Restricted Tower of Hanoi Analysis

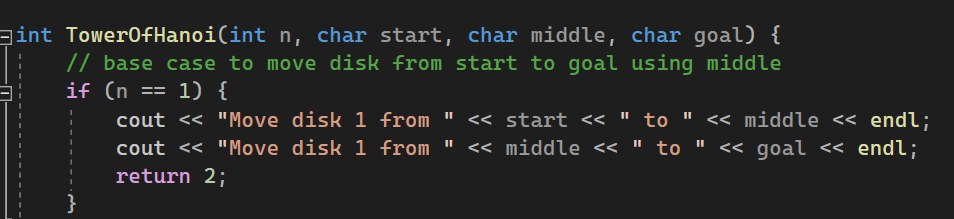
Quick introduction:-

In my program I implemented a different variation of the normal Tower of Hanoi problem using recursion. In this version, any move must go through the middle peg (B), which means that the disks cannot move directly between peg A and peg C.

Another difference between this version and the normal tower of Hanoi problem is that the total moves for the normal version is: 2ⁿ - 1. For the restricted tower of Hanoi the total moves is: 3ⁿ - 1.

Code Explanation:-

For the code I have a base case, which if there is only one disk it moves it from peg A to peg B, then from peg B to peg C in two moves. This makes sure that the middle peg must be used.



Then there is the recursion part of the code where the number of disks (n) is greater than one:  
 - Move (n−1) disks from peg A to peg C, passing through peg B.

  
 - Move the largest disk (n) from peg A to peg B.

  
 - Move (n−1) disks back from peg C to peg A, passing through peg B.

  
 - Move the largest disk (n) from peg B to peg C. (reached the goal)

  
 - Move (n−1) disks from peg A to peg C, passing through peg B. (reaching the goal)



Complexity Analysis:-

Time Complexity: O(3ⁿ), we know this from the number of moves (3ⁿ - 1). This is much worse than the time complexity of the normal tower of Hanoi problem which is: O(2ⁿ). I implemented it using recursion, however if it was done iteratively, it would still have had the same time complexity as the same number of moves would still be required.

Summary:-

Algorithm: Restricted tower of Hanoi using recursion.  
Formula: T(n) = 3T(n-1) + 2 = 3ⁿ - 1   
Moves Required: 3^n - 1  
Time Complexity: O(3^n)  
Difference from Standard Hanoi: All moves must pass through peg B, which increases total moves compared to the normal version, and increases the time drastically especially when we increase the number of disks.

GitHub Link:-

<https://github.com/Abdulrahman-Elzemety/Task-2-Tower-of-Hanoi-/tree/main>