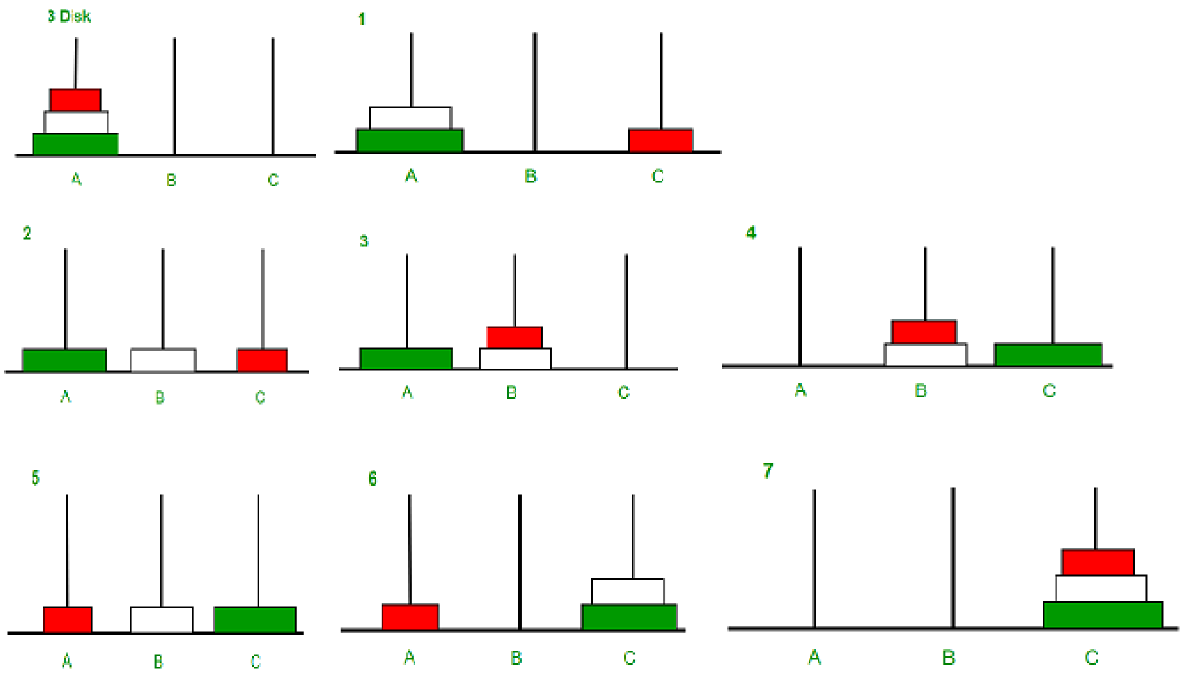
Tower of Hanoi (Strategy: Recursion)

The **Tower of Hanoi** is a classic problem that can be elegantly solved using recursion. The problem involves three rods and a number of disks of different sizes which can slide onto any rod. The puzzle starts with the disks neatly stacked in ascending order of size on one rod, the smallest at the top, making a conical shape.

The objective is to move the entire stack to another rod, obeying the following simple rules:

1. Only one disk can be moved at a time.
2. Each move consists of taking the top disk from one of the stacks and placing it on top of another stack or on an empty rod.
3. No disk may be placed on top of a smaller disk.



Here is a detailed step-by-step explanation of how to solve the Tower of Hanoi problem using recursion, along with Java code that illustrates each iteration.

Step-by-Step Explanation

1. **Understand the Base Case**:
   - If there is only one disk, move it directly from the source rod to the target rod.
2. **Recursive Case**:
   - To move n disks from the source rod to the target rod using an auxiliary rod:
     1. Move the top n-1 disks from the source rod to the auxiliary rod.
     2. Move the nth disk from the source rod to the target rod.
     3. Move the n-1 disks from the auxiliary rod to the target rod.

Detailed Iteration for 3 Disks

Let's see how the code works for 3 disks (numberOfDisks = 3):

1. **Initial Call**:

   solveTowerOfHanoi(3, 'A', 'C', 'B');
   - Move 2 disks from A to B, using C as auxiliary.
   - Move disk 3 from A to C.
   - Move 2 disks from B to C, using A as auxiliary.

2. **First Recursive Call**:

   solveTowerOfHanoi(2, 'A', 'B', 'C');
   - Move 1 disk from A to C, using B as auxiliary.
   - Move disk 2 from A to B.
   - Move 1 disk from C to B, using A as auxiliary.

3. **Second Recursive Call**:

   solveTowerOfHanoi(1, 'A', 'C', 'B');
   - Move disk 1 from A to C.

4. **Move Disk 2**:

   System.out.println("Move disk 2 from rod A to rod B");

5. **Third Recursive Call**:

   solveTowerOfHanoi(1, 'C', 'B', 'A');
   - Move disk 1 from C to B.

6. **Move Disk 3**:

   System.out.println("Move disk 3 from rod A to rod C");

7. **Fourth Recursive Call**:

   solveTowerOfHanoi(2, 'B', 'C', 'A');
   - Move 1 disk from B to A, using C as auxiliary.
   - Move disk 2 from B to C.
   - Move 1 disk from A to C, using B as auxiliary.

8. **Fifth Recursive Call**:

   solveTowerOfHanoi(1, 'B', 'A', 'C');
   - Move disk 1 from B to A.

9. **Move Disk 2**:

   ```
   System.out.println("Move disk 2 from rod B to rod C");
   ```

10. **Sixth Recursive Call**:

    ```
    solveTowerOfHanoi(1, 'A', 'C', 'B');
    ```
    - Move disk 1 from A to C.

Output

The program will produce the following output:

```
Move disk 1 from rod A to rod C
Move disk 2 from rod A to rod B
Move disk 1 from rod C to rod B
Move disk 3 from rod A to rod C
Move disk 1 from rod B to rod A
Move disk 2 from rod B to rod C
Move disk 1 from rod A to rod C
```

This output shows the step-by-step process of moving the disks from rod A to rod C using rod B as an auxiliary.