Recursion in Practice

Recursion is a powerful technique used in programming to solve complex problems. It is a process where a function calls itself until a base condition is met. Recursion has its place in programming but is not always necessary. **For any recursive solutions, there is usually a non-recursive approach that solves the problem with matching or better time efficiency.**

Recursion is particularly useful in solving problems that involve dividing a big problem into smaller sub-problems. These sub-problems are solved recursively, and their solutions are combined to produce the final result. Examples of problems that can be solved using recursion include factorial computation, and the Fibonacci sequence.

Factorial computation involves multiplying a number with all the numbers that come before it. Recursion can be used to implement this by defining the base case as the factorial of 0 which is 1, and the recursive case as the factorial of n which is n multiplied by the factorial of n-1.

The Fibonacci sequence is a series of numbers in which each number is the sum of the two preceding numbers. Recursion can be used to implement this by defining the base case as the first two numbers of the sequence, and the recursive case as the sum of the previous two numbers in the sequence.

General guidelines for using recursion are as follows: do NOT use recursion if a solution can be met using an iterative approach that is developed easily and concisely. There is NO need to use recursion in processing a linear collection or for a nonlinear structure. Do NOT use recursion in a situation where something is repeatedly computed. Use recursion ONLY if time efficiency matches the time of an iterative approach and results in an easier and more concise solution. USE recursion in a divide and conquer approach that cannot be easily approached with an iterative approach. USE recursion in tasks that require processing (traversal, nonlinear structures, tree structure, etc.).

There is also a very strong relationship between recursion and mathematical induction. Induction is often employed to prove properties about recursive algorithms. Such an example is the n-queens problem. The n-queens problem is a classic problem in Comp Sci that's expanded on in the textbook. The problem involves placing n queens on an n x n chessboard in such a way that no two queens can attack each other. Specifically, no two queens should be placed in the same row, column, or diagonal. The n-queens problem can be solved using a recursion.

While recursion is a powerful technique, it may not always be the best solution for certain problems. Iterative solutions are preferred when the problem can be solved using a loop, and the input size is large. This is because recursion involves the creation of multiple function calls and maintaining a stack, which can lead to stack overflow errors for large input sizes. Examples of problems that should be solved iteratively include traversing a large tree structure, sorting an array using a loop, and searching for an element in an unsorted array.