Decrease and conquer

Math recap

Let's take a simple math problem of adding four numbers together:

$$1 + 2 + 3 + 4 = ?$$

Here, you can start by adding up the first numbers, then add the sum to the next number, and so eventually you'll get the total:

$$1+ 2 + 3 + 4 = (1 + 2 + 3) + 4 = ((1 + 2) + 3) + 4 = (3 + 3) + 4 = 6 + 4 = 10$$

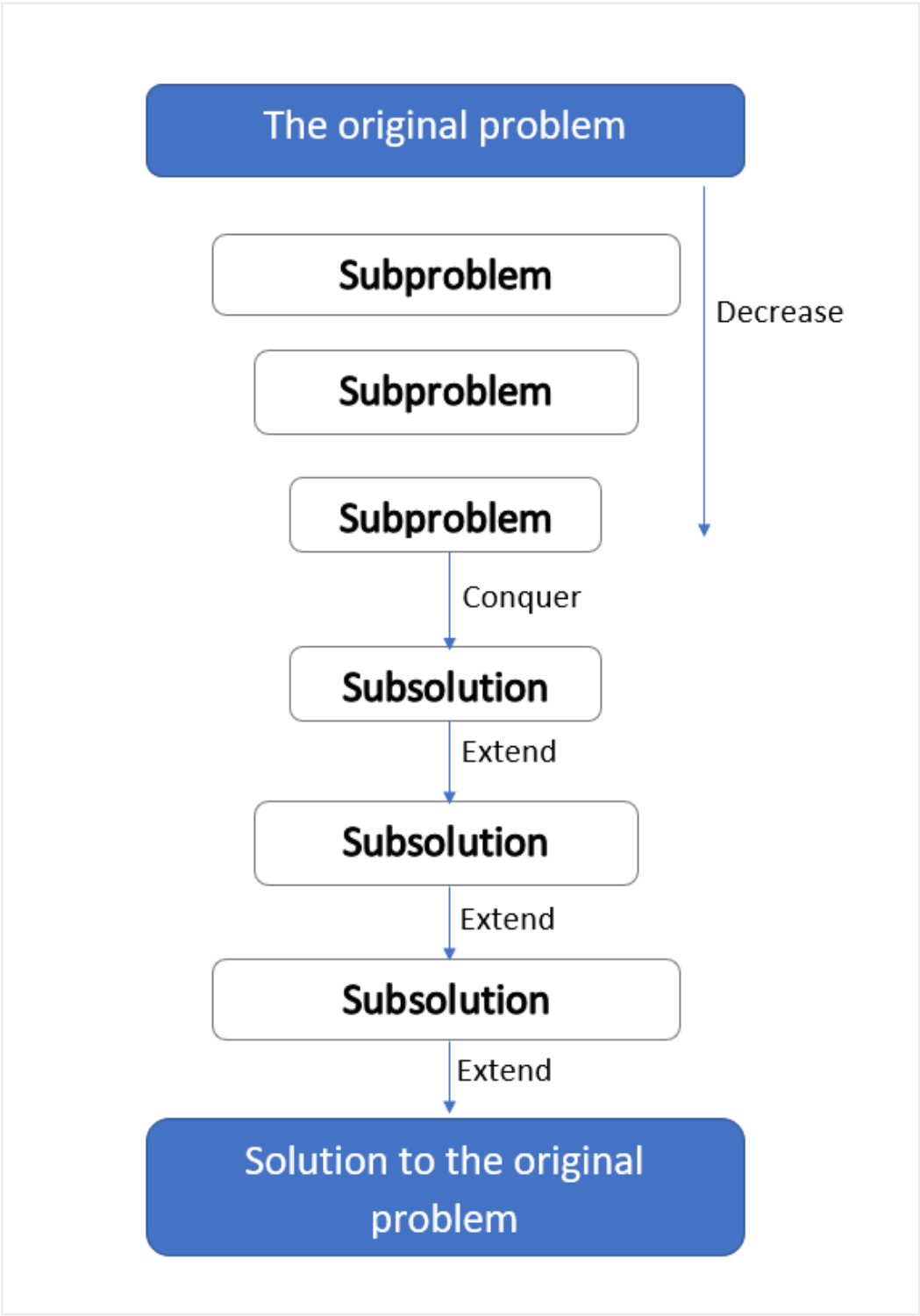
This process of reducing a problem to a subproblem and solving it one step at a time is the core concept of the decrease-and-conquer algorithm.

Decrease-and-conquer is an algorithm design paradigm in which a problem is transformed into a smaller subproblem and then that subproblem is solved first. Unlike the case of Divide and conquer algorithms, the reduction process here generates a single subproblem. The transformation is applied either iteratively or recursively until the sub-problem becomes simple enough to be solved directly as a base case. Finally, the solutions of all sub-problems are extended to get a solution to the original problem.

The steps of the decrease-and-conquer algorithm

Generally, the decrease-and-conquer approach implies the following three steps:

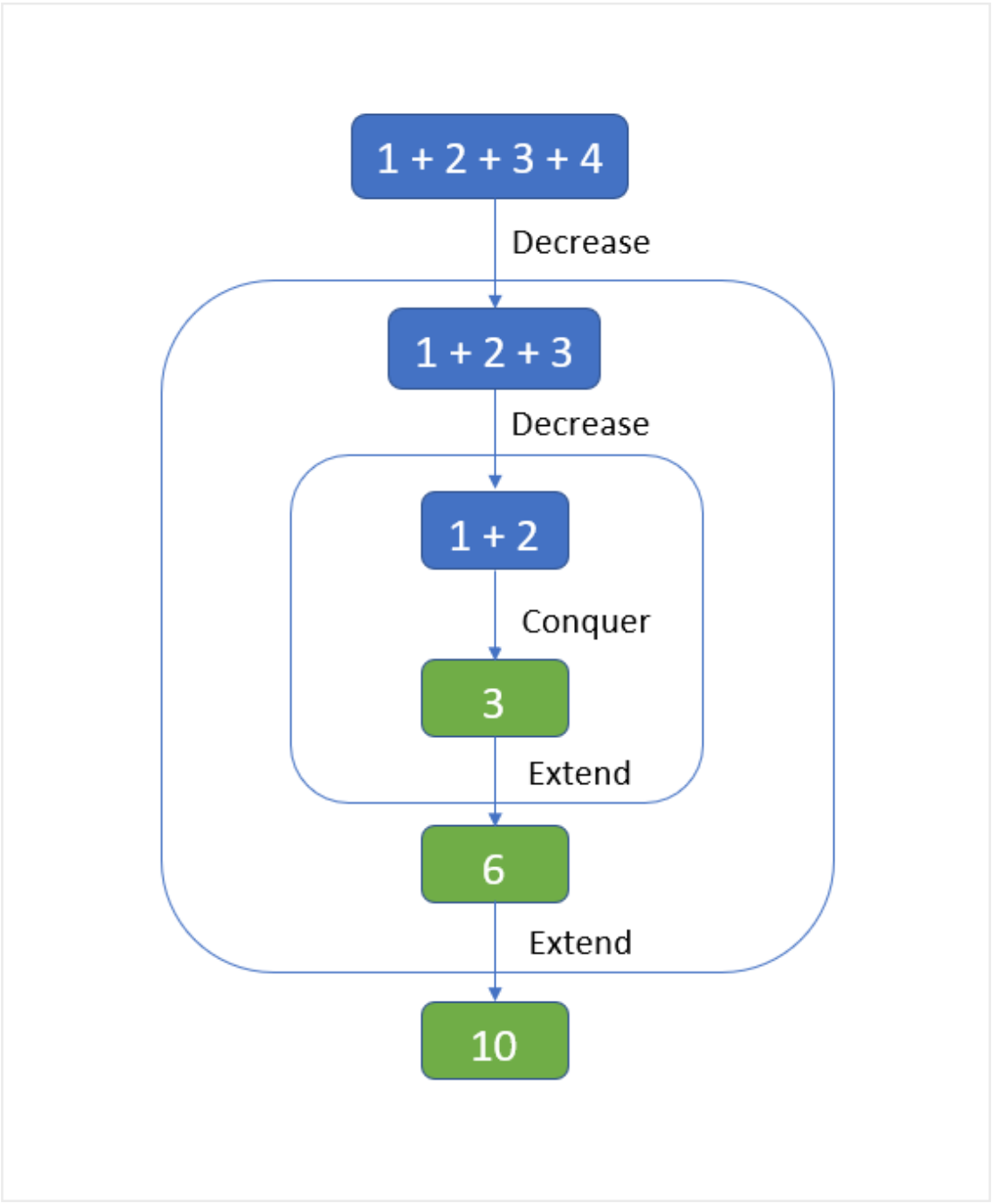
1. Decrease: reduce a problem to a smaller instance of the same problem.
2. Conquer: iteratively or recursively solve the sub-problem.
3. Extend: apply the sub-problem solution to the next sub-problem to get a solution to the original problem.



As shown above, we first reduce the original problem to a smaller sub-problem. Then we move on to the conquer step by solving that single subproblem. Then, we extend that sub-solution to another subproblem in order to obtain a subsolution to that subproblem. This process continues until we have a solution to the original problem.

A simple example: the sum of elements in an array
Let's see how the decrease-and-conquer-based algorithm can solve the

problem of finding the total sum of numbers in an array.
The first step of this algorithm is to decrease the original problem to a smaller subproblem. So, we reduce the problem to subproblem recursion. We slice the array one element at a time and pass the sliced array to the same function, thereby decreasing the problem by a single constant. The base case for this would be the case when the length of the array is one: in this case, the sum of the subarray with one element is simply the value of that element. Then we conquer and extend the sub-solution to the next subproblem by adding one element that is returned by the recursion at a time. An illustration of this algorithm is as follows:



As shown above, given an array of four numbers, the algorithm decreases the problem by slicing the array down to one single element, namely 1. Then, 1 is added to 2, and this sub-total is extended to the next addition (plus 3). The process is continued until the total sum of four numbers is calculated.

Examples of decrease-and-conquer-based algorithms

Decrease-and-conquer-based algorithms are widely applied in many real-world problems. Kahn's algorithm and binary search are just two examples of the algorithm's efficiency in data sorting and searching. As you go further into algorithm topics, you will definitely get to know them.

Conclusion

In this topic, you've learned about the decrease-and-conquer algorithm design paradigm:

- The three steps of decrease-and-conquer-based algorithms are: decrease, conquer, and extend.
- Unlike the case of divide-and-conquer paradigms, decrease-and-conquer reduction generates one subproblem.
- With the decrease-and-conquer-based approach, you can solve a problem either iteratively or recursively.
- There are many use cases of decrease-and-conquer-based algorithms, so you may soon find yourself applying such algorithm design, too!