Comparing Complexities
Between Recursive And Stack Algorithm

Stack algorithm:

This algorithm makes use of dynamically allocated memory and actually move legitimate data within the memory to solve the puzzle. The stack.c defines a structure called Stack and its helper functions. Here's a link to help you understand what a stack is: https://www.cs.cmu.edu/~adamchik/15-121/lectures/Stacks%20and%20Queues/Stacks%20and%20Queues.html
In this program, I made allocated space in memory for a stack to store integer type items which will be the disks. The pattern of the solution is well explained in the DESIGN pdf under the solve_stack function.

Recursive Function:

This solution uses recursion to solve the puzzle. The base case is that when there is only 1 disk, we print to move it from source peg to destination peg. Else we call the function to move n-1 disks from source peg to auxiliary peg then from aux to destination peg. The pattern for the recursive function is also explained in the DESIGN pdf.

Comparison:

The stack solution is easily a very inefficient solution and a long solution compared to the recursive one. This is easily seen in the comparison in the No. of clicks it takes to execute both solutions. Let's set the disk number to 8. Below is the console output for the recursive solution. It takes around 0.00484 sec to execute.

```
Move disk 1 from peg C to peg B
Move disk 2 from peg C to peg A
Move disk 1 from peg B to peg A
Move disk 5 from peg C to peg B
Move disk 1 from peg A to peg C
Move disk 2 from peg A to peg B
Move disk 1 from peg C to peg B
Move disk 3 from peg A to peg C
Move disk 1 from peg B to peg A
Move disk 2 from peg B to peg C
Move disk 1 from peg A to peg C
Move disk 4 from peg A to peg B
Move disk 1 from peg C to peg B
Move disk 2 from peg C to peg A
Move disk 1 from peg B to peg A
Move disk 3 from peg C to peg B
Move disk 1 from peg A to peg C
Move disk 2 from peg A to peg B
Move disk 1 from peg C to peg B
No. of clicks 484 clicks (0.000484 seconds).
```

```
Move disk 1 from peg B to peg C.
Move disk 2 from peg B to peg A.
Move disk 1 from peg C to peg A.
Move disk 5 from peg B to peg C.
Move disk 1 from peg A to peg B.
Move disk 2 from peg A to peg C.
Move disk 1 from peg B to peg C.
Move disk 3 from peg A to peg B.
Move disk 1 from peg C to peg A.
Move disk 2 from peg C to peg B.
Move disk 1 from peg A to peg B.
Move disk 4 from peg A to peg C.
Move disk 1 from peg B to peg C.
Move disk 2 from peg B to peg A.
Move disk 1 from peg C to peg A.
Move disk 3 from peg B to peg C.
Move disk 1 from peg A to peg B.
Move disk 2 from peg A to peg C.
Move disk 1 from peg B to peg C.
No. of clicks 541 clicks (0.000541 seconds).
```

The second, or the image above, is the console output when running the stack solution. As we can see, it takes 0.000541s to execute which is a bit more than the recursive function. This could be explained by the fact that the stack solution has more loops, conditionals, and calls to other functions like stack_create() which costs more cycles from all the jumping and stack pushing and popping. So it requires more memory allocation and a longer time to execute making it an inefficient solution in comparison.