Divide and Conquer Algorithm

A **divide and conquer algorithm** is a strategy of solving a large problem by

1. breaking the problem into smaller sub-problems
2. solving the sub-problems, and
3. combining them to get the desired output.

To use the divide and conquer algorithm, **recursion** is used.

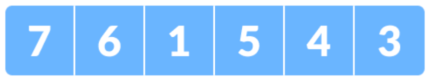
How Divide and Conquer Algorithms Work?

Here are the steps involved:

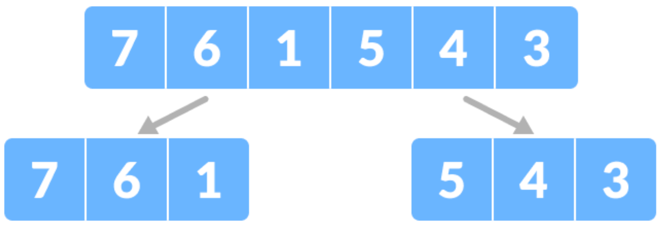
1. **Divide**: Divide the given problem into sub-problems using recursion.
2. **Conquer**: Solve the smaller sub-problems recursively. If the subproblem is small enough, then solve it directly.
3. **Combine:** Combine the solutions of the sub-problems that are part of the recursive process to solve the actual problem.

Let us understand this concept with the help of an example.

Here, we will sort an array using the divide and conquer approach (ie. merge sort).

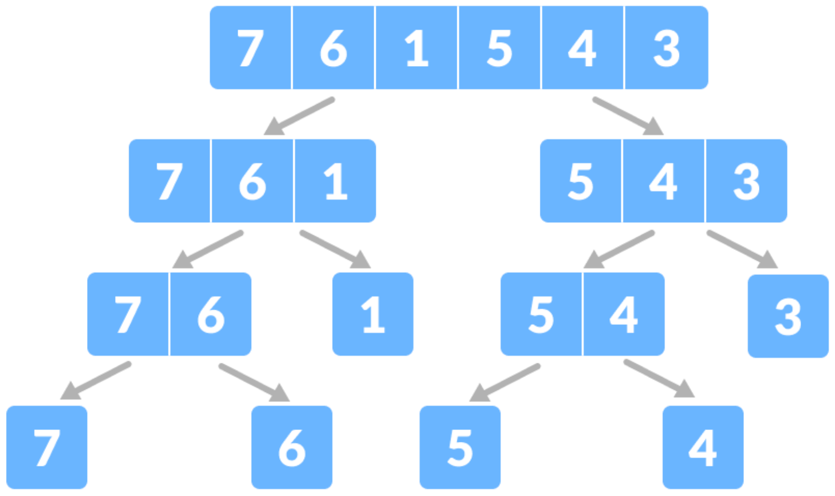


1. Let the given array be:
   Array for merge sort
2. **Divide** the array into two halves.

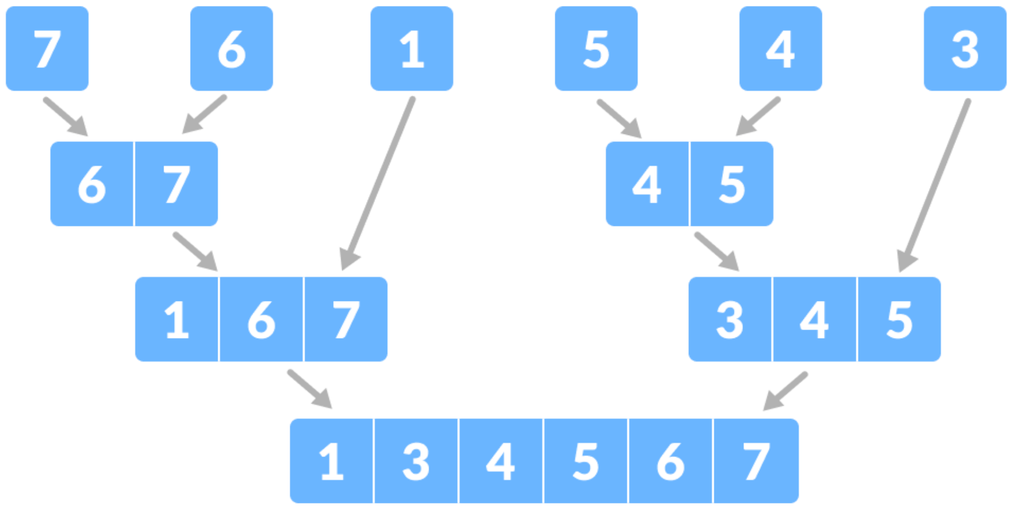


3. Divide the array into two subparts

4. Again, divide each subpart recursively into two halves until you get individual elements.



5. Divide the array into smaller subparts
6. Now, combine the individual elements in a sorted manner. Here, **conquer** and **combine** steps go side by side.



7. Combine the subparts

The divide and conquer approach divides a problem into smaller subproblems; these subproblems are further solved recursively. The result of each subproblem is not stored for future reference, whereas, in a dynamic approach, the result of each subproblem is stored for future reference.

Use the divide and conquer approach when the same subproblem is not solved multiple times. Use the dynamic approach when the result of a subproblem is to be used multiple times in the future.

Advantages of Divide and Conquer Algorithm

- The complexity for the multiplication of two matrices using the naive method is $O(n^3)$, whereas using the divide and conquer approach (i.e. Strassen's matrix multiplication) is $O(n^{2.8074})$. This approach also simplifies other problems, such as the Tower of Hanoi.
- This approach is suitable for multiprocessing systems.
- It makes efficient use of memory caches.

Divide and Conquer Applications

- Binary Search
- Merge Sort
- Quick Sort
- Strassen's Matrix multiplication
- Karatsuba 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.
4. **Strassen's Algorithm** is an efficient algorithm to multiply two matrices. A simple method to multiply two matrices needs 3 nested loops and is O(n^3). Strassen's algorithm multiplies two matrices in O(n^2.8974) time.
5. **Cooley–Tukey Fast Fourier Transform (FFT) algorithm** is the most common algorithm for FFT. It is a divide and conquer algorithm which works in O(N log N) time.
6. **Karatsuba algorithm for fast multiplication** does the multiplication of two *n*-digit numbers in at most

Advantages of Divide and Conquer Algorithm:

- The difficult problem can be solved easily.
- It divides the entire problem into subproblems thus it can be solved parallelly ensuring multiprocessing
- Efficiently uses cache memory without occupying much space
- Reduces time complexity of the problem
- **Solving difficult problems:** Divide and conquer technique is a tool for solving difficult problems conceptually. e.g. Tower of Hanoi puzzle. It requires a way of breaking the problem into sub-problems, and solving all of them as an individual cases and then combining sub- problems to the original problem.
- **Algorithm efficiency:** The divide-and-conquer paradigm often helps in the discovery of efficient algorithms. It was the key, for example to the Quicksort and Mergesort algorithms, and fast Fourier transforms. In all these examples, the D and C approach led to an improvement in the asymptotic cost of the solution. In particular, if the base cases have constant-bounded size, the work of splitting the problem and combining the partial solutions is proportional to the problem's size n, and there are a bounded number p of subproblems of size-n/p at each stage, then the cost of the divide-and-conquer algorithm will be O(n log n).
- **Parallelism:** Normally DAC algorithms are used in multi-processor machines having shared-memory systems where the communication of data between processors does not need to be planned in advance, because distinct sub-problems can be executed on different processors.
- **Memory access:** These algorithms naturally make an efficient use of memory caches. Since the subproblems are small enough to be solved in cache without using the main memory that is slower one. Any algorithm that uses cache efficiently is called cache oblivious.
- **Roundoff control:** In computations with rounded arithmetic, e.g. with floating point numbers, a divide-and-conquer algorithm may yield more accurate results than a superficially equivalent iterative method. For example, one can add N numbers either by a simple loop that adds each datum to a single variable, or by a D And C algorithm that breaks the data set into two halves, recursively computes the sum of each half, and then adds the two sums. While the second method performs the same number of additions as the first, and pays the overhead of the recursive calls, it is usually more accurate.

Disadvantages of Divide and Conquer Algorithm:

- It involves recursion which is sometimes slow
- Efficiency depends on the implementation of logic
- It may crash the system if the recursion is performed rigorously.
- **Overhead:** The process of dividing the problem into subproblems and then combining the solutions can require additional time and resources. This overhead can be significant for problems that are already relatively small or that have a simple solution.

- **Complexity:** Dividing a problem into smaller subproblems can increase the complexity of the overall solution. This is particularly true when the subproblems are interdependent and must be solved in a specific order.
- **Difficulty of implementation:** Some problems are difficult to divide into smaller subproblems or require a complex algorithm to do so. In these cases, it can be challenging to implement a divide and conquer solution.
- **Memory limitations:** When working with large data sets, the memory requirements for storing the intermediate results of the subproblems can become a limiting factor.
- **Suboptimal solutions:** Depending on how the subproblems are defined and how they are combined, a divide and conquer solution may not always produce the most optimal solution. In some cases, it may be necessary to apply additional optimization techniques to improve the final solution.
- **Difficulty in parallelization:** In some cases, dividing the problem into subproblems and solving them independently may not be easily parallelizable, leading to inefficient use of computational resources.

Divide and Conquer is a popular algorithmic technique in computer science that involves breaking down a problem into smaller sub-problems, solving each sub-problem independently, and then combining the solutions to the sub-problems to solve the original problem. The basic idea behind this technique is to divide a problem into smaller, more manageable sub-problems that can be solved more easily.

The divide and conquer algorithm typically consists of three steps:

1. Divide: The problem is divided into smaller sub-problems that are similar to the original problem but of smaller size. The division is done until the sub-problems become small enough to be solved directly.
2. Conquer: The sub-problems are solved recursively using the same algorithm until they become simple enough to be solved directly.
3. Combine: The solutions to the sub-problems are combined to solve the original problem.
4. This technique is used in many algorithms, including quicksort, mergesort, binary search, and Karatsuba's algorithm for multiplying large integers.

The advantages of using the divide and conquer algorithm are:

1. It is easy to implement and understand.
2. It reduces the time complexity of the algorithm by breaking down the problem into smaller sub-problems.
3. It can be used to solve a wide range of problems, including sorting, searching, and optimization.
4. It can be used in parallel computing to improve performance.

The disadvantages of using the divide and conquer algorithm are:

1. It can be difficult to determine the base case or stopping condition for the recursive calls.
2. It may not be the most efficient algorithm for all problems.
3. It may require more memory than other algorithms because it involves storing intermediate results.

Applications

● Merge Sort and Quicksort

● Median Finding

● Min and Max finding

● Matrix Multiplication

● Closest Pair problem