Winter 2016
COMP-250: Introduction to Computer Science

Lecture 11, February 16, 2016

Please participate in a 10-minute survey on the undergraduate student experience in SOCS!

Complete the survey before Feb. 22: surveys.mcgill.ca/limesurvey/index.php?sid=94659

This study is conducted on behalf of the Undergraduate Committee and the Women@SOCS Committee in the School of Computer Science in collaboration with the Computer Science Undergraduate Society (CSUS). The purpose of this survey is to assess the strengths and weaknesses of SOCS, and to recommend measures to improve the student experience.

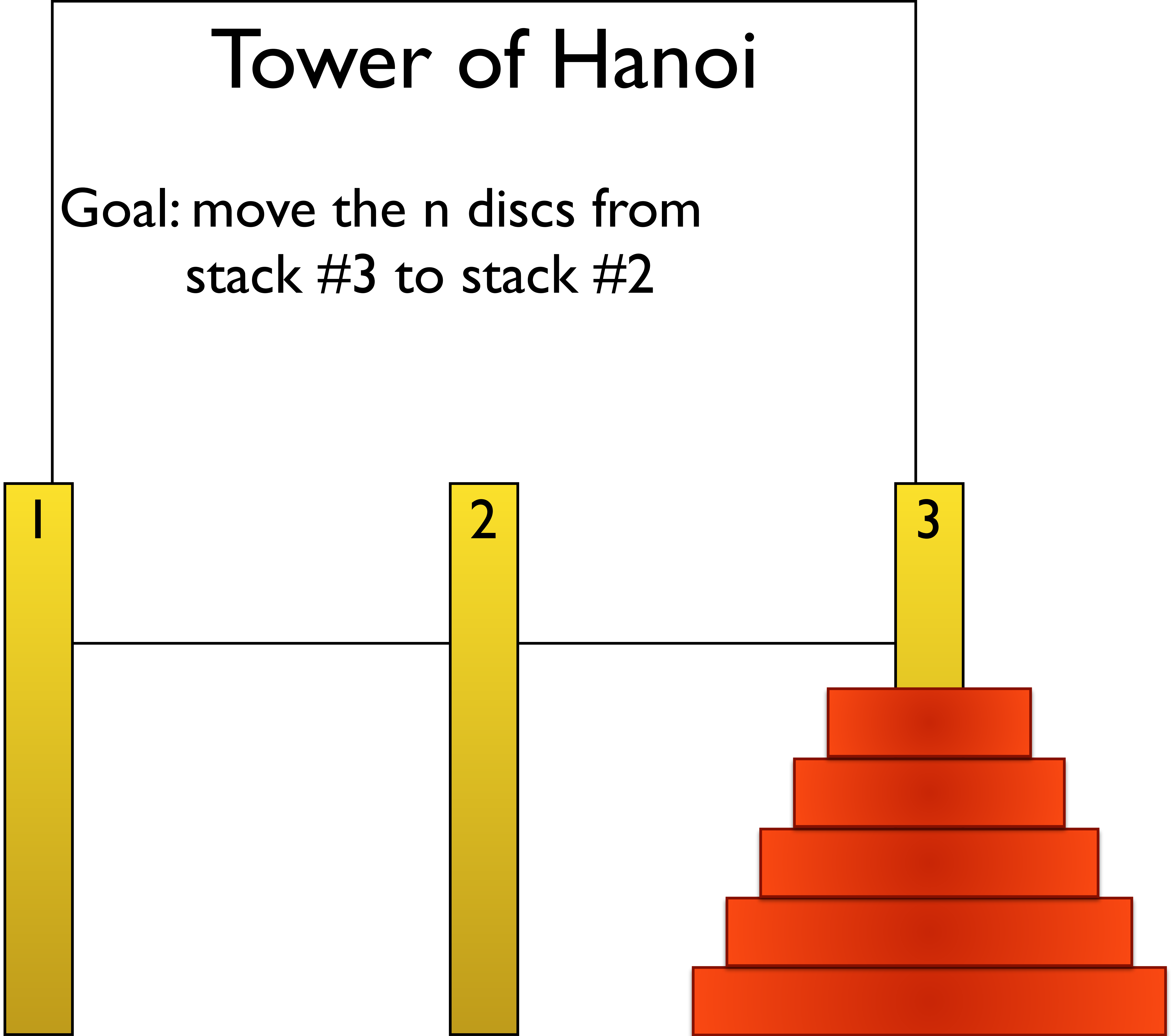
CSUS

For any questions, email Brigitte Pientka at bpientka@cs.mcgill.ca

McGill
School of Computer Science

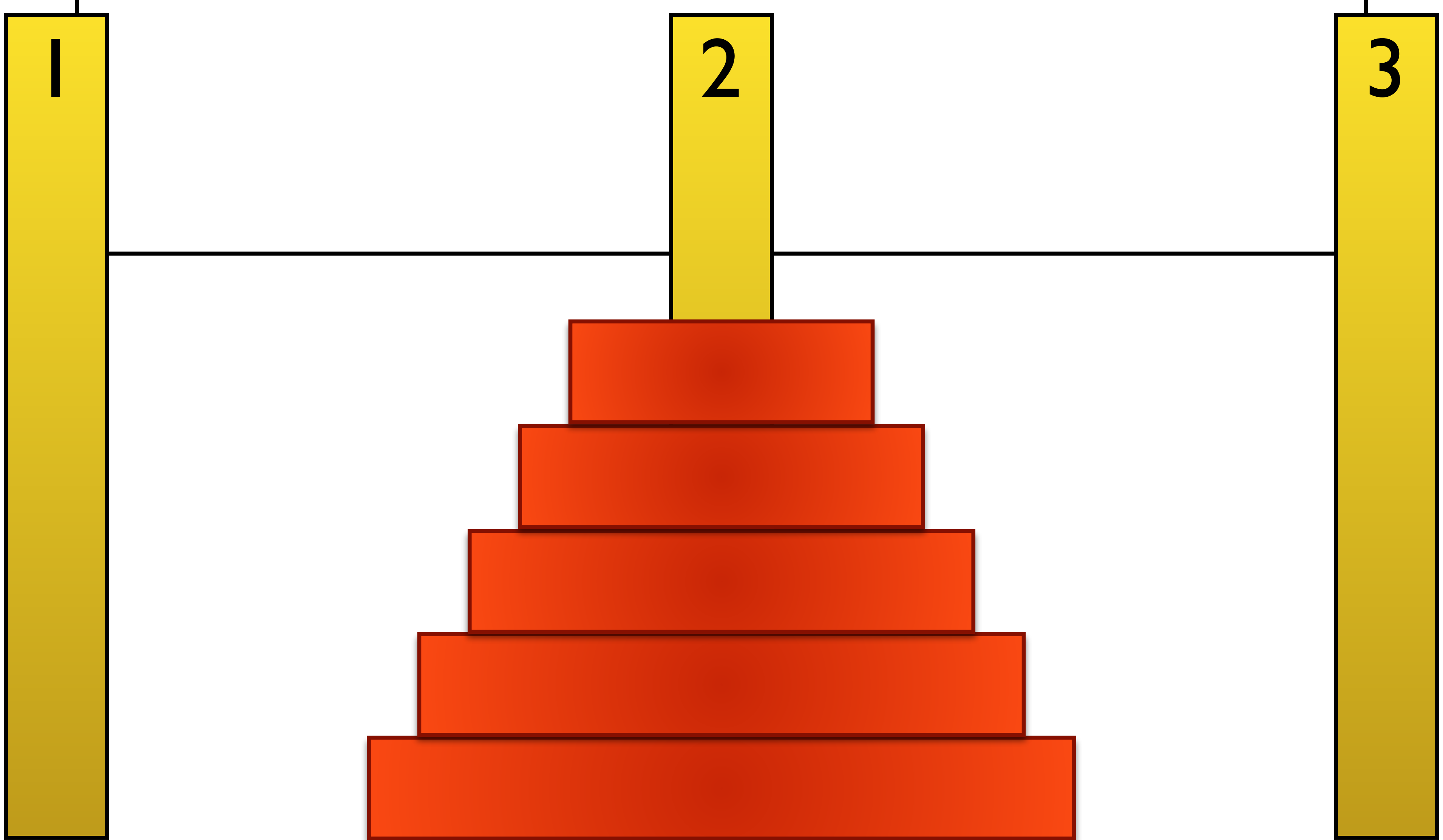
Tower of Hanoi

Goal: move the n discs from stack #3 to stack #2



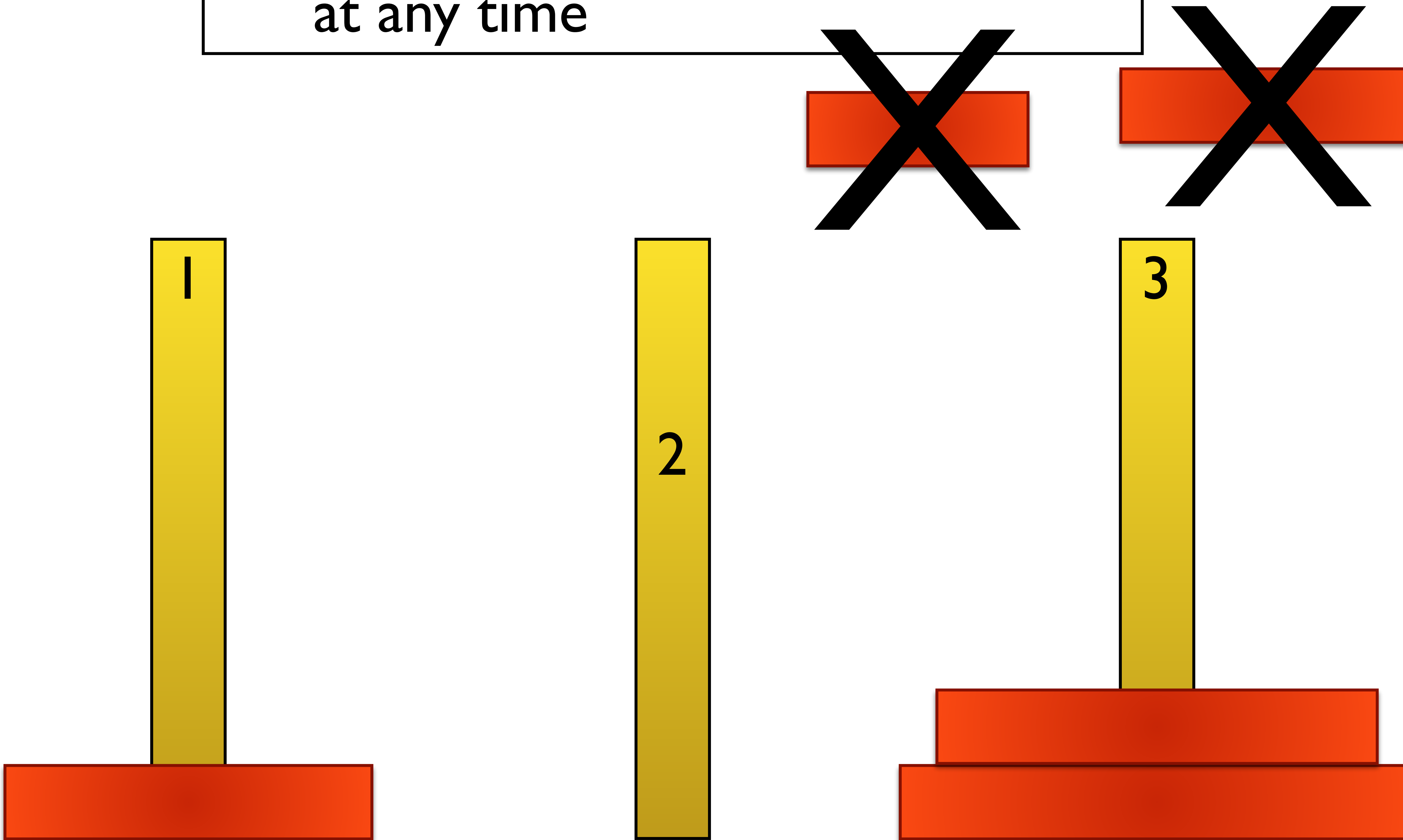
Tower of Hanoi

Goal: move the n discs from stack #3 to stack #2



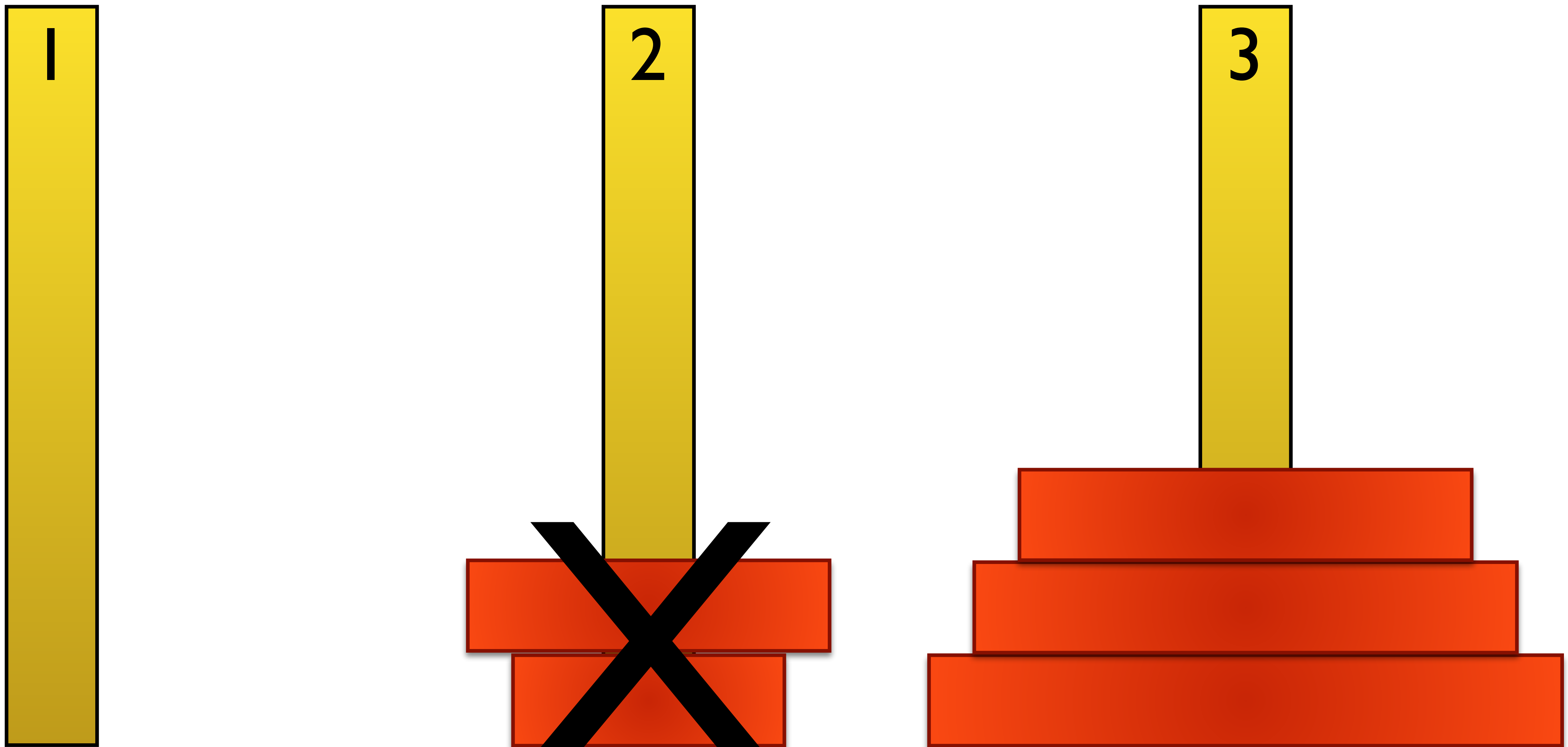
Tower of Hanoi

- allowing only one disc removed at any time



Tower of Hanoi

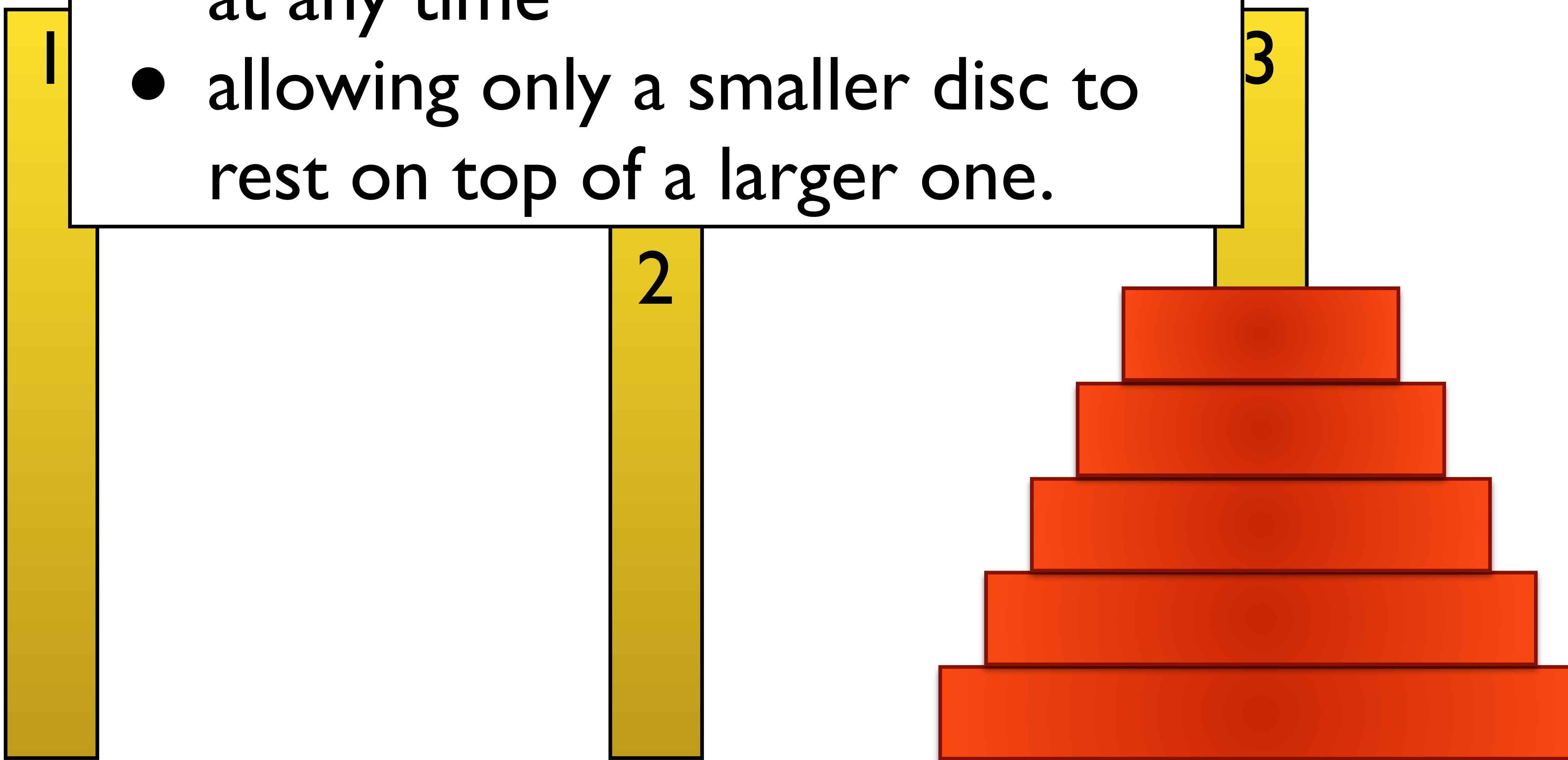
- allowing only a smaller disc to rest on top of a larger one.



Tower of Hanoi

Goal: move the n discs from stack #3 to stack #2 while

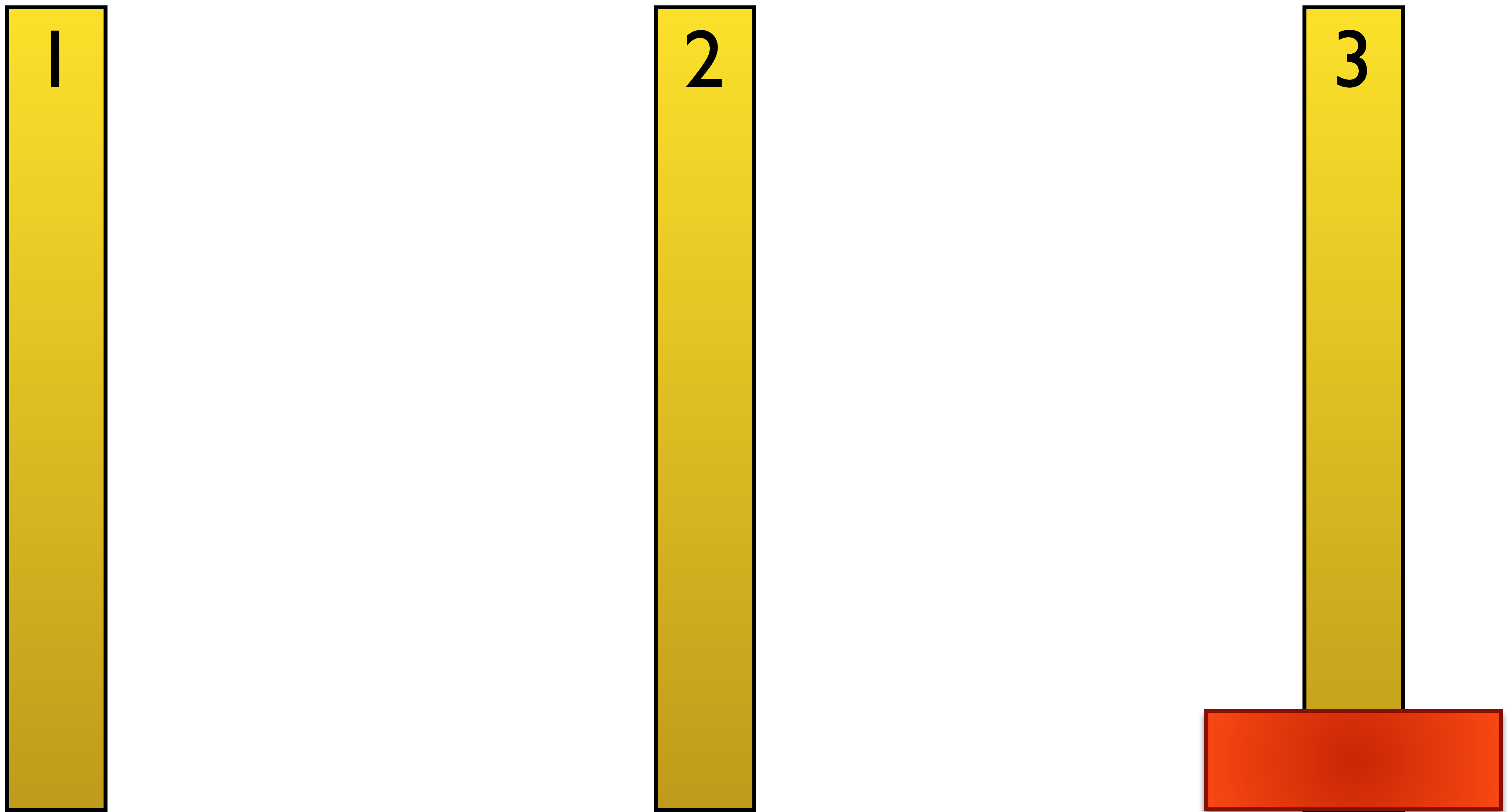
- allowing only one disc removed at any time
- allowing only a smaller disc to rest on top of a larger one.



Hanoi(1,S3,S2,S1)

<u>Base case:</u>

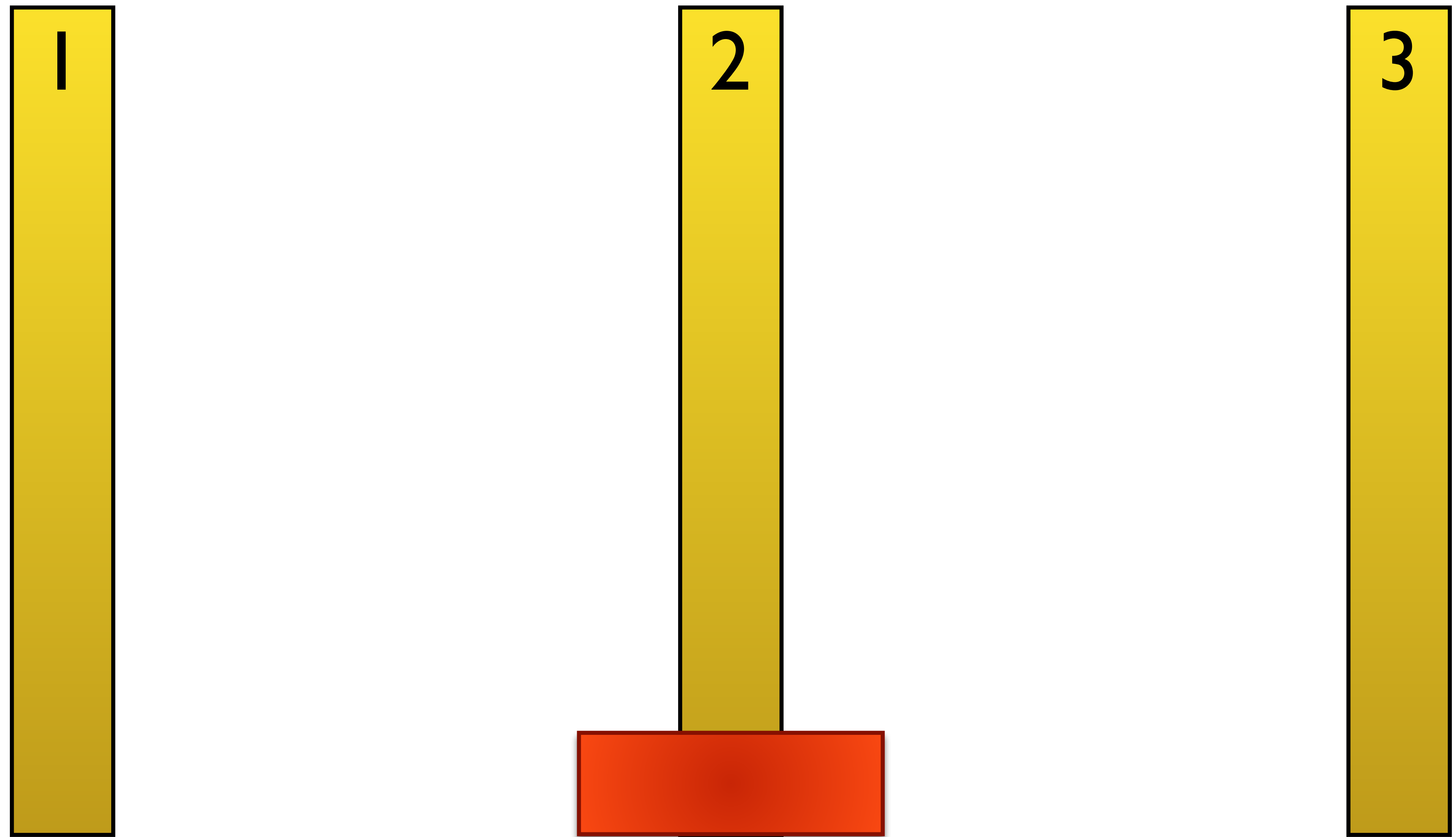
move disc 1 from S3 to S2



Hanoi(1,S3,S2,S1)

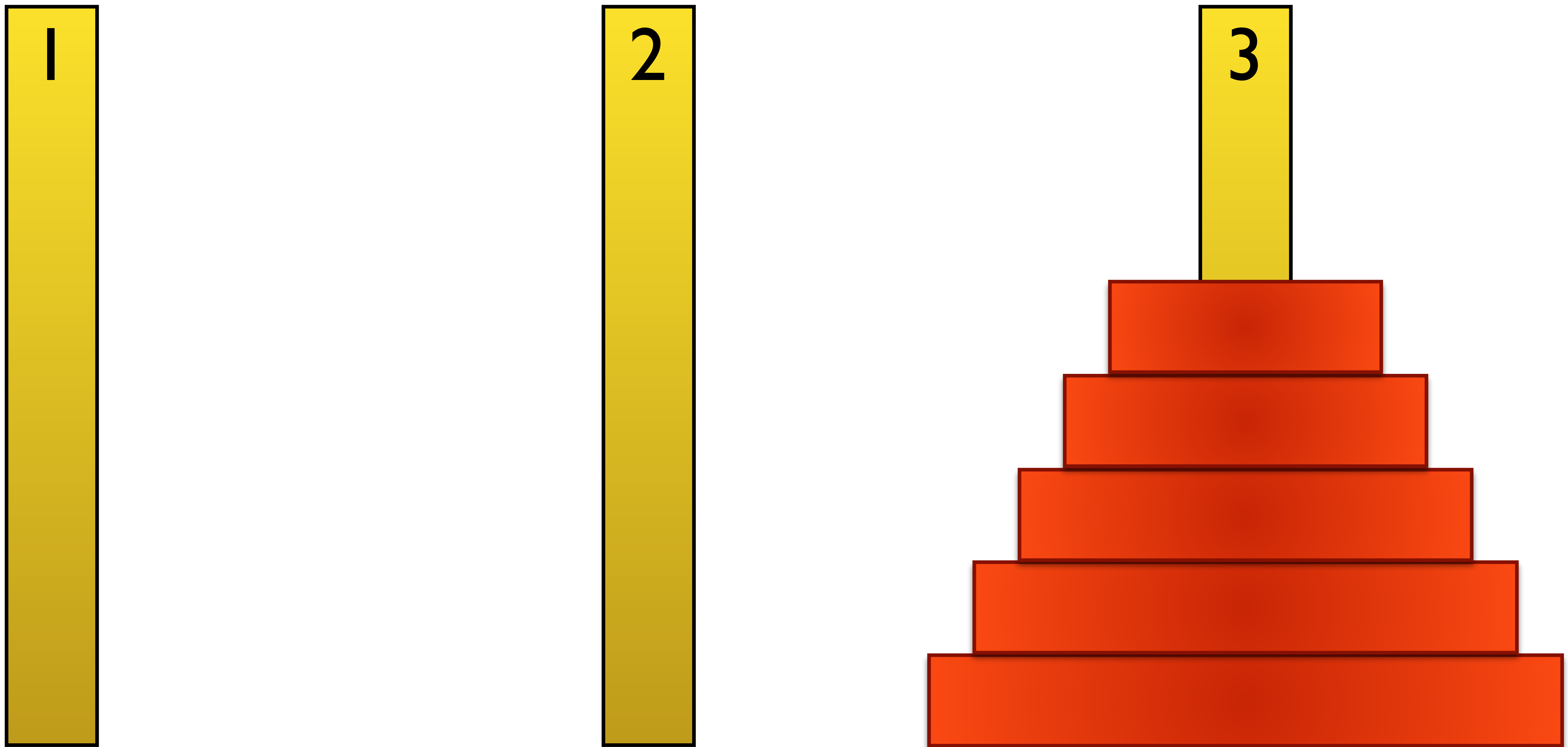
<u>Base case:</u>

move disc 1 from S3 to S2



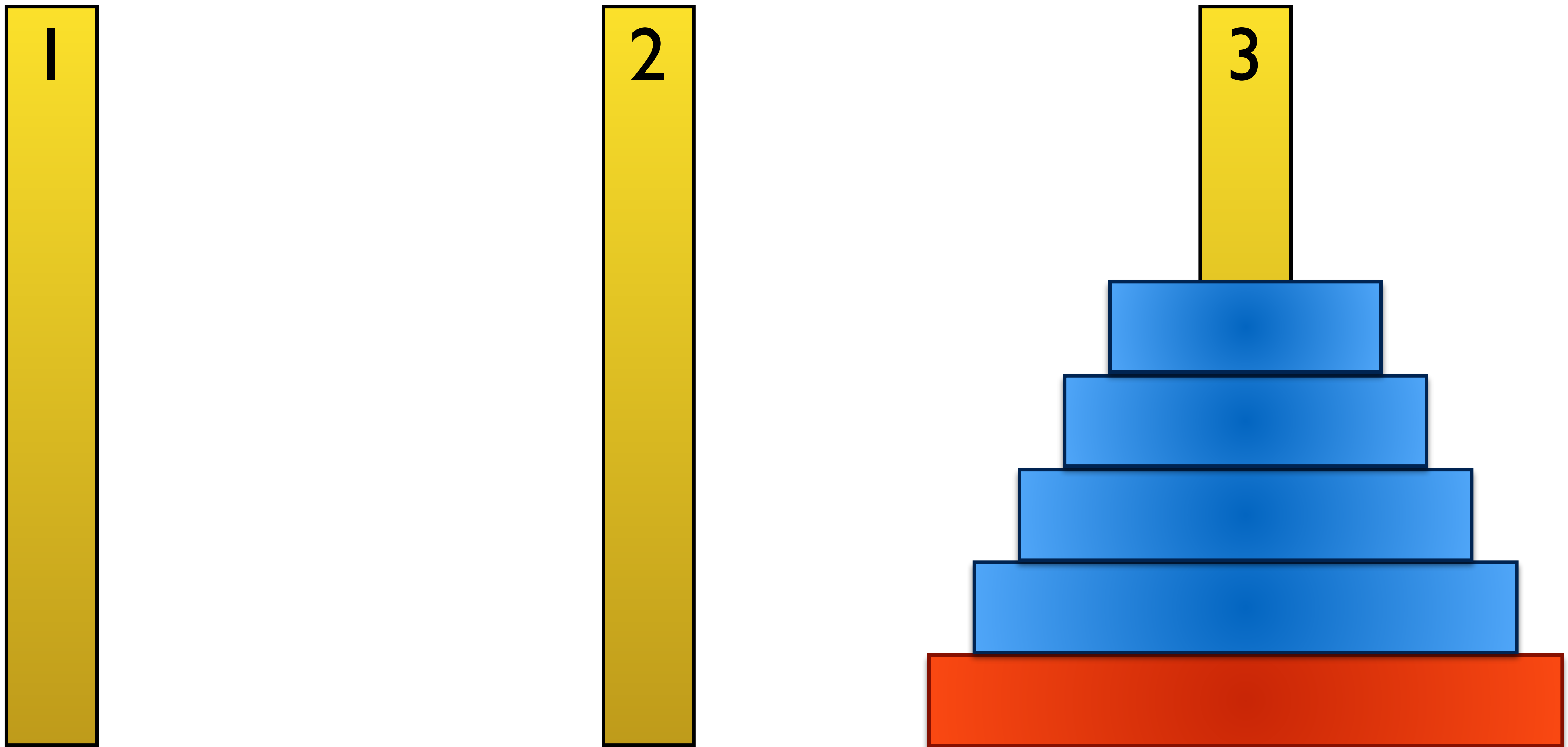
Hanoi(n,S3,S2,S1) // n≥1

if n>1then Hanoi(n-1,S3,S1,S2)
move disc n from S3 to S2
if n>1then Hanoi(n-1,S1,S2,S3)



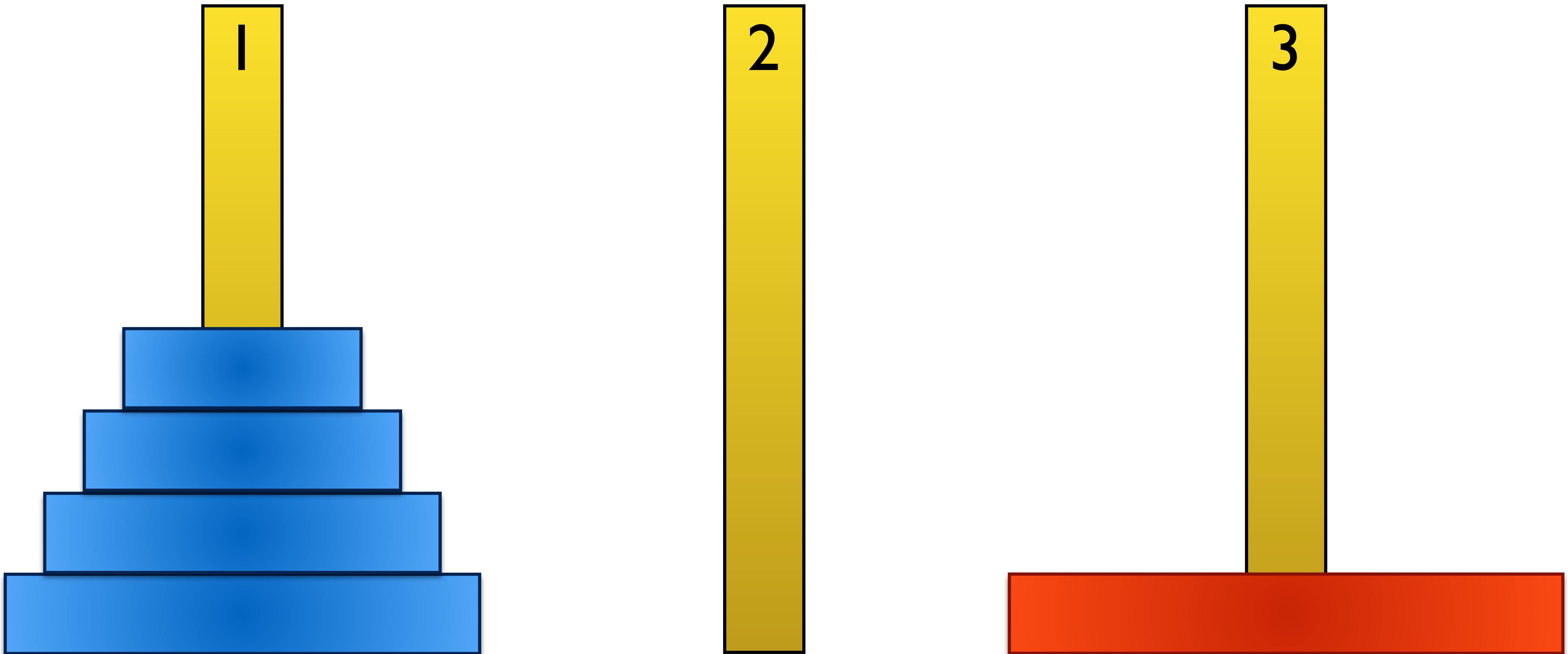
Hanoi(n,S3,S2,S1)   // $n \geq 1$

if n>1 then Hanoi(n-1,S3,S1,S2)
move disc n from S3 to S2
if n>1 then Hanoi(n-1,S1,S2,S3)



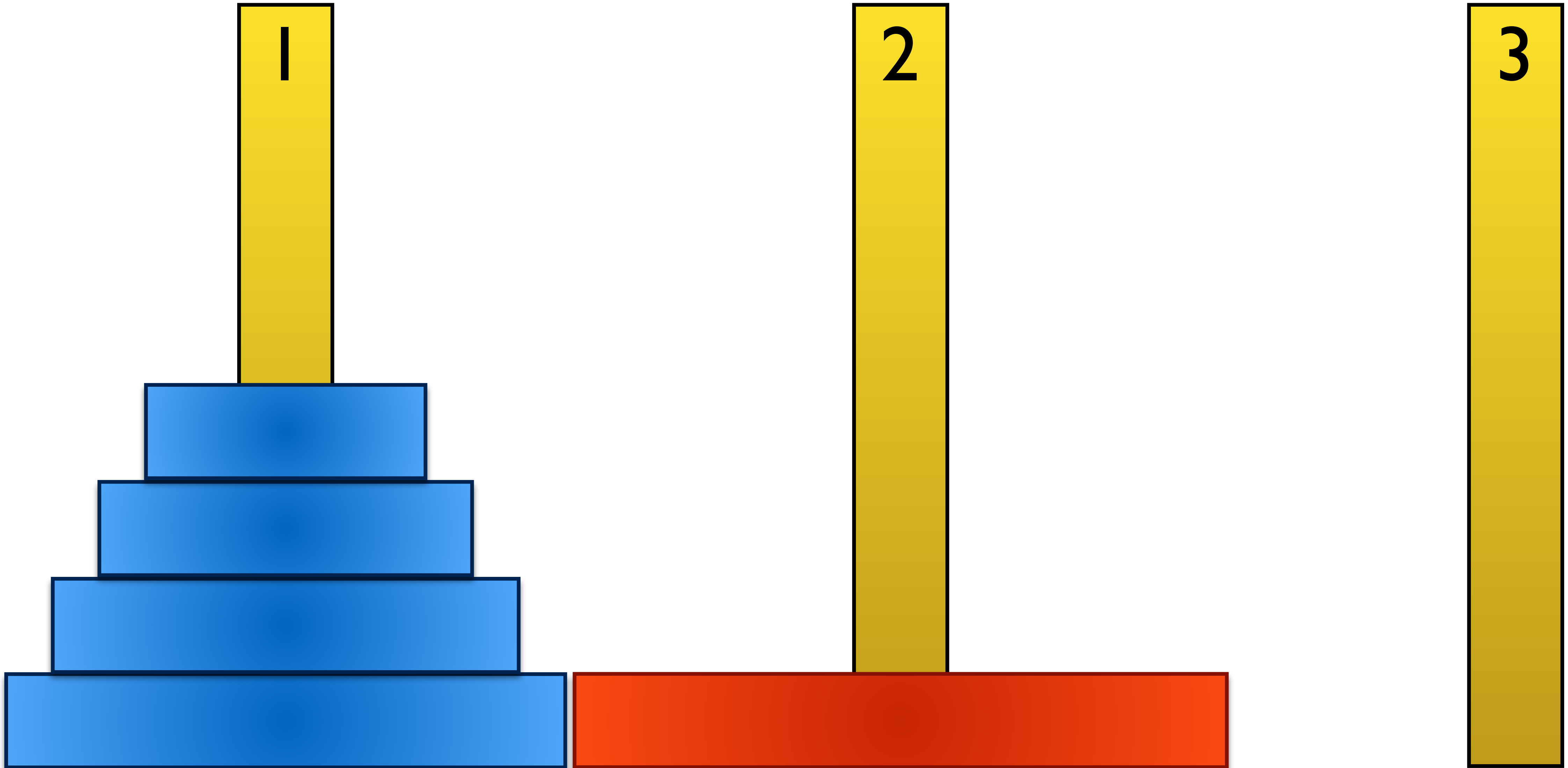
Hanoi(n,S3,S2,S1)   // n≥1

if n>1 then Hanoi(n-1,S3,S1,S2)
move disc n from S3 to S2
if n>1 then Hanoi(n-1,S1,S2,S3)



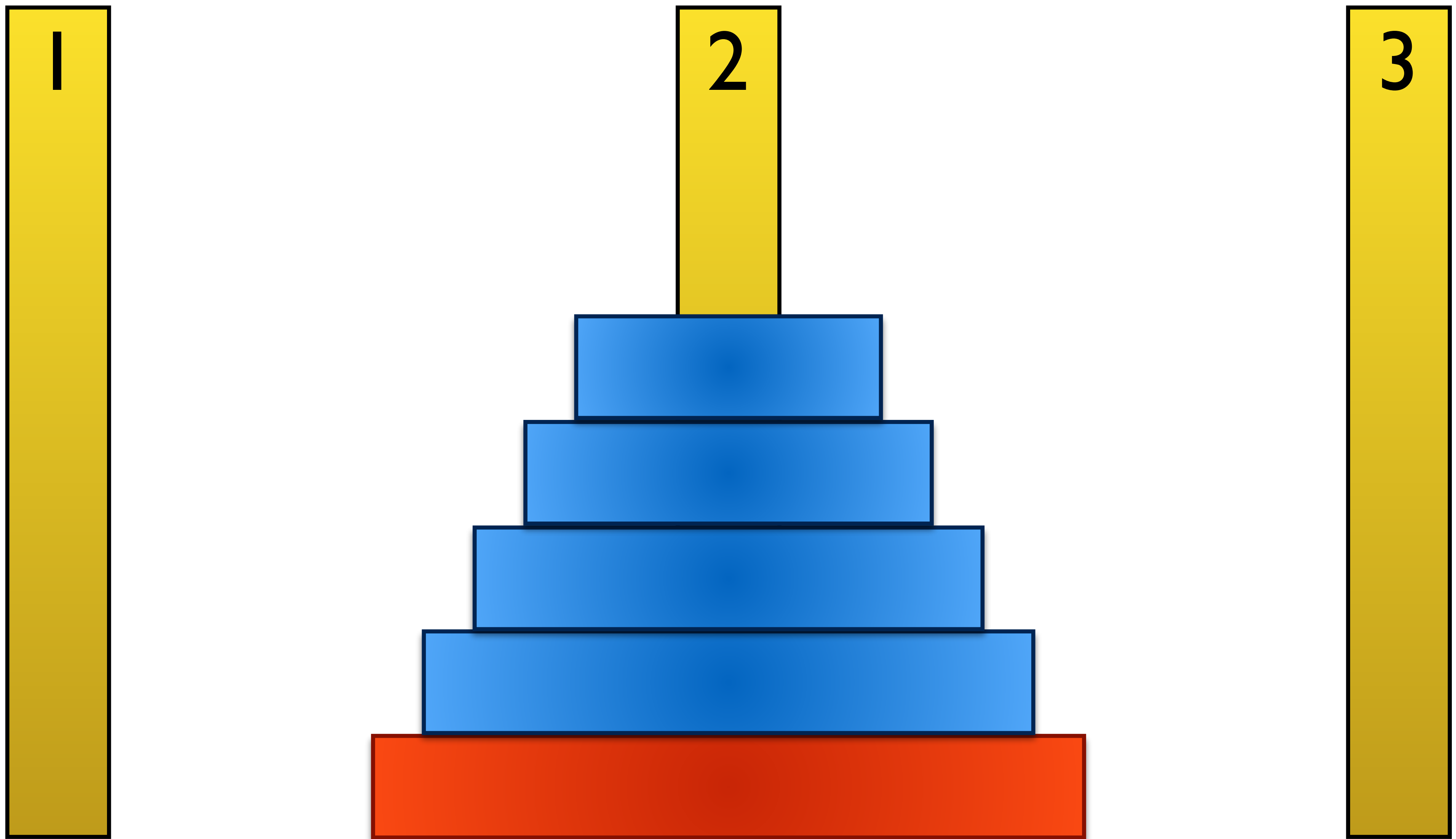
Hanoi(n,S3,S2,S1)   // n≥1

if n>1 then Hanoi(n-1,S3,S1,S2)
move disc n from S3 to S2
if n>1 then Hanoi(n-1,S1,S2,S3)



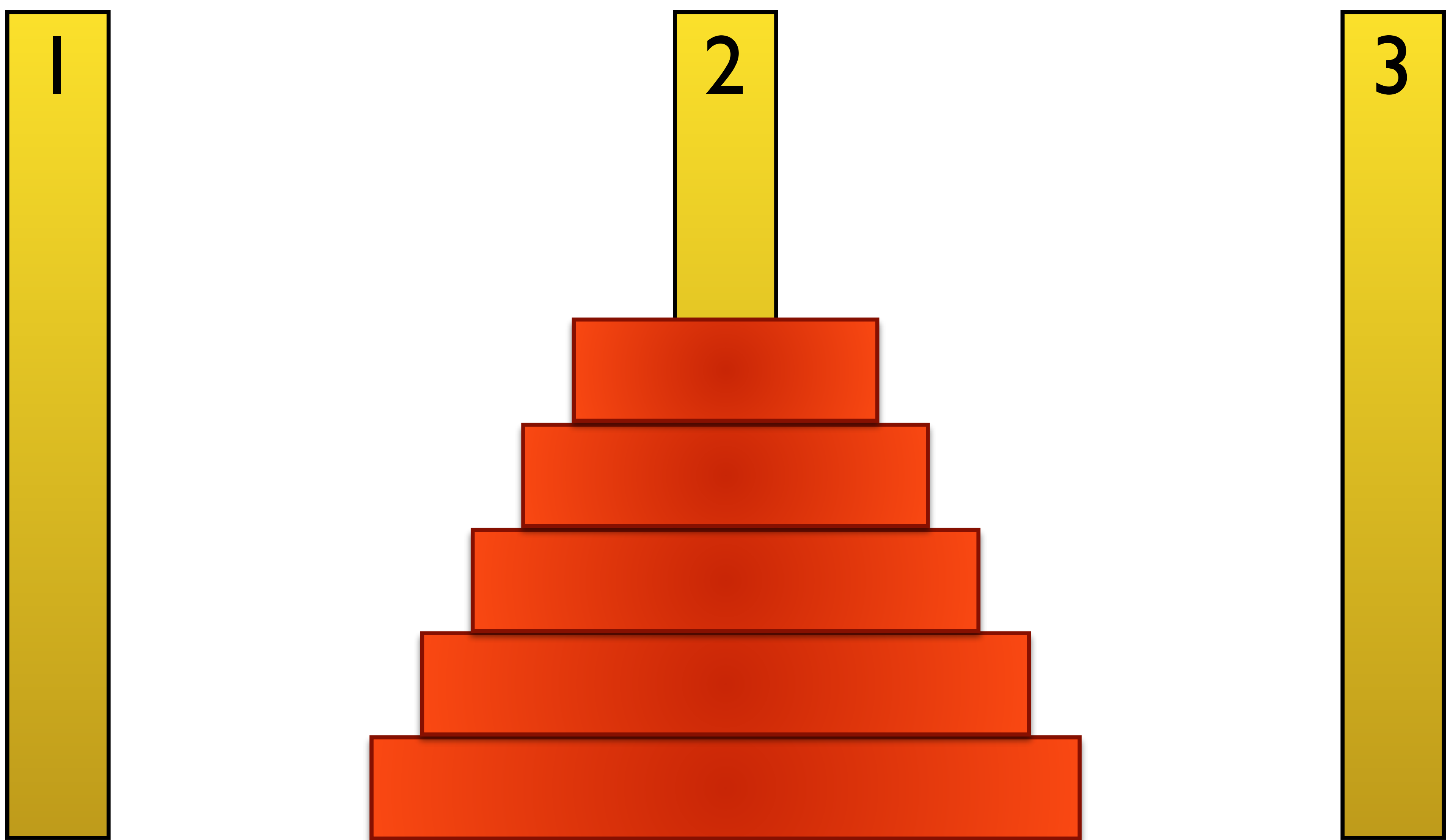
Hanoi(n,S3,S2,S1) // n≥1

if n>1then Hanoi(n-1,S3,S1,S2)
move disc n from S3 to S2
if n>1then Hanoi(n-1,S1,S2,S3)



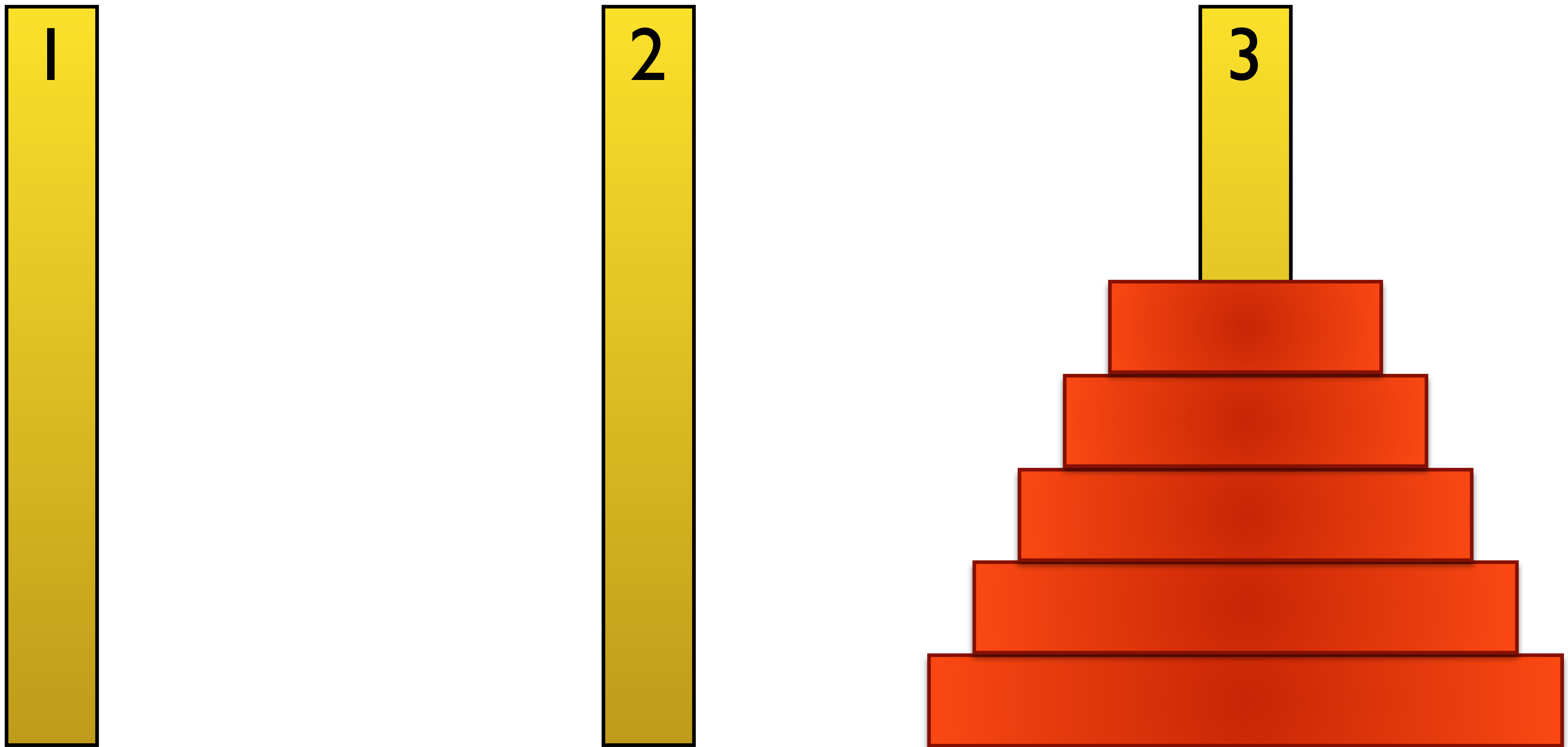
Hanoi(n,S3,S2,S1) // n≥1

if n>1 then Hanoi(n-1,S3,S1,S2)
move disc n from S3 to S2
if n>1 then Hanoi(n-1,S1,S2,S3)



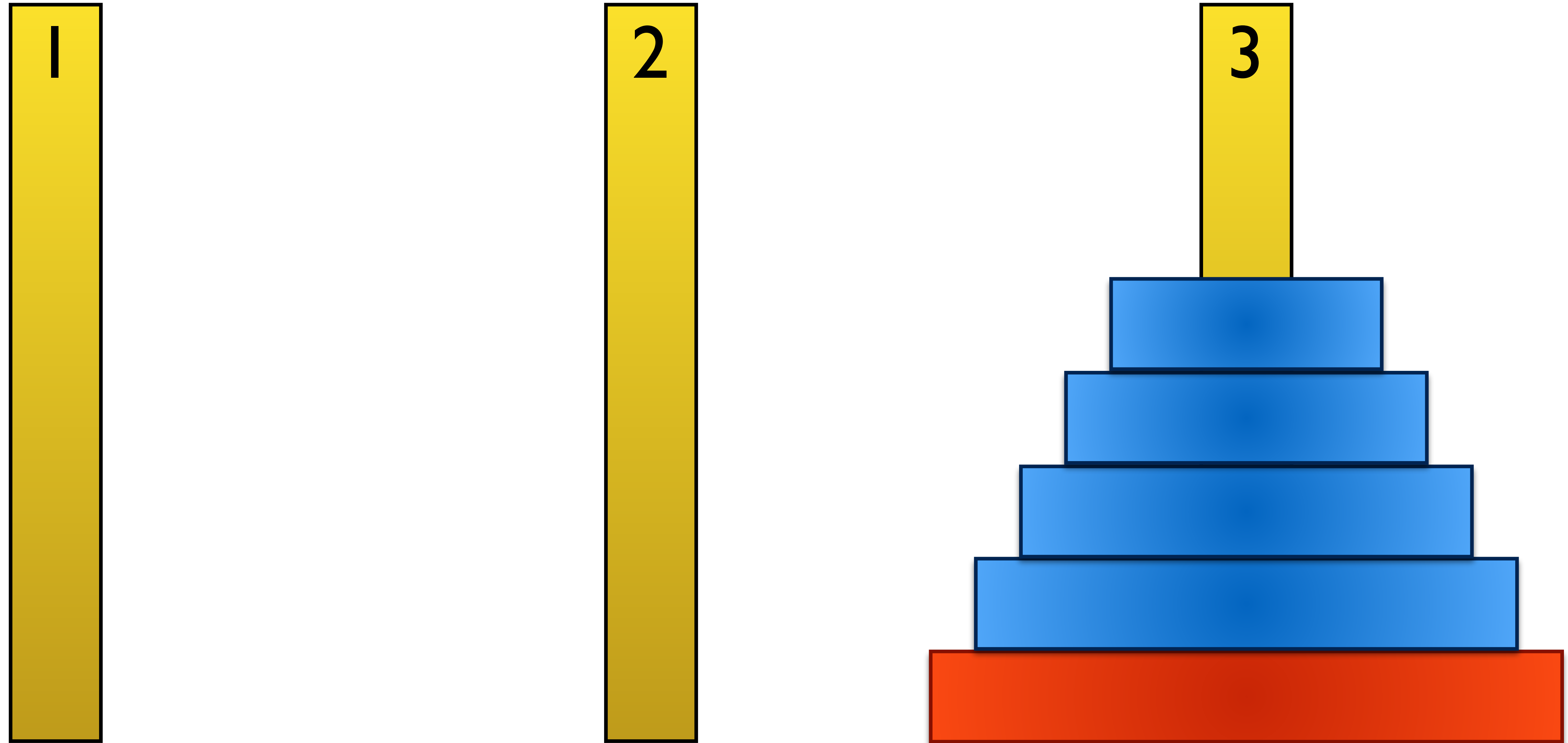
Hanoi(5,S3,S2,S1) // n≥1

if n>1 then Hanoi(4,S3,S1,S2)
move disc n from S3 to S2
if n>1 then Hanoi(4,S1,S2,S3)



Hanoi(5,S3,S2,S1) // n≥1

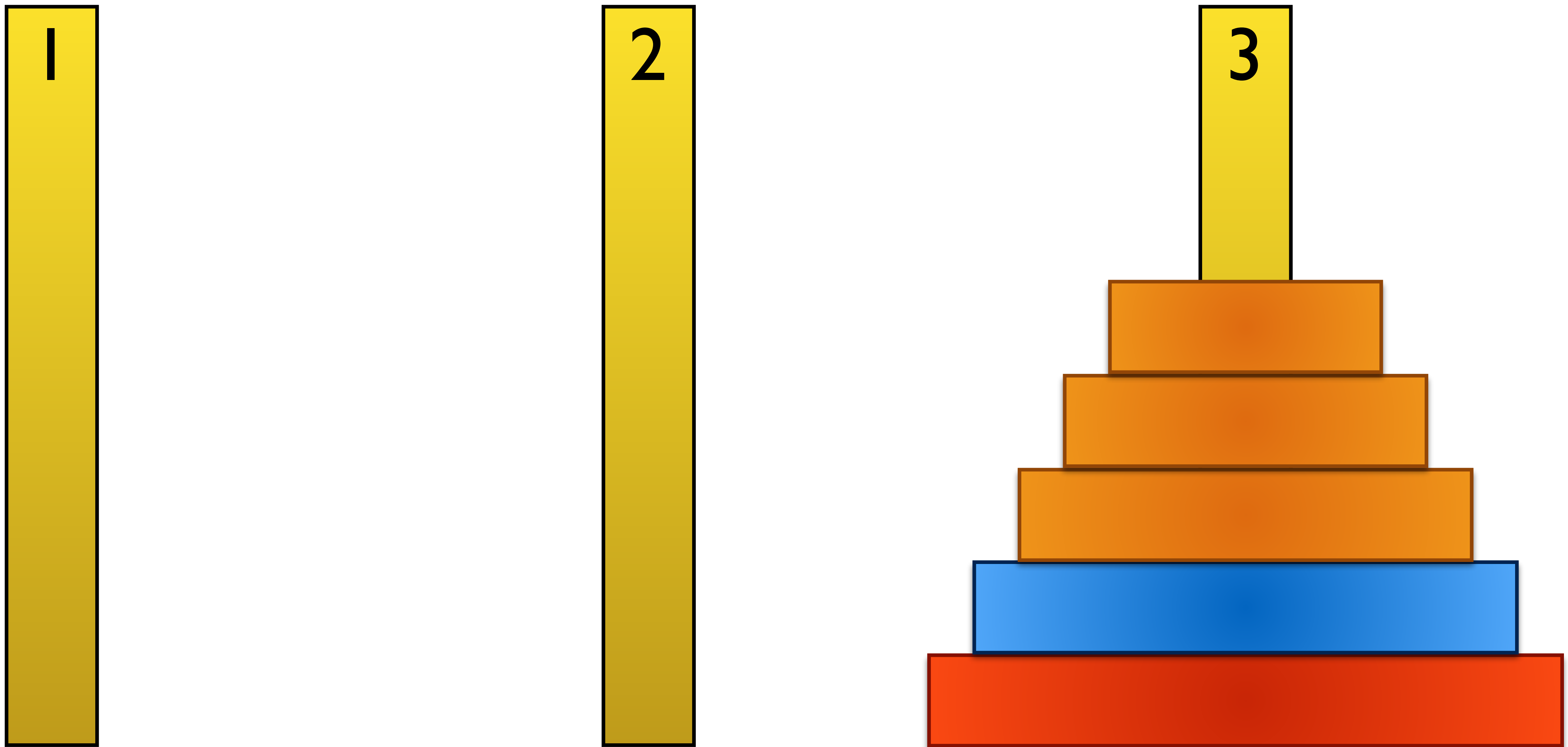
if 5>1then Hanoi(4,S3,S1,S2)



Hanoi(5,S3,S2,S1) // n≥1

if 5>1then Hanoi(4,S3,S1,S2)

if 4>1then Hanoi(3,S3,S2,S1)

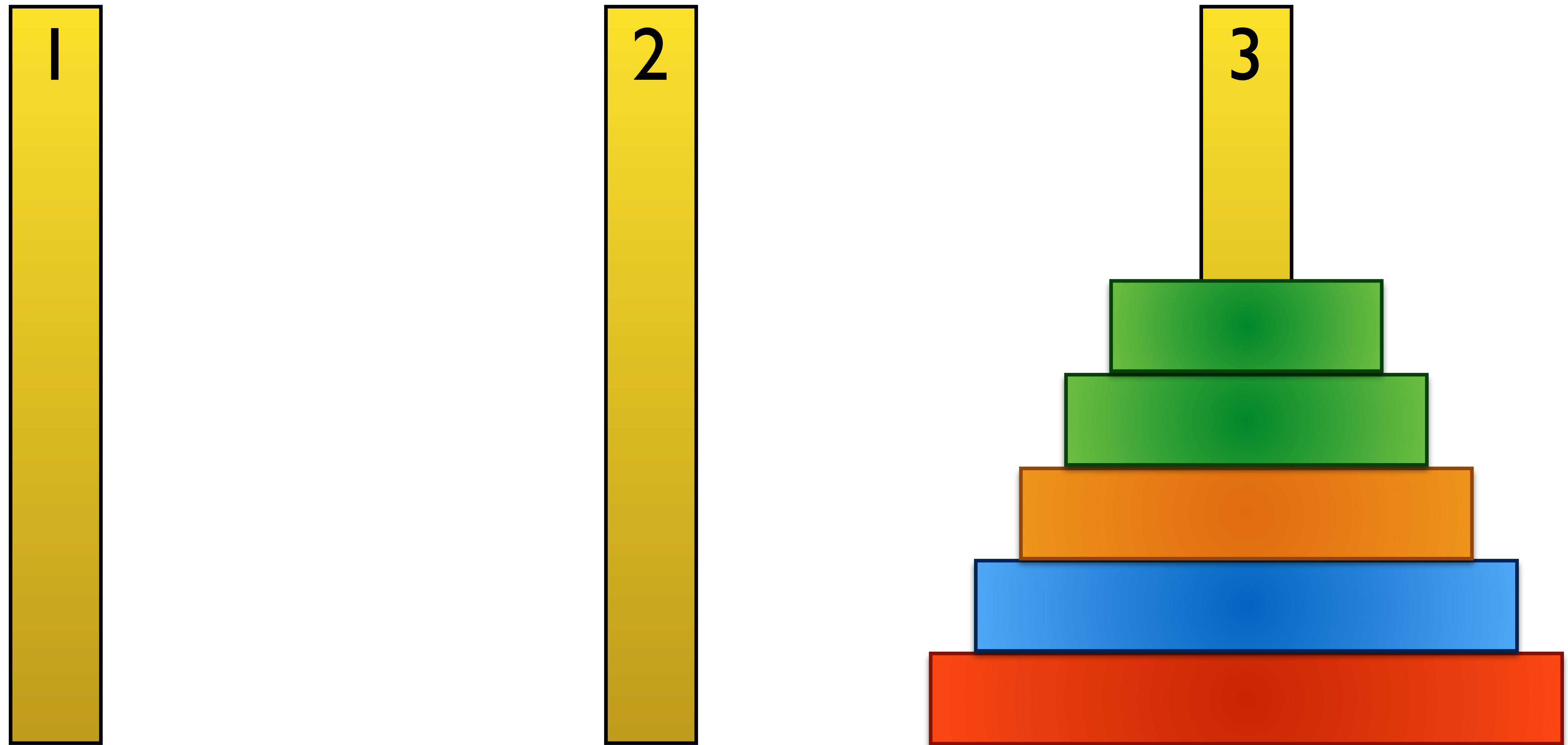


Hanoi(5,S3,S2,S1) // n≥1

if 5>1then Hanoi(4,S3,S1,S2)

if 4>1then Hanoi(3,S3,S2,S1)

if 3>1then Hanoi(2,S3,S1,S2)



Hanoi(5,S3,S2,S1) // $n \geq 1$

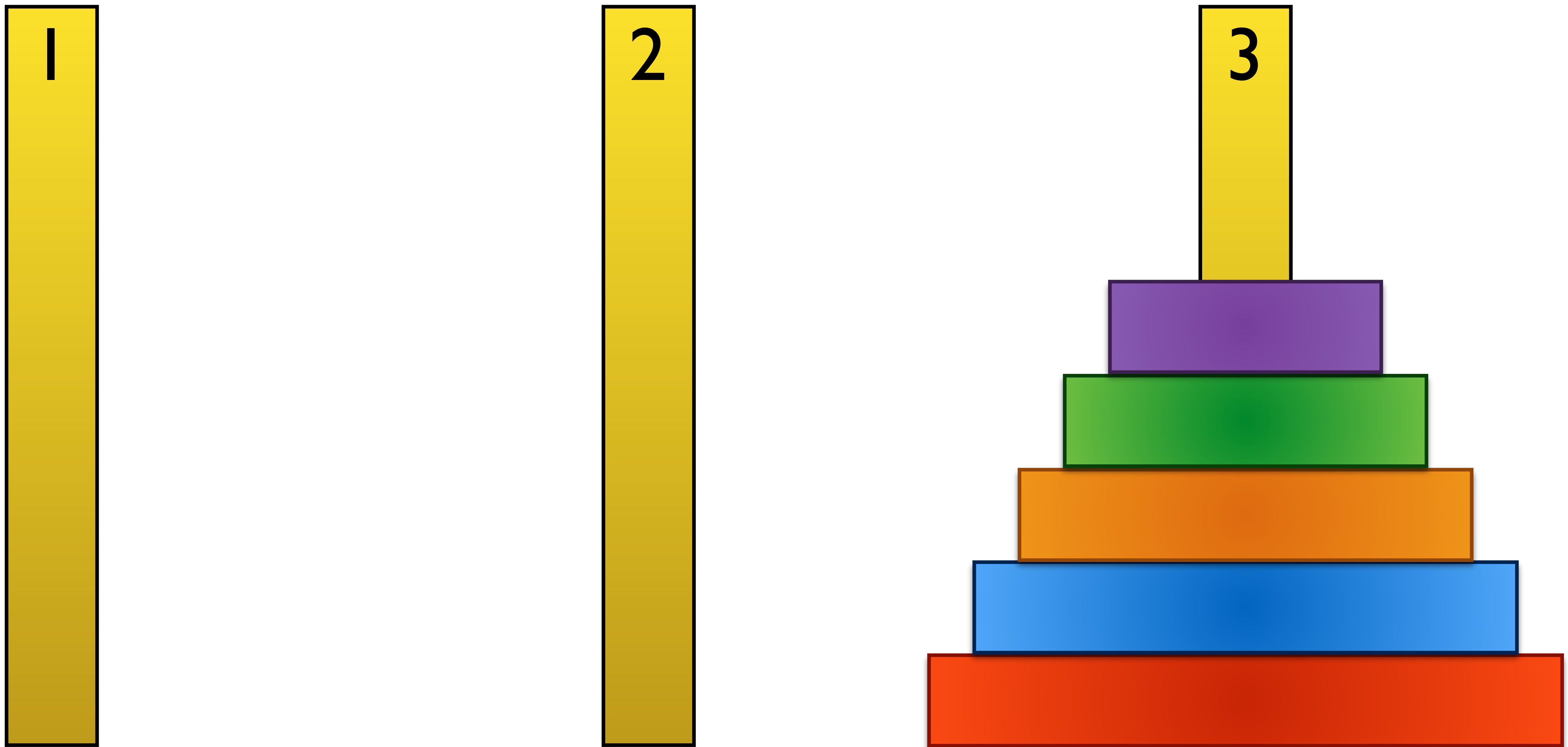
if 5>1then Hanoi(4,S3,S1,S2)

if 4>1then Hanoi(3,S3,S2,S1)

if 3>1then Hanoi(2,S3,S1,S2)

if 2>1then Hanoi(1,S3,S2,S1)

move disc 1 from S3 to S2



Hanoi(5,S3,S2,S1) // n≥1

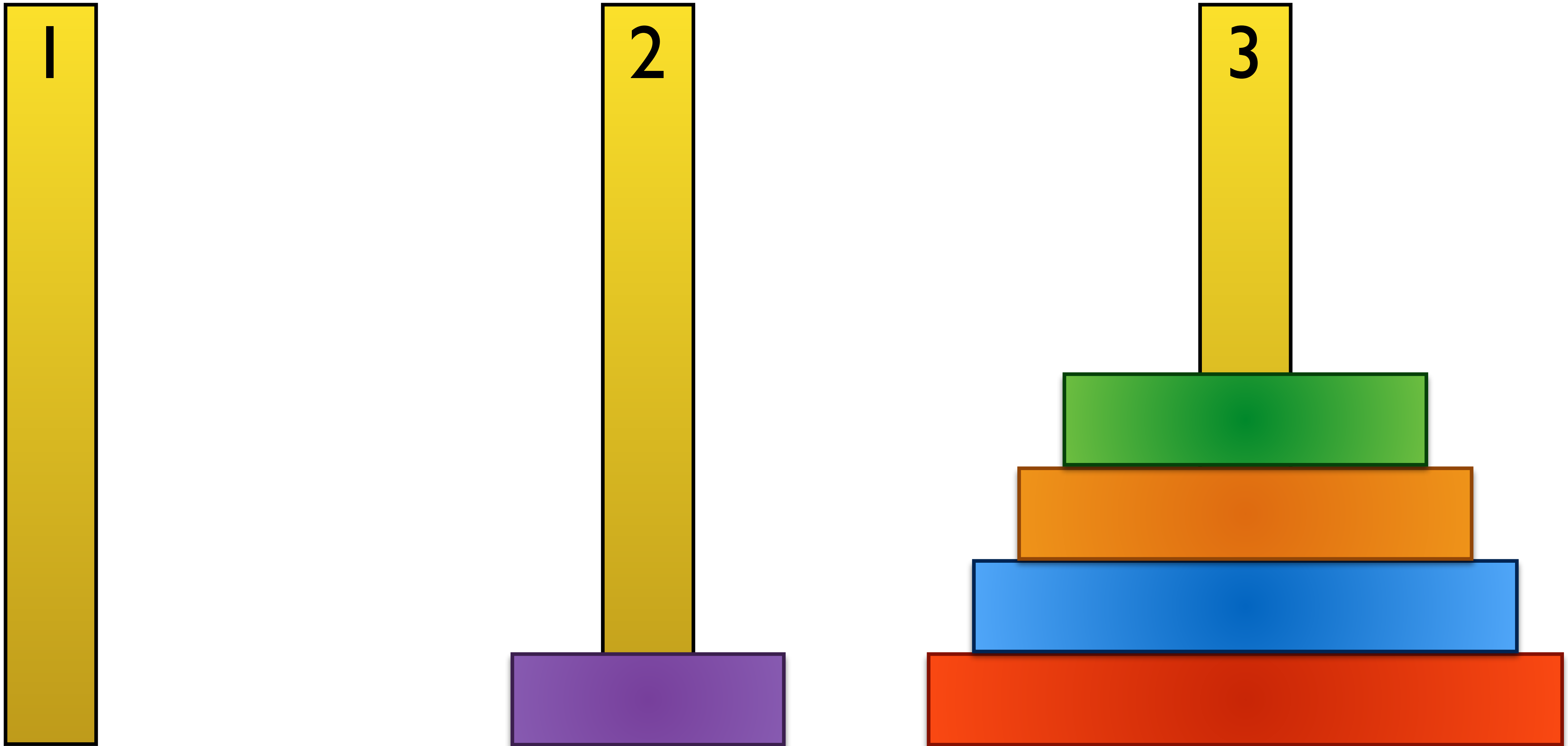
if 5>1then Hanoi(4,S3,S1,S2)

if 4>1then Hanoi(3,S3,S2,S1)

if 3>1then Hanoi(2,S3,S1,S2)

if 2>1then Hanoi(1,S3,S2,S1)

move disc 1 from S3 to S2



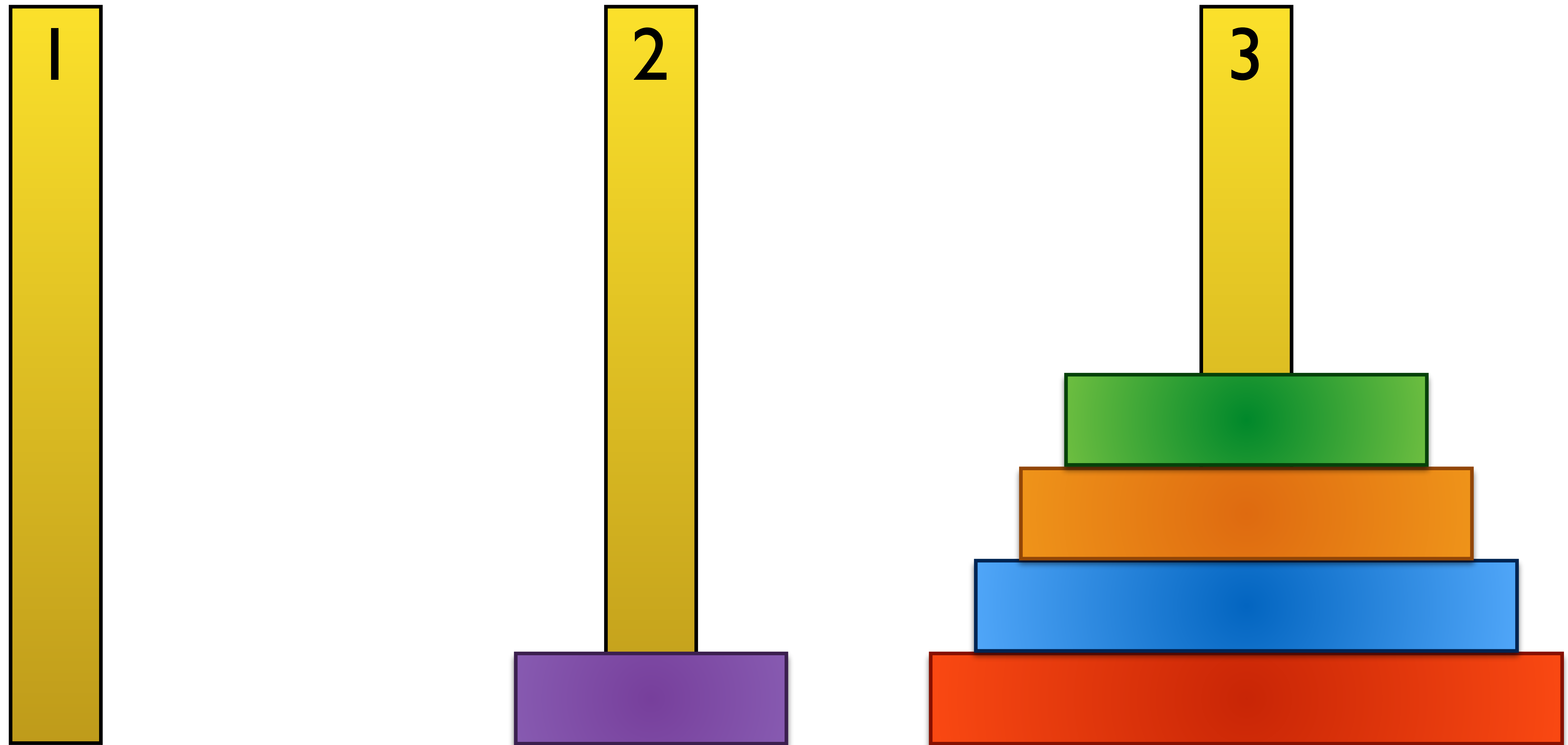
Hanoi(5,S3,S2,S1) // n≥1

if 5>1then Hanoi(4,S3,S1,S2)

if 4>1then Hanoi(3,S3,S2,S1)

if 3>1then Hanoi(2,S3,S1,S2)

move disc 2 from S3 to S1



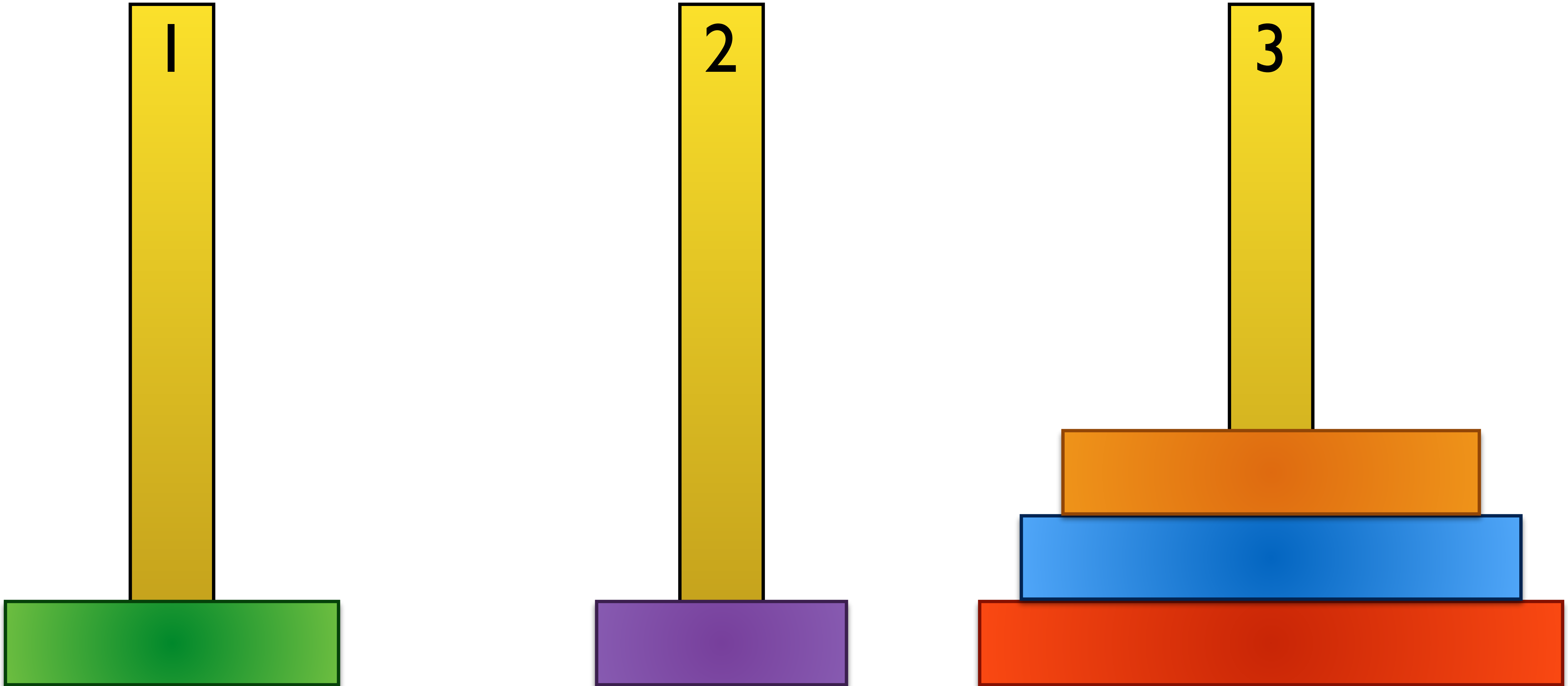
Hanoi(5,S3,S2,S1) // n≥1

if 5>1then Hanoi(4,S3,S1,S2)

if 4>1then Hanoi(3,S3,S2,S1)

if 3>1then Hanoi(2,S3,S1,S2)

move disc 2 from S3 to S1



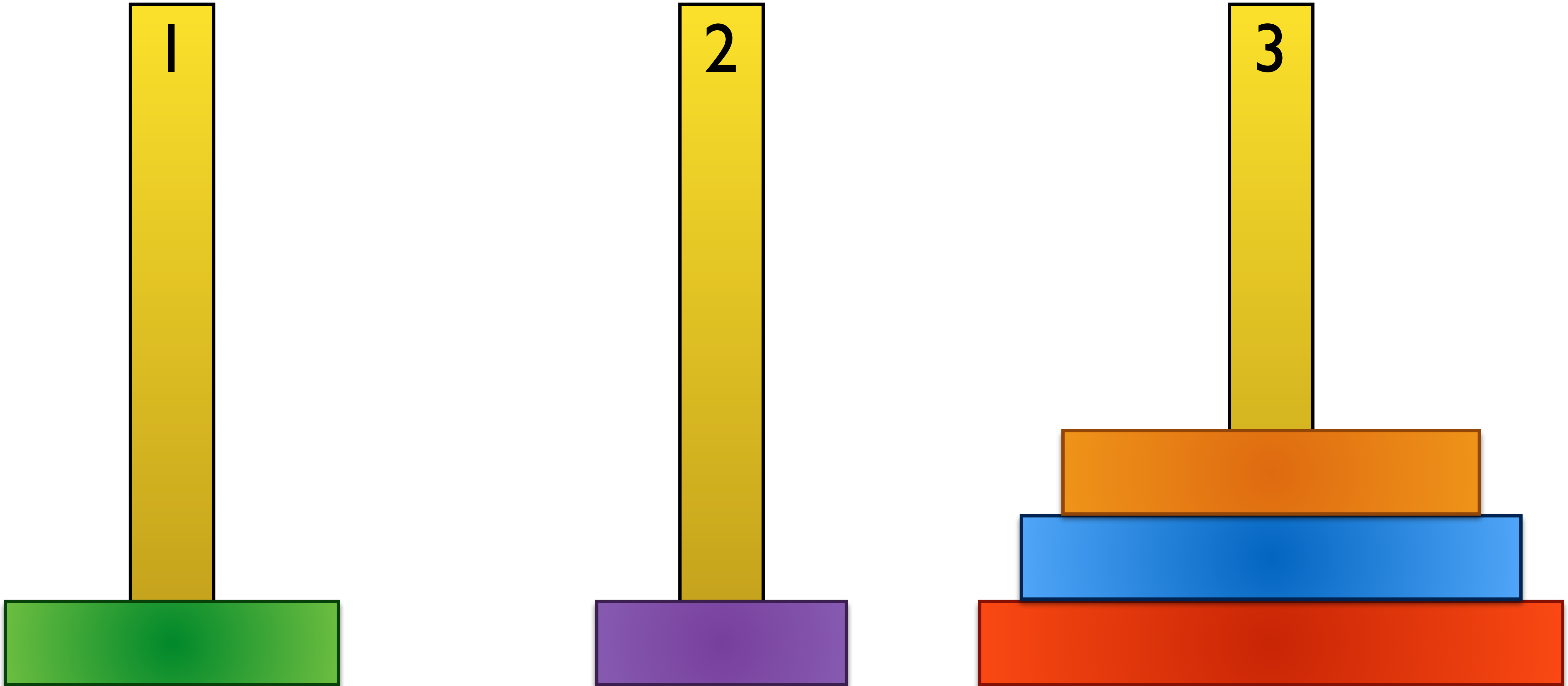
Hanoi(5,S3,S2,S1) // n≥1

if 5>1then Hanoi(4,S3,S1,S2)

if 4>1then Hanoi(3,S3,S2,S1)

if 3>1then Hanoi(2,S3,S1,S2)

if 2>1then Hanoi(1,S2,S1,S3)



Hanoi(5,S3,S2,S1) // n≥1

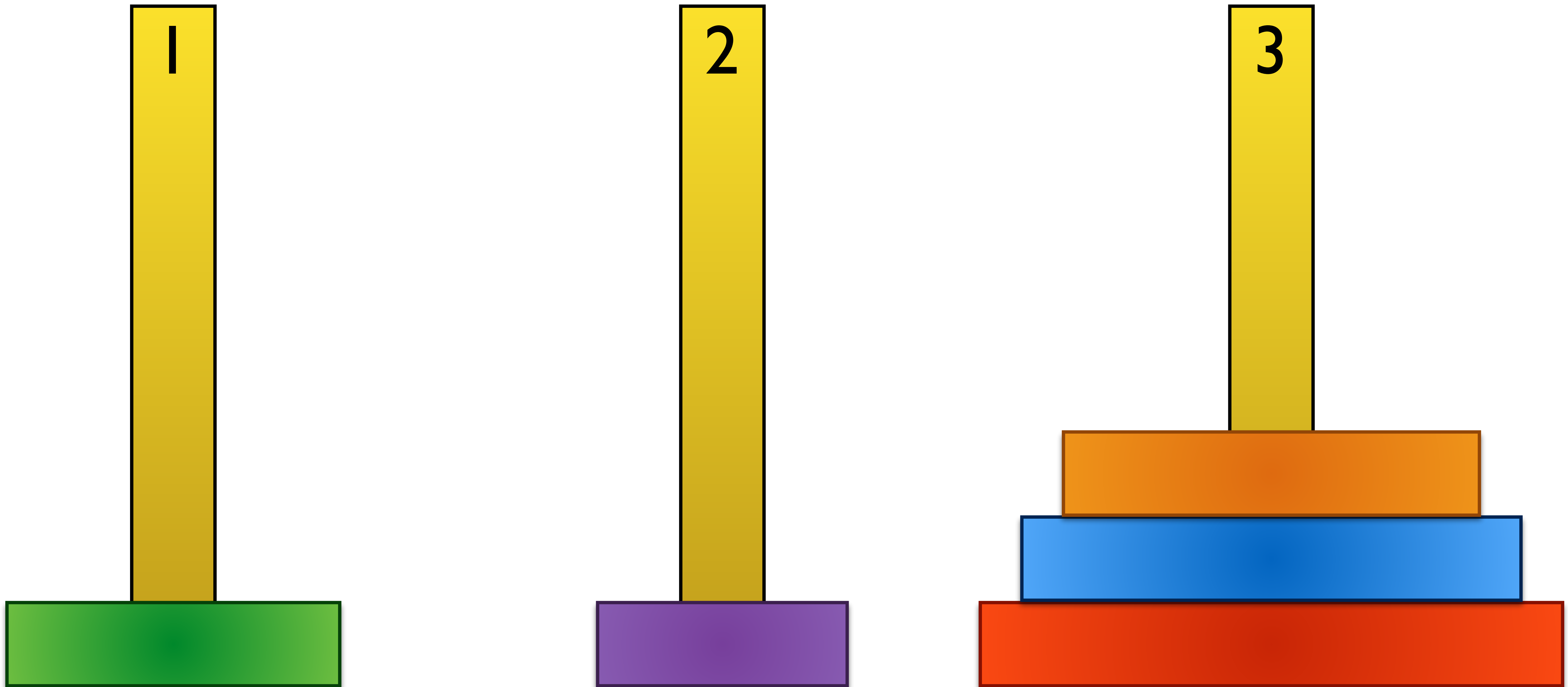
if 5>1then Hanoi(4,S3,S1,S2)

if 4>1 then Hanoi(3,S3,S2,S1)

if 3>1then Hanoi(2,S3,S1,S2)

if 2>1then Hanoi(1,S2,S1,S3)

move disc 1 from S2 to S1



Hanoi(5,S3,S2,S1) // n≥1

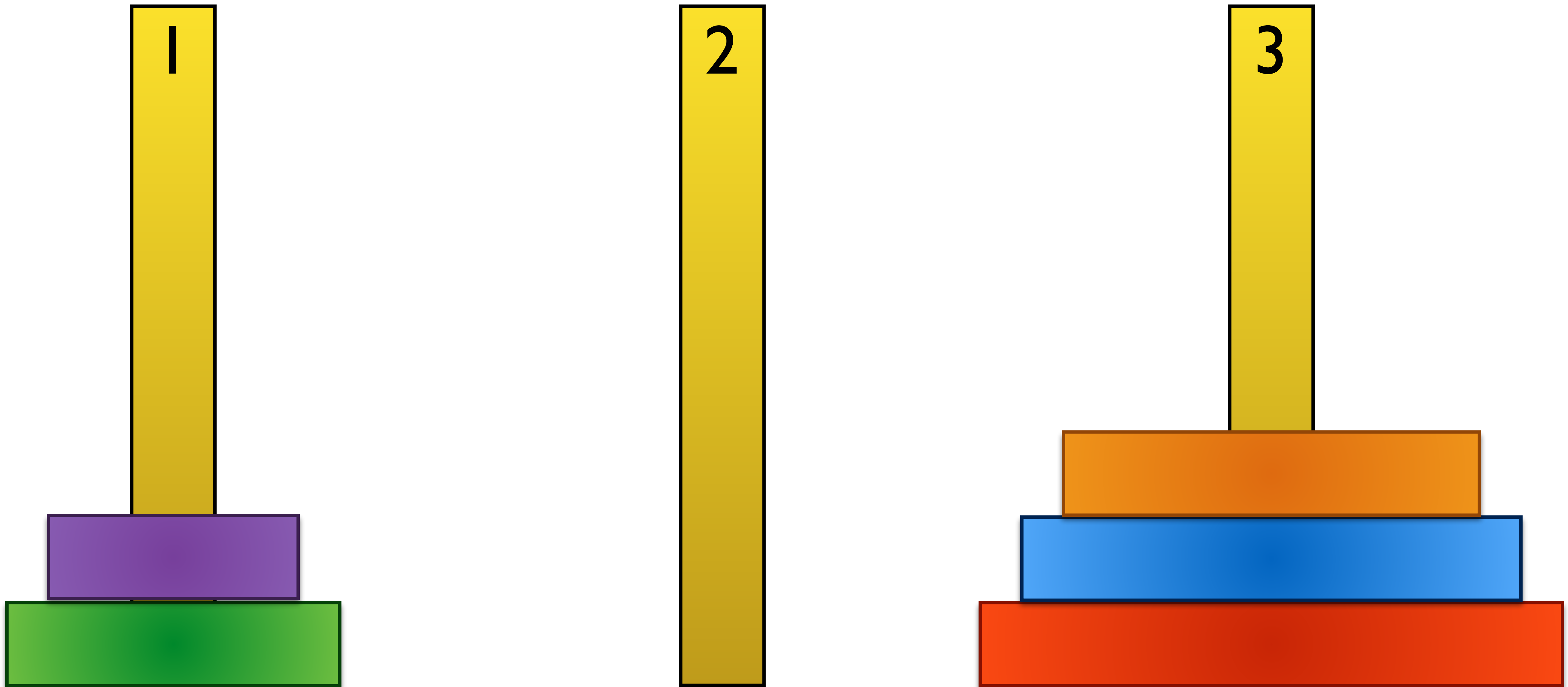
if 5>1then Hanoi(4,S3,S1,S2)

if 4>1 then Hanoi(3,S3,S2,S1)

if 3>1then Hanoi(2,S3,S1,S2)

if 2>1then Hanoi(1,S2,S1,S3)

move disc 1 from S2 to S1

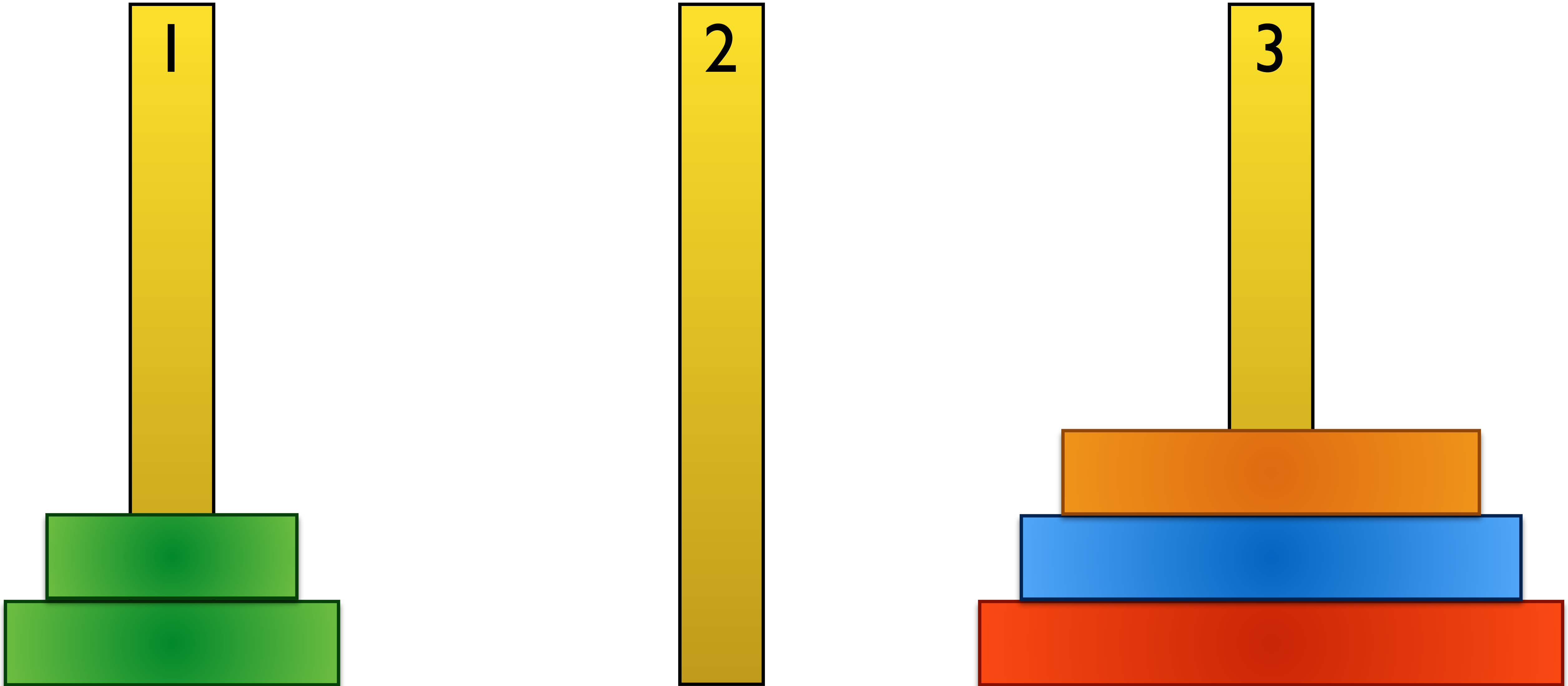


Hanoi(5,S3,S2,S1) // n≥1

if 5>1then Hanoi(4,S3,S1,S2)

if 4>1 then Hanoi(3,S3,S2,S1)

move disc 3 from S3 to S2

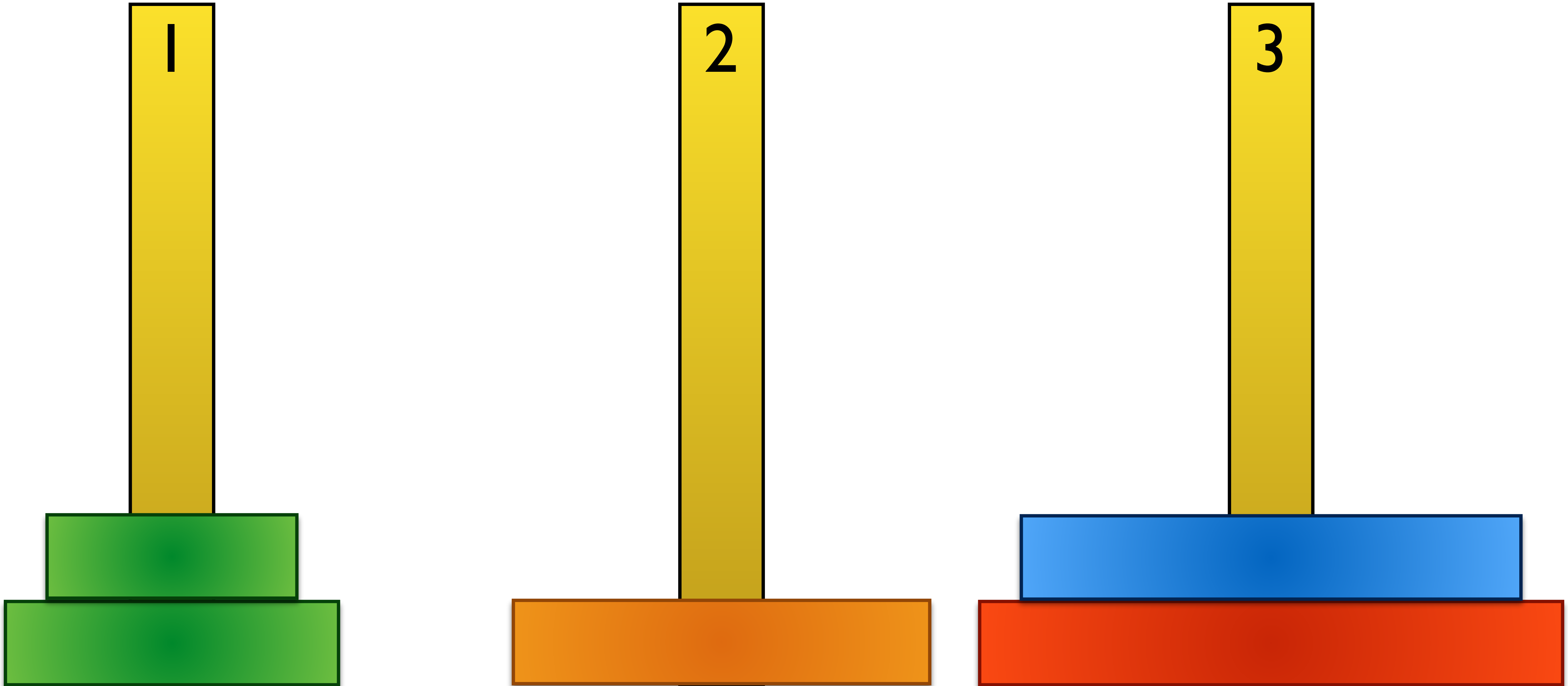


Hanoi(5,S3,S2,S1) // n≥1

if 5>1then Hanoi(4,S3,S1,S2)

if 4>1 then Hanoi(3,S3,S2,S1)

move disc 3 from S3 to S2

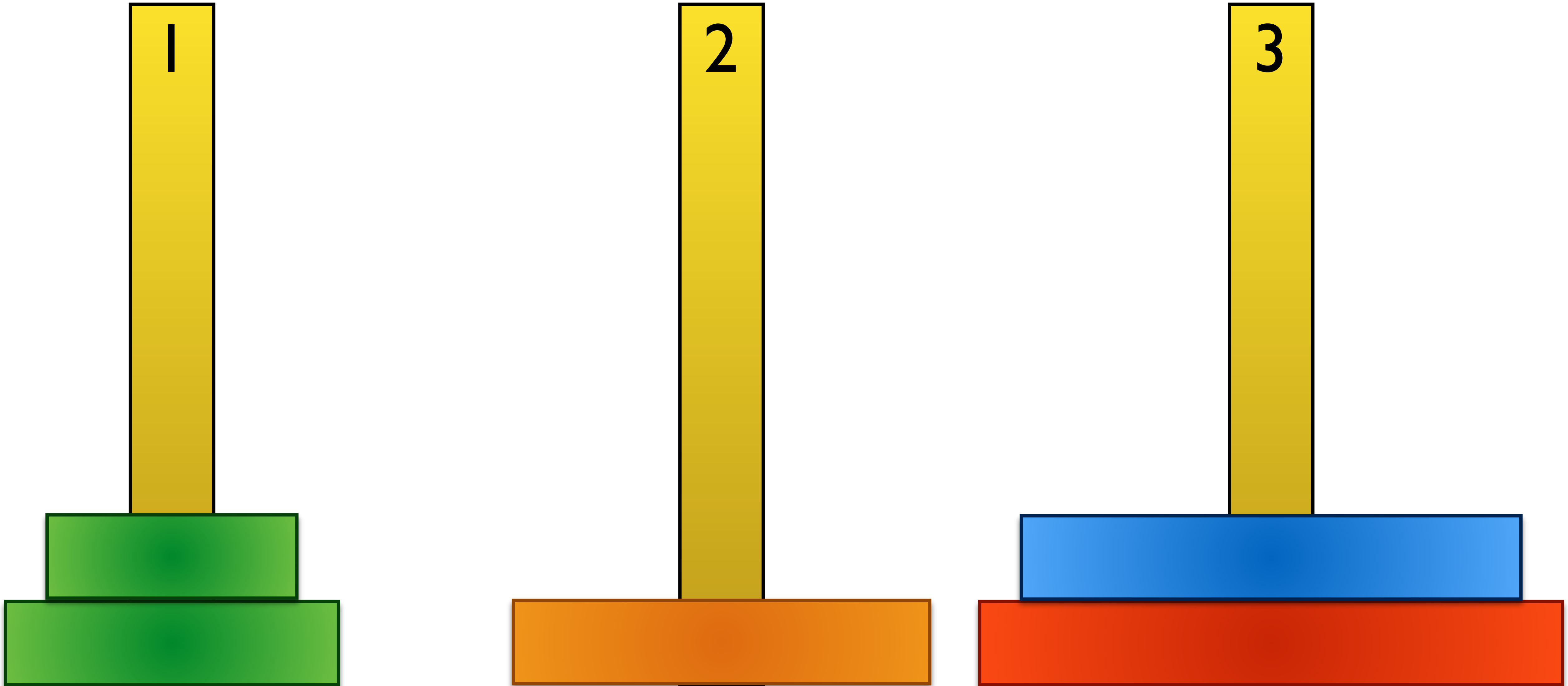


Hanoi(5,S3,S2,S1) // n≥1

if 5>1then Hanoi(4,S3,S1,S2)

if 4>1 then Hanoi(3,S3,S2,S1)

if 3>1then Hanoi(2,S1,S2,S3)

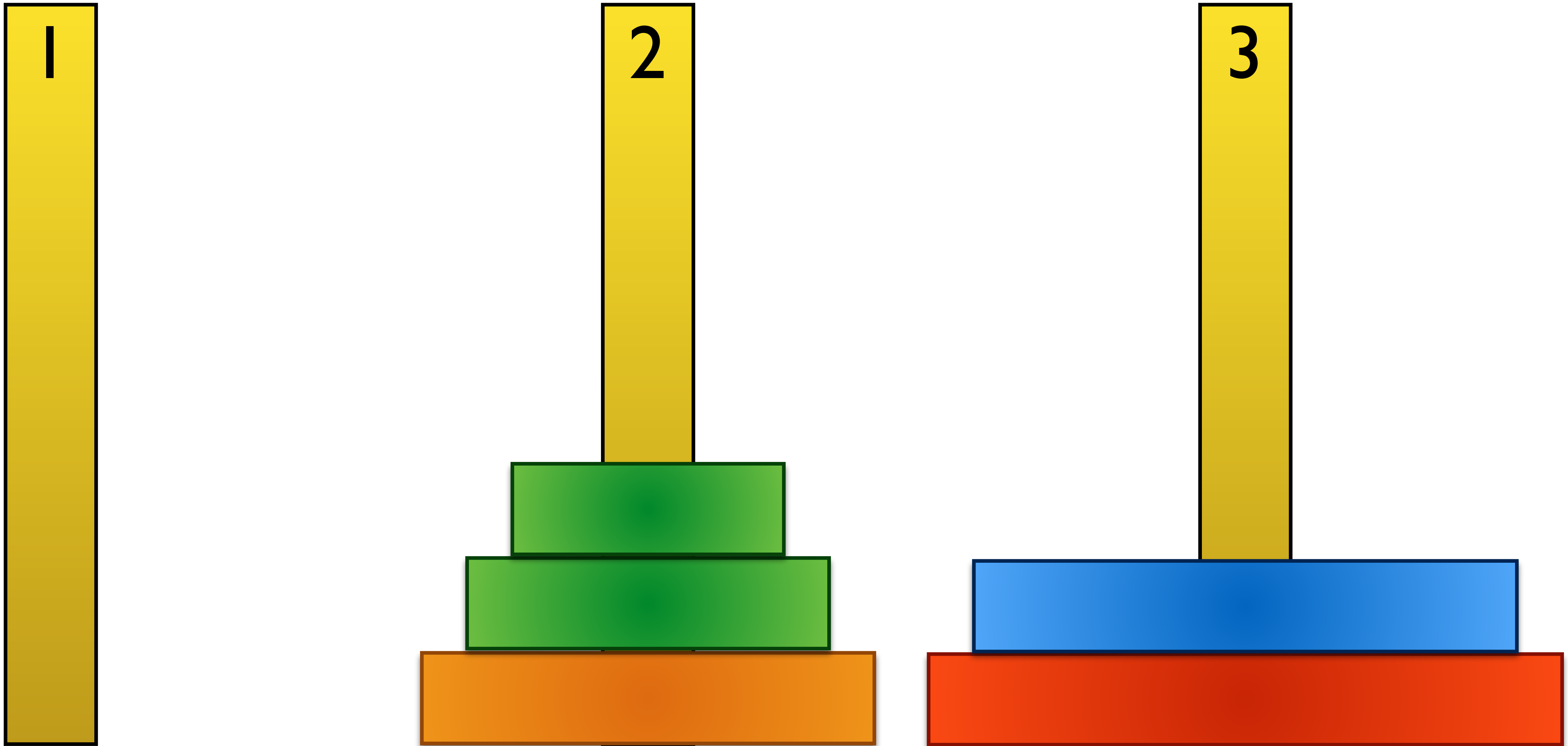


Hanoi(5,S3,S2,S1) // n≥1

if 5>1then Hanoi(4,S3,S1,S2)

if 4>1 then Hanoi(3,S3,S2,S1)

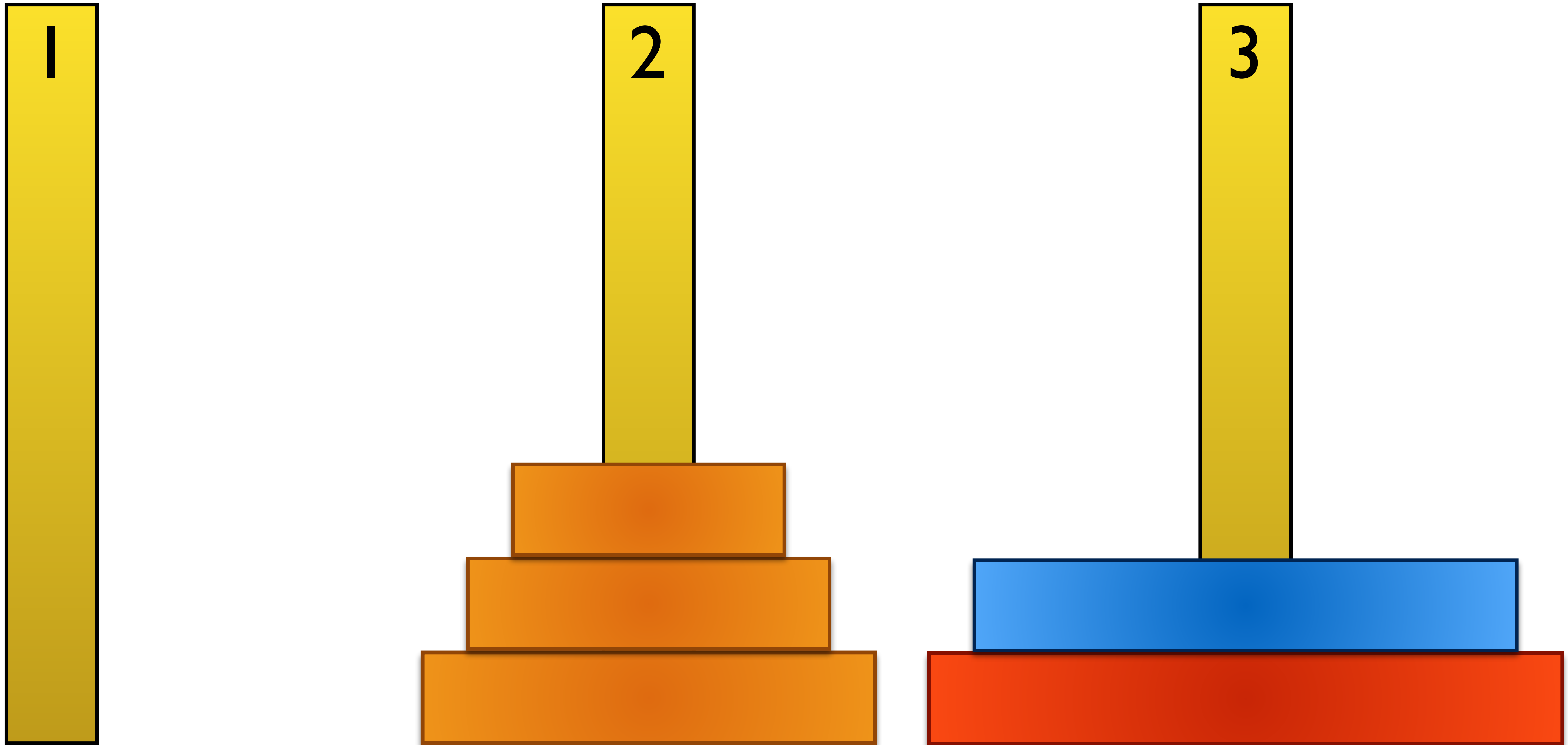
if 3>1then Hanoi(2,S1,S2,S3)



Hanoi(5,S3,S2,S1)   // n≥1

if 5>1 then Hanoi(4,S3,S1,S2)

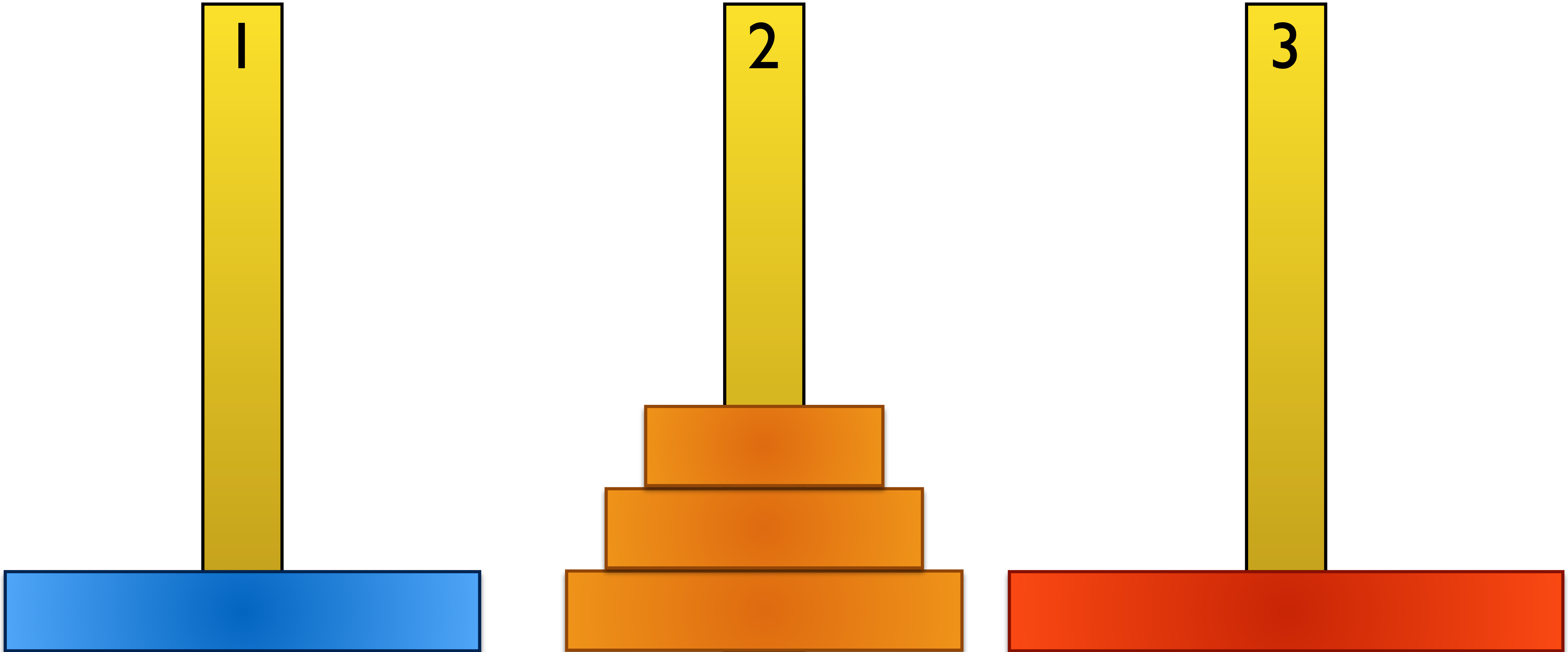
move disc 4 from S3 to S1



Hanoi(5,S3,S2,S1)   // n≥1

if 5>1 then Hanoi(4,S3,S1,S2)

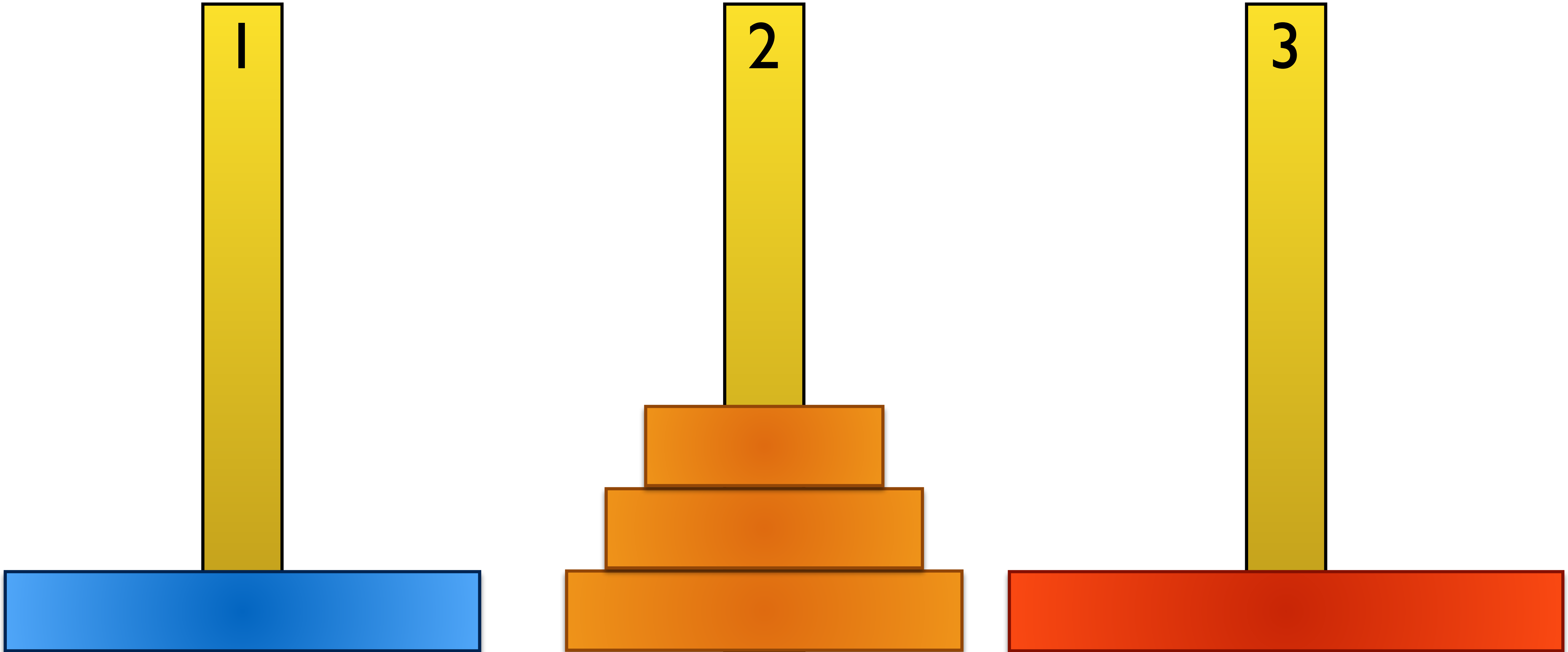
move disc 4 from S3 to S1



Hanoi(5,S3,S2,S1)   // n≥1

if 5>1 then Hanoi(4,S3,S1,S2)

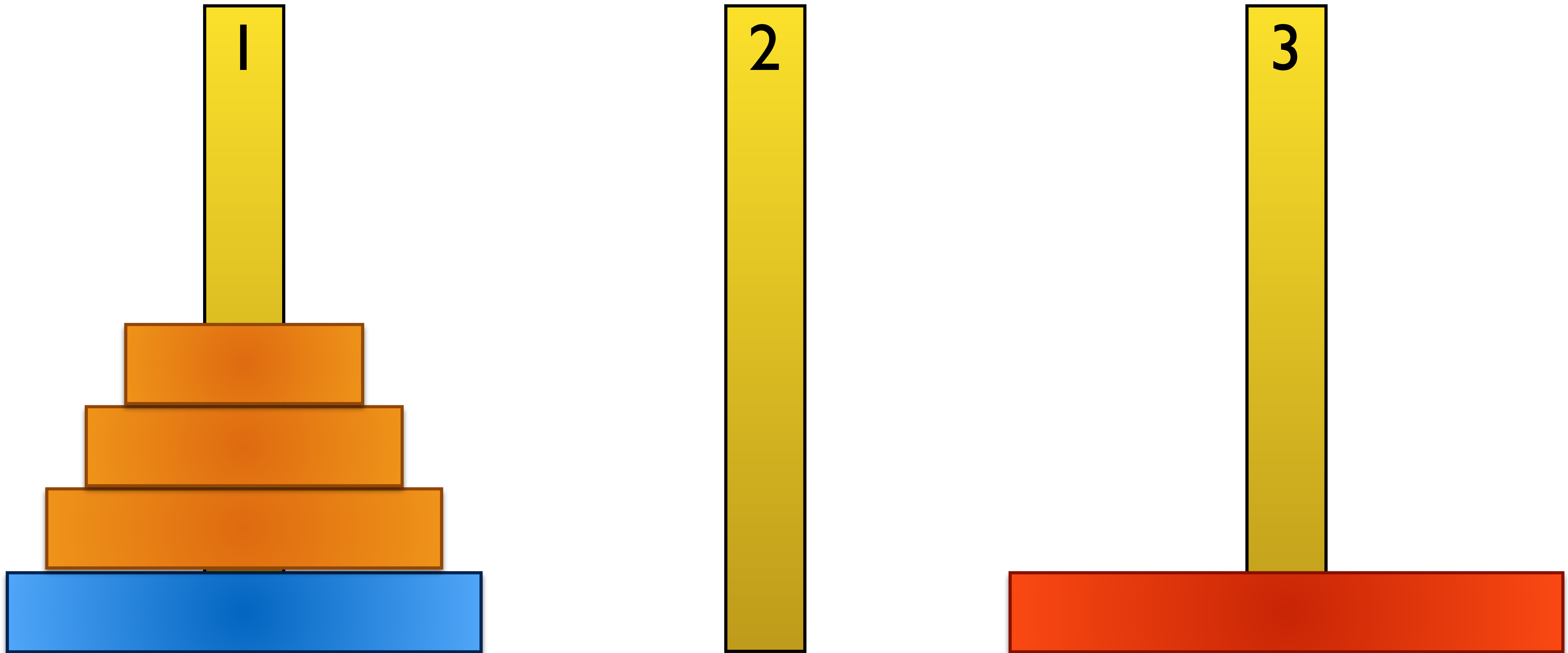
if 4>1 then Hanoi(3,S2,S1,S3)



Hanoi(5,S3,S2,S1) // $n \geq 1$

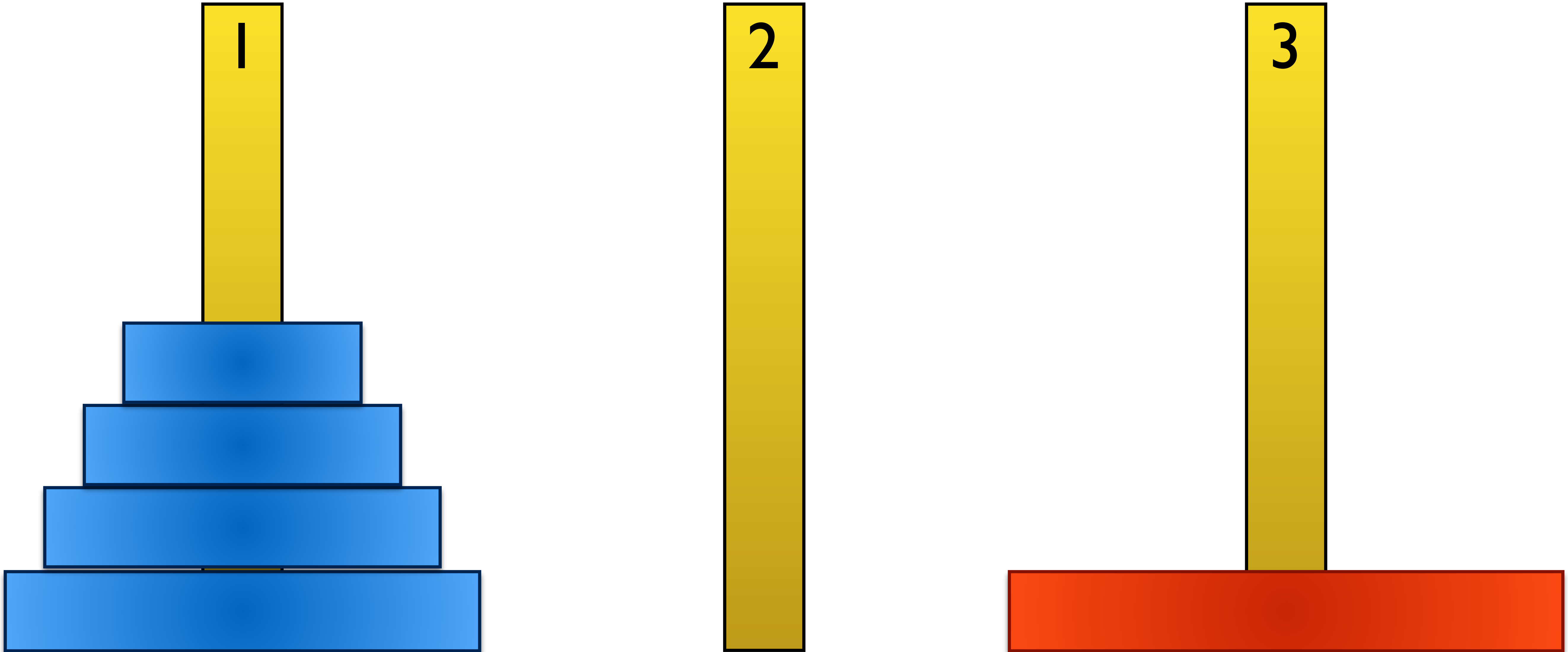
if 5>1then Hanoi(4,S3,S1,S2)

if 4>1 then Hanoi(3,S2,S1,S3)



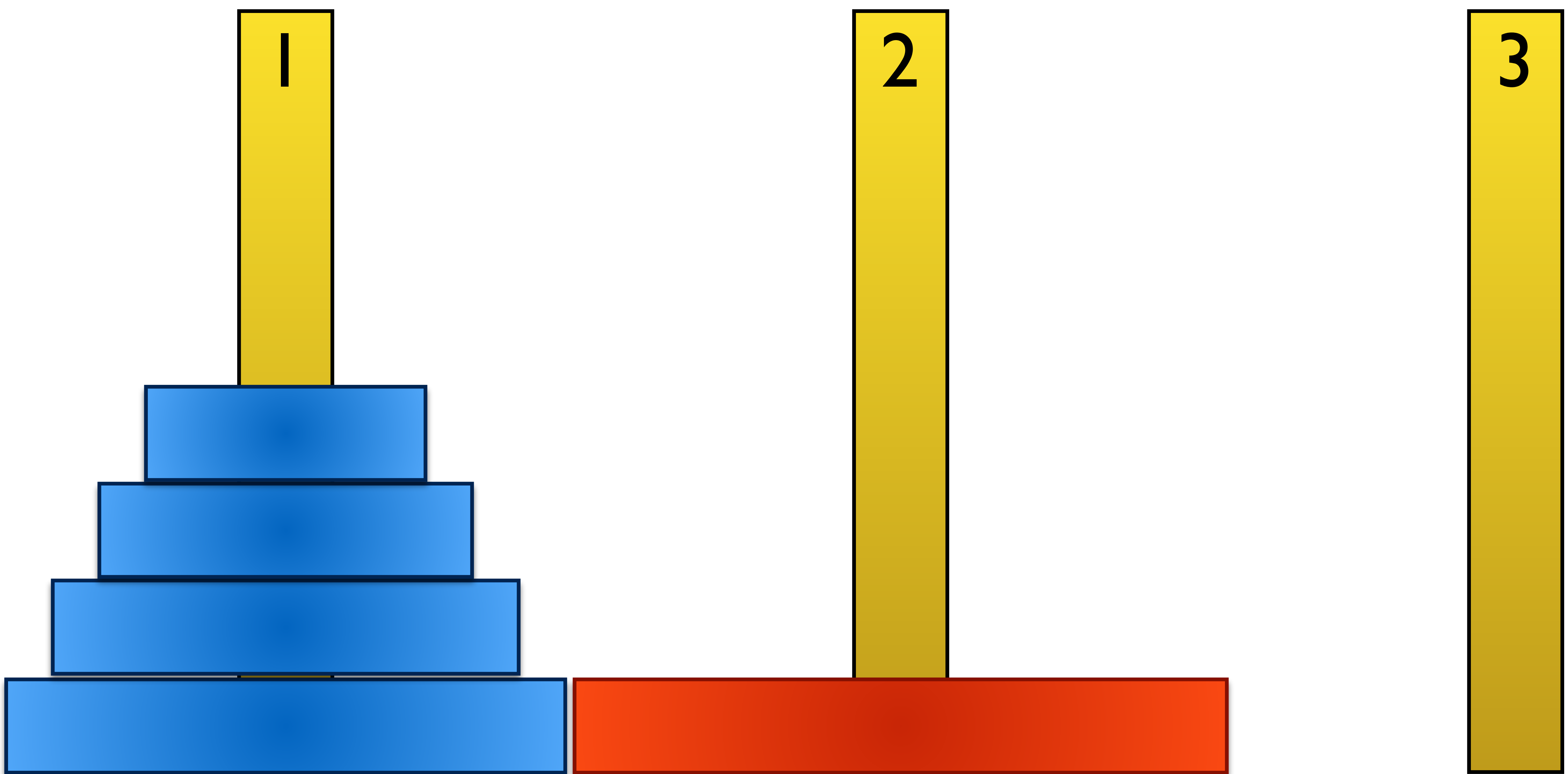
Hanoi(5,S3,S2,S1) // n≥1

move disc 5 from S3 to S2



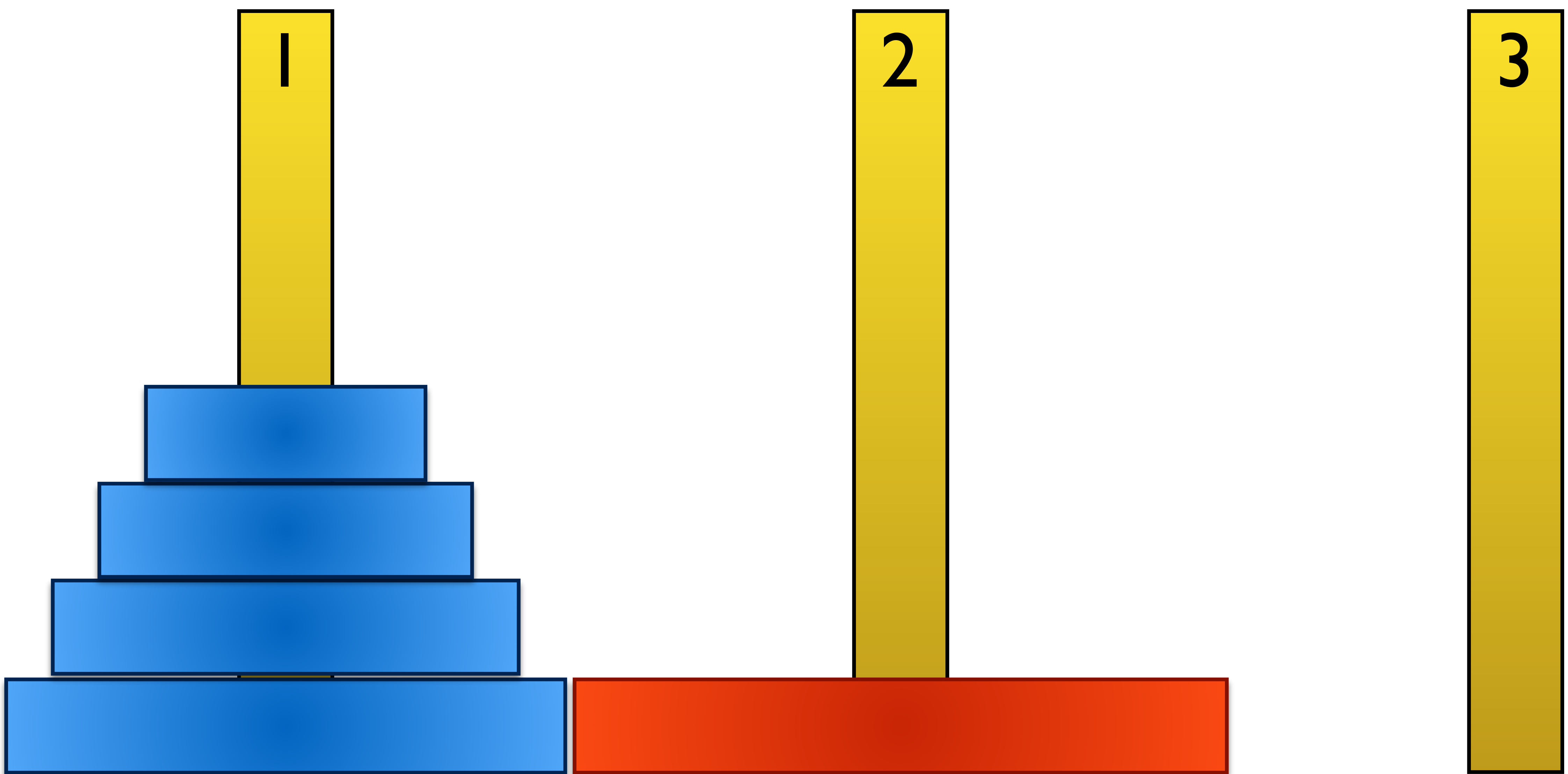
Hanoi(5,S3,S2,S1) // n≥1

move disc 5 from S3 to S2



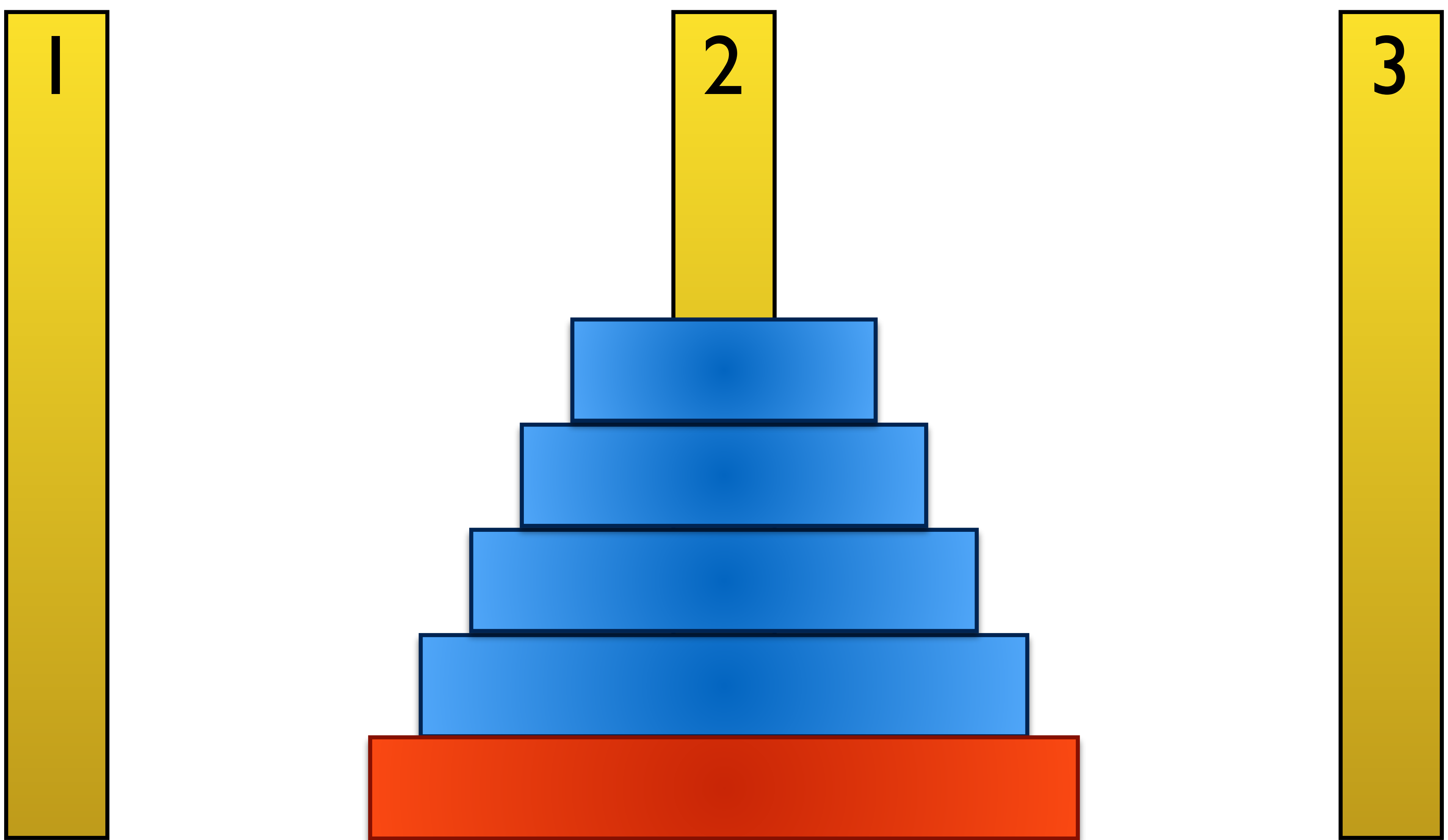
Hanoi(5,S3,S2,S1)   // $n \geq 1$

if 5>1then Hanoi(4,S1,S2,S3)

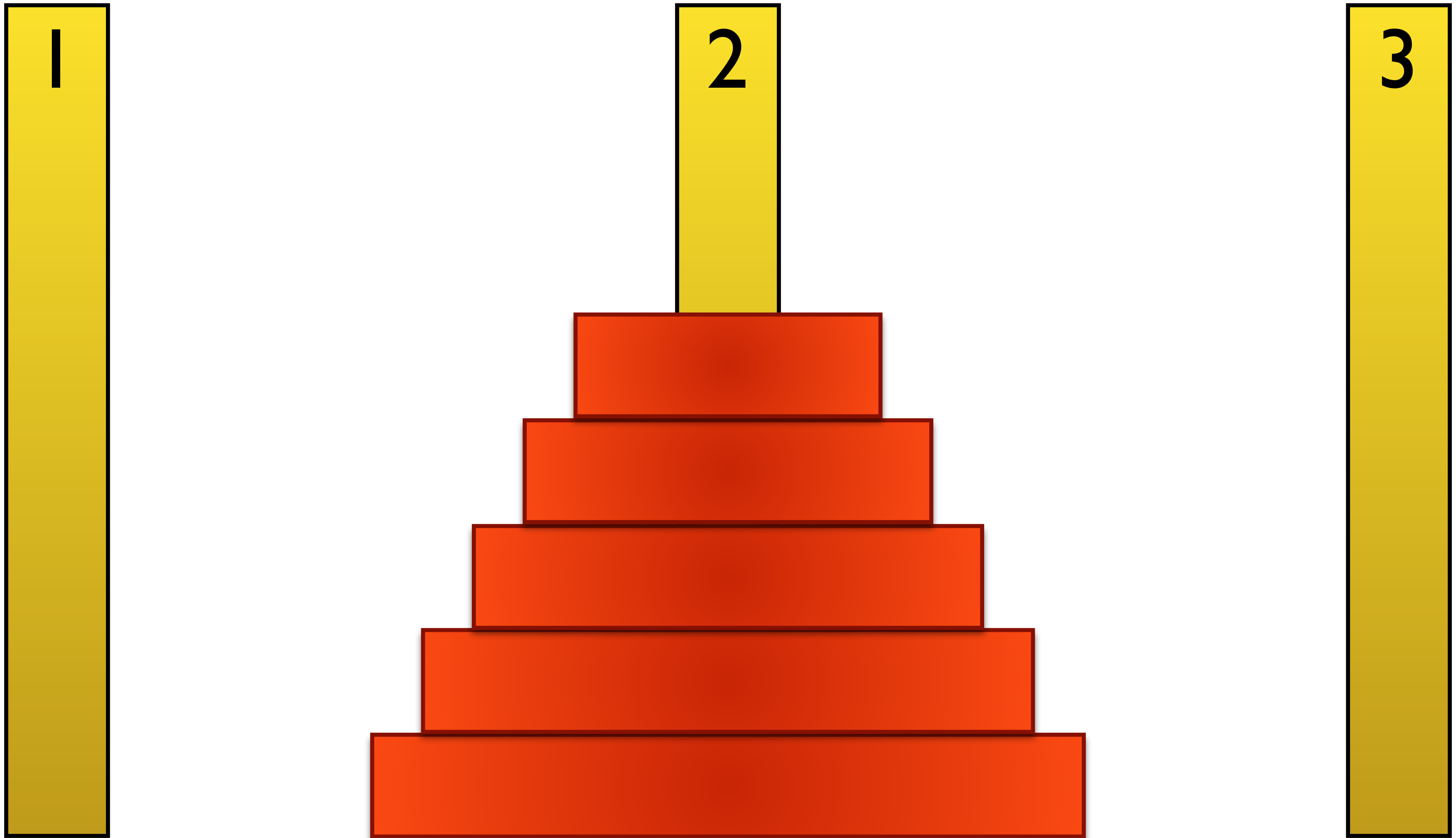


Hanoi(5,S3,S2,S1) // n≥1

if 5>1 then Hanoi(4,S1,S2,S3)



Hanoi(5,S3,S2,S1) // n≥1



Recurrence Relation

Def. T(n) = number of moves to Hanoi of n.

Hanoi recurrence.

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2T(n\text{-}1) + 1 & \text{if } n > 1 \end{cases}$$

Solution. T(n) is $O(2^n)$.

Assorted proofs. We describe several ways to prove this recurrence.

Telescoping Proof

Claim. If T(n) satisfies this recurrence, then $T(n) = 2^n - 1$.

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2T(n-1) + 1 & \text{if } n > 1 \end{cases}$$

Pf. For $n > 1$:

$$\begin{aligned} T(n) &= 2T(n-1) + 1 \\ &= 2(\ 2T(n-2) + 1\ ) + 1 \\ &= 4T(n-2) + 2 + 1 \\ &= 4(\ 2T(n-3) + 1\ ) + 2 + 1 \\ &= 8T(n-3) + 4 + 2 + 1 \\ &\ldots \\ &= 2^k T(n-k) + 2^{k-1} + \ldots + 2 + 1 \\ &\ldots \\ &= 2^{n-1} T(1) + 2^{n-2} + \ldots + 2 + 1 \\ &= 2^n - 1. \end{aligned}$$

Induction Proof

Claim. If T(n) satisfies this recurrence, then $T(n) = 2^n - 1$.

$$T(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2T(n-1) + 1 & \text{if } n > 1 \end{cases}$$

Pf. (by induction on n)

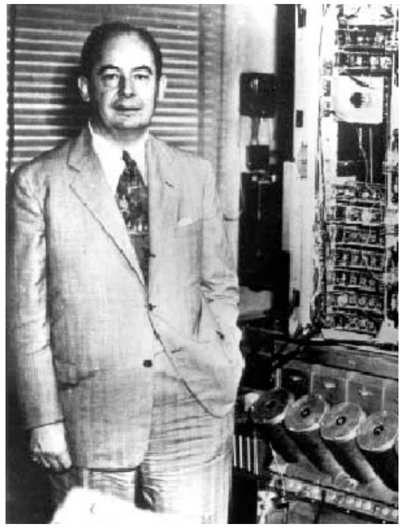
- Base case: $n = 1 = 2^1 - 1$.
- Inductive hypothesis: for $n \geq 1$, $T(n) = 2^n - 1$.
- Goal: show that $T(n+1) = 2^{n+1} - 1$.

$$\begin{aligned} T(n+1) &= 2T(n) + 1 && \text{by definition} \\ &= 2(\,2^n - 1\,) + 1 && \text{by I.H.} \\ &= 2^{n+1} - 2 + 1 \\ &= 2^{n+1} - 1. \end{aligned}$$

Merge Sort

Mergesort.

- Divide array into two halves.
- Recursively sort each half.
- Merge two halves to make sorted whole.



Jon von Neumann
(1945)

A	L	G	O	R	I	T	H	M	S	
A	L	G	O	R	I	T	H	M	S	divide O(1)
A	G	L	O	R	H	I	M	S	T	sort 2T(n/2)
A	G	H	I	L	M	O	R	S	T	merge O(n)

Recurrence Relation

Def. T(n) = number of comparisons to mergesort an input of size n.

Mergesort recurrence.

$$T(n) \le \begin{cases} 0 & \text{if } n=1 \\ \underbrace{T(\lceil n/2 \rceil)}_{\text{solve left half}} + \underbrace{T(\lfloor n/2 \rfloor)}_{\text{solve right half}} + \underbrace{n}_{\text{merging}} & \text{otherwise} \end{cases}$$

Solution. T(n) is $O(n \log_2 n)$.

Assorted proofs. We describe several ways to prove this recurrence. Initially we assume n is a power of 2 and replace ≤ with =.

Telescoping Proof

Claim. If T(n) satisfies this recurrence, then $T(n) = n \log_2 n$.

assumes n is a power of 2

$$T(n) = \begin{cases} 0 & \text{if } n = 1 \\ \underbrace{2T(n/2)}_{\text{sorting both halves}} + \underbrace{n}_{\text{merging}} & \text{otherwise} \end{cases}$$

Pf. For n > 1:

$$\begin{aligned} \frac{T(n)}{n} &= \frac{2T(n/2)}{n} + 1 \\ &= \frac{T(n/2)}{n/2} + 1 \\ &= \frac{T(n/4)}{n/4} + 1 + 1 \\ &\cdots \\ &= \frac{T(n/n)}{n/n} + \underbrace{1 + \cdots + 1}_{\log_2 n} \\ &= \log_2 n \end{aligned}$$

Induction Proof

Claim. If T(n) satisfies this recurrence, then T(n) = n $\log_2$ n.

assumes n is a power of 2

$$T(n) = \begin{cases} 0 & \text{if } n = 1 \\ \underbrace{2T(n/2)}_{\text{sorting both halves}} + \underbrace{n}_{\text{merging}} & \text{otherwise} \end{cases}$$

Pf. (by induction on k such that $n=2^k$)

- Base case: $n = 2^0 = 1$.
- Inductive hypothesis: $T(n) = T(2^k) = n \log_2 n$.
- Goal: show that $T(2n) = T(2^{k+1}) = 2n \log_2 (2n)$.

$$\begin{aligned} T(2n) &= 2T(n) + 2n \\ &= 2n\log_2 n + 2n \\ &= 2n\left(\log_2(2n) - 1\right) + 2n \\ &= 2n\log_2(2n) \end{aligned}$$

Generalized Induction Proof

Claim. If T(n) satisfies the following recurrence, then T(n) $\leq n \lceil \lg n \rceil$.

$\lg n$: $\log_2 n$

$$T(n) \leq \begin{cases} 0 & \text{if } n = 1 \\ \underbrace{T(\lceil n/2 \rceil)}_{\text{solve left half}} + \underbrace{T(\lfloor n/2 \rfloor)}_{\text{solve right half}} + \underbrace{n}_{\text{merging}} & \text{otherwise} \end{cases}$$

Pf. (by induction on n)

- Base case: n = 1. $T(1) = 0 = 1 \lceil \lg 1 \rceil$.
- Define $n_1 = \lfloor n/2 \rfloor$, $n_2 = \lceil n/2 \rceil$. (note $1 \leq n_1 < n$, $1 \leq n_2 < n$)
- Induction step: Let $n \geq 2$, assume true for 1, 2, ... , n–1.

$$\begin{aligned} T(n) &\leq T(n_1) + T(n_2) + n \\ &\leq n_1 \lceil \lg n_1 \rceil + n_2 \lceil \lg n_2 \rceil + n \\ &\leq n_1 \lceil \lg n_2 \rceil + n_2 \lceil \lg n_2 \rceil + n \\ &= n \lceil \lg n_2 \rceil + n \\ &\leq n(\lceil \lg n \rceil - 1) + n \\ &= n \lceil \lg n \rceil \end{aligned}$$

$$\begin{aligned} n_2 &= \lceil n/2 \rceil \\ &\leq \left\lceil 2^{\lceil \lg n \rceil} / 2 \right\rceil \\ &= 2^{\lceil \lg n \rceil} / 2 \\ \Rightarrow \lg n_2 &\leq \lceil \lg n \rceil - 1 \end{aligned}$$

Winter 2016
COMP-250: Introduction to Computer Science

Lecture 11, February 16, 2016