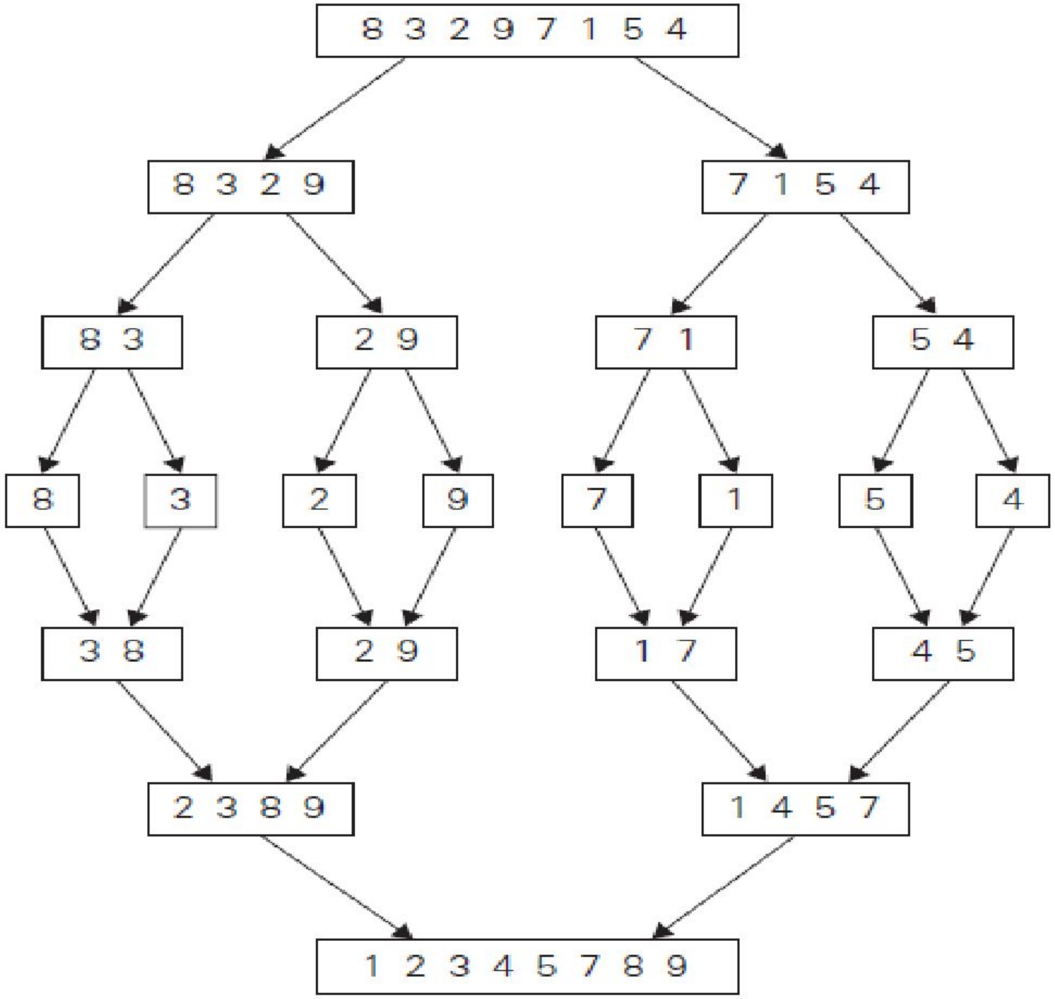
Mergesort Algorithm

- It sorts a given list of numbers by first dividing them into two equal halves, sorting each half separately by recursion, and then combining the results of these recursive calls—in the form of the two sorted halves—using the linear-time algorithm for merging sorted lists

Mergesort Algorithm

- Divide the input into two pieces of equal size; solve the two subproblems on these pieces separately by recursion; and then combine the two results into an overall solution, spending only linear time for the initial division and final recombining.
- Once the input has been reduced to size 2, we stop the recursion and sort the two elements by simply comparing them to each other.

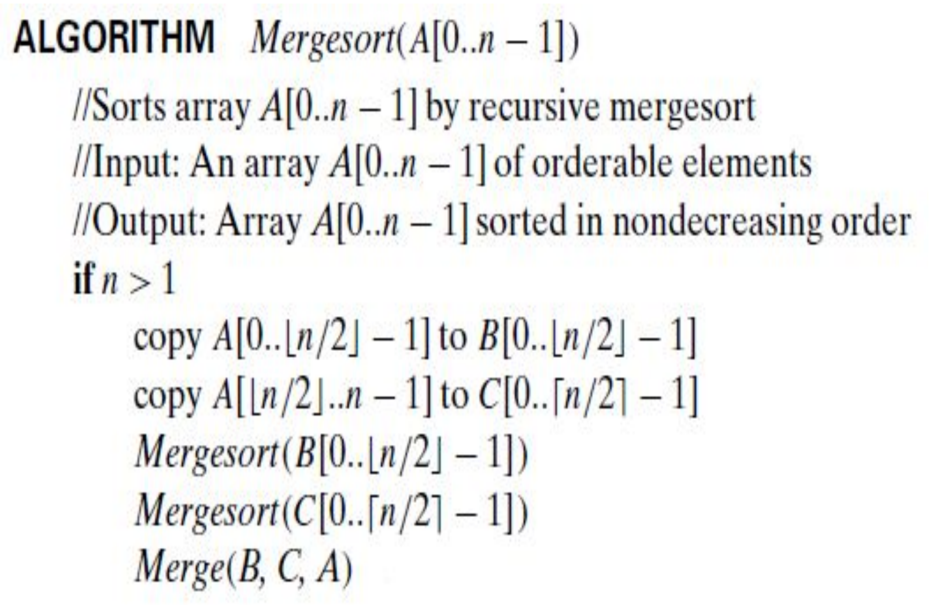
Mergesort Algorithm



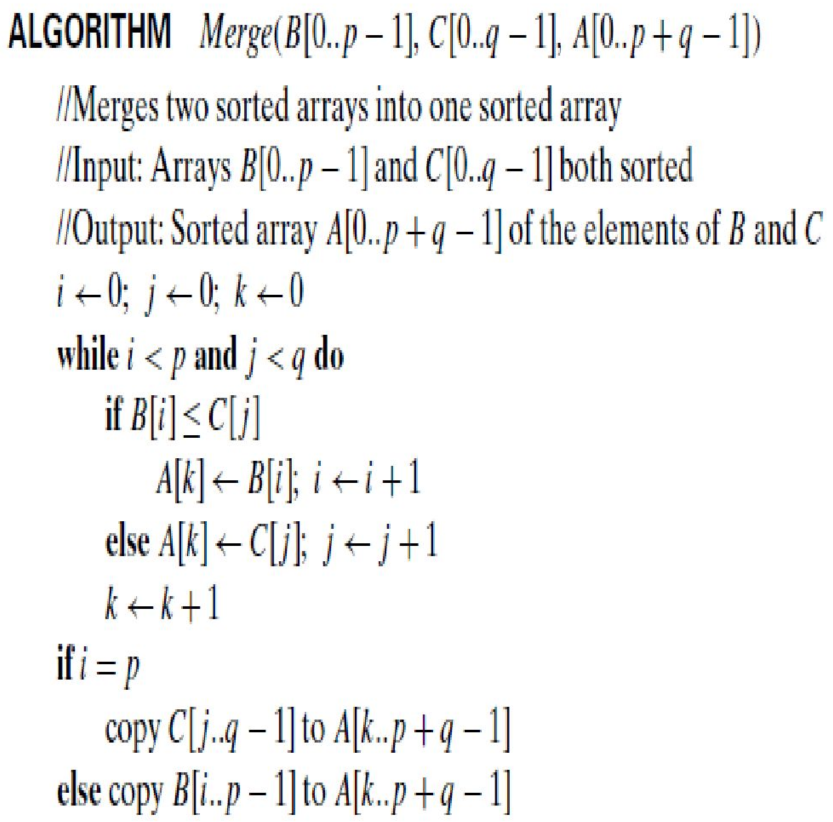
Mergesort Algorithm

- Recurrence solving is a task that has been incorporated into a number of standard computer algebra systems, and the solution to many standard recurrences can now be found by automated means.
- To understand the process of solving recurrences and to recognize which recurrences lead to good running times, since the design of an efficient divide-and-conquer algorithm is heavily intertwined with an understanding of how a recurrence relation determines a running time.

Mergesort Algorithm



```
ALGORITHM  Mergesort(A[0..n − 1])
    //Sorts array A[0..n − 1] by recursive mergesort
    //Input: An array A[0..n − 1] of orderable elements
    //Output: Array A[0..n − 1] sorted in nondecreasing order
    if n > 1
        copy A[0..⌊n/2⌋ − 1] to B[0..⌊n/2⌋ − 1]
        copy A[⌊n/2⌋..n − 1] to C[0..⌈n/2⌉ − 1]
        Mergesort(B[0..⌊n/2⌋ − 1])
        Mergesort(C[0..⌈n/2⌉ − 1])
        Merge(B, C, A)
```



```
ALGORITHM  Merge(B[0..p − 1], C[0..q − 1], A[0..p + q − 1])
    //Merges two sorted arrays into one sorted array
    //Input: Arrays B[0..p − 1] and C[0..q − 1] both sorted
    //Output: Sorted array A[0..p + q − 1] of the elements of B and C
    i ← 0; j ← 0; k ← 0
    while i < p and j < q do
        if B[i] ≤ C[j]
            A[k] ← B[i]; i ← i + 1
        else A[k] ← C[j]; j ← j + 1
        k ← k + 1
    if i = p
        copy C[j..q − 1] to A[k..p + q − 1]
    else copy B[i..p − 1] to A[k..p + q − 1]
```

Analysis of Mergesort Algorithm

- Solved

(5.1) *For some constant c,*

$$T(n) \leq 2T(n/2) + cn$$

when $n > 2$, and

$$T(2) \leq c.$$