TOWER OF HANOI

Recursive Algorithm

Origins

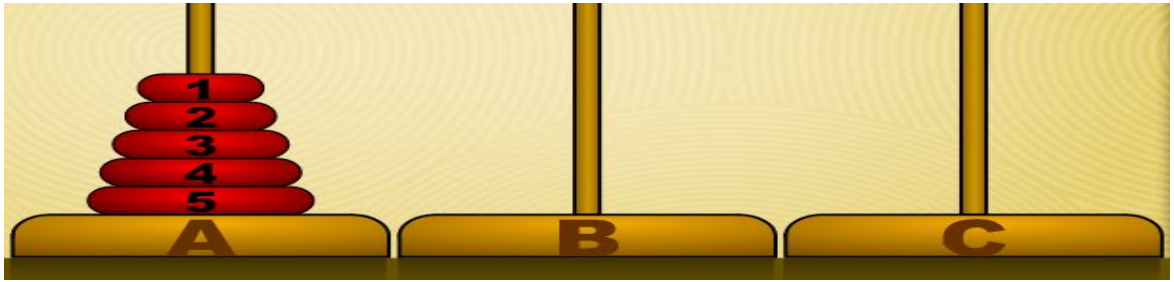
The french mathematician Édouard Lucas invented the Tower of Hanoi puzzle around 1883.

It is associated with a legend of a Hindu temple:

- It was used to increase mental discipline of young priests.
- 64 gold rings stacked on one of three post.
- Recreate the stack on another post: move 1 ring at a time, smaller ring on top of a larger one.
- How long would it take?

  At a rate of one movement per second -> 2^{64} – 1 sec. = **585 billion years** > 42 times the age of a universe

Édouard Lucas (1842-1891)



Rules

- **Move one ring at a time**

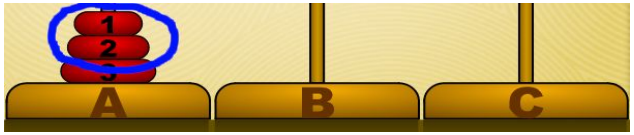
- **No ring may be placed on top of a ring that is smaller than it.**

Activities

https://www.mathplayground.com/logic_tower_of_hanoi.html (mute the tab so you do not get annoyed by the sound)

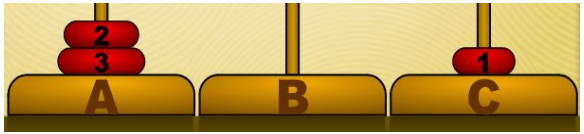
1. How many moves for 1 ring from tower A to tower C?
2. How many moves for 2 rings from tower A to tower C?
3. How many moves for 3 rings from tower A to tower C?

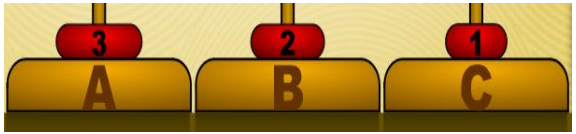
Solution for 3 rings

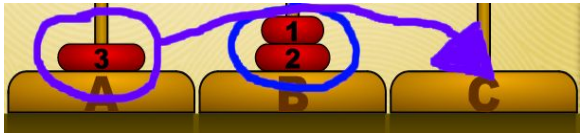


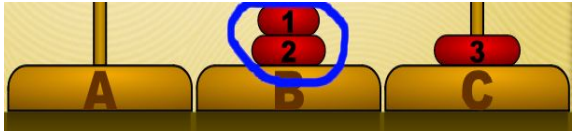
T(2, A, B, C)

1. Small from A to C
2. Medium from A to B
3. Small from C to B







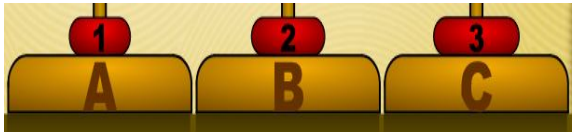


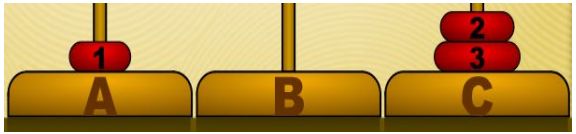
T(1, A, C, B)

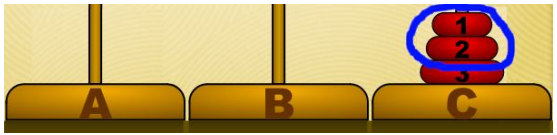
4. Large from A to C

T(2, B, C, A)

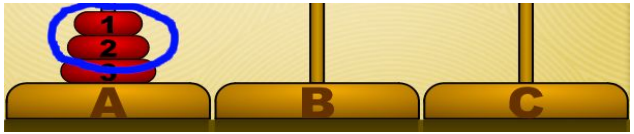
5. Small from B to A
6. Medium from B to C
7. Small from A to C





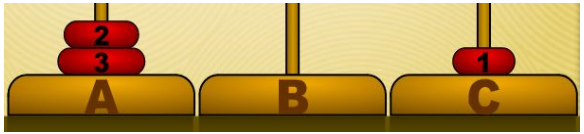


Solution for 3 rings

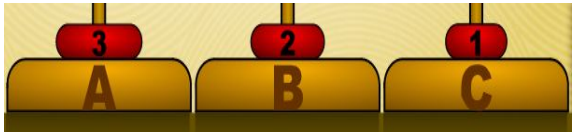


T(2, A, B, C)

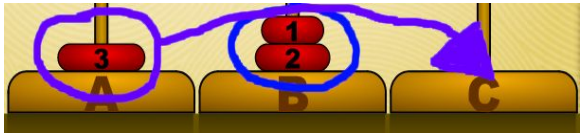
1. Small from A to C



2. Medium from A to B

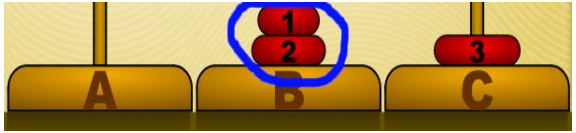


3. Small from C to B



T(1, A, C, B)

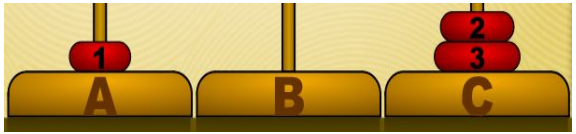
4. Large from A to C



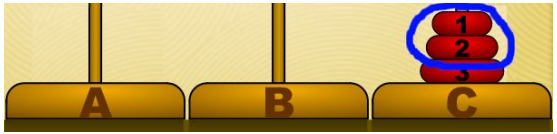
T(2, B, C, A)

5. Small from B to A

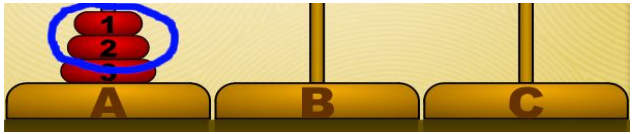
6. Medium from B to C



7. Small from A to C



Solution for 3 rings



T(3, A, C, B)

T(2, A, B, C)

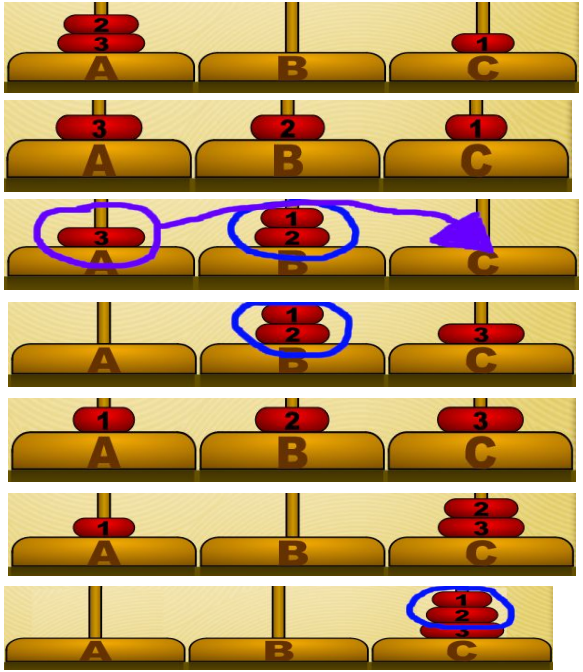
1. Small from A to C ⇒ T(1, A, C, B)
2. Medium from A to B ⇒ T(1, A, B, C)
3. Small from C to B ⇒ T(1, C, B, A)

T(1, A, C, B)

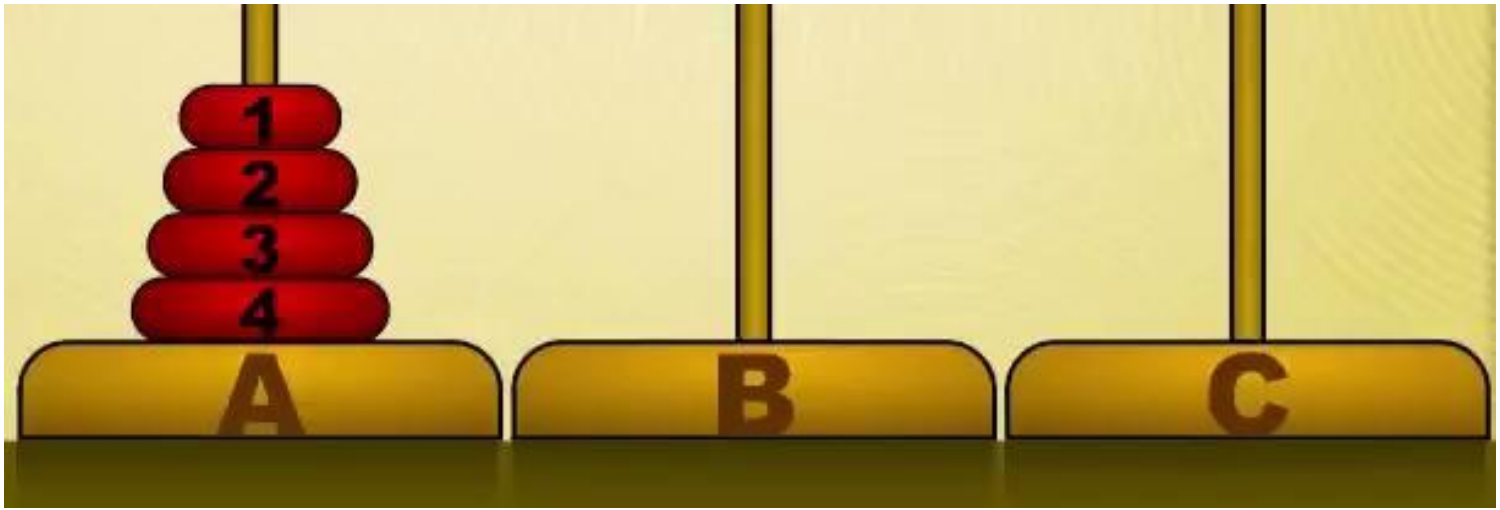
4. Large from A to C

T(2, B, C, A)

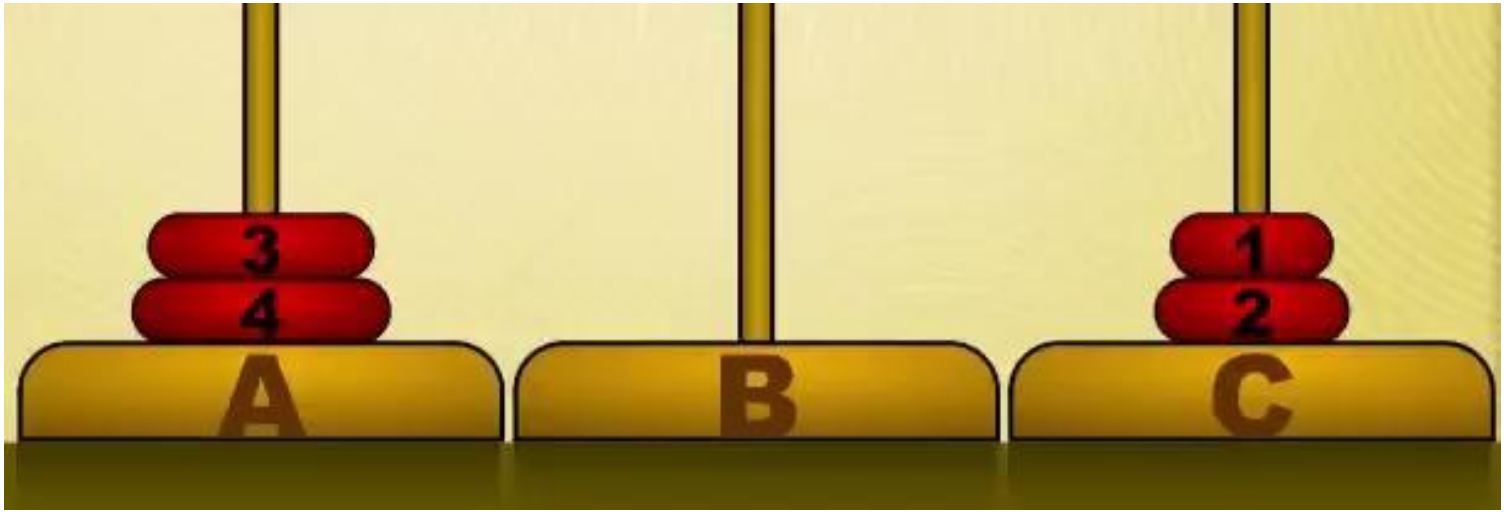
5. Small from B to A ⇒ T(1, B, A, C)
6. Medium from B to C ⇒ T(1, B, C, A)
7. Small from A to C ⇒ T(1, A, C, B)



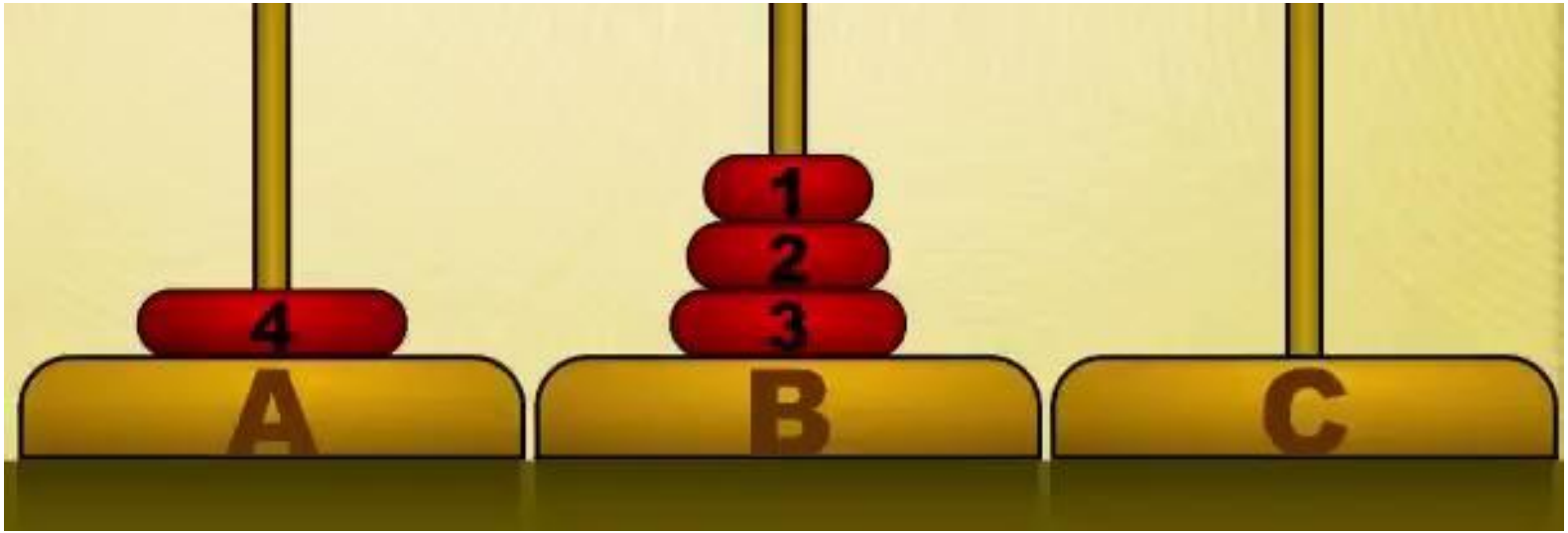
Solution for 4 ring (first step)



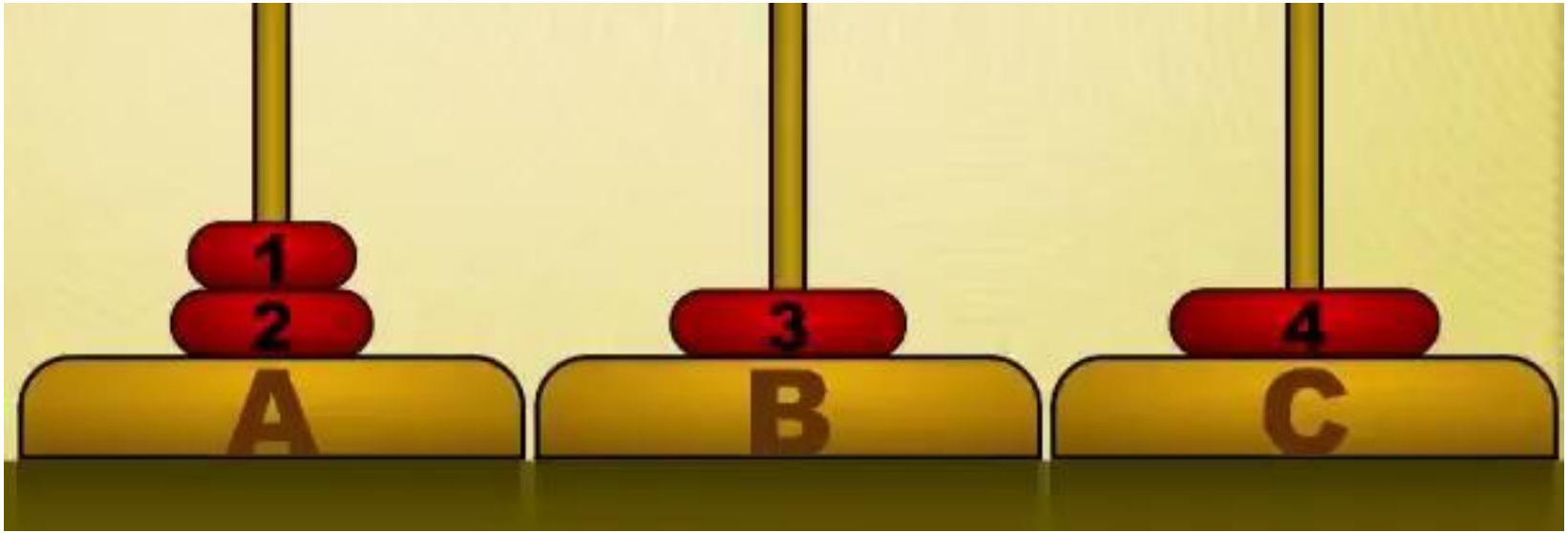
Solution for 4 ring (second step)



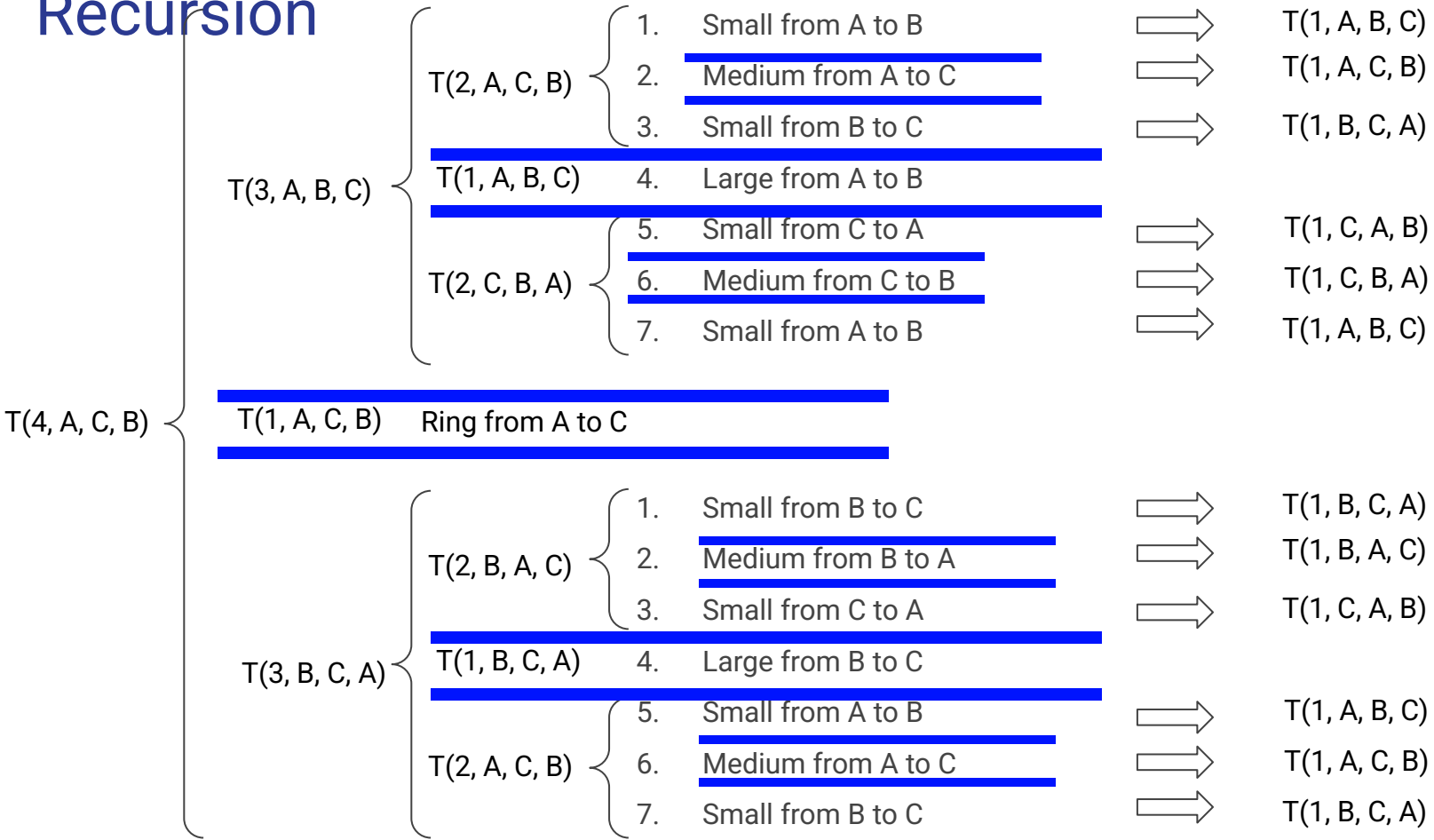
Solution for 4 ring (third step)



Solution for 4 ring (last step)



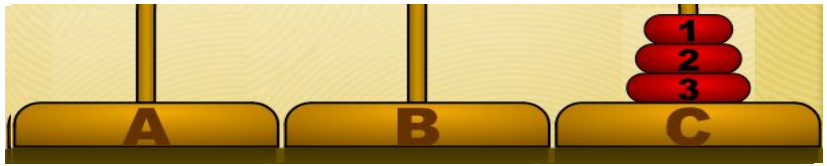
Recursion


T(4, A, C, B)
T(3, A, B, C)
T(2, A, C, B)
1. Small from A to B
2. Medium from A to C
3. Small from B to C
T(1, A, B, C)
4. Large from A to B
T(2, C, B, A)
5. Small from C to A
6. Medium from C to B
7. Small from A to B
T(1, A, B, C)
T(1, A, C, B)
Ring from A to C
T(3, B, C, A)
T(2, B, A, C)
1. Small from B to C
2. Medium from B to A
3. Small from C to A
T(1, B, C, A)
4. Large from B to C
T(2, A, C, B)
5. Small from A to B
6. Medium from A to C
7. Small from B to C
T(1, A, B, C)
T(1, A, C, B)
T(1, B, C, A)
T(1, C, A, B)
T(1, C, B, A)
T(1, A, B, C)
T(1, B, C, A)
T(1, B, A, C)
T(1, C, A, B)
T(1, A, B, C)
T(1, A, C, B)
T(1, B, C, A)

Recursion

T(n, A, C, B):

- T(n-1, A, B, C)
- Move ring from A to C
- T(n-1, B, C, A)



T(4, A, C, B)

- T(3, A, B, C)
  - T(2, A, C, B)
    1. Small from A to B
    2. Medium from A to C
    3. Small from B to C
  - T(1, A, B, C)
    4. Large from A to B
  - T(2, C, B, A)
    5. Small from C to A
    6. Medium from C to B
    7. Small from A to B
- T(1, A, C, B) Large from A to C
- T(3, B, C, A)
  - T(2, B, A, C)
    1. Small from B to C
    2. Medium from B to A
    3. Small from C to A
  - T(1, B, C, A)
    4. Large from B to C
  - T(2, B, C, A)
    5. Small from A to B
    6. Medium from A to C
    7. Small from B to C

Numbers of moves

Minimal number of moves required = $2^n - 1$

Where, n= number of rings.

For 3 rings: $2^3 - 1 = 7$ movements

Number of rings (N)	Number of Moves ((2**N)-1)	2**N
1	1	1
2	3	4
3	7	8
4	15	16
5	31	32
6	63	64
7	127	128
8	255	256

Coding Time !!!

Save your work here: **.../APCSA1/apcsa-assignments-fall-YourUsername/classwork/01_13_hanoi/Hanoi.java**

Write a function to move n rings from source rod to destination rod, print the moves of each ring.

```
public static void hanoiMild(int n, char source_rod, char destination_rod, char aux_rod){
     YOUR CODE HERE

}
```

Output with 3 rings:

Move ring 1 from source A to destination C
Move ring 2 from source A to destination B
Move ring 1 from source C to destination B
Move ring 3 from source A to destination C
Move ring 1 from source B to destination A
Move ring 2 from source B to destination C
Move ring 1 from source A to destination C

Coding Time!!!

Write a function to move n rings from source rod to destination rod, print the moves of each ring, and the total number of moves. Check if your solution used the least possible number of moves.

```
public static int hanoiMedium(int n, char source_rod, char destination_rod, char
aux_rod){
     YOUR CODE HERE

}
```

Output with 3 rings:

(Print moves as shown in previous method)

Total number of moves: 7

The total number of moves (7) is equal to 2**n - 1 (2**3 - 1)

Coding Time!!!

Write a function to move n rings from source rod to destination rod, print the moves of each ring, and the list of rings on each rod after each move.

```
public static void hanoiSpicy(int n, char source_rod, char destination_rod, char
aux_rod){
     YOUR CODE HERE

}
```

Example:

Current state of rods:
Rod A: 3 2 1
Rod B:
Rod C:

Move disk 1 from Rod A to Rod C
Current state of rods:
Rod A: 3 2
Rod B:
Rod C: 1
....

Coding Time!!! (Optional)

Write a function to move n rings from source rod to destination rod, print the moves of each ring, and the list of rings on each rod after each move.

```
public static void hanoiSpicy(int n, char source_rod, char destination_rod, char
aux_rod){
     YOUR CODE HERE

}
```

Example:

Current state of rods:
Rod A: 3 2 1
Rod B:
Rod C:

Move disk 1 from Rod A to Rod C
Current state of rods:
Rod A: 3 2
Rod B:
Rod C: 1
....

Hints

You probably need a structure to represent the rods => ArrayList:

```
rods ==> [[3, 2, 1], [], []]
```

Possible base case (there is more than one way to implement this algorithm):

```
if (n == 1) {
    Get the structure that represents source_rod.
    Get the top ring by removing it from the source rod. Store that ring in a variable.
    Add that ring to the destination tower.
    Print "Move ring <the one stored in your variable> from source <source> to destination <destination>"
    return;
}
```