5CCS2FC2: Foundations of Computing II

Divide-and-Conquer and Recursion

Week 3

Dr Christopher Hampson

Department of Informatics

King's College London

Divide-and-Conquer

Divide-and-Conquer Techninque

Divide-and-Conquer

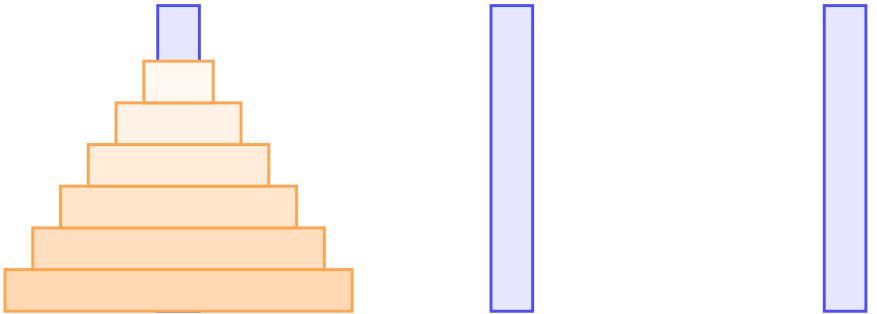
Divide) Divide the problem into several 'self-similar' but smaller sub-problems,

Conquer) Solve these sub-problems recursively,

Combine) Recombine the sub-problems into a solution for the whole problem.

The Towers of Hanoi

The Towers of Hanoi



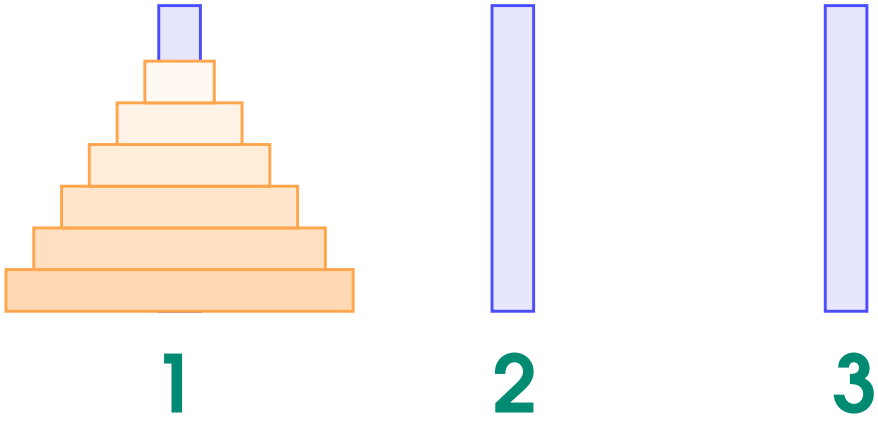
The Towers of Hanoi

Constraint 1) You can only move **one block** at a time.

Constraint 2) You cannot place any block on top of a **smaller** block.

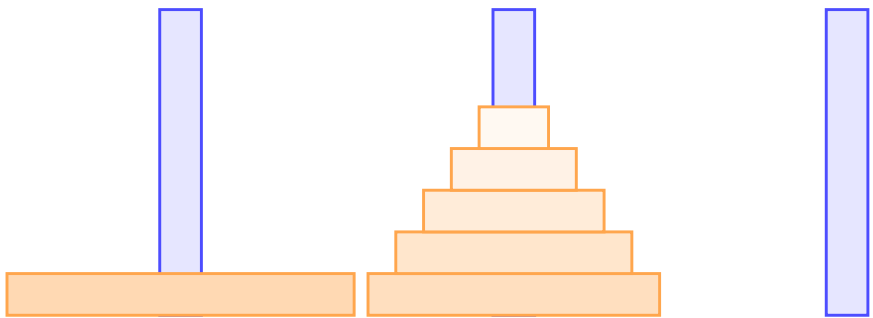
The Towers of Hanoi

MoveTower(n, 1, 3):



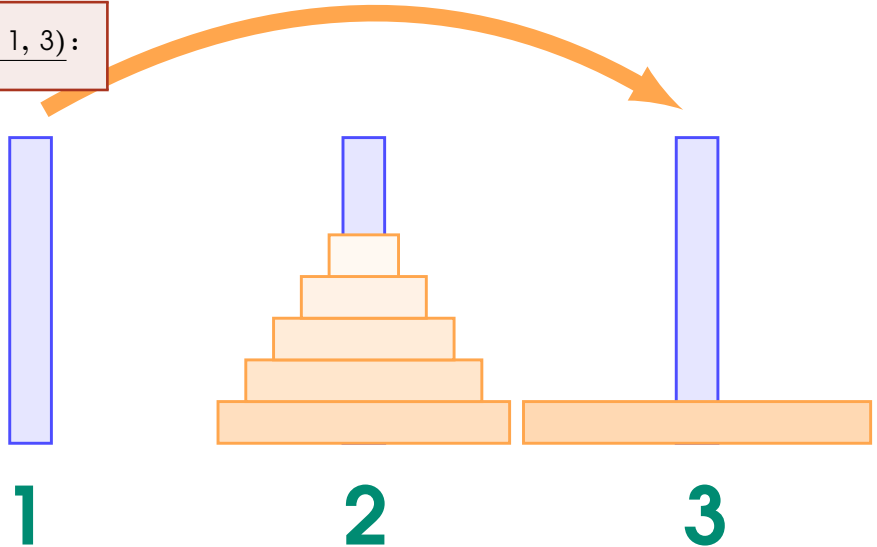
The Towers of Hanoi

MoveTower(n, 1, 3):



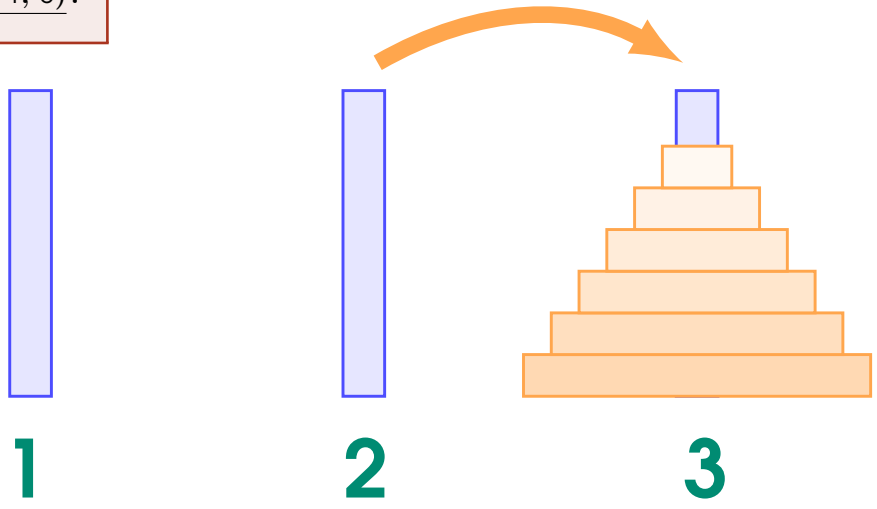
The Towers of Hanoi

MoveTower(n, 1, 3):



The Towers of Hanoi

MoveTower(n, 1, 3):

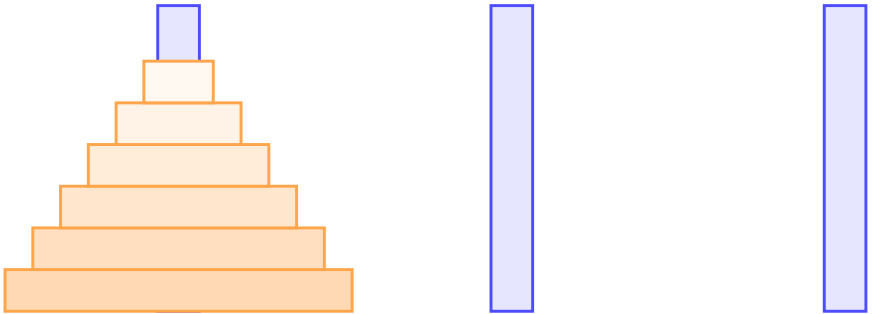


The Towers of Hanoi

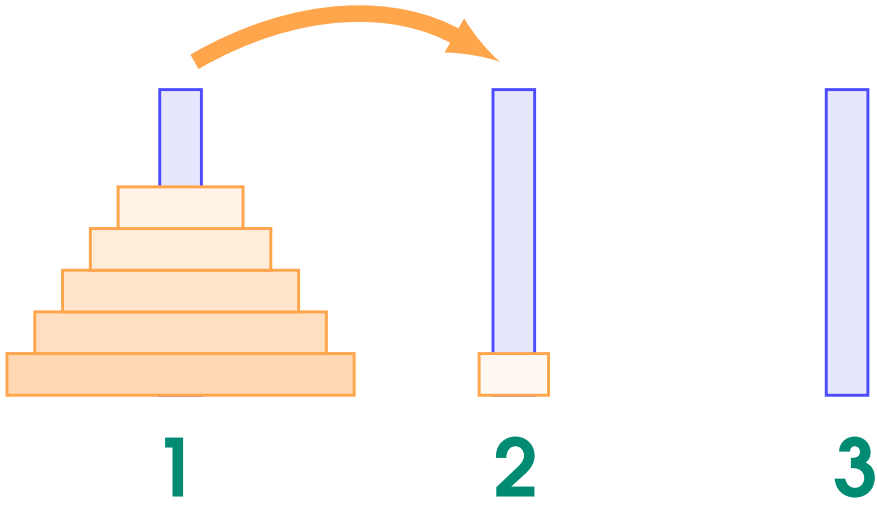
MoveTower(n, 1, 3):

1. **If** $n = 1$ **then** Move(1, 3)
2. **Else**
    3. MoveTower($n - 1$, 1, 2)
    4. Move(1, 3)
    5. MoveTower($n - 1$, 2, 3)

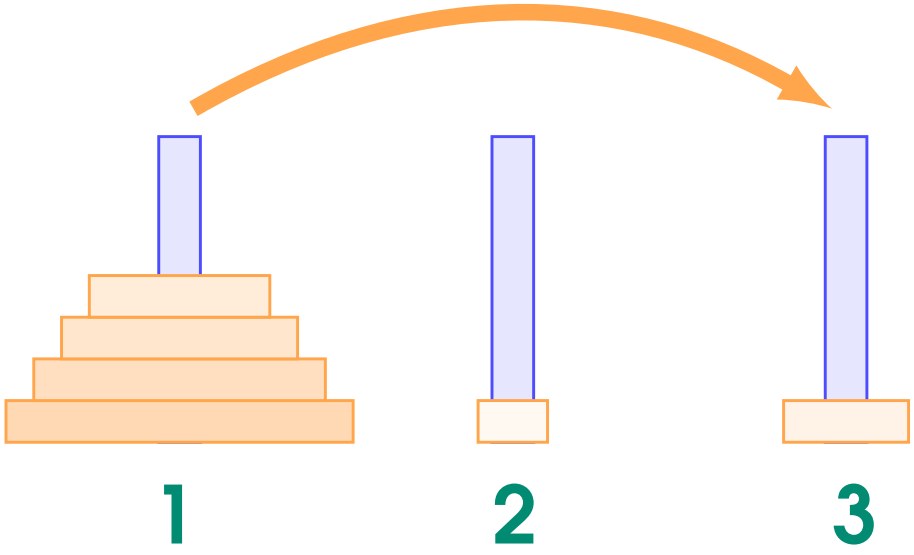
The Towers of Hanoi



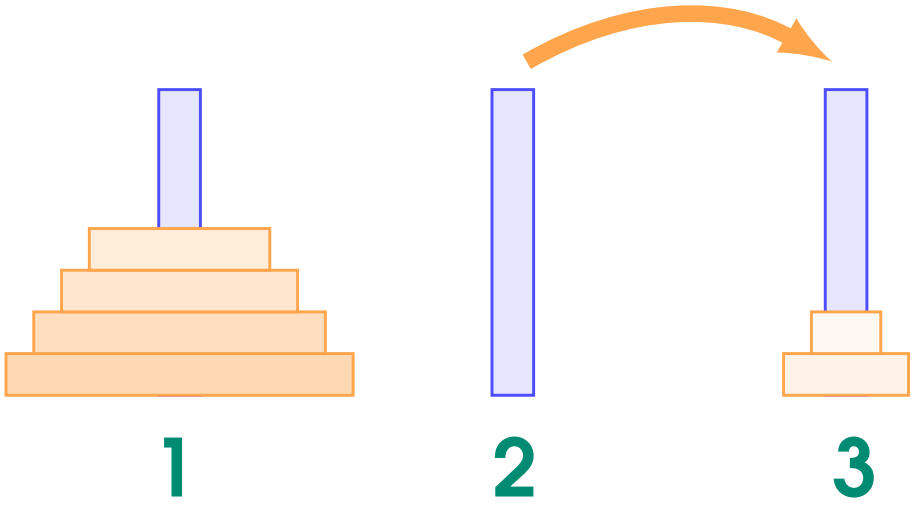
The Towers of Hanoi



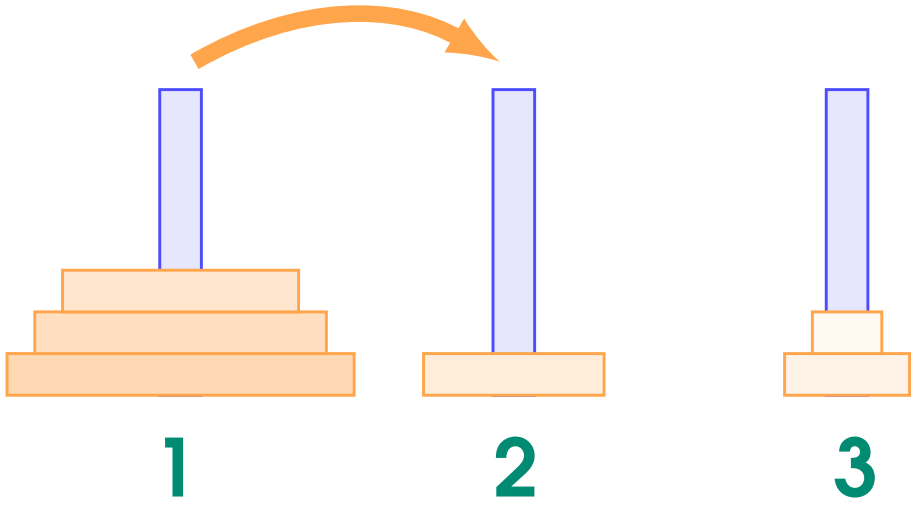
The Towers of Hanoi



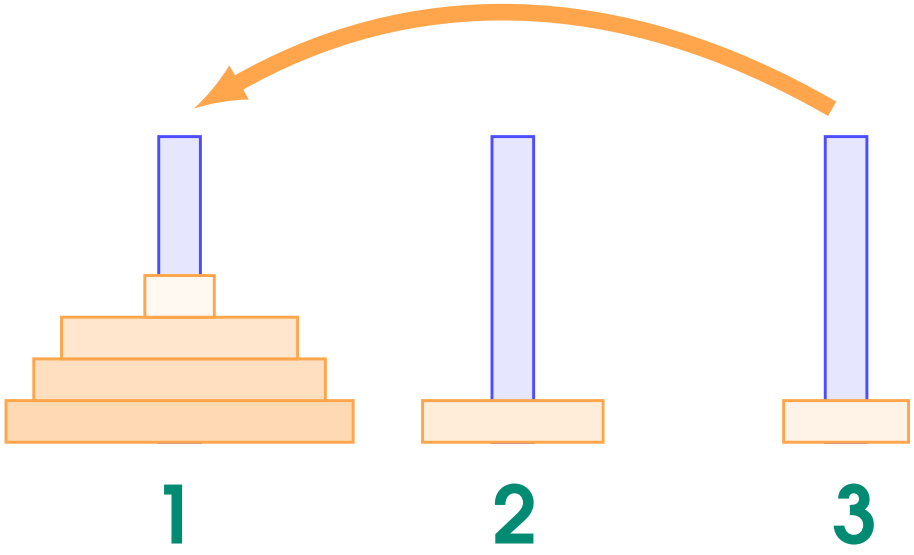
The Towers of Hanoi



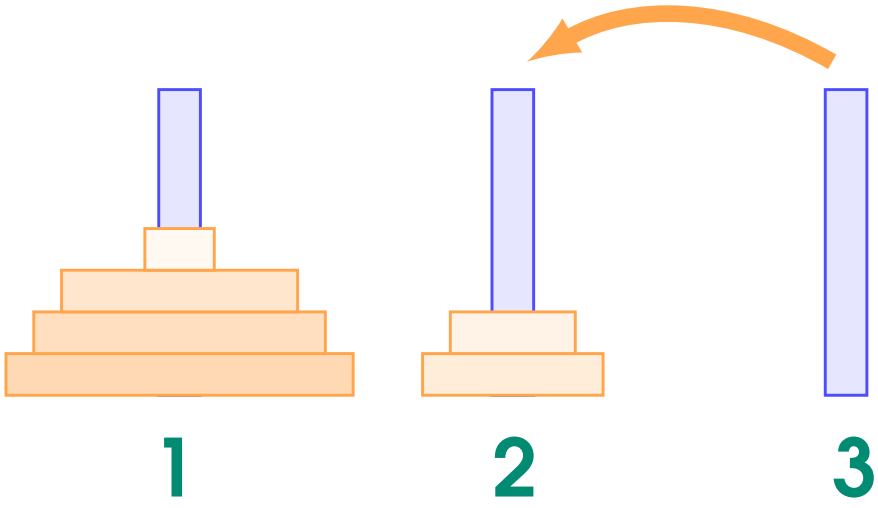
The Towers of Hanoi



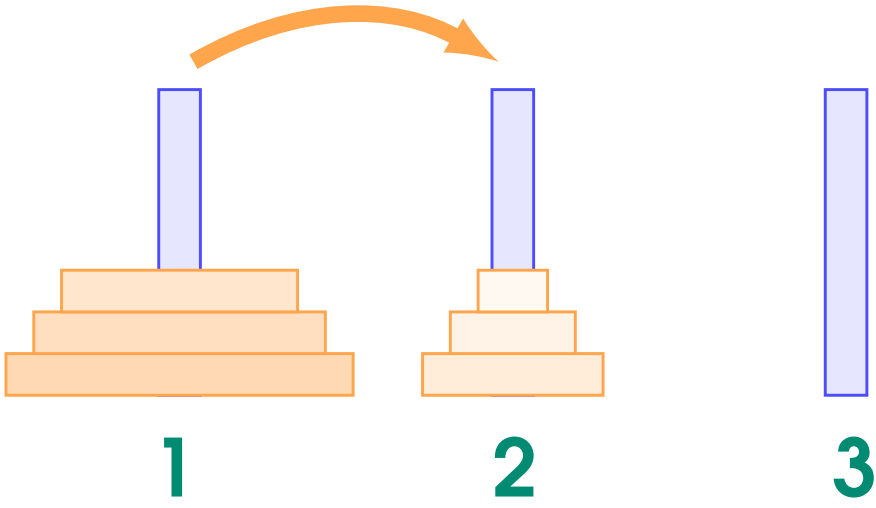
The Towers of Hanoi



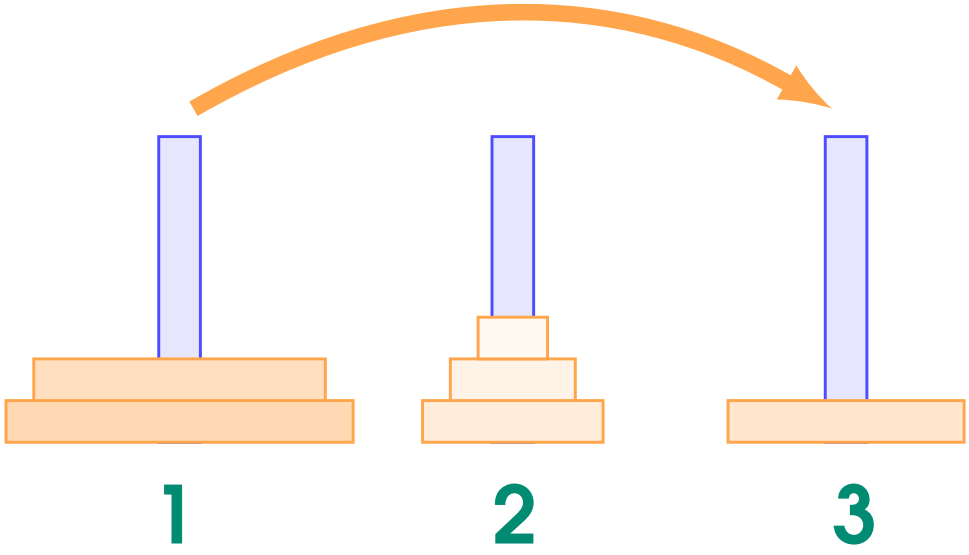
The Towers of Hanoi



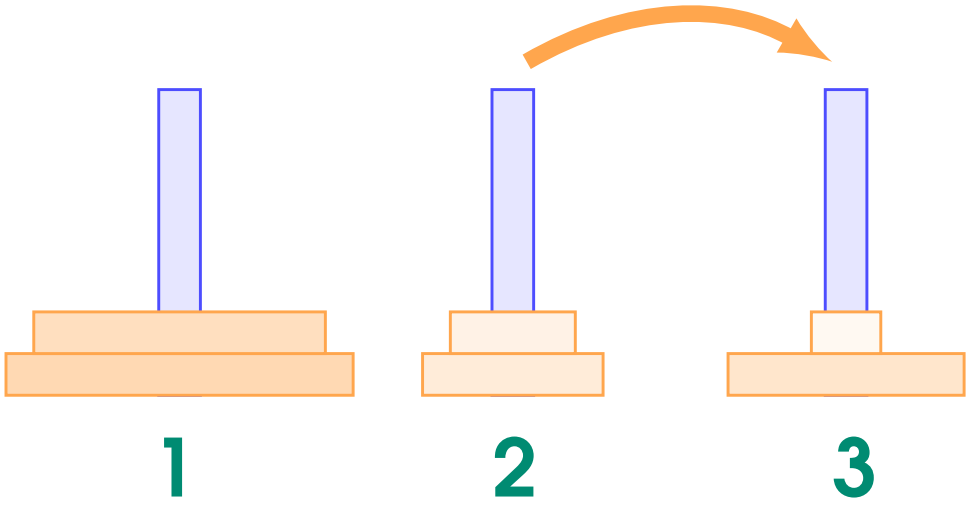
The Towers of Hanoi



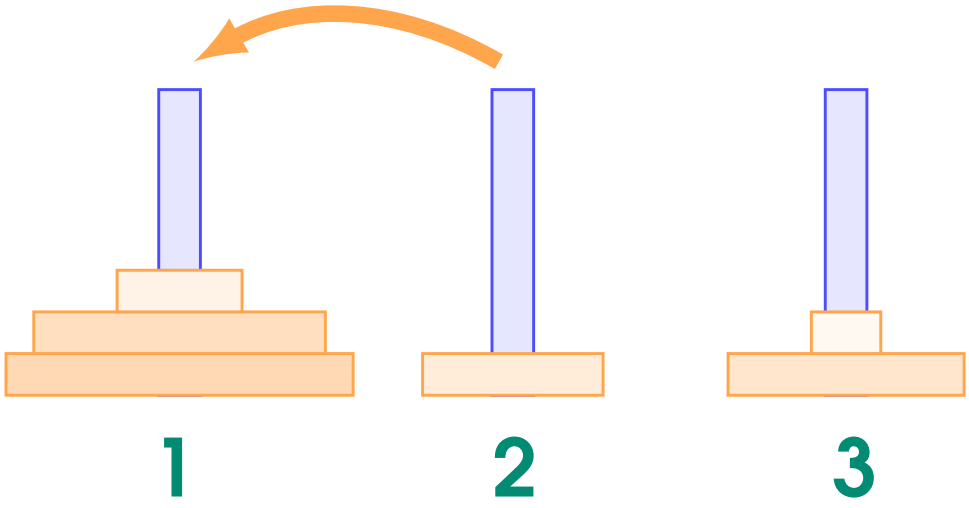
The Towers of Hanoi



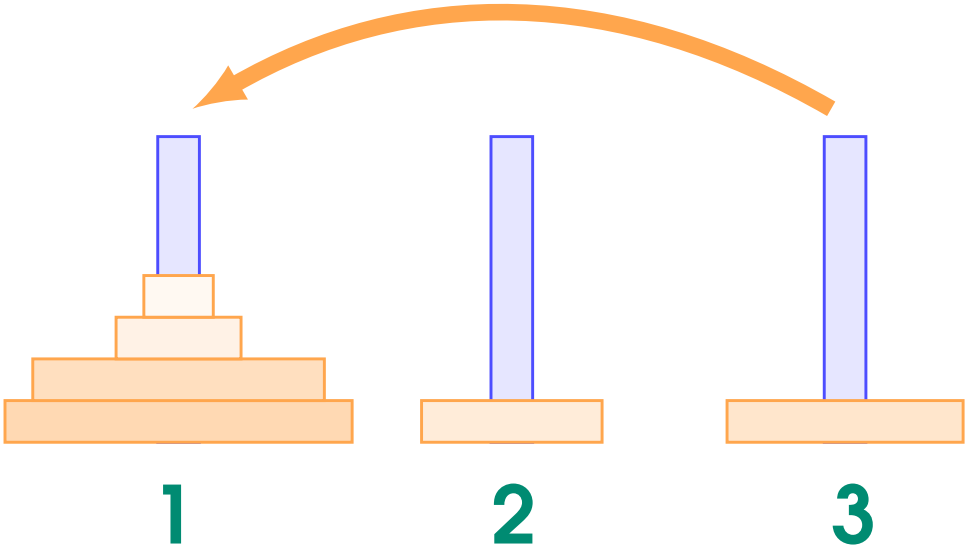
The Towers of Hanoi



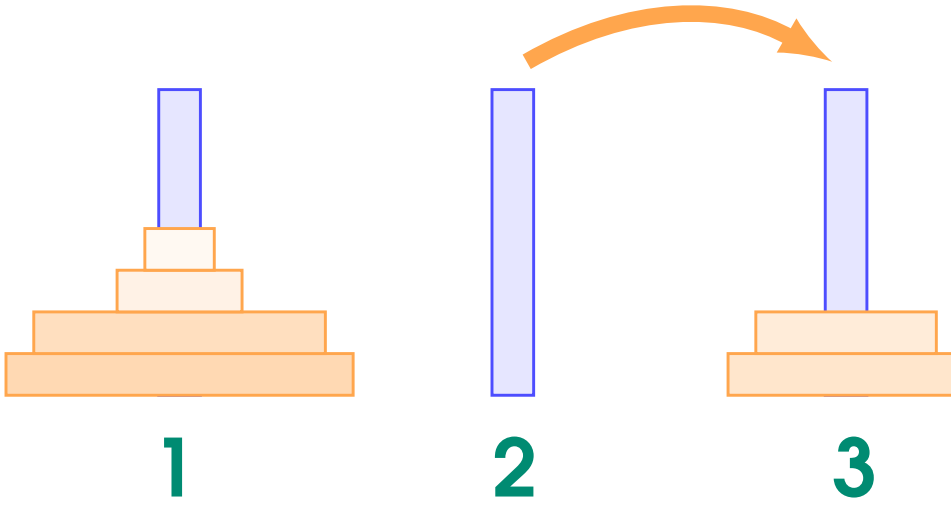
The Towers of Hanoi



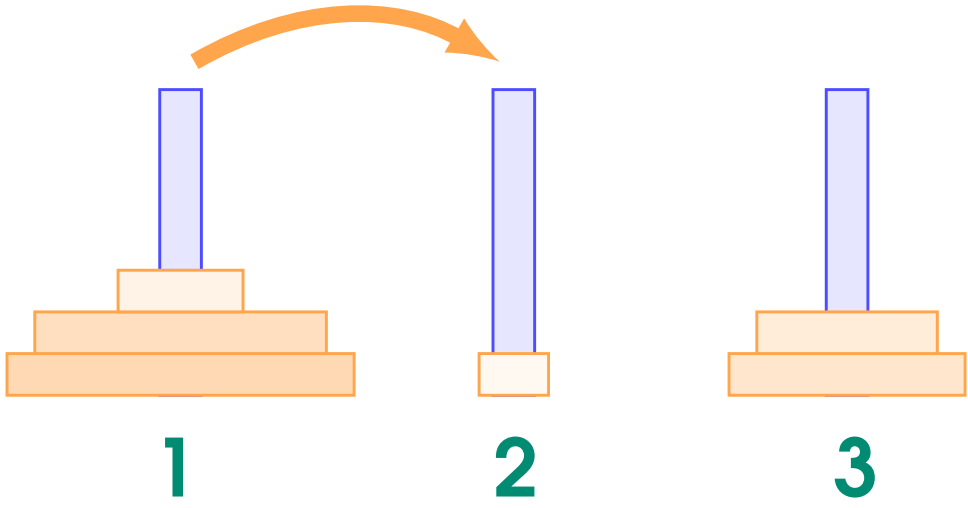
The Towers of Hanoi



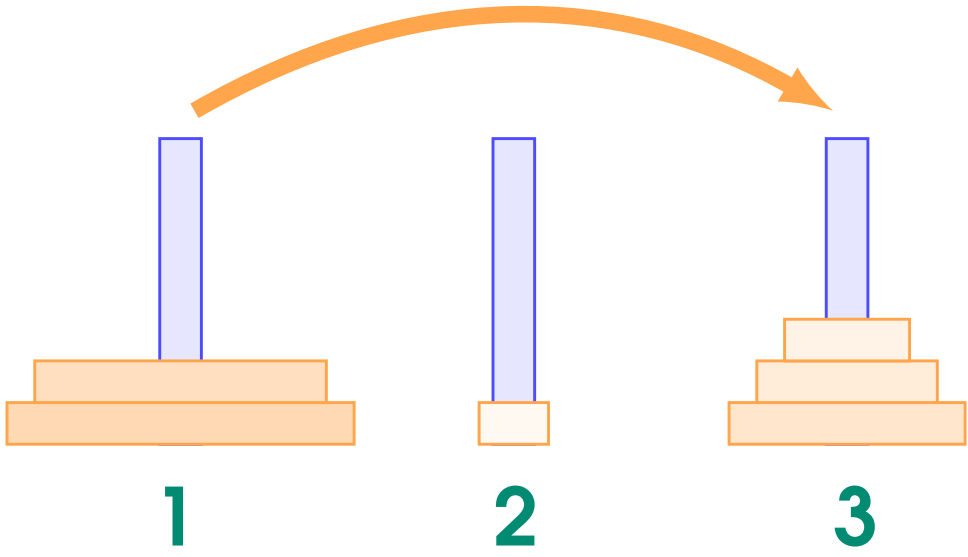
The Towers of Hanoi



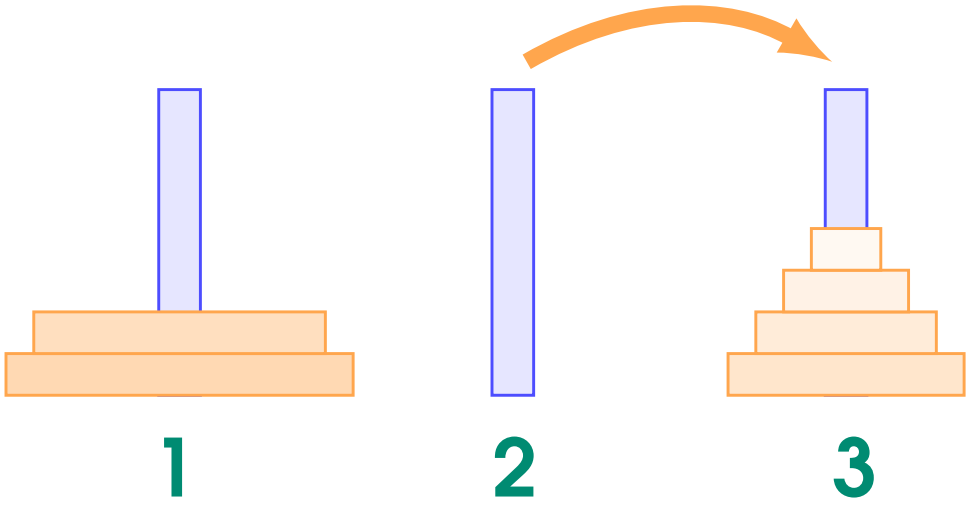
The Towers of Hanoi



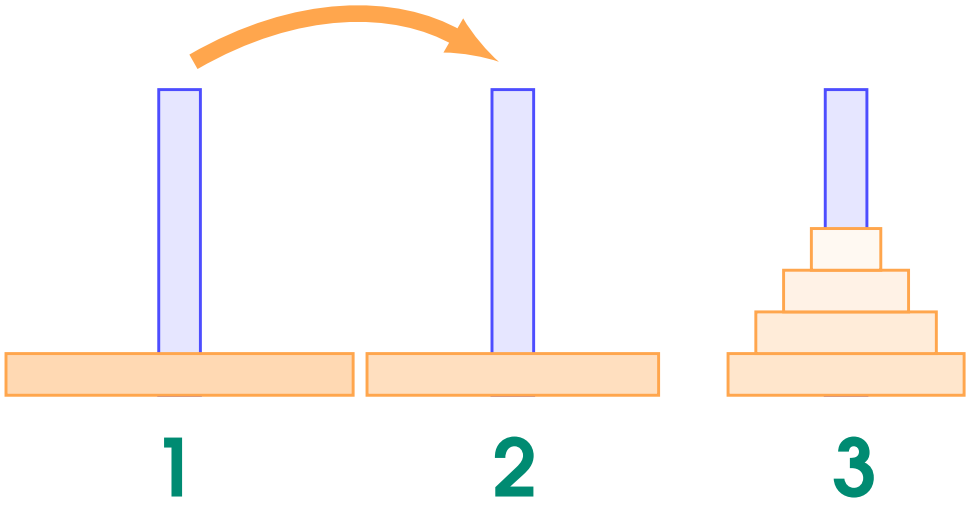
The Towers of Hanoi



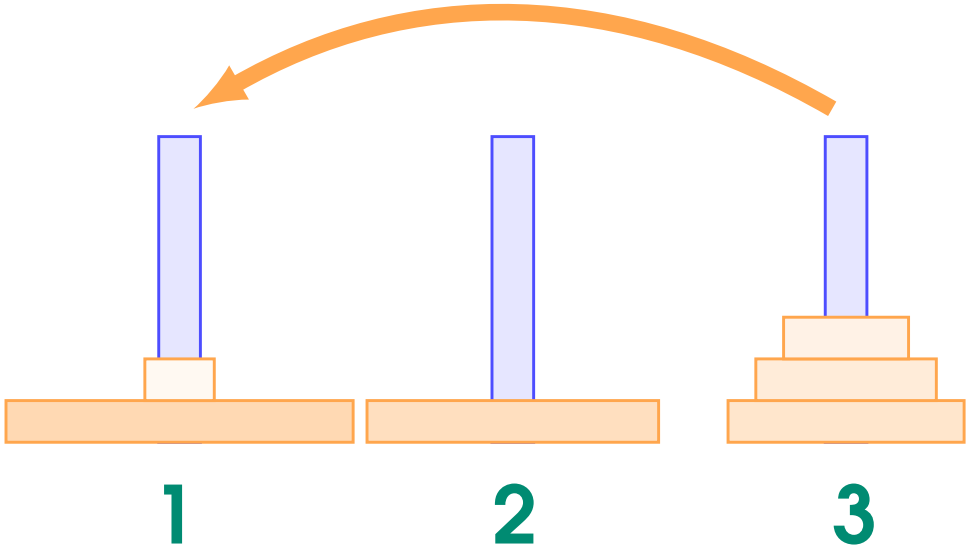
The Towers of Hanoi



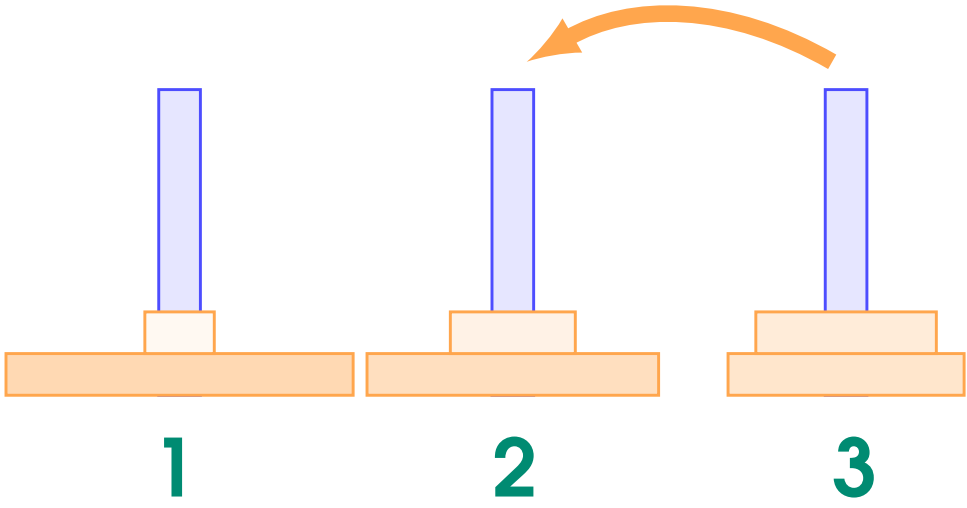
The Towers of Hanoi



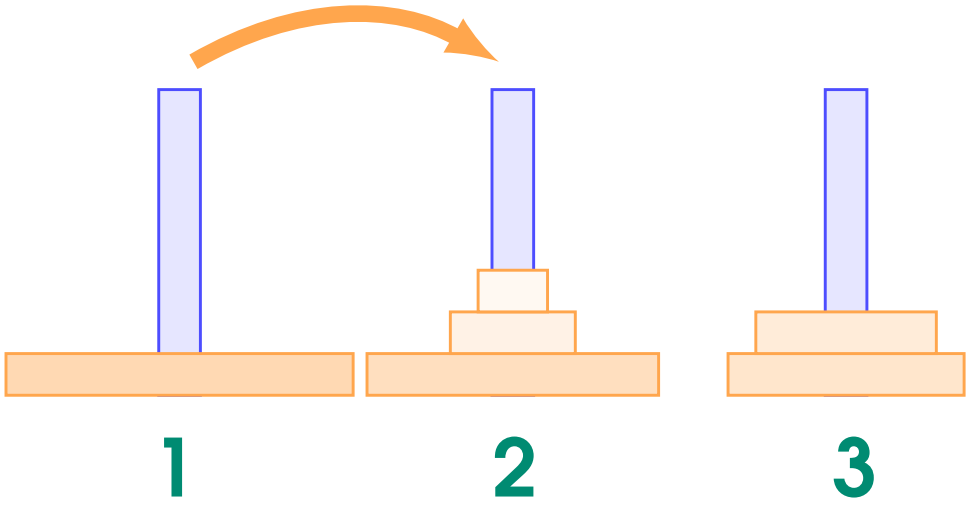
The Towers of Hanoi



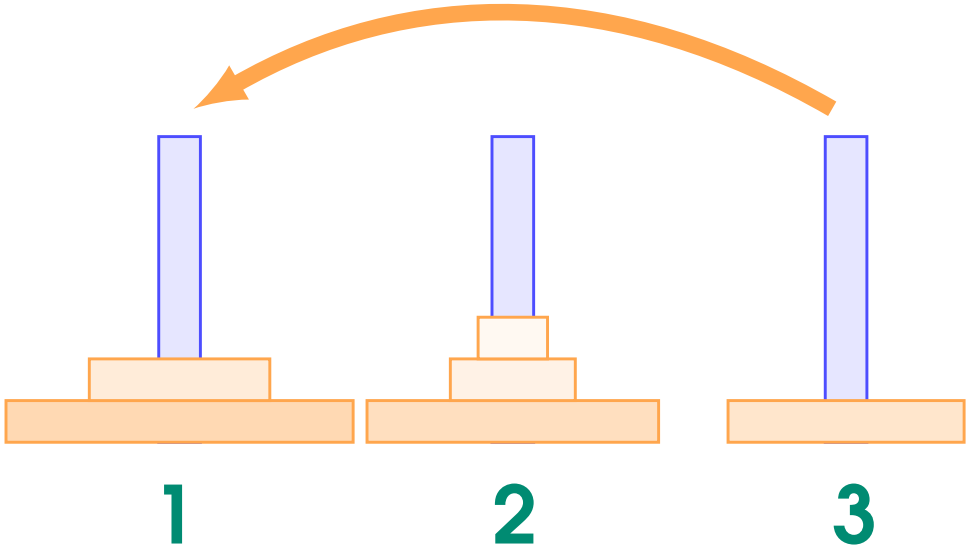
The Towers of Hanoi



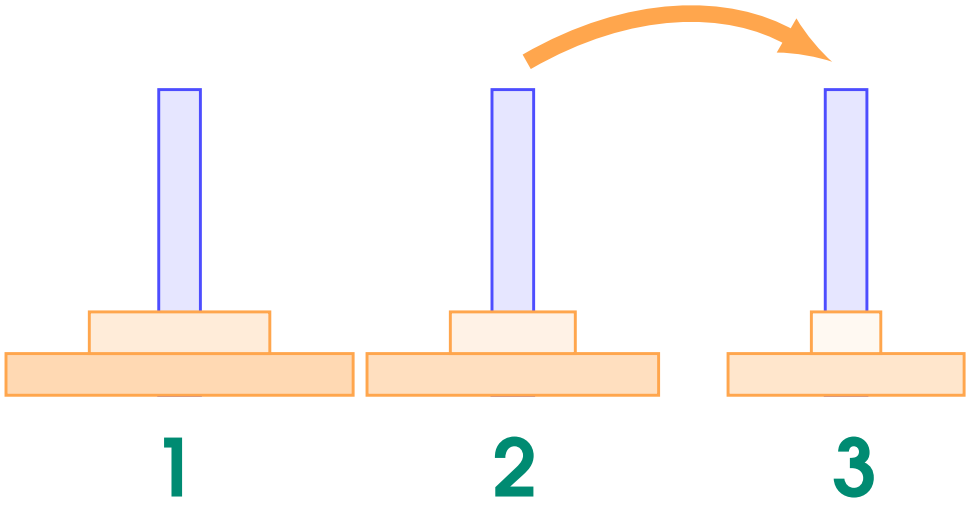
The Towers of Hanoi



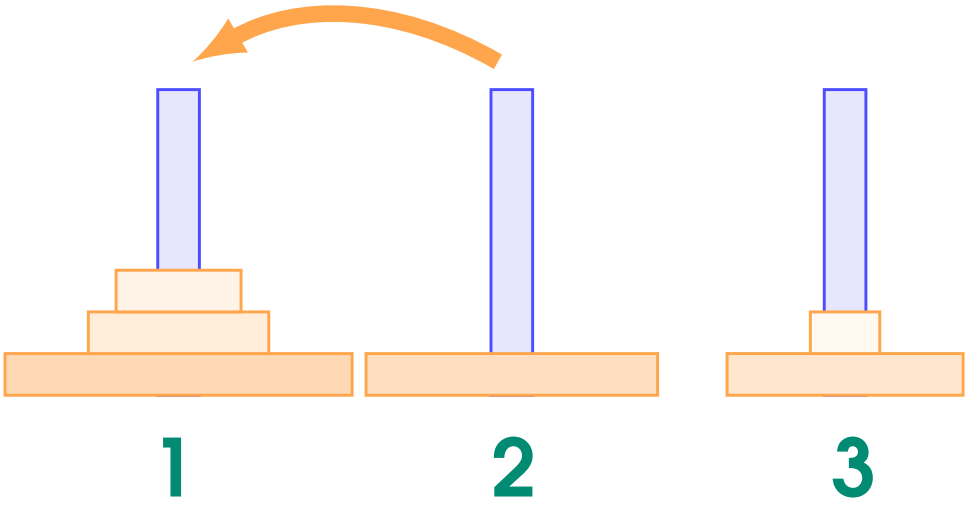
The Towers of Hanoi



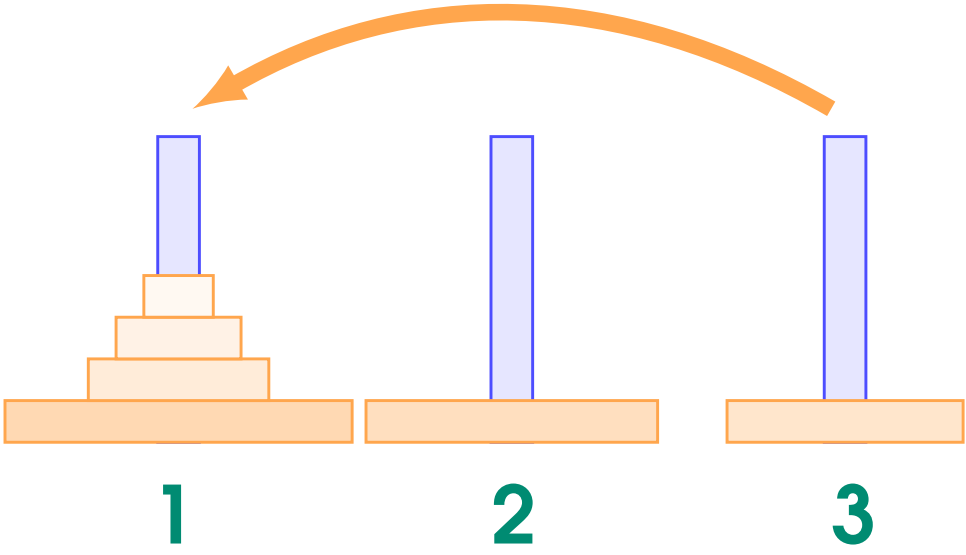
The Towers of Hanoi



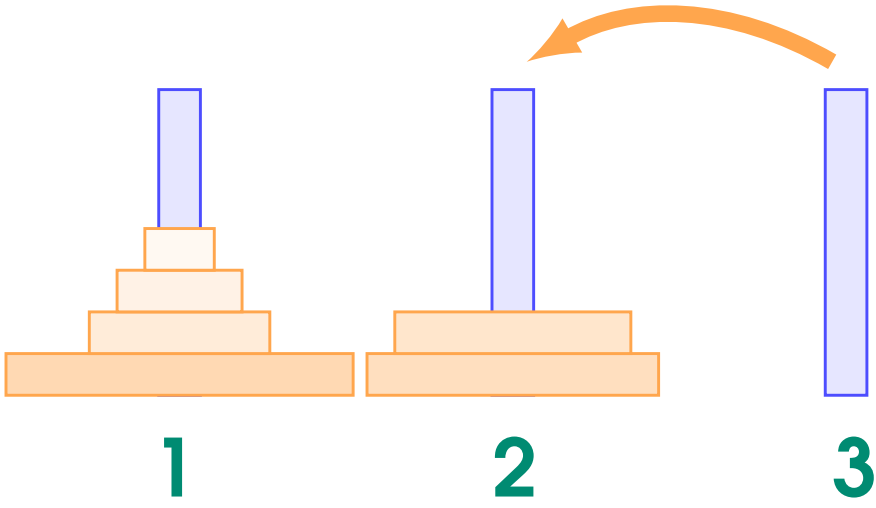
The Towers of Hanoi



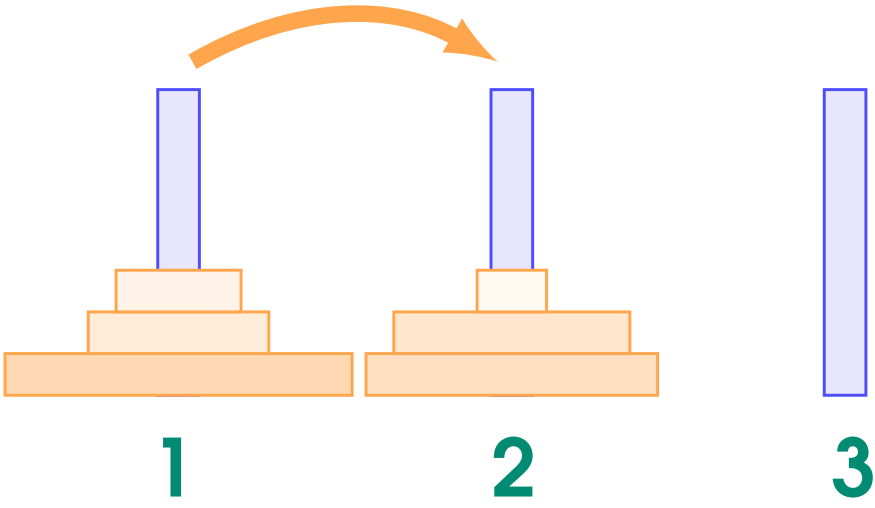
The Towers of Hanoi



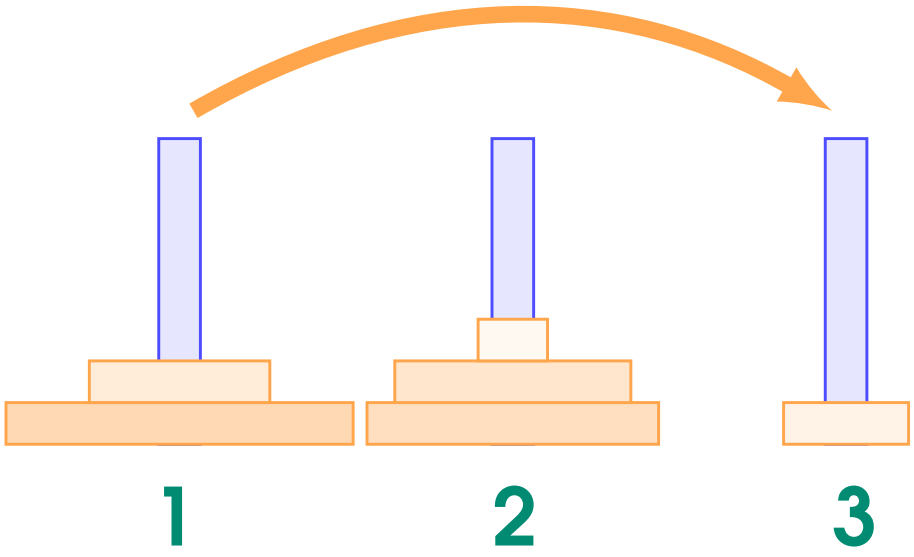
The Towers of Hanoi



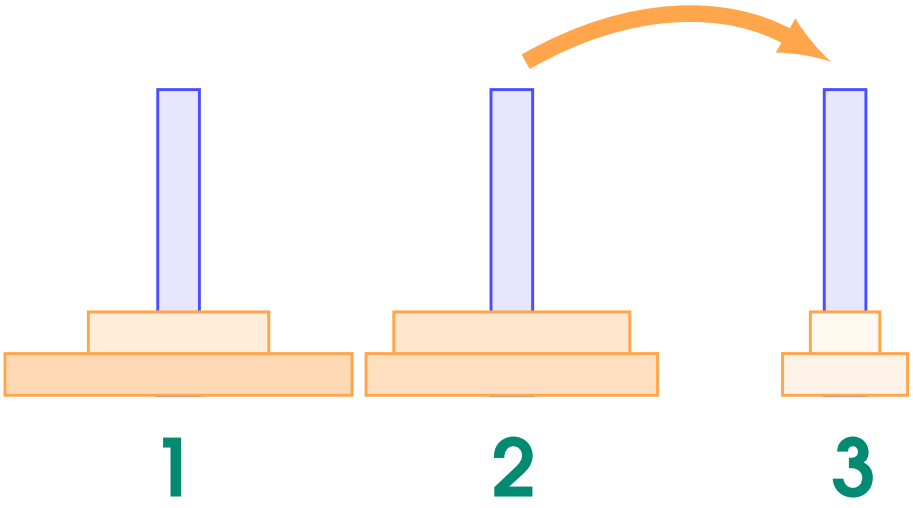
The Towers of Hanoi



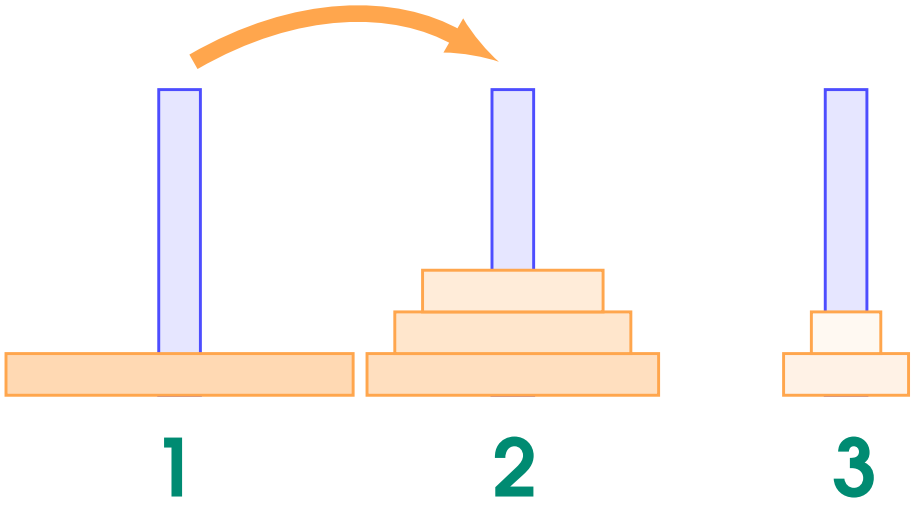
The Towers of Hanoi



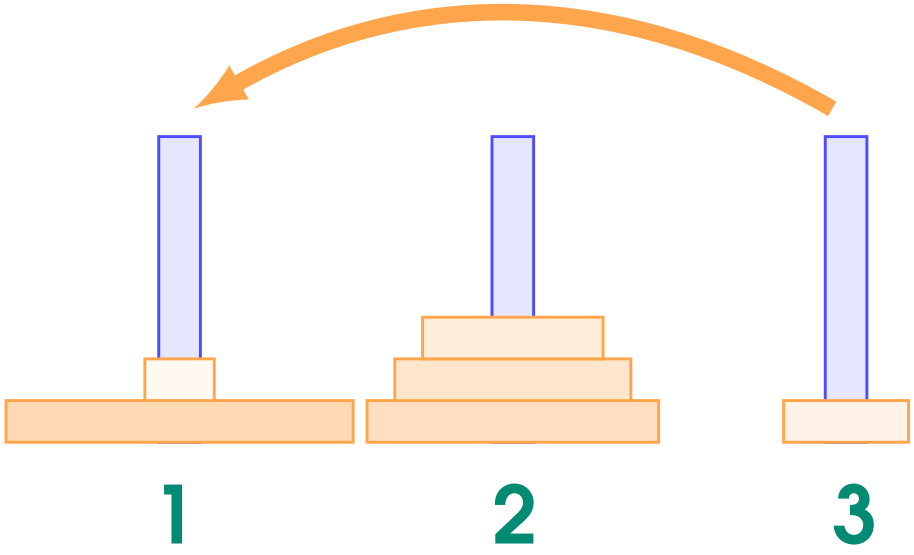
The Towers of Hanoi



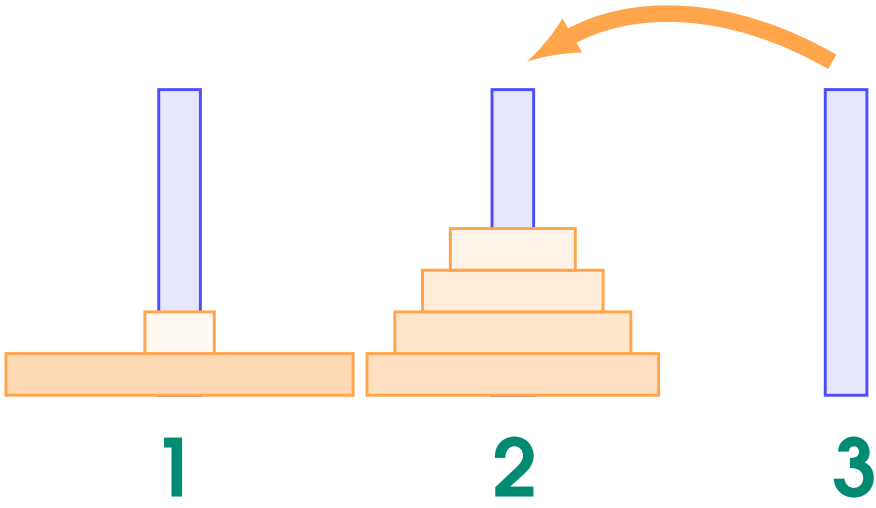
The Towers of Hanoi



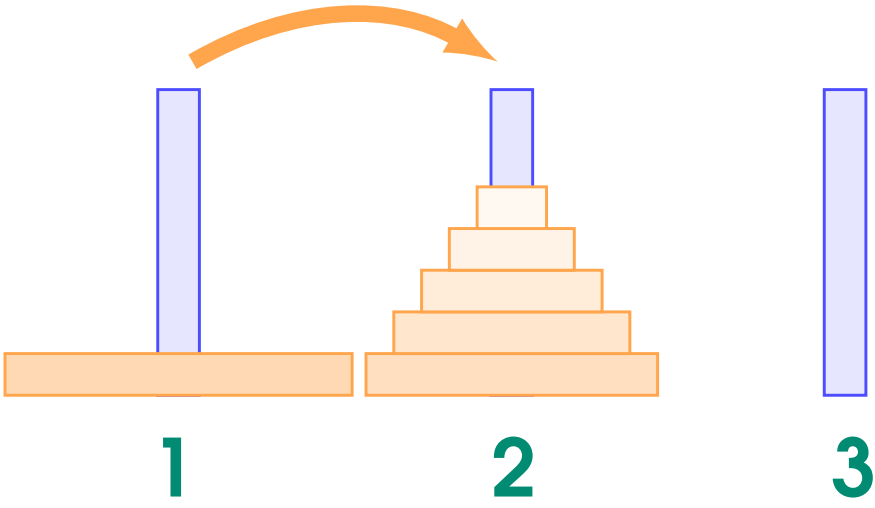
The Towers of Hanoi



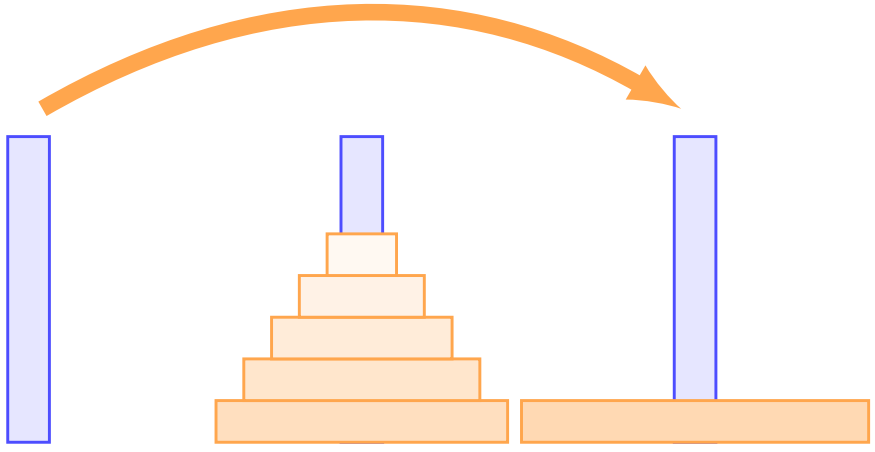
The Towers of Hanoi



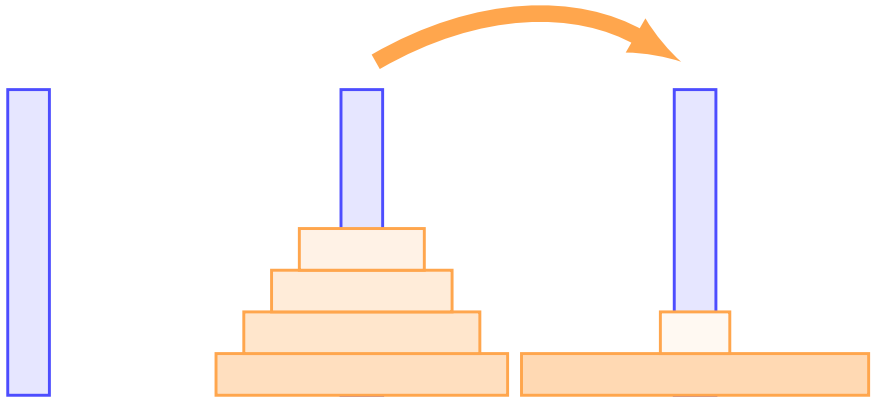
The Towers of Hanoi



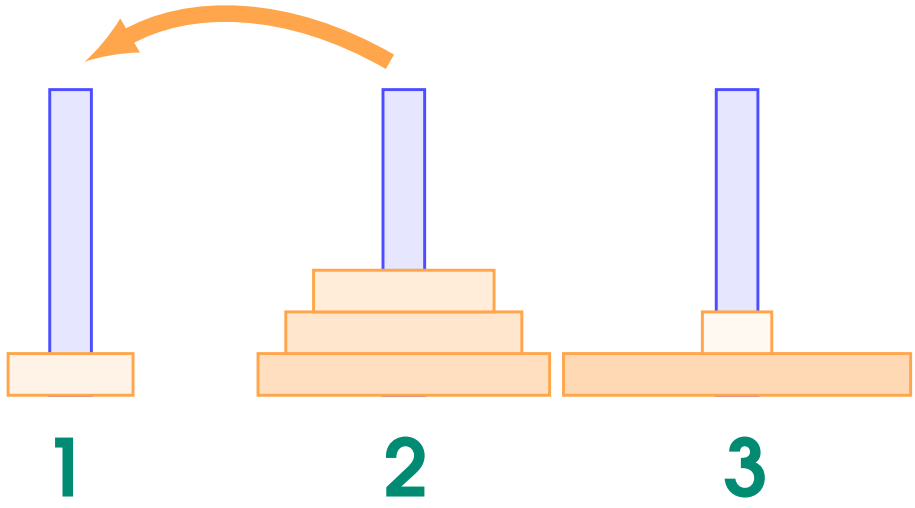
The Towers of Hanoi



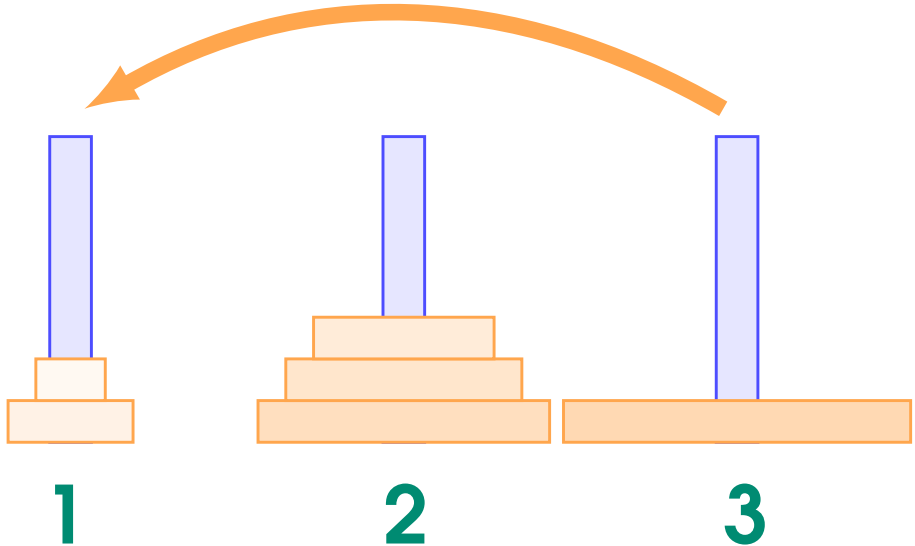
The Towers of Hanoi



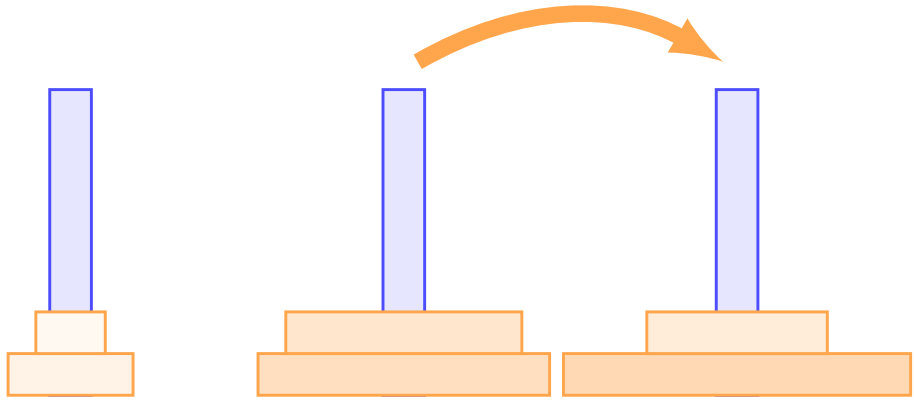
The Towers of Hanoi



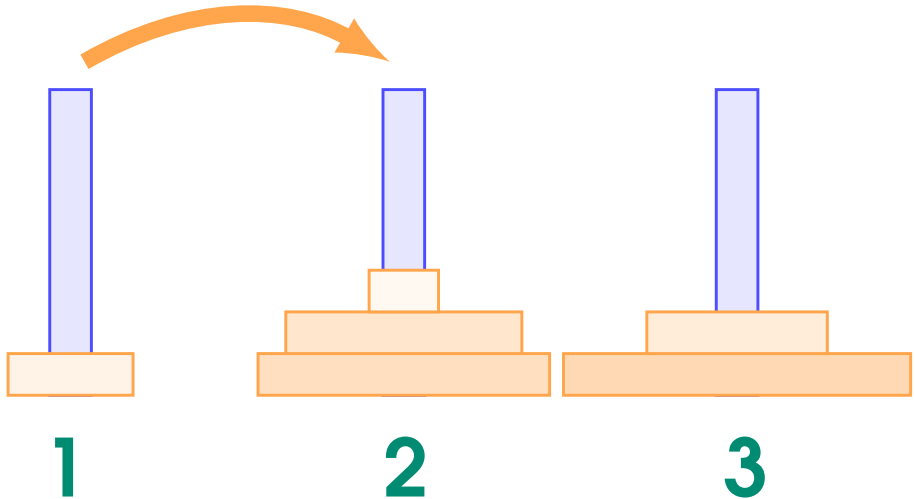
The Towers of Hanoi



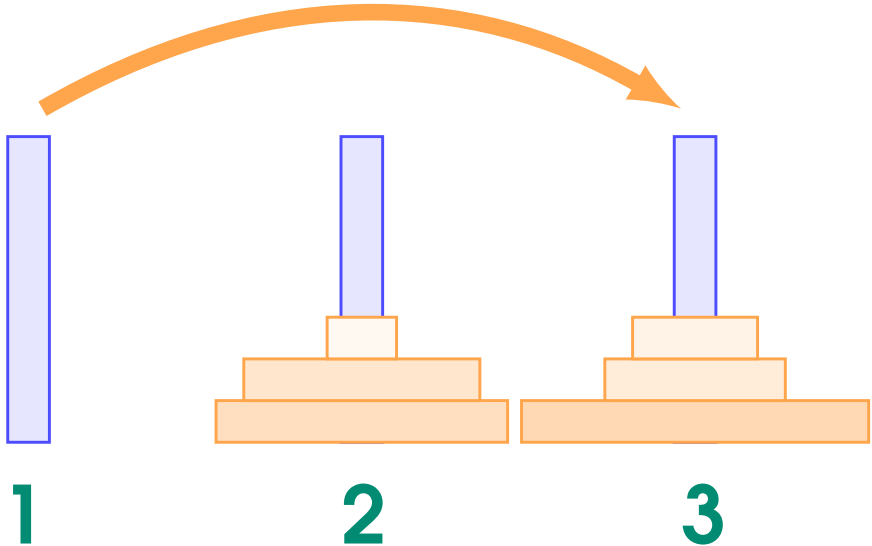
The Towers of Hanoi



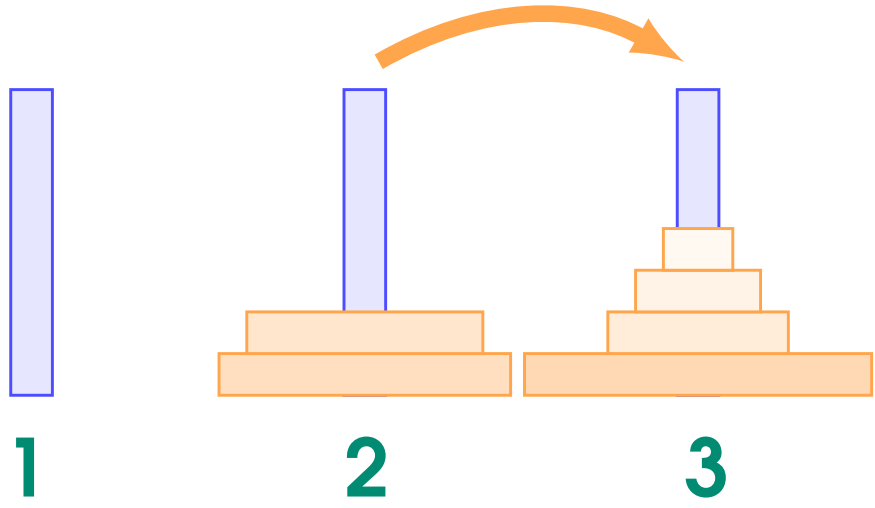
The Towers of Hanoi



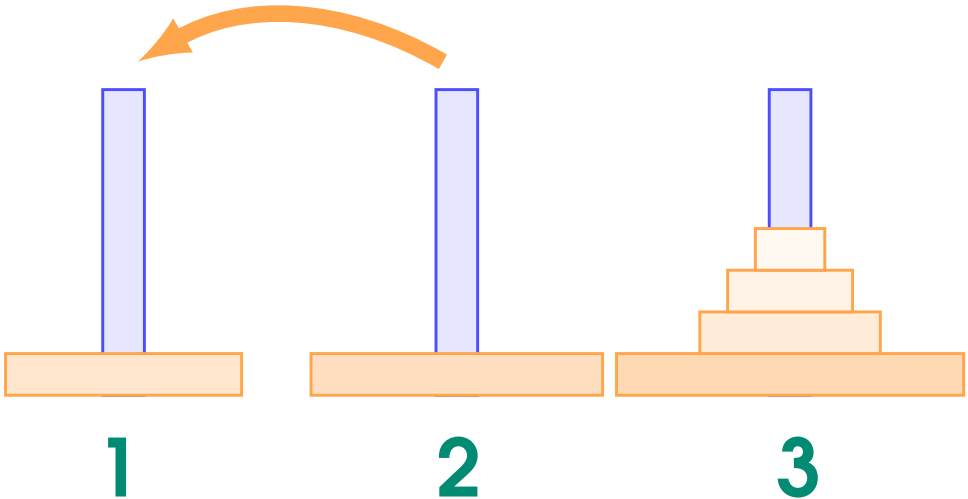
The Towers of Hanoi



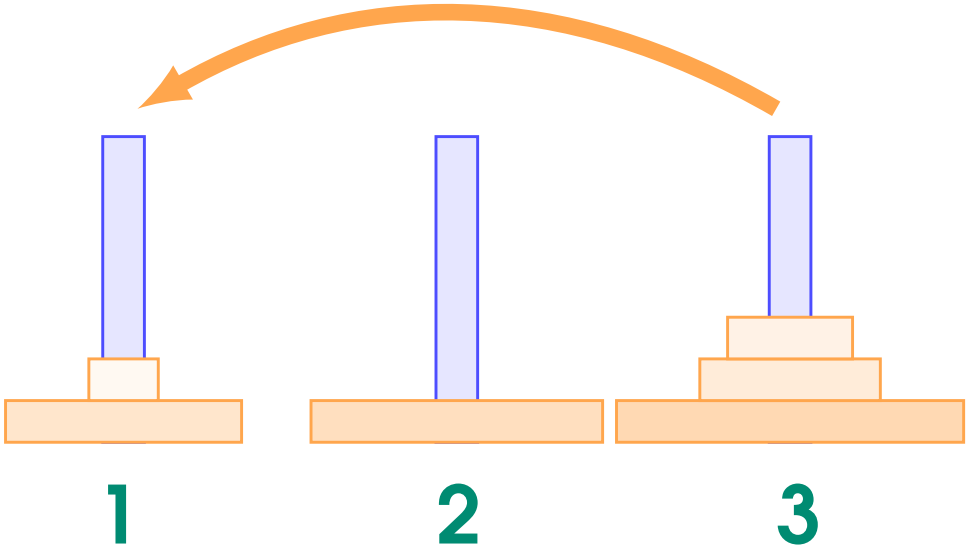
The Towers of Hanoi



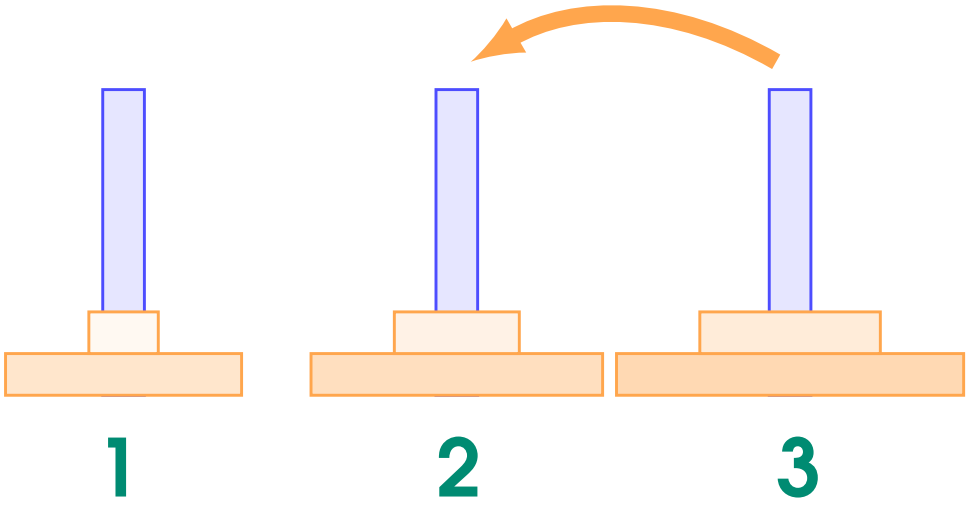
The Towers of Hanoi



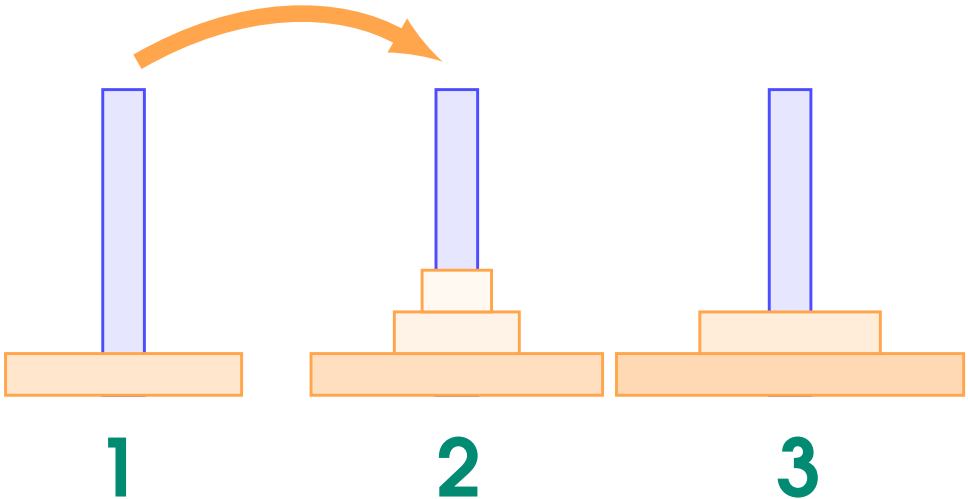
The Towers of Hanoi



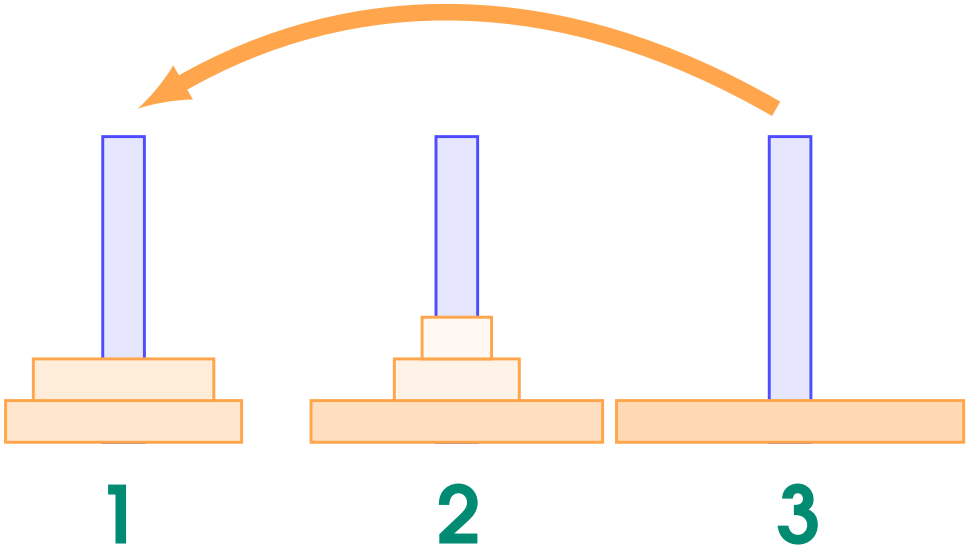
The Towers of Hanoi



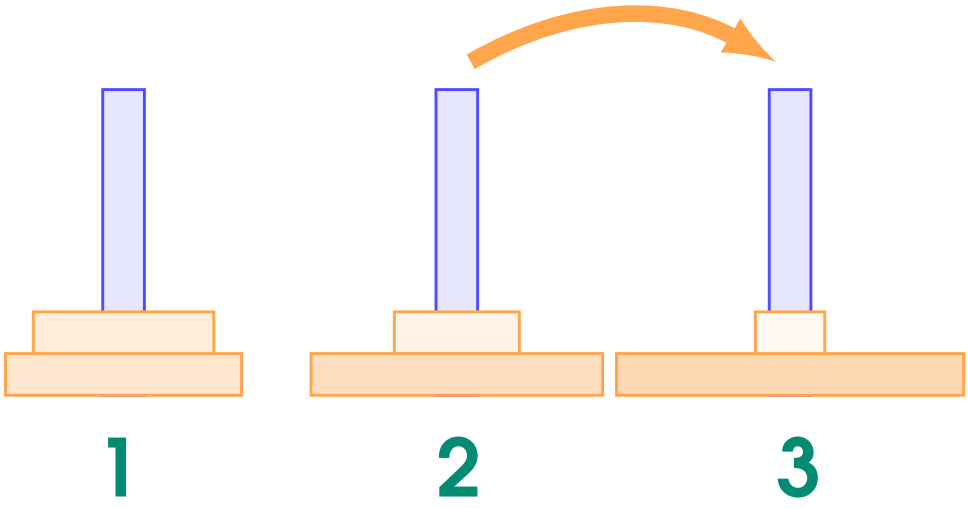
The Towers of Hanoi



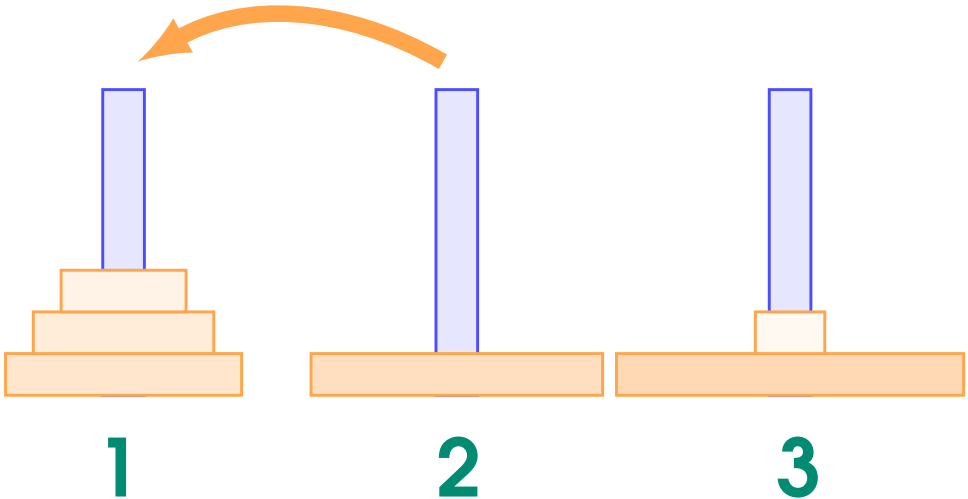
The Towers of Hanoi



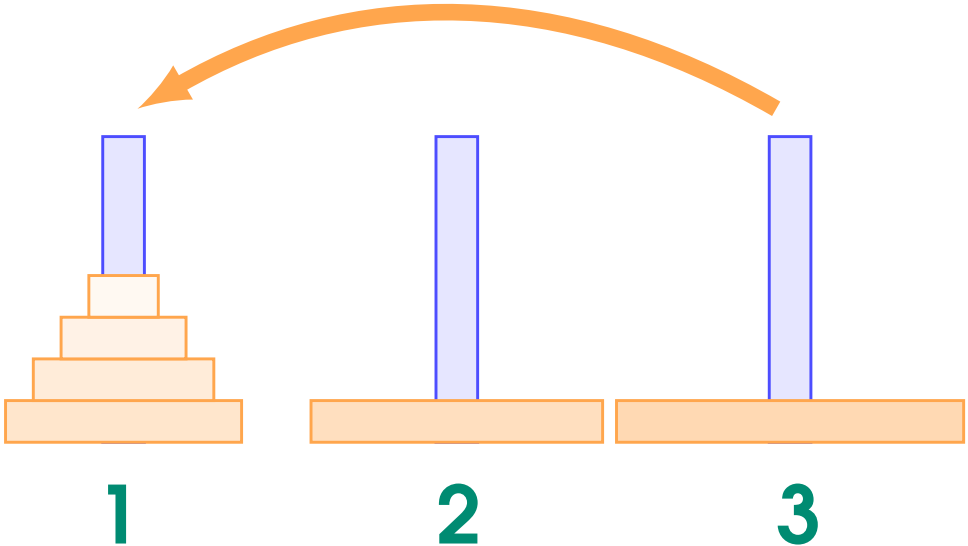
The Towers of Hanoi



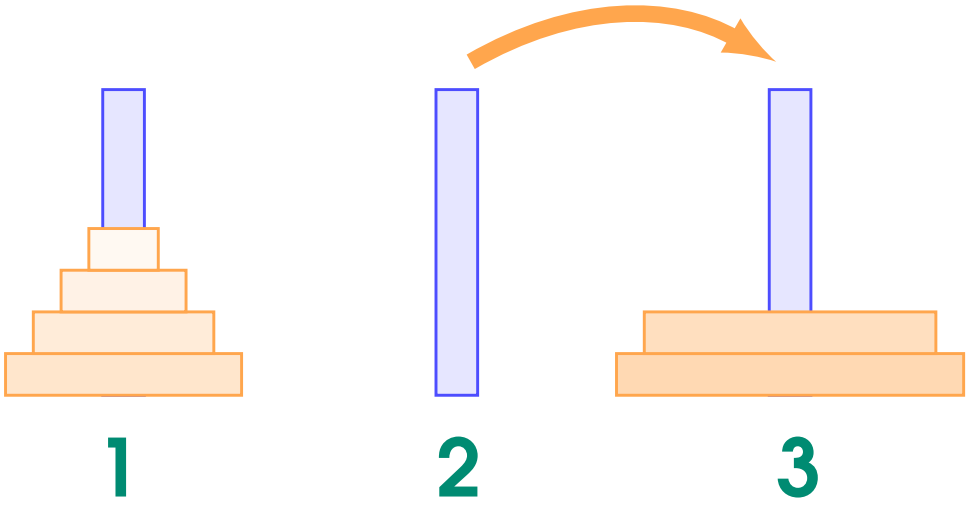
The Towers of Hanoi



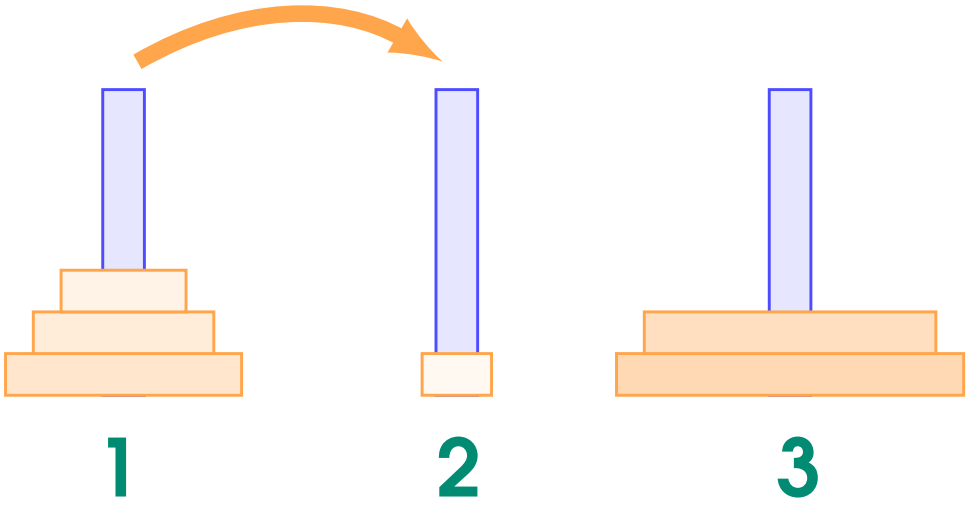
The Towers of Hanoi



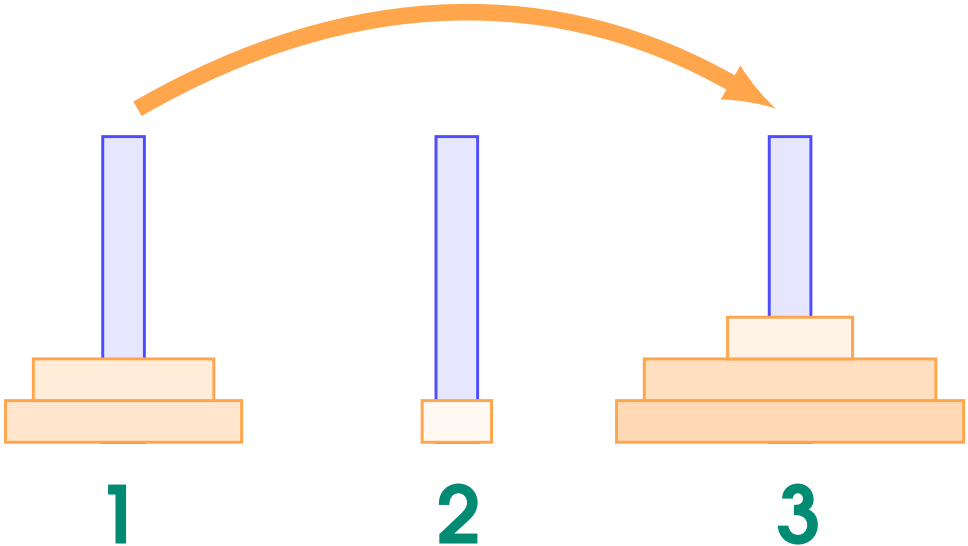
The Towers of Hanoi



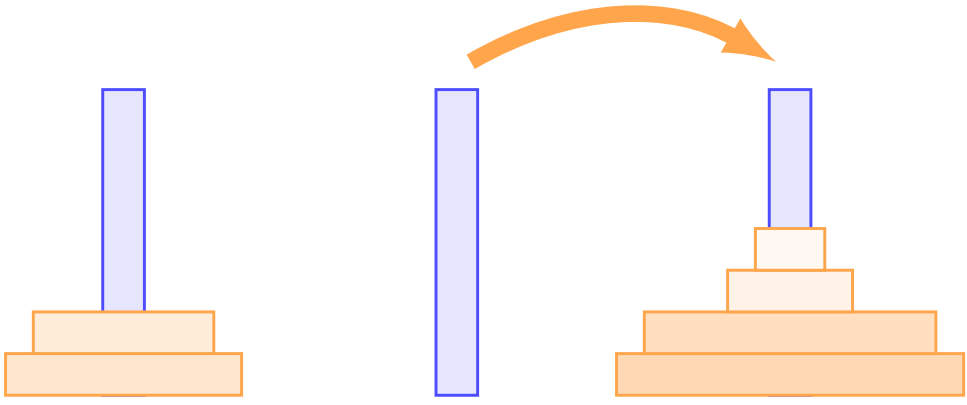
The Towers of Hanoi



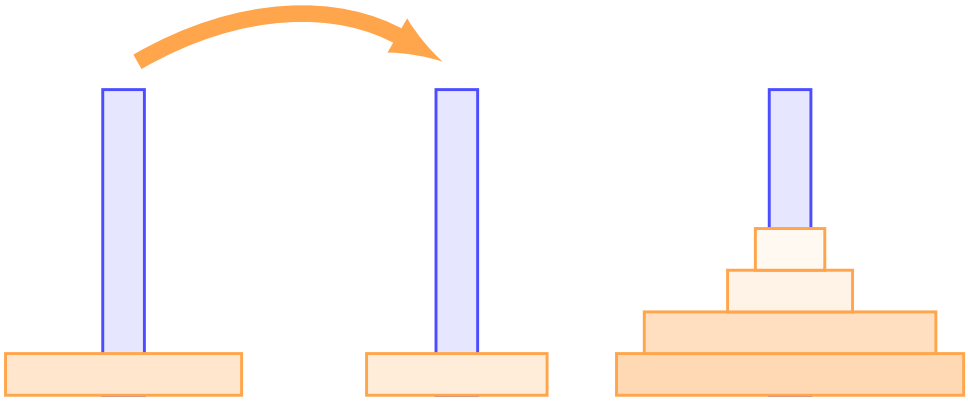
The Towers of Hanoi



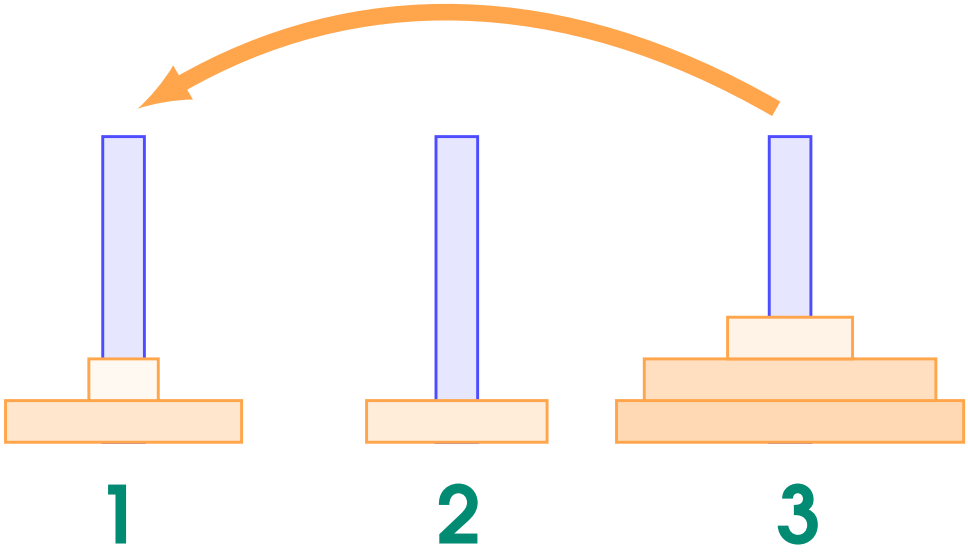
The Towers of Hanoi



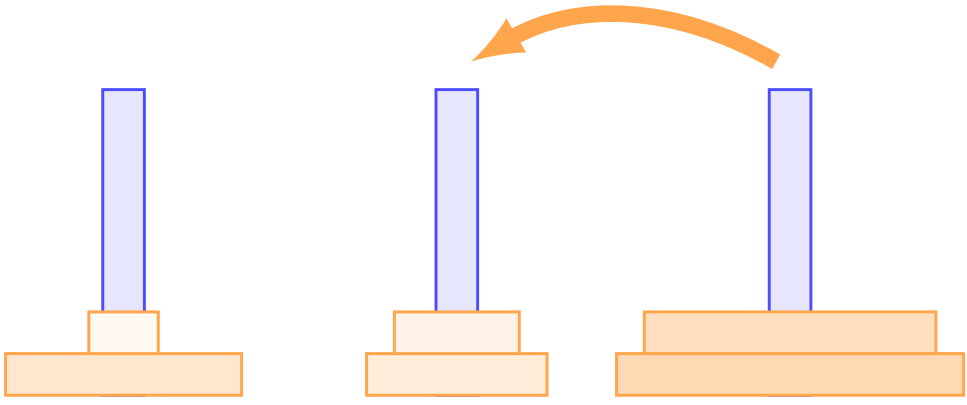
The Towers of Hanoi



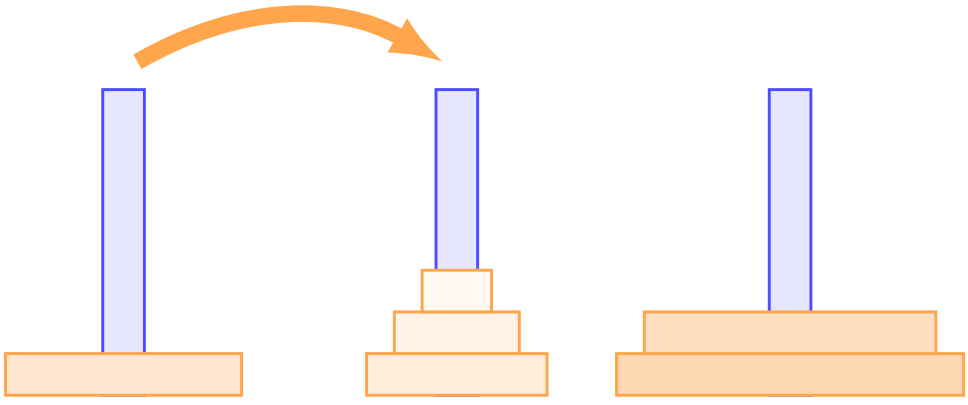
The Towers of Hanoi



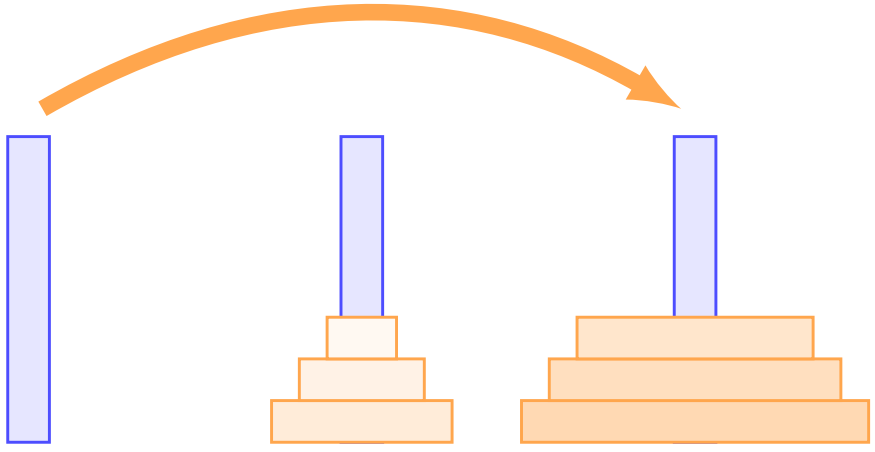
The Towers of Hanoi



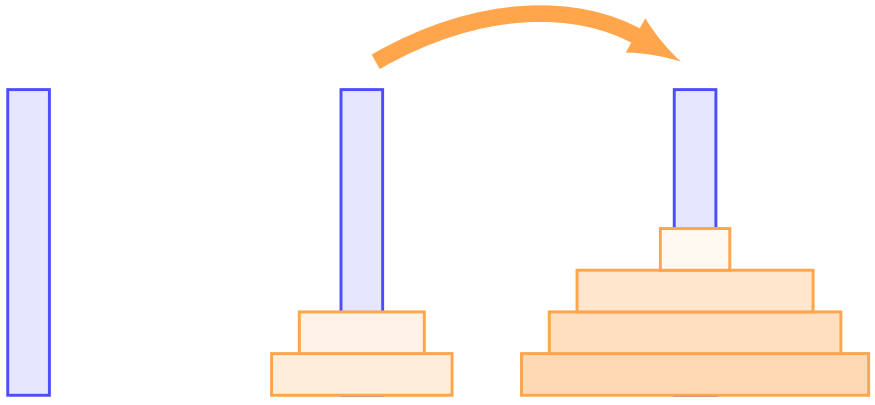
The Towers of Hanoi



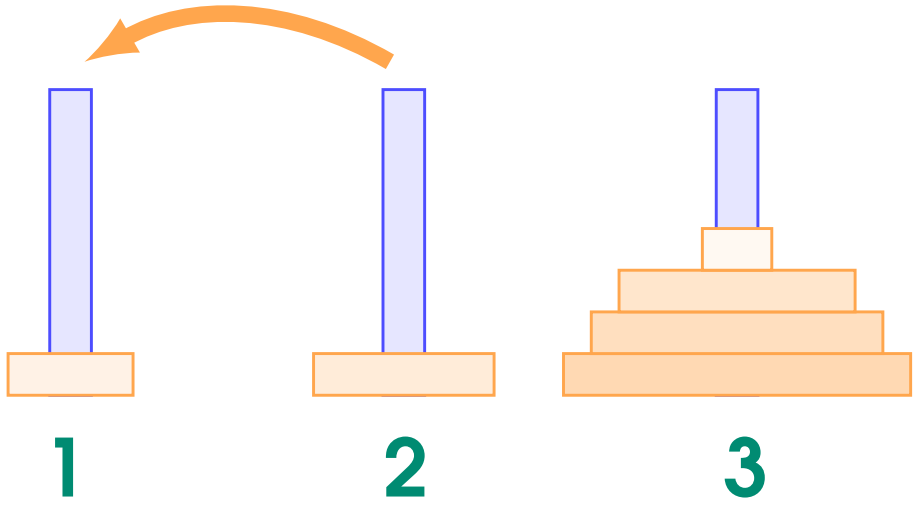
The Towers of Hanoi



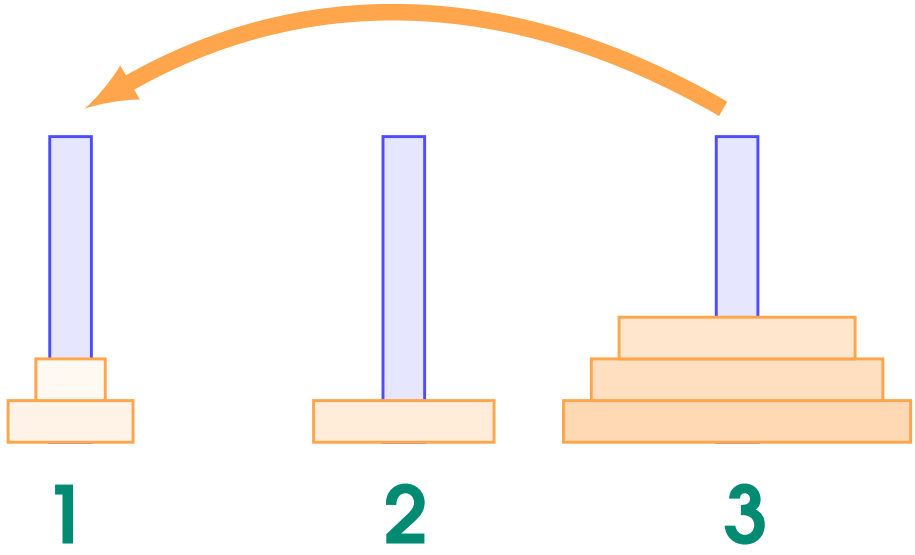
The Towers of Hanoi



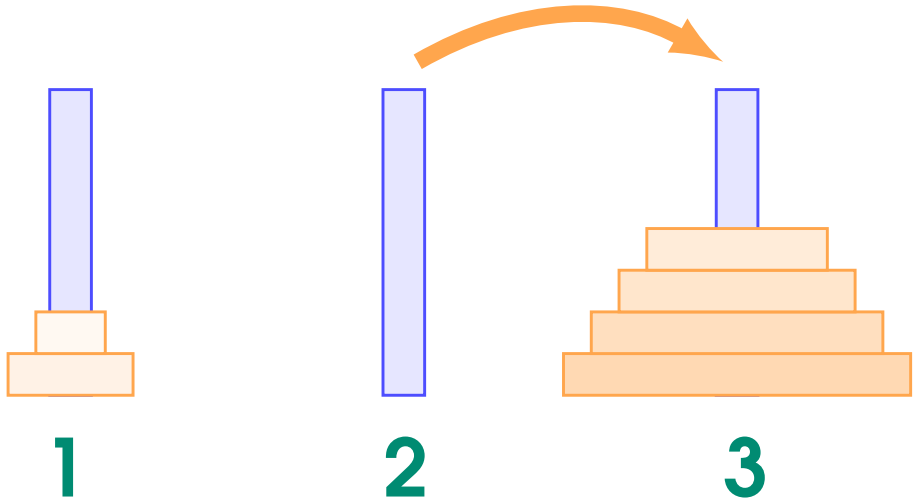
The Towers of Hanoi



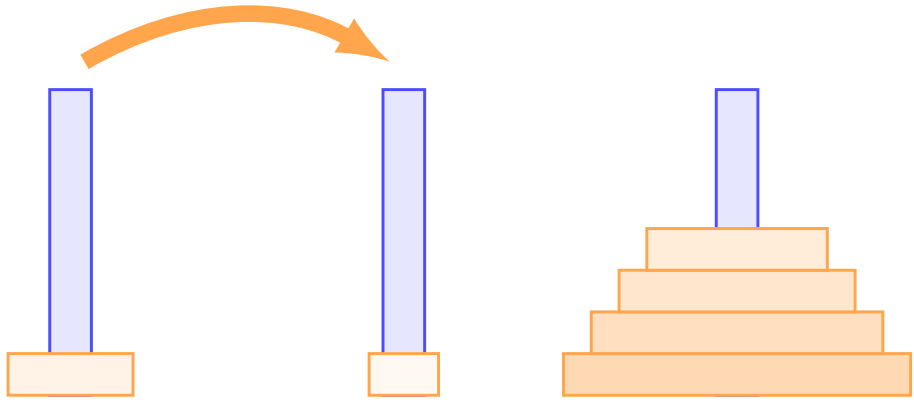
The Towers of Hanoi



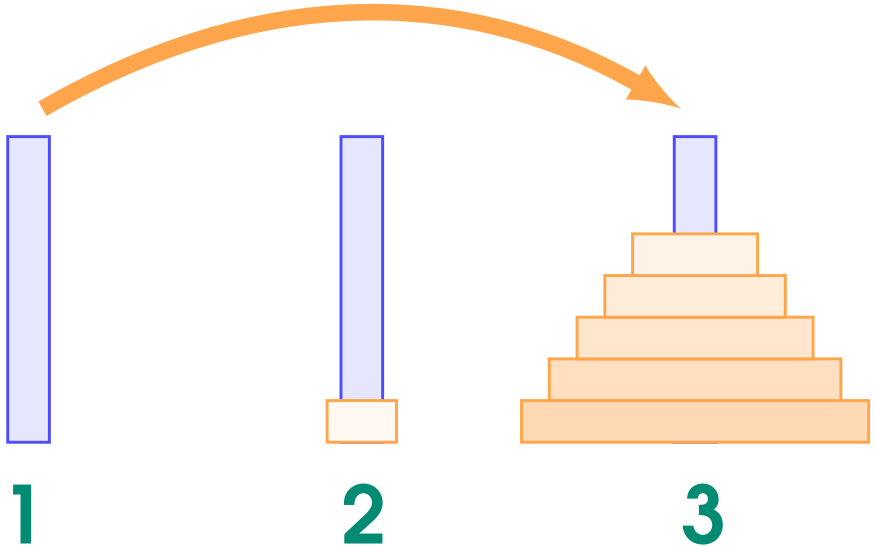
The Towers of Hanoi



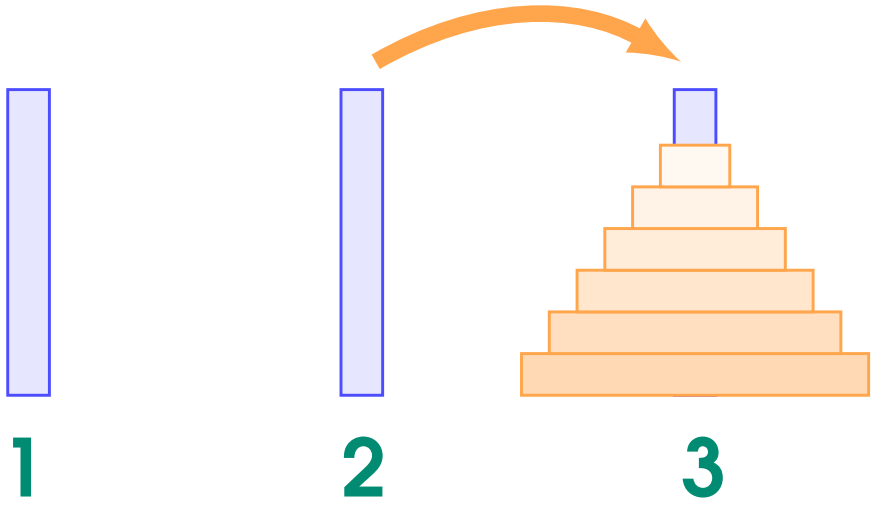
The Towers of Hanoi



The Towers of Hanoi



The Towers of Hanoi



The Towers of Hanoi

Theorem The number of steps required for the MoveTower algorithm satisfies the following recurrence relation:

$$T(1) = 1, \qquad T(n) = 2T(n-1) + 1$$

for $n > 1$.

The Towers of Hanoi

MoveTower(n, 1, 3):

1. **If** $n = 1$ **then** Move(1, 3)
2. **Else**
   3. MoveTower($n - 1$, 1, 2)
   4. Move(1, 3)
   5. MoveTower($n - 1$, 2, 3)

$$T(n) = T(n-1) + 1 + T(n-1)$$

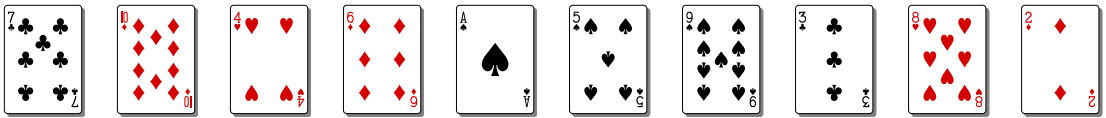
The Towers of Hanoi

n	1	2	3	4	5	6	7	8	$\cdots$
$T(n)$	1	3	7	15	31	63	127	255	$\cdots$

The Merge-Sort Algorithm

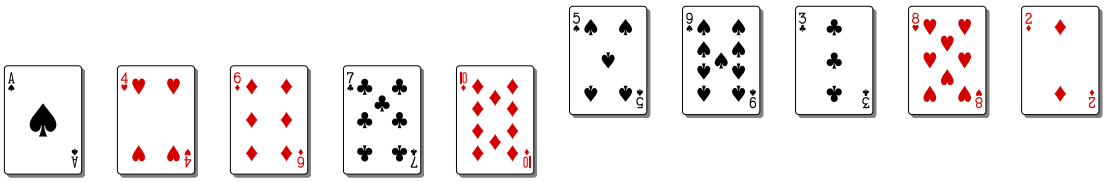
The Merge-Sort Algorithm

MERGESORT $([a_1, \ldots, a_n])$:



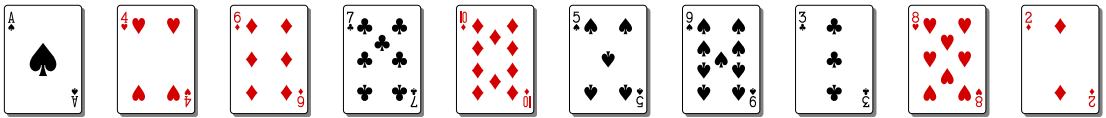
The Merge-Sort Algorithm

MERGESORT $([a_1, \ldots, a_n])$:



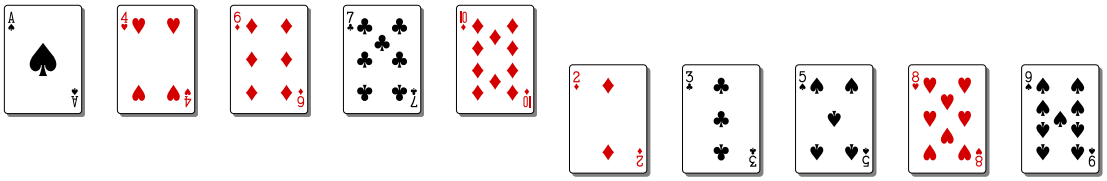
The Merge-Sort Algorithm

MERGESORT $([a_1, \ldots, a_n])$:



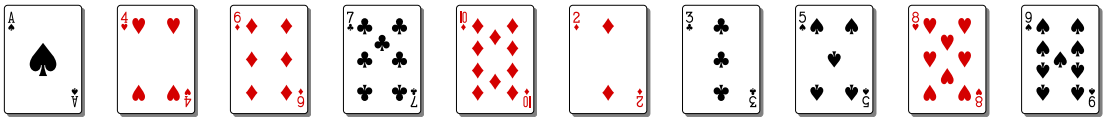
The Merge-Sort Algorithm

MERGESORT $([a_1, \ldots, a_n])$:



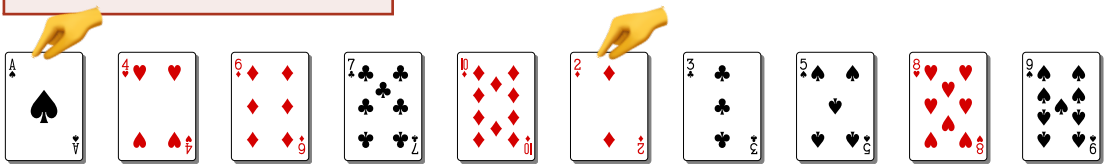
The Merge-Sort Algorithm

MERGESORT $([a_1, \ldots, a_n])$:



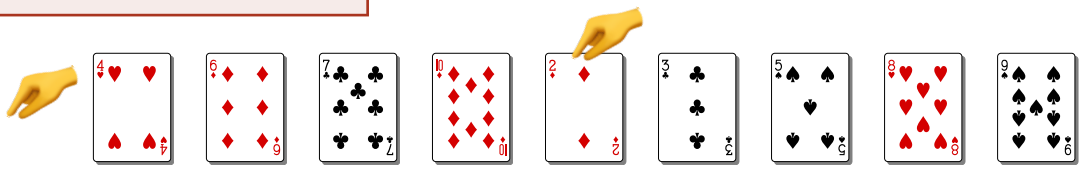
The Merge-Sort Algorithm

MERGESORT $([a_1, \ldots, a_n])$:



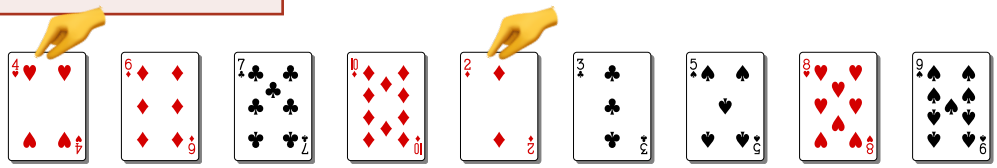
The Merge-Sort Algorithm

MERGESORT $([a_1, \ldots, a_n])$:



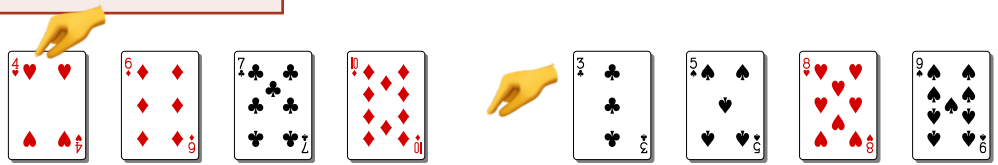
The Merge-Sort Algorithm

MERGESORT $([a_1, \ldots, a_n])$:



The Merge-Sort Algorithm

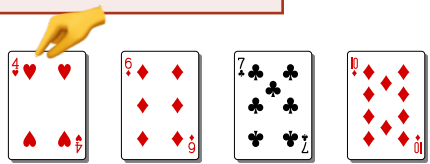
MERGESORT $([a_1, \ldots, a_n])$:

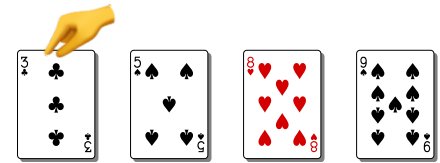


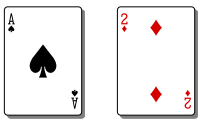
<footer>5CCS2FC2 Foundations of Computing II C.Hampson, King's College London 14 / 21</footer>

The Merge-Sort Algorithm

MERGESORT $([a_1, \ldots, a_n])$:

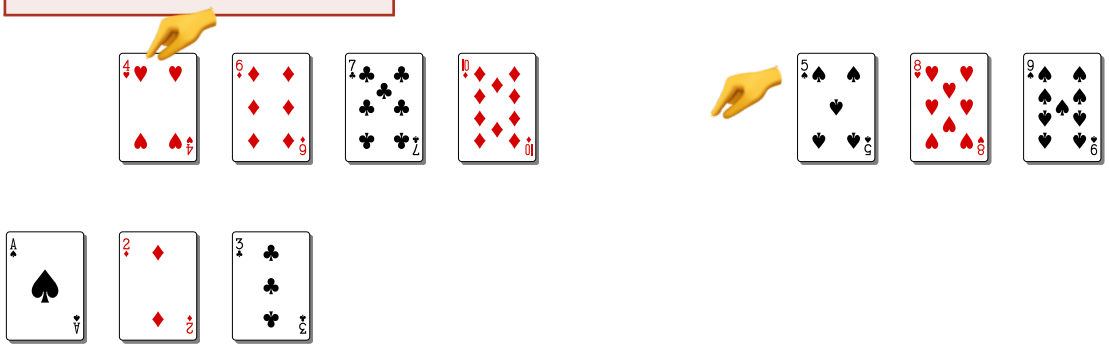






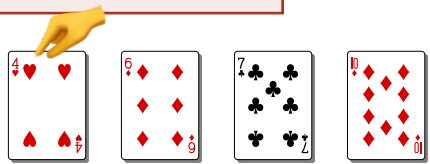
The Merge-Sort Algorithm

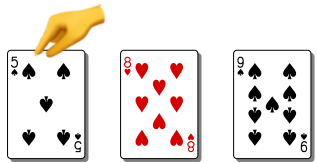
MERGESORT $([a_1, \ldots, a_n])$:



The Merge-Sort Algorithm

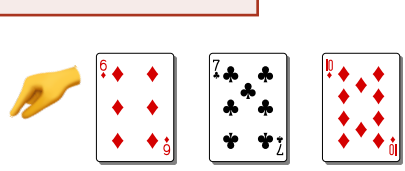
MERGESORT $([a_1, \ldots, a_n])$:

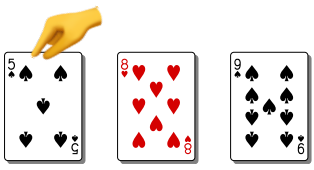




The Merge-Sort Algorithm

MERGESORT $([a_1, \ldots, a_n])$:

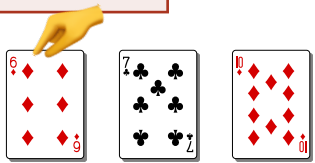


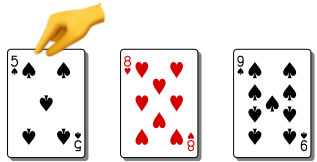


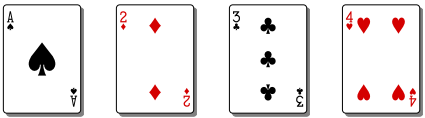
<footer>5CCS2FC2 Foundations of Computing II C.Hampson, King's College London 14 / 21</footer>

The Merge-Sort Algorithm

MERGESORT $([a_1, \ldots, a_n])$:

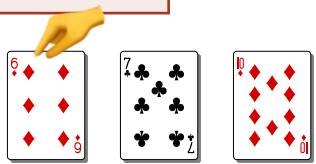


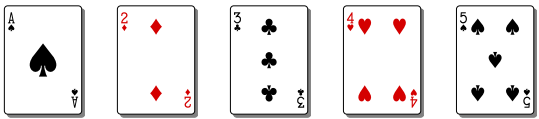


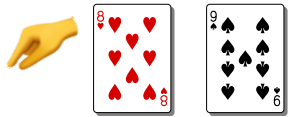


The Merge-Sort Algorithm

MERGESORT $([a_1, \ldots, a_n])$:

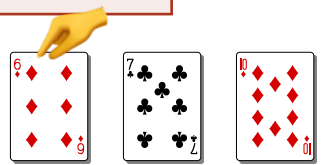


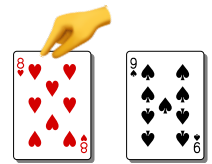




The Merge-Sort Algorithm

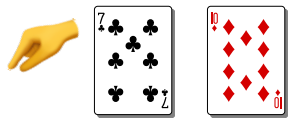
MERGESORT $([a_1, \ldots, a_n])$:

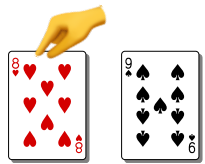


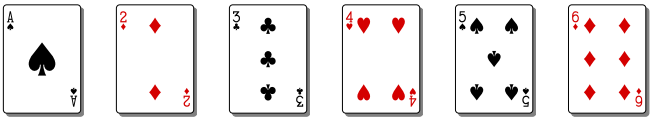


The Merge-Sort Algorithm

MERGESORT $([a_1, \ldots, a_n])$:

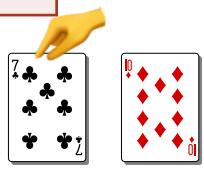


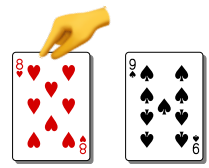


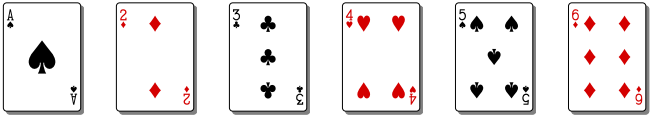


The Merge-Sort Algorithm

MERGESORT $([a_1, \ldots, a_n])$:

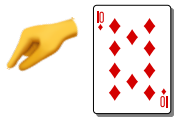


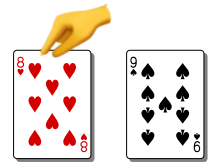


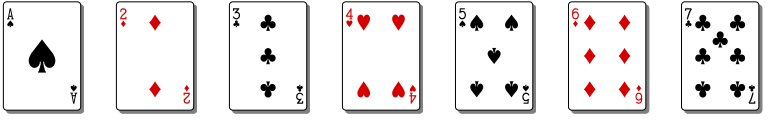


The Merge-Sort Algorithm

MERGESORT $([a_1, \ldots, a_n])$:

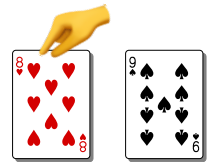






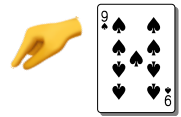
The Merge-Sort Algorithm

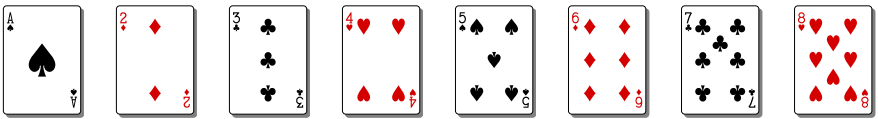
MERGESORT $([a_1, \ldots, a_n])$:



The Merge-Sort Algorithm

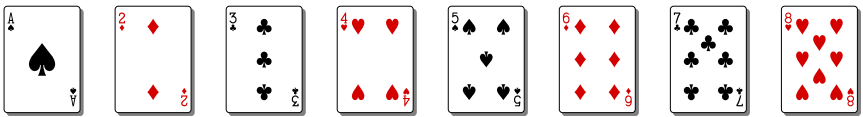
MERGESORT $([a_1, \ldots, a_n])$:





The Merge-Sort Algorithm

MERGESORT $([a_1, \ldots, a_n])$:

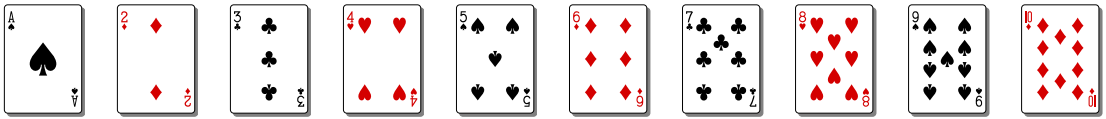


The Merge-Sort Algorithm

MERGESORT $([a_1, \ldots, a_n])$:

The Merge-Sort Algorithm

MERGESORT $([a_1, \ldots, a_n])$:



The Merge-Sort Algorithm

MERGESORT $([a_1, \ldots, a_n])$:

1. **If** $n = 1$ **then Return** $[a_1]$
2. **Else**
   3. $L \leftarrow$ MERGESORT $\left(\left[a_1, \ldots, a_{\lfloor n/2 \rfloor}\right]\right)$
   4. $R \leftarrow$ MERGESORT $\left(\left[a_{\lfloor n/2 \rfloor + 1}, \ldots, a_n\right]\right)$
   5. $A \leftarrow$ MERGE(L, R)
   6. **Return** A

The Merge-Sort Algorithm

Theorem The number of steps required for the MERGESORT algorithm satisfies the following recurrence relation:

$$T(1) = 1, \qquad T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + n$$

for $n > 1$.

The Merge-Sort Algorithm

Theorem The number of steps required for the MERGESORT algorithm satisfies the following recurrence relation:

$$T(1) = 1, \qquad T(n) \approx 2T\left(\left\lceil \frac{n}{2} \right\rceil\right) + n$$

for $n > 1$.

The Merge-Sort Algorithm

MERGESORT $([a_1, \ldots, a_n])$:

1. **If** $n = 1$ **then Return** $[a_1]$
2. **Else**
3. $L \leftarrow$ MERGESORT $([a_1, \ldots, a_{\lfloor n/2 \rfloor}])$
4. $R \leftarrow$ MERGESORT $([a_{\lfloor n/2 \rfloor + 1}, \ldots, a_n])$
5. $A \leftarrow$ MERGE(L, R)
6. **Return** A

$$T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + n$$

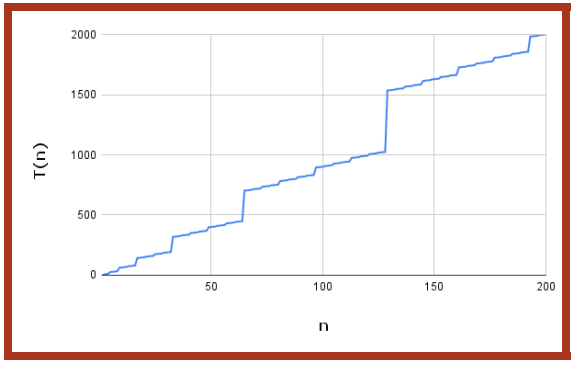
The Merge-Sort Algorithm

n	1	2	3	4	5	6	7	8	$\cdots$
$T(n)$	1	4	11	12	27	28	31	32	$\cdots$

The Merge-Sort Algorithm

B3 | fx | =2*VLOOKUP(ceiling(A3/2),A:B,2)+A3

	A	B	C	D
1	n	T(n)		
2	1	1		
3	2	4		
4	3	11		
5	4	12		
6	5	27		
7	6	28		
8	7	31		
9	8	32		
10	9	63		
11	10	64		
12	11	67		
13	12	68		
14	13	75		
15	14	76		
16	15	79		
17	16	80		
18	17	143		
19	18	144		



Disclaimer: Spreadsheets should not be used for recording and/or storing critical or important business or government data. Use with caution.

End of Slides!