In the *Towers of Hanoi* problem, we can understand how the program parses the task at hand, moving a stack of *n* disks onto another pole, by understanding the human tendency to solve this problem. To start a new stack of disks on a pole, the largest disk must be the first to make that pile. Once the initial pole has been shifted to allow this move, the same task must occur again, in that the next largest disk must be placed on the new pole before any future placements are made. In this way, recursion seems to be the most logical way to solve the problem, as it incorporates two essential parts in its own calculations.

Firstly, it allows for efficient repetition of the code sequence. The function can call itself any number of times until the problem is marked as solved, when all the disks are moved to the next pole. The information about the placement of each disk is updated at the end of each movement, so that the next call of the function will have the truth values for each disk to perform the next movement.

Secondly, the code sifts through each disk by checking its placement amongst the other disks in the problem by keeping track of the moves it made before the current movement. As each function call is nested within another, the code provisions each disk to be moved multiple times iteratively, depending on whether it is in the proper place at the right time. The tree of function calls continue until the disk is in the right spot, at which the code jumps out of that tree and moves on to another disk to be moved.