ALPII Objektorientierte Programmierung

für das 5. Übungsblatt

2012

Prof. Dr. Margarita Esponda

Sieb des Eratosthenes

3. Jahrhundert v. Chr.

Das Sieb des Eratosthenes ist ein sehr bekannter Algorithmus, der für ein vorgegebenes **N** alle Primzahlen findet, die kleiner gleich **N** sind.

Der Algorithmus verwendet ein Feld **p** aus booleschen Werten, mit dem Ziel, dem Element **p[i]** den Wert **1** zuzuweisen, falls **i** eine Primzahl ist, und anderenfalls den Wert **0**.

Ziel:

			P ↓	**P** ↓		**P** ↓		**P** ↓				**P** ↓
	0	1	**2**	**3**	4	**5**	6	**7**	8	9	10	**11**
P	0	0	**1**	**1**	0	**1**	0	**1**	0	0	0	**1**

Sieb des Eratosthenes

Anfang:

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

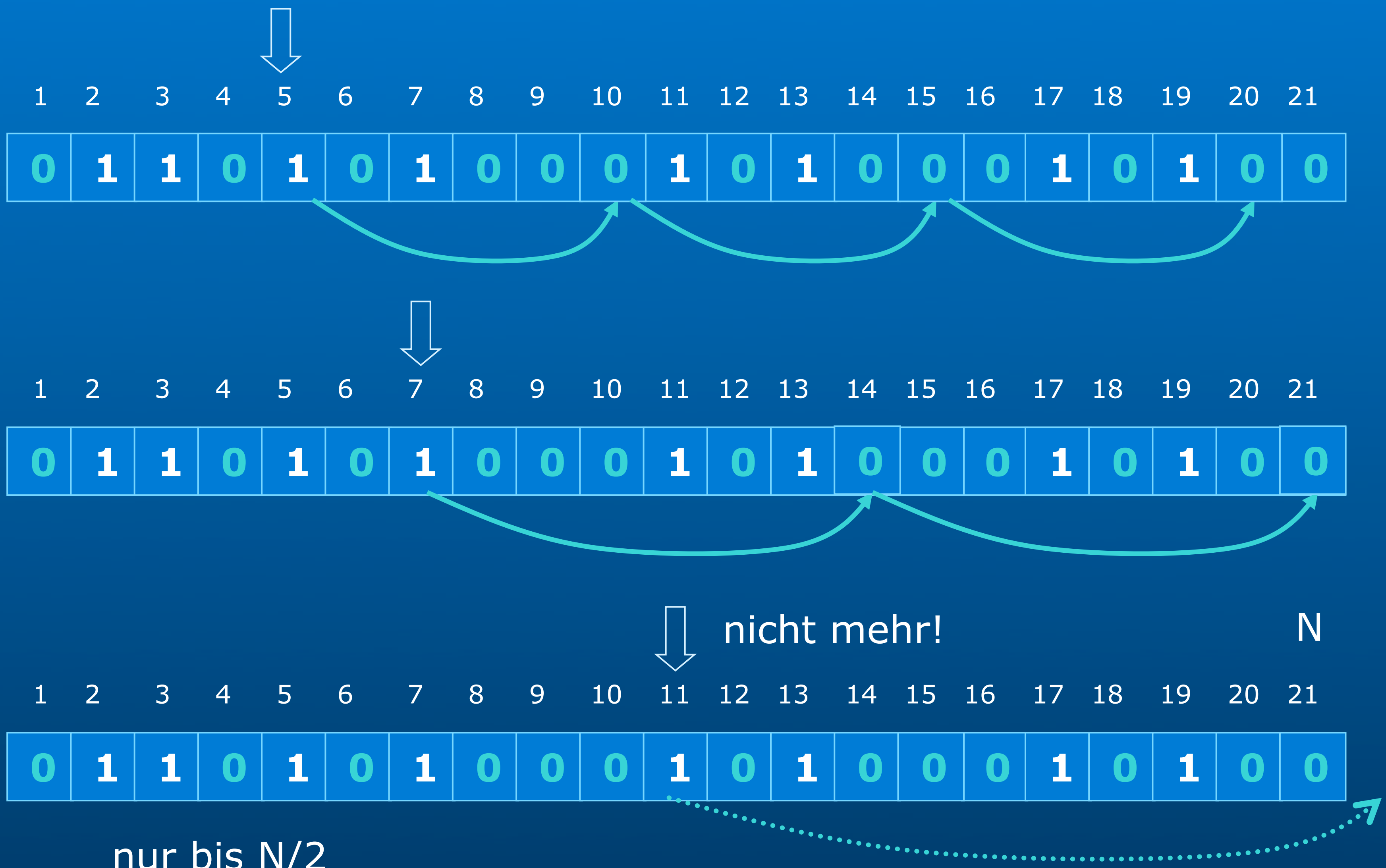
⇩

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
0	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

⇩

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
0	1	1	0	1	0	1	0	0	0	1	0	1	0	0	0	1	0	1	0	0

Sieb des Eratosthenes



nur bis N/2

noch besser ist nur bis $\sqrt[2]{N}$

Sieb des Eratosthenes

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
0	1	1	0	1	0	1	0	0	0	1	0	1	0	0	0	1	0	1	0	0

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
0	1	1	0	1	0	1	0	0	0	1	0	1	0	0	0	1	0	1	0	0

N

1	**2**	**3**	4	**5**	6	**7**	8	9	10	**11**	12	**13**	14	15	16	**17**	18	**19**	20	21
0	1	1	0	1	0	1	0	0	0	1	0	1	0	0	0	1	0	1	0	0

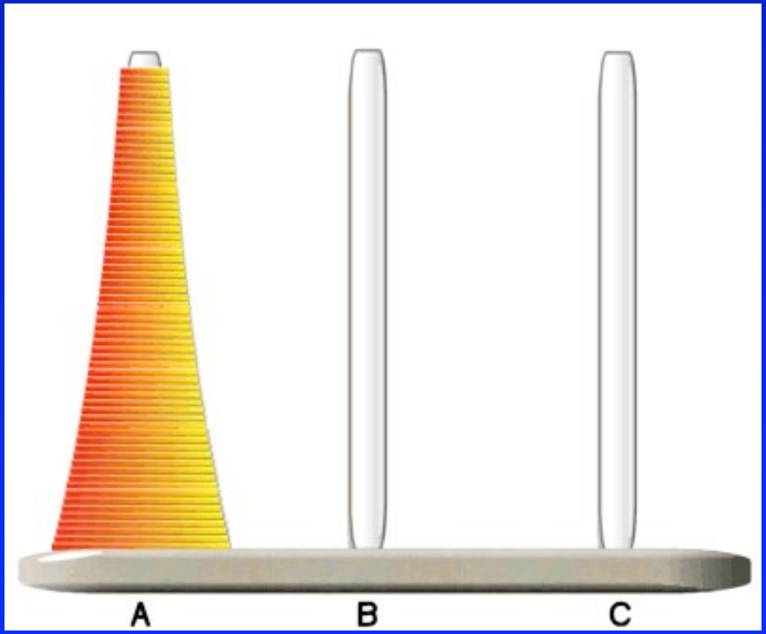
Sieb des Eratosthenes

```
from math import sqrt

def erathostenes(grenze):
    if grenze >=2:
        primzahlen = [True for i in range(grenze)]
        primzahlen[0] = False
        primzahlen[1] = False

        berechnungsgrenze = int(sqrt(grenze))+1

        for i in range(2,berechnungsgrenze):
            if primzahlen[i] == True:
                step = i
                for j in range(2*i,grenze,i):
                    primzahlen[j] = False
        return primzahlen
    else:
        return None
```

Hanoi-Turm

Edouard Lucas (1883)

- Tempel von Benares
- Turm von Brahama
- 64 Scheiben aus Gold



Ziel:

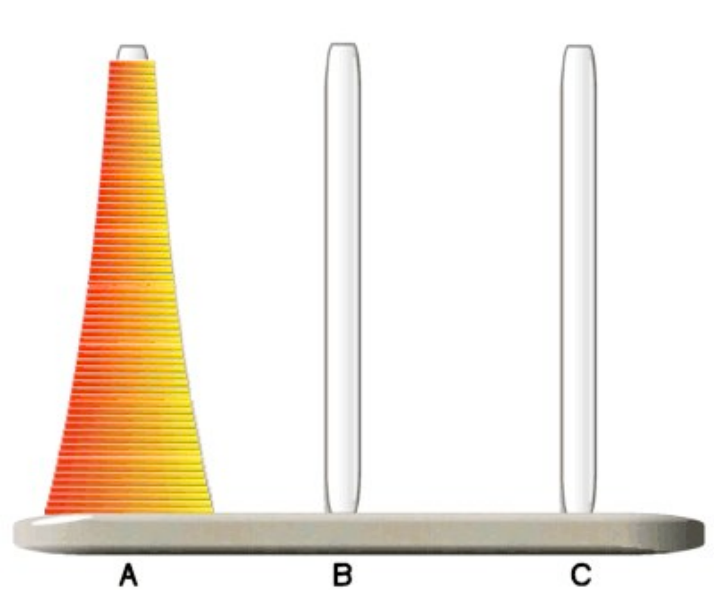
Alle Scheiben von Nadel **A** zu Nadel **C** umzusetzen.

Regeln:

1. Nur eine Scheibe darf auf einmal umgesetzt werden.
2. Es darf sich nie eine kleinere Scheibe unter einer größeren befinden.

Sobald alle 64 Scheiben auf einer anderen Nadel sind, wird die Welt untergehen.

Hanoi-Turm



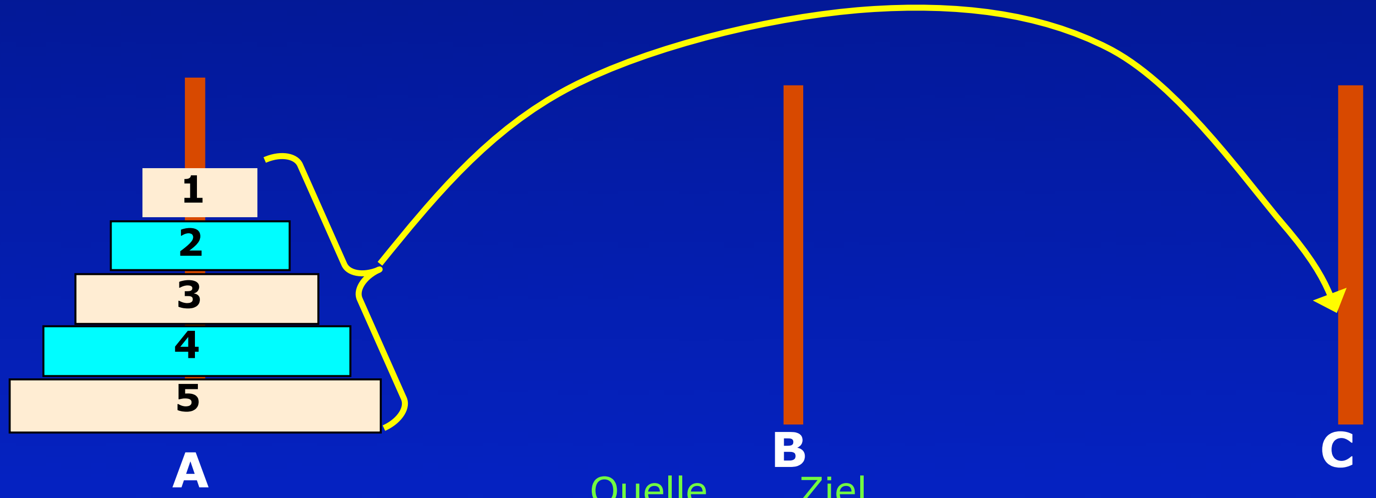
Welche ist die minimale Bewegungssequenz ohne Wiederholungen und Endlosschleifen?

Eingabegröße n = Anzahl der Scheiben

Berechnungsschritte Züge und rekursive Aufrufe

Komplexitätsanalyse Z(n) = Die minimale Anzahl von Zügen + rekursiven Aufrufen

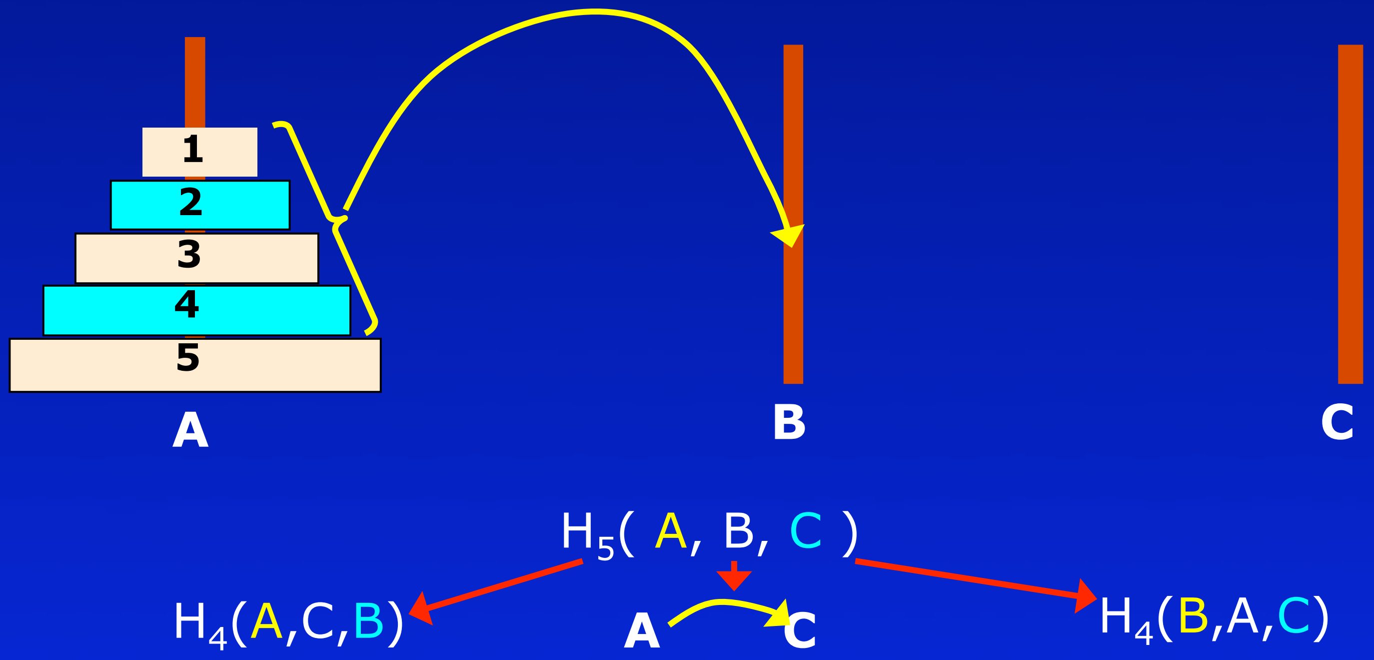
Hanoi-Turm



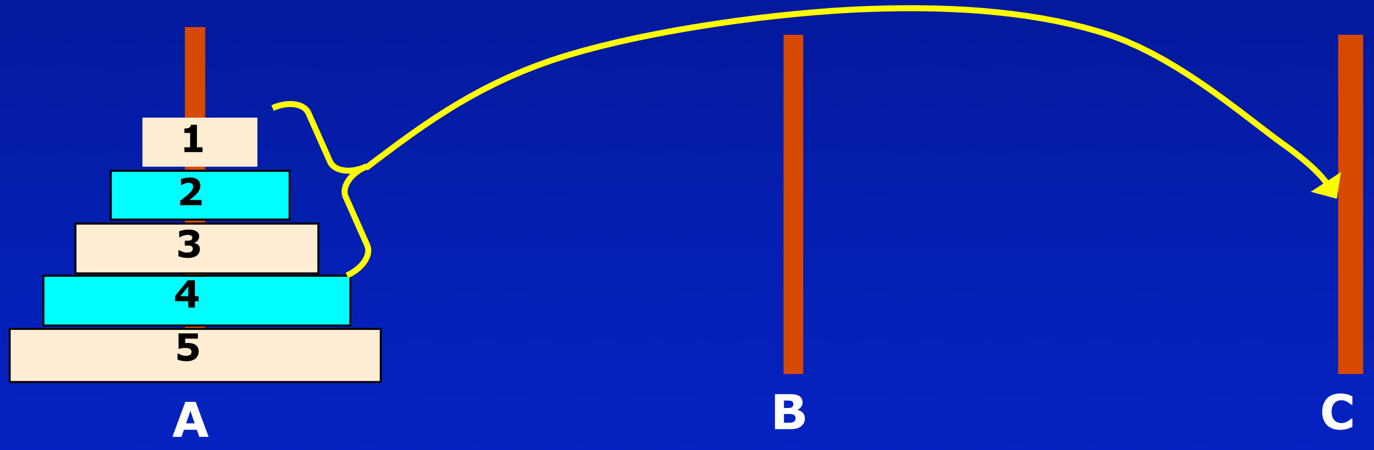
Quelle Ziel

$$H_5(A, B, C)$$

Hanoi-Turm



Hanoi-Turm

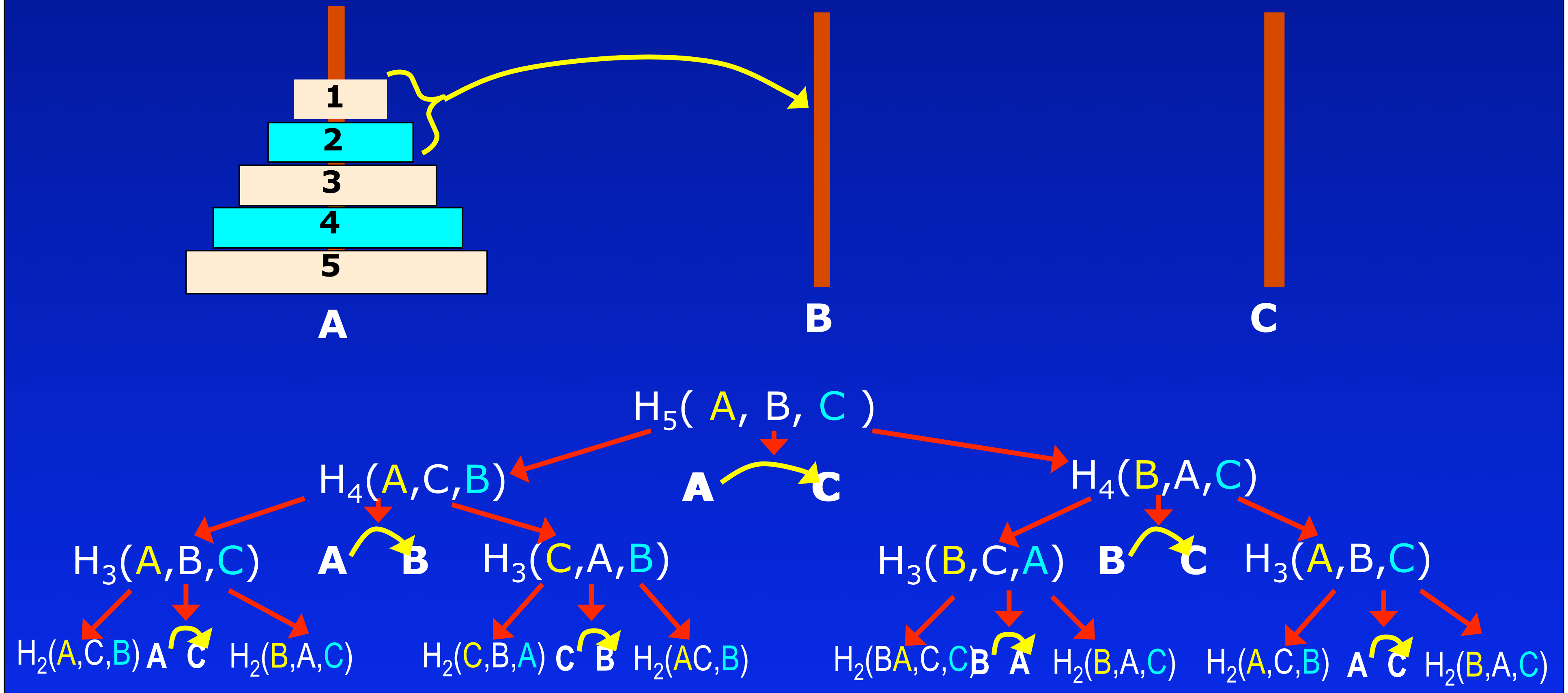


$H_5(A, B, C)$

$H_4(A,C,B)$ $A \rightarrow C$ $H_4(B,A,C)$

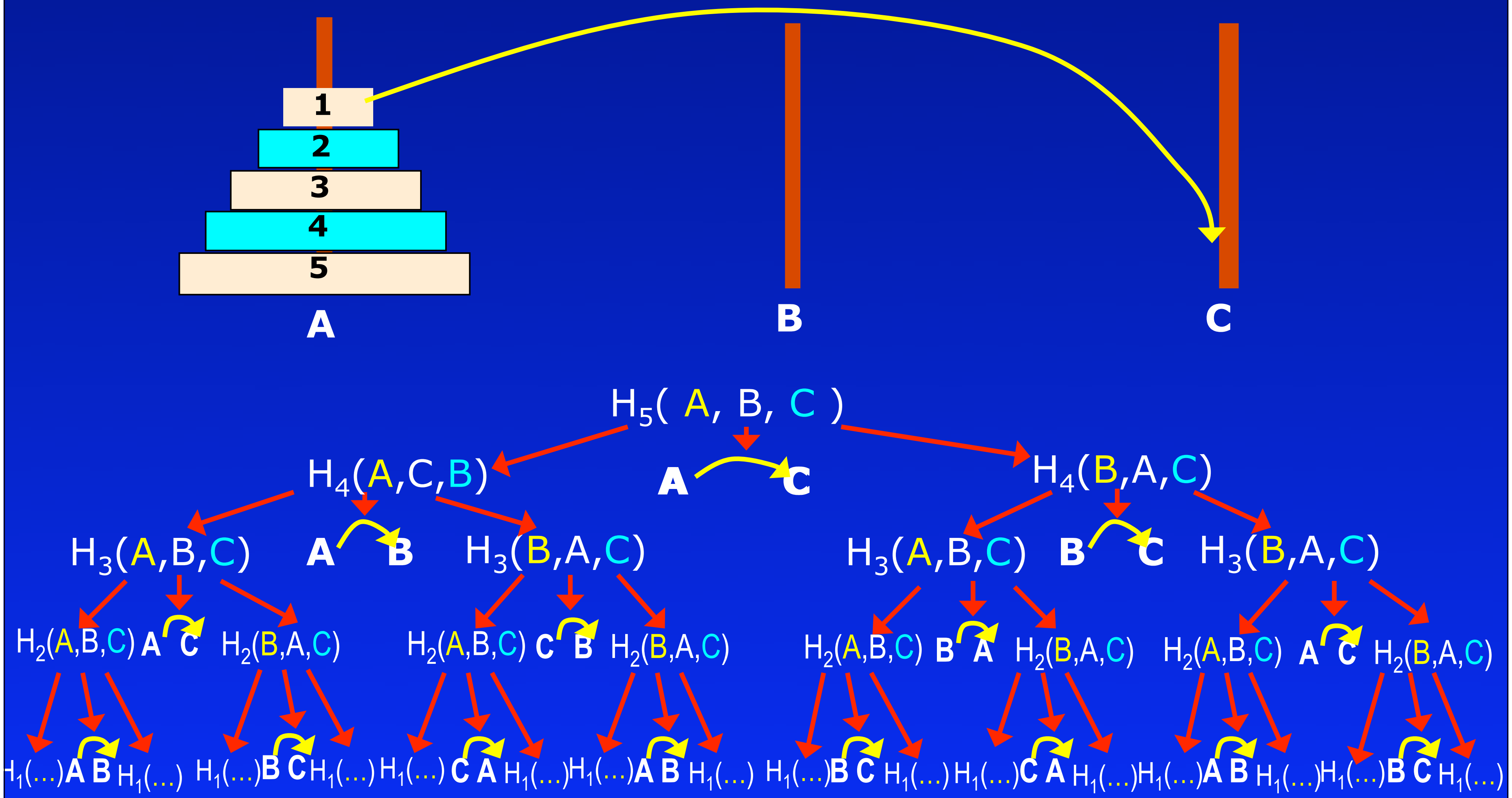
$H_3(A,B,C)$ $A \rightarrow B$ $H_3(C,A,B)$ $H_3(B,C,A)$ $B \rightarrow C$ $H_3(A,B,C)$

Hanoi-Turm



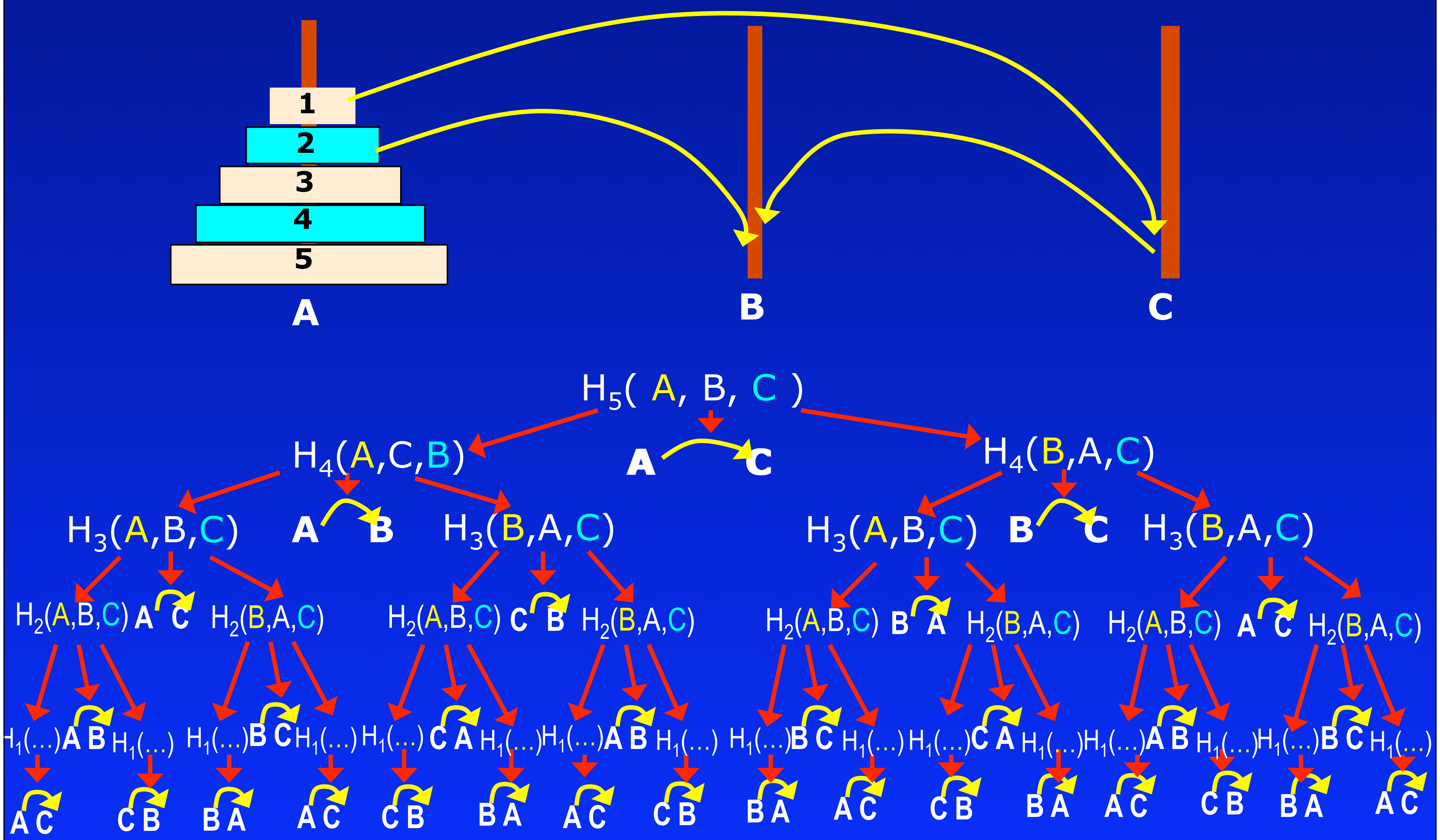
1
2
3
4
5
A
B
C
H_5(A, B, C)
H_4(A,C,B)
A C
H_4(B,A,C)
H_3(A,B,C)
A B
H_3(C,A,B)
H_3(B,C,A)
B C
H_3(A,B,C)
H_2(A,C,B) A C H_2(B,A,C)
H_2(C,B,A) C B H_2(AC,B)
H_2(BA,C,C B A H_2(B,A,C)
H_2(A,C,B) A C H_2(B,A,C)

Hanoi-Turm



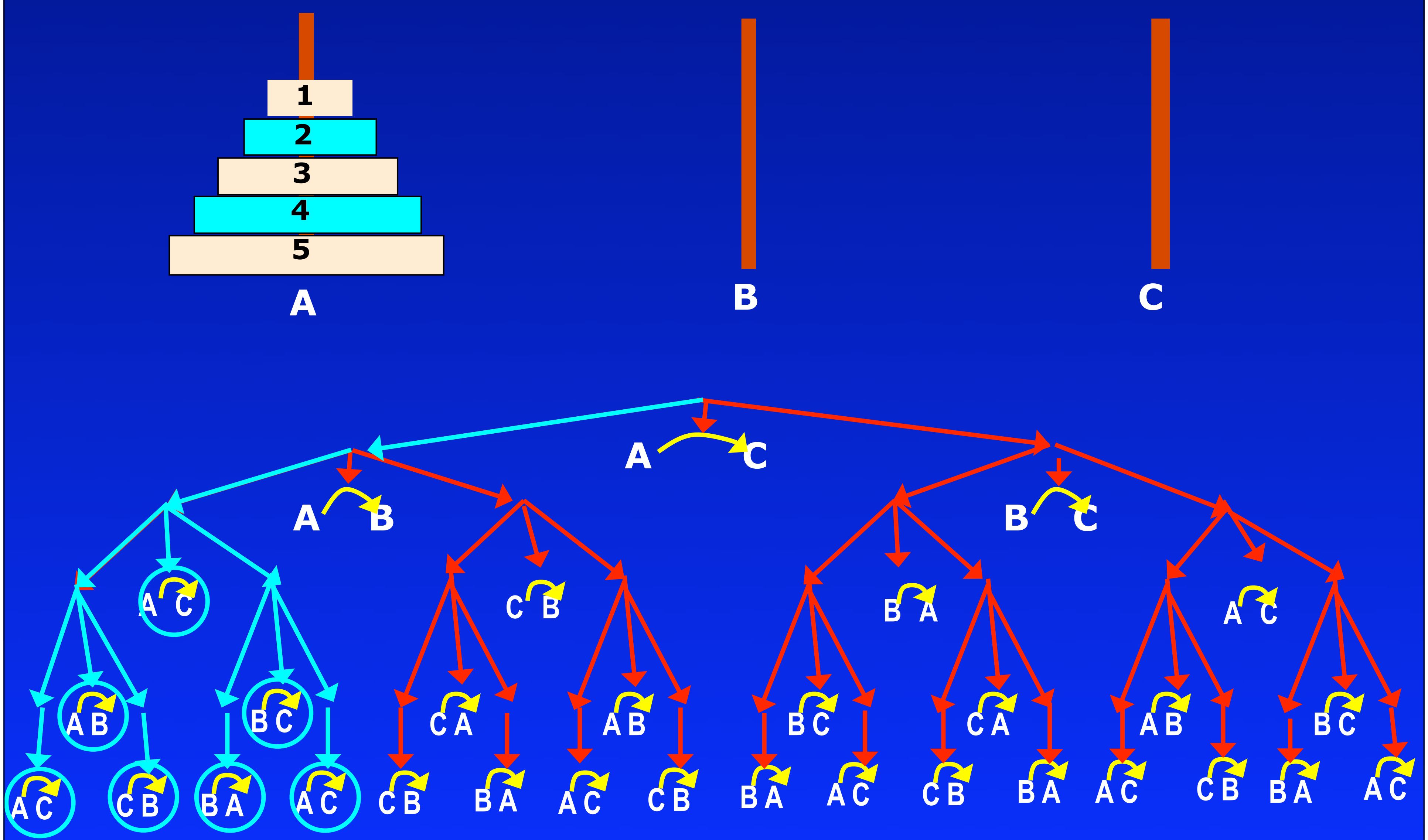
1
2
3
4
5
A
B
C
H5(A, B, C)
H4(A,C,B)
A C
H4(B,A,C)
H3(A,B,C) A B H3(B,A,C)
H3(A,B,C) B C H3(B,A,C)
H2(A,B,C) A C H2(B,A,C)
H2(A,B,C) C B H2(B,A,C)
H2(A,B,C) B A H2(B,A,C)
H2(A,B,C) A C H2(B,A,C)
H1(...) A B H1(...) H1(...) B C H1(...) H1(...) C A H1(...) H1(...) A B H1(...) H1(...) B C H1(...) H1(...) C A H1(...) H1(...) A B H1(...) H1(...) B C H1(...)

Hanoi-Turm



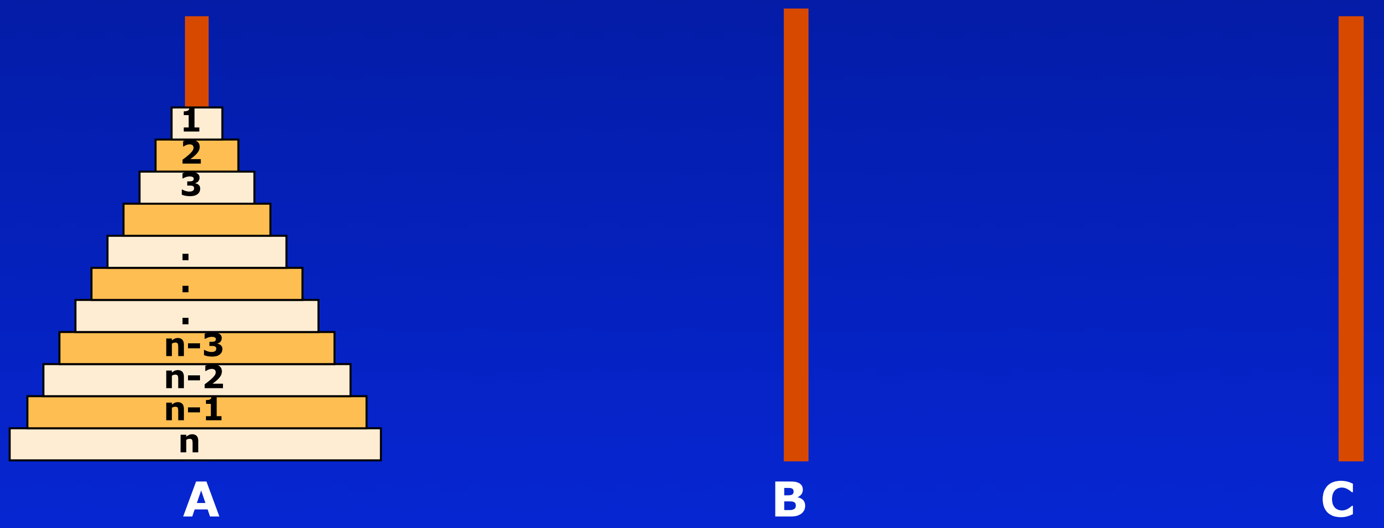
1
2
3
4
5
A
B
C
H5(A, B, C)
H4(A,C,B)
A C
H4(B,A,C)
H3(A,B,C)
A B
H3(B,A,C)
H3(A,B,C)
B C
H3(B,A,C)
H2(A,B,C) A C H2(B,A,C)
H2(A,B,C) C B H2(B,A,C)
H2(A,B,C) B A H2(B,A,C)
H2(A,B,C) A C H2(B,A,C)
H1(...)A B H1(...) H1(...)B C H1(...) H1(...) C A H1(...)H1(...)A B H1(...) H1(...)B C H1(...) H1(...)C A H1(...)H1(...)A B H1(...)H1(...)B C H1(...)
A C C B B A A C C B B A A C C B B A A C C B B A A C C B B A A C

Hanoi-Turm



1
2
3
4
5
A
B
C
A C
A B
B C
A C
C B
B A
A B
A C
C B
B A
A C
B C
C A
C B
B A
A B
C A
A B
B C
B A
A C
C A
C B
B A
A C
A B
C B
B A
B C
A C

Hanoi-Turm

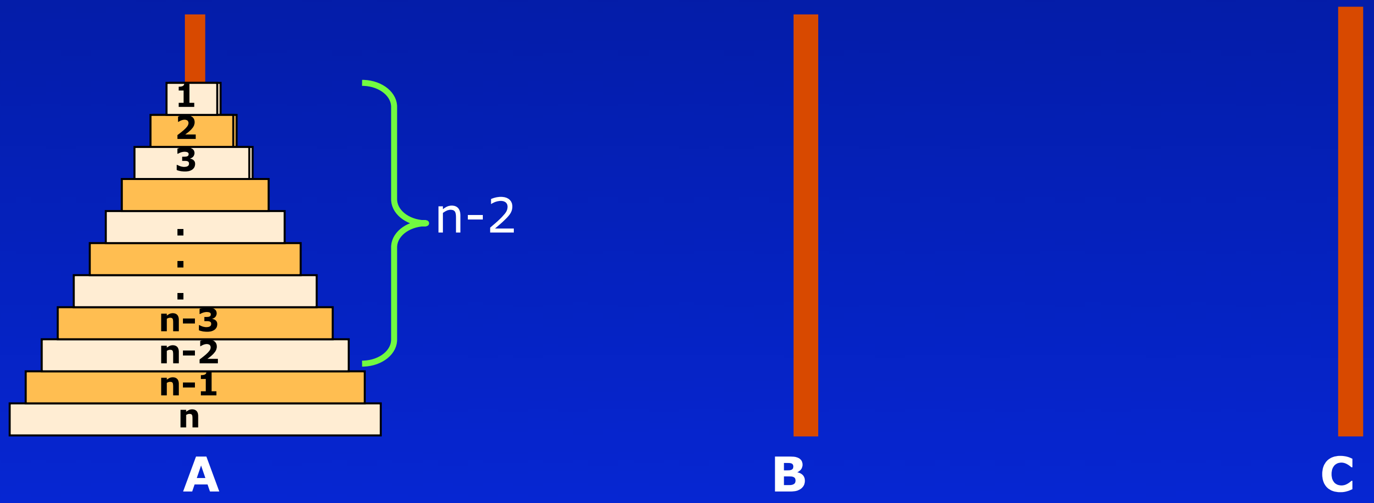


Wenn wir alle Scheiben von **A** nach **C** bewegen möchten, müssen wir zuerst das Problem, **n-1** Scheiben nach **B** zu bewegen, lösen, damit die Scheibe **n** nach **C** bewegt werden kann. Zum Schluss müssen wir noch mal die **n-1** Scheiben von **B** nach **C** bewegen.

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

$$Z(n) = 1 + 2 \cdot \left(Z(n-1) \right)$$

Hanoi-Turm



Aber um **n-1** Scheiben von **A** nach **B** zu bewegen, müssen wir zuerst **n-2** Scheiben von **A** nach **C** bewegen, die **(n-1)-**Scheibe von **A** nach **B** und zum Schluss die **n-2** Scheiben von **C** nach **B**

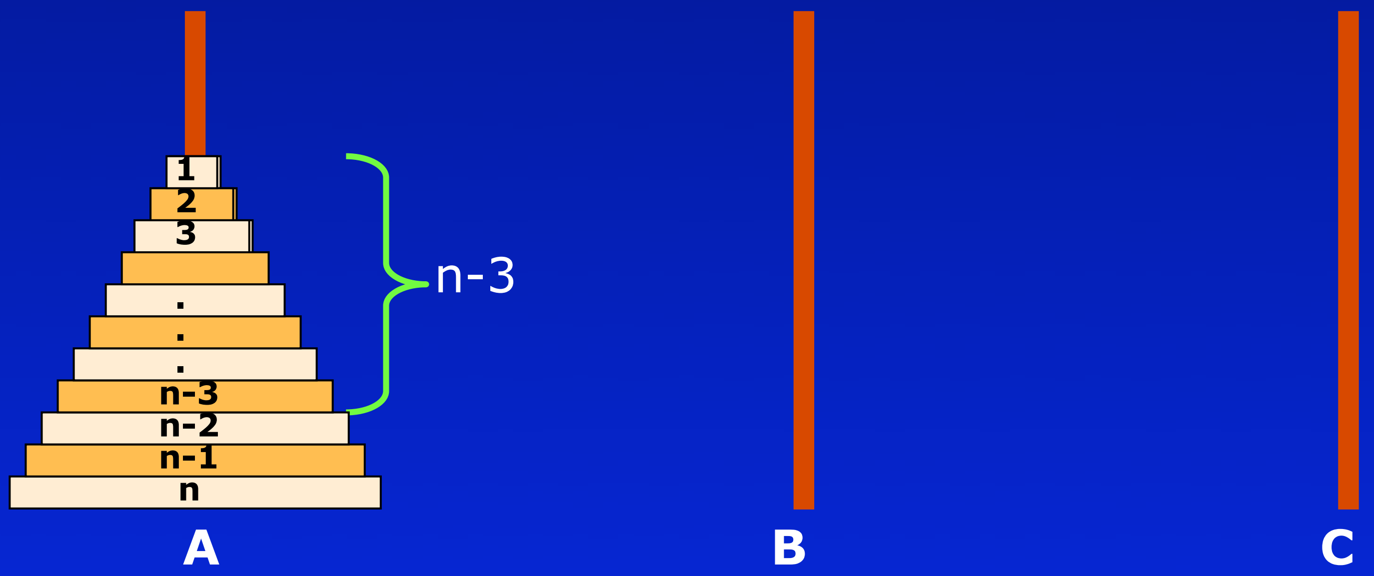
$$Z(n) = 1 + 2 \cdot (Z(n-1))$$

$$Z(n) = 1 + 2 \cdot (Z(n-2) + 1 + Z(n-2))$$

$$Z(n) = 1 + 2 \cdot (1 + 2 \cdot (Z(n-2)))$$

$$Z(n) = 1 + 2 + 2^2 \cdot (Z(n-2)))$$

Hanoi-Turm



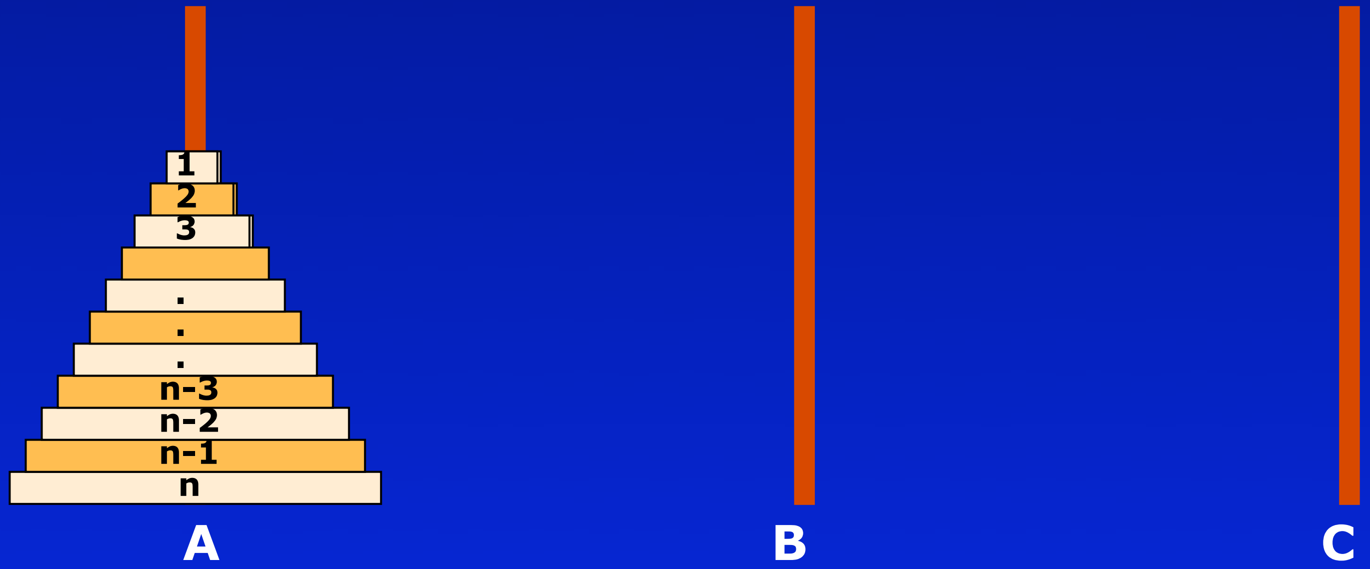
Aber um **n-2** Scheiben von **A** nach **C** zu bewegen, müssen wir zuerst **n-3** Scheiben von **A** nach **B** bewegen, die **(n-2)-**Scheibe von **A** nach **C** und zum Schluss die **n-3** Scheiben von **B** nach **C**

$$Z(n) = 1 + 2 + 2^2 \cdot Z(n-2)\ )$$

$$Z(n) = 1 + 2 + 2^2 \cdot (\ 1 + 2 \cdot Z(n-3)\ )$$

$$Z(n) = 1 + 2 + 2^2 + 2^3 \cdot Z(n-3)$$

Hanoi-Turm



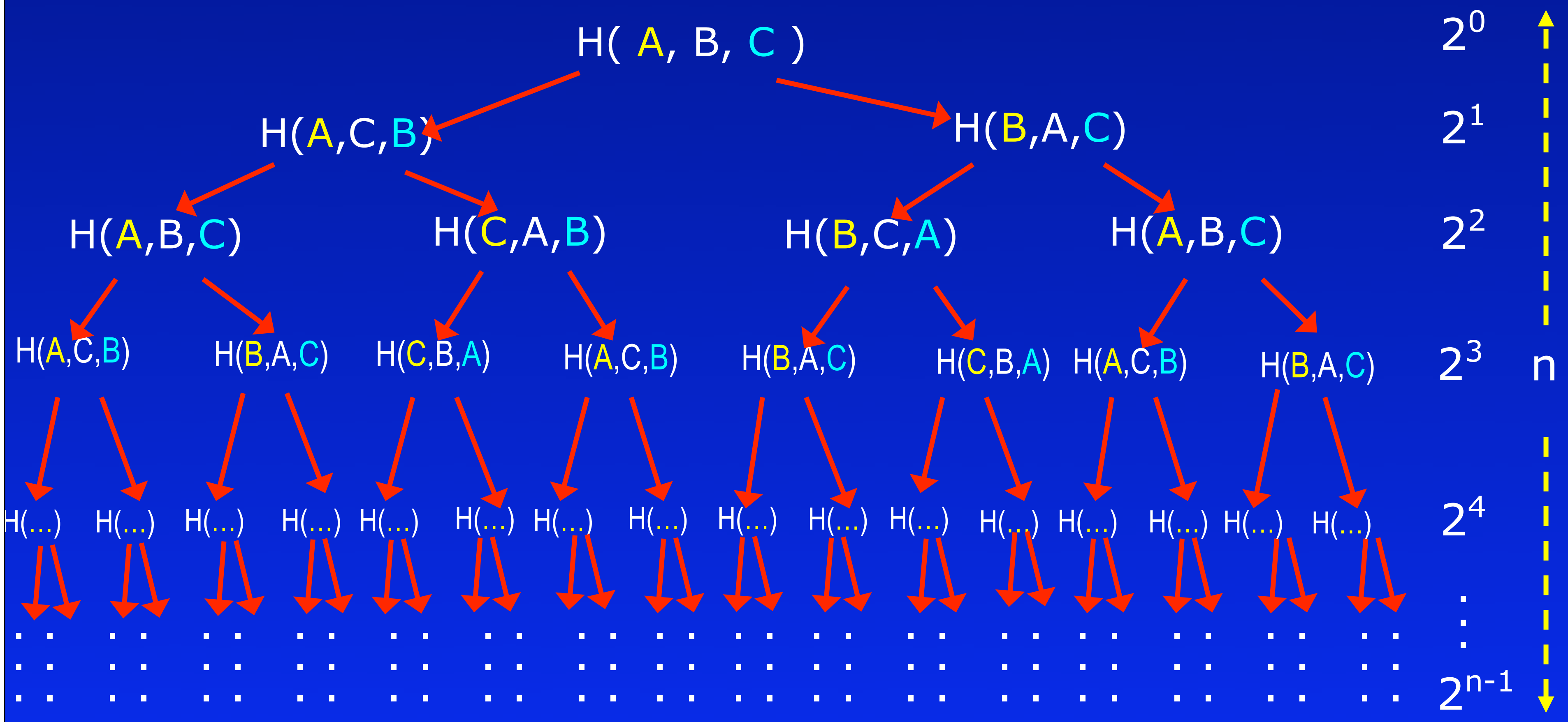
$$Z(n) = 1 + 2 + 2^2 + 2^3 + . . . +2^{n-1} \cdot Z(n-(n-1))$$

$$Z(n) = 1 + 2 + 2^2 + 2^3 + . . . +2^{n-1} \cdot Z(1)))$$

$$Z(n) = 1 + 2 + 2^2 + 2^3 + . . . +2^{n-1}$$

$$Z(n) = 2^n - 1$$

Hanoi-Turm



$$\mathbf{Z(n)} = 1 + 2 + 2^2 + 2^3 + \ldots + 2^{n-1} = \mathbf{2^n - 1}$$

Hanoi-Turm

$$\mathbf{Z(n)} = 1 + 2 + 2^2 + 2^3 + \ldots + 2^{n-1} = \mathbf{2^n - 1}$$

$$1 + 1 + 2 + 2^2 + 2^3 + \ldots + 2^{n-1} = \mathbf{2^n - 1} + 1$$

$$2 + 2 + 2^2 + 2^3 + \ldots + 2^{n-1} = \mathbf{2^n}$$

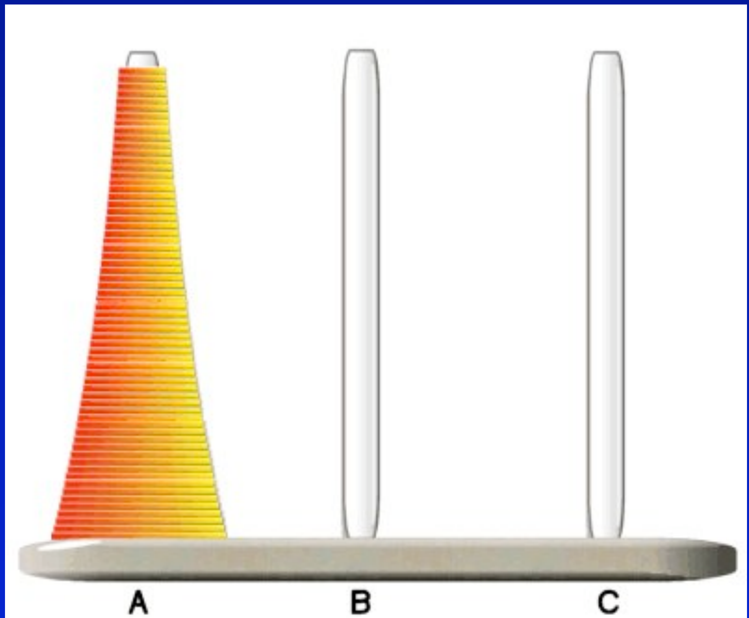
$$2^2 + 2^2 + 2^3 + \ldots + 2^{n-1} = \mathbf{2^n}$$

$$2^3 + 2^3 + \ldots + 2^{n-1} = \mathbf{2^n}$$

$$\ldots$$

$$2^{n-1} + 2^{n-1} = \mathbf{2^n}$$

Hanoi-Turm



$Z(n)$ = Anzahl der Züge

Komplexität

$O(2^n)$

$Z(n) = 2^n - 1$

$Z(64) = 2^{64} - 1$

$Z(64) =$ 18 446 744 073 709 551 615 Züge

Wenn jede Scheibe innerhalb einer Sekunde umgesetzt wird, dauert es

584 942 417 355 Jahre

Das Universum ist **12 000 000 000 Jahre** alt.

Die Erde ist **4 500 000 000 Jahre** alt.

Das Sonnenlicht reicht nur für weitere **5 000 000 000 Jahre**.

Die Priester werden leider im Dunkeln weiter arbeiten müssen!

Hanoi-Turm

Rekursive Python-Implementierung:

```
def hanoi(n,s,m,q):
    if n==0:
        return []
    else:
        return hanoi((n-1),s,q,m)+[(n,s,q)]+hanoi((n-1),m,s,q)
```

Hanoi-Turm

Iterativer Algorithmus von Buneman und Levy (1980)

Start:

if (n 'mod' 2) == 0

then bewegen Sie die kleinste Scheibe zur Hilfsstange

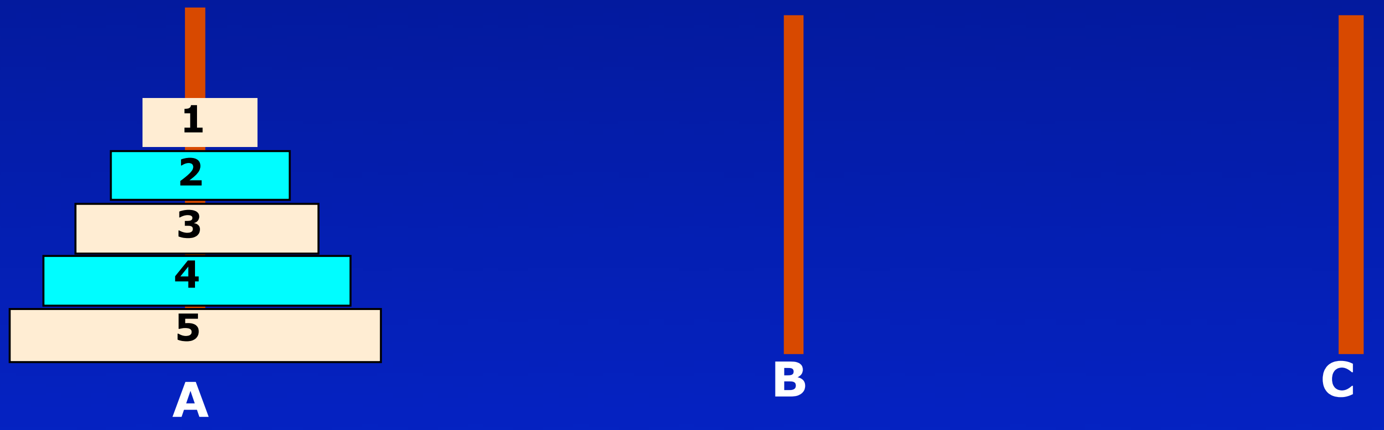
else bewegen Sie die kleinste Scheibe zur Zielstange

So lange nicht alle Scheiben am Ziel sind:

1. bewegen Sie die kleinste Scheibe zu der Stange, wo diese als letztes nicht gewesen ist.
2. machen Sie eine legale Bewegung, ohne die kleinste Scheibe zu bewegen (nur eine Bewegung ist immer möglich).

Hanoi-Turm

Fall n>2

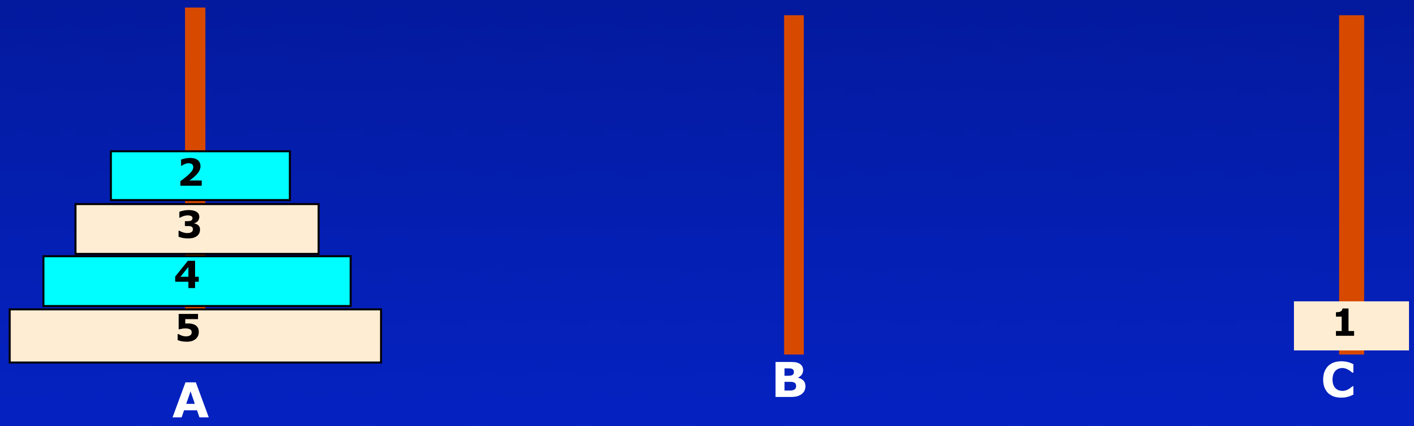


Quelle Ziel

$H_5(A, B, C)$

Hanoi-Turm

Start

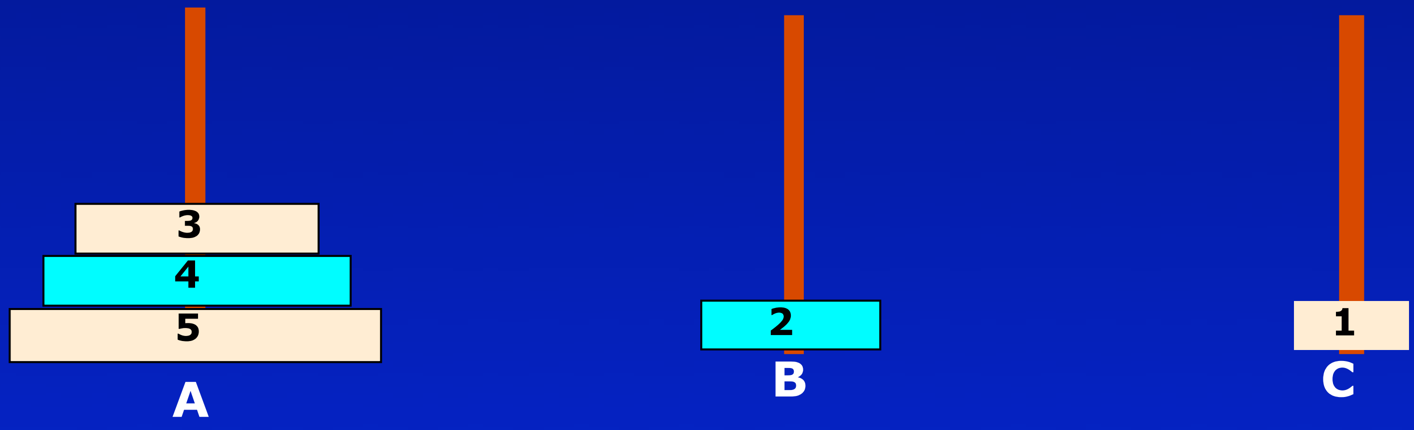


Quelle Ziel

$H_5(A, B, C)$

Hanoi-Turm

Start

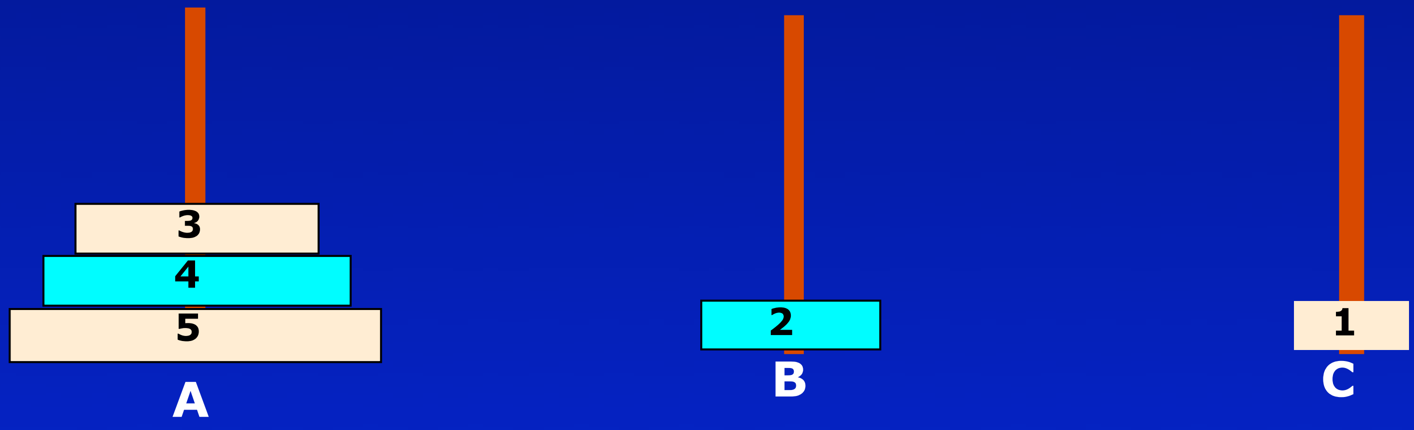


Quelle Ziel

$H_5(A, B, C)$

Hanoi-Turm

Loop



Quelle Ziel

$$H_5(A, B, C)$$

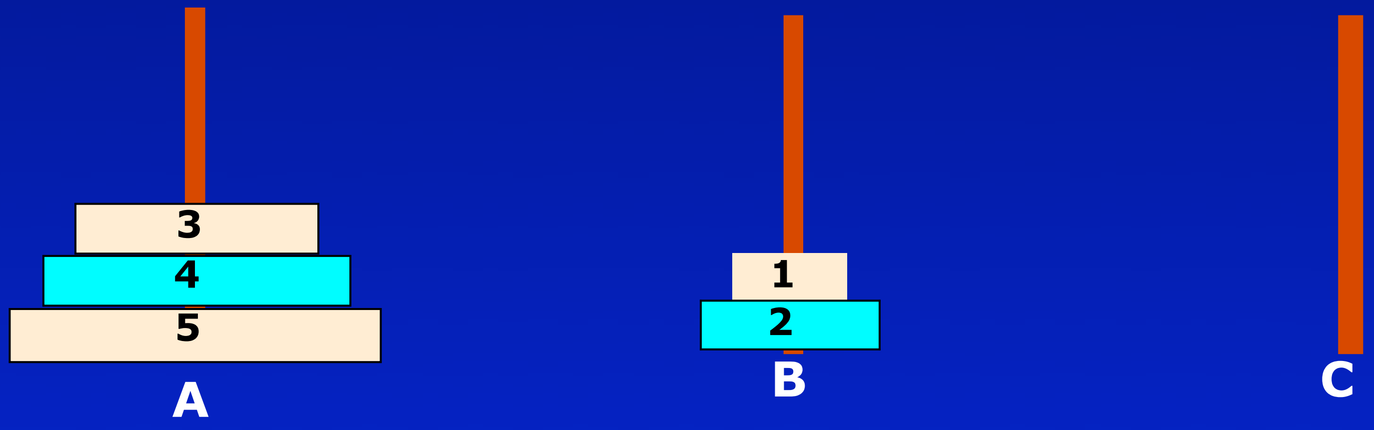
So lange nicht alle Scheiben am Ziel sind:

1. bewegen Sie die kleinste Scheibe zu der Stange, wo diese als letztes nicht gewesen ist.

2. machen Sie eine legale Bewegung, ohne die kleinste Scheibe zu bewegen (nur eine Bewegung ist immer möglich).

Hanoi-Turm

Loop



$H_5($A$, B, C)$

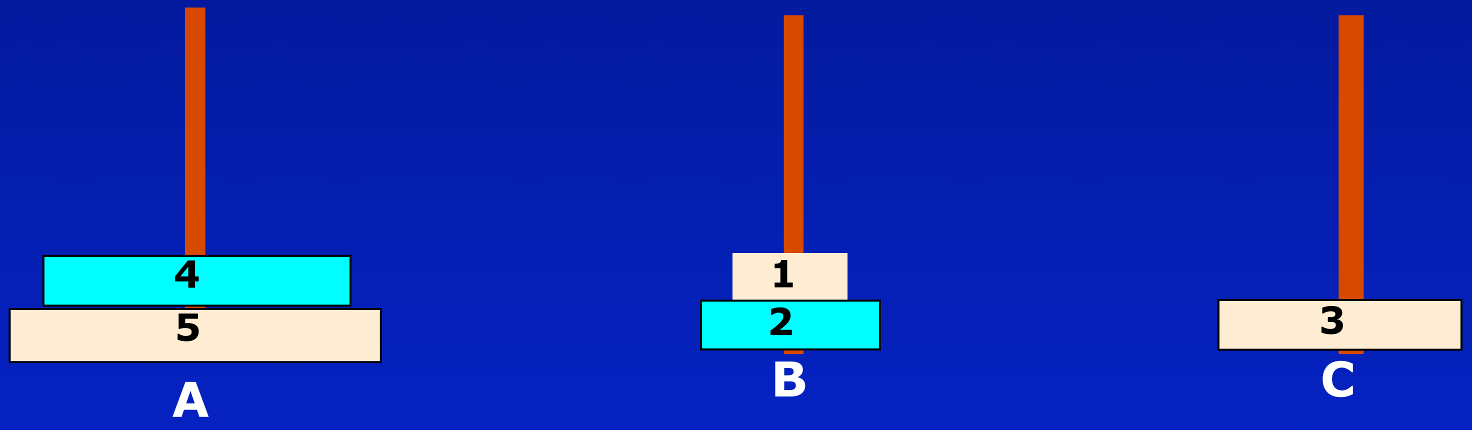
So lange nicht alle Scheiben am Ziel sind:

1. **bewegen Sie die kleinste Scheibe zu der Stange, wo diese als letztes nicht gewesen ist.**

2. machen Sie eine legale Bewegung, ohne die kleinste Scheibe zu bewegen (nur eine Bewegung ist immer möglich).

Hanoi-Turm

Loop



Quelle Ziel

$$H_5(A, B, C)$$

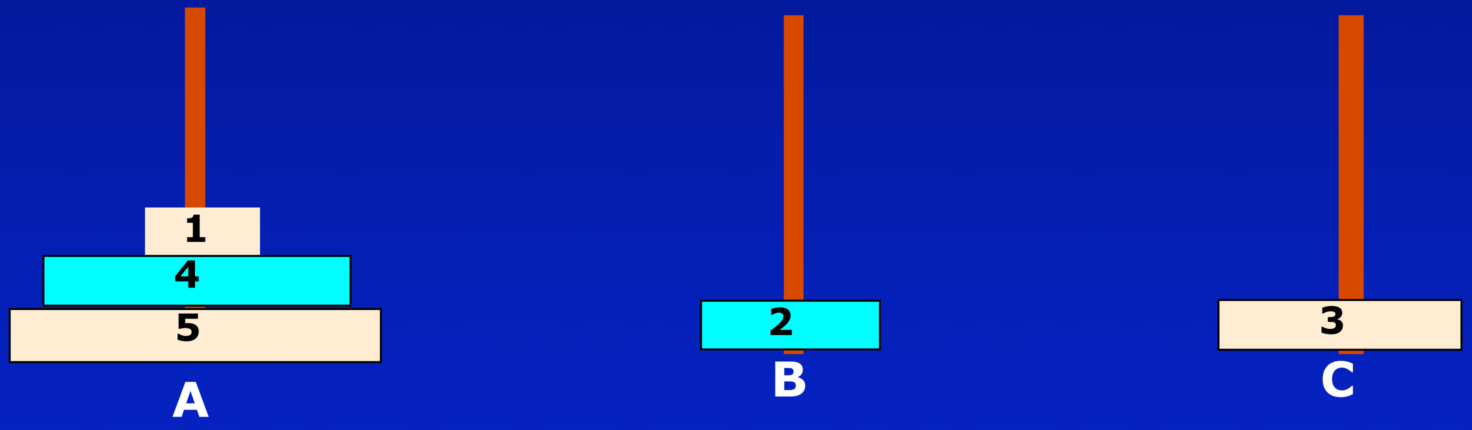
So lange nicht alle Scheiben am Ziel sind:

1. bewegen Sie die kleinste Scheibe zu der Stange, wo diese als letztes nicht gewesen ist.

2. **machen Sie eine legale Bewegung, ohne die kleinste Scheibe zu bewegen (nur eine Bewegung ist immer möglich).**

Hanoi-Turm

Loop



Quelle Ziel

$H_5(A, B, C)$

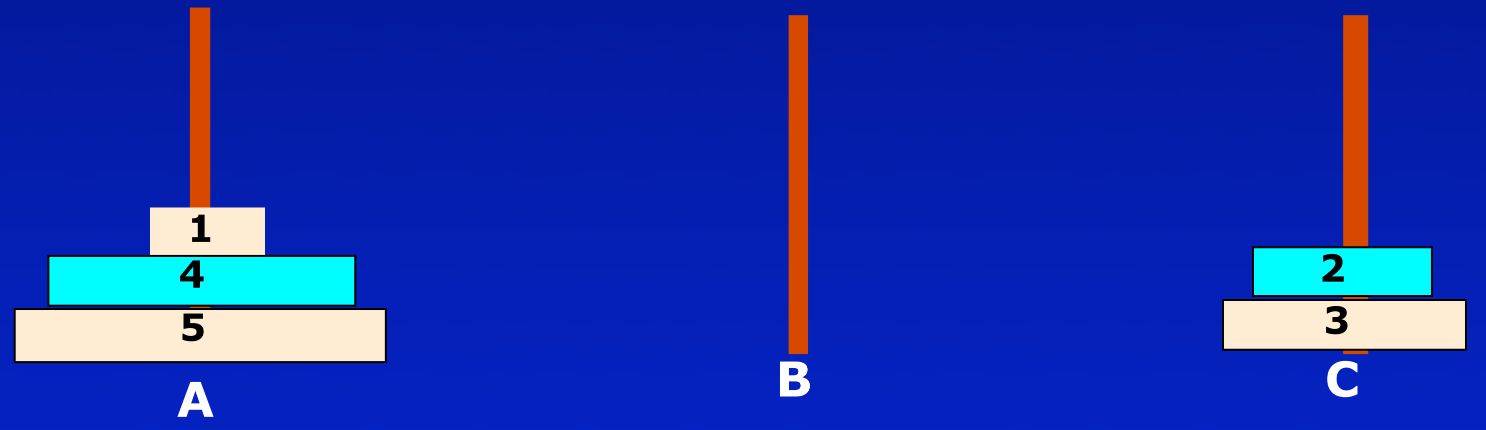
So lange nicht alle Scheiben am Ziel sind:

1. **bewegen Sie die kleinste Scheibe zu der Stange, wo diese als letztes nicht gewesen ist.**

2. machen Sie eine legale Bewegung, ohne die kleinste Scheibe zu bewegen (nur eine Bewegung ist immer möglich).

Hanoi-Turm

Loop



Quelle Ziel

$H_5(A, B, C)$

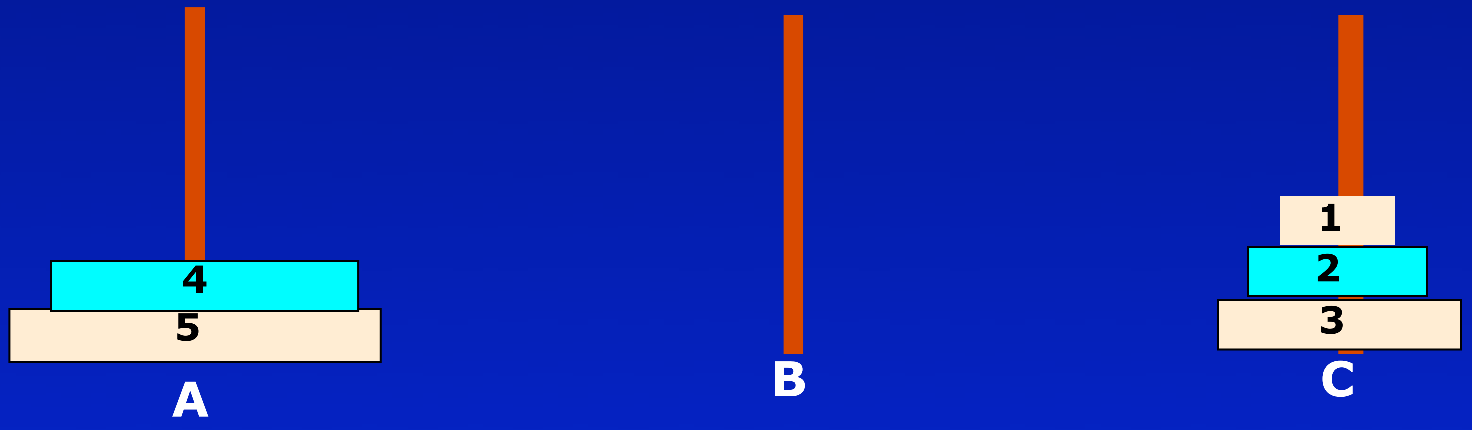
So lange nicht alle Scheiben am Ziel sind:

1. bewegen Sie die kleinste Scheibe zu der Stange, wo diese als letztes nicht gewesen ist.

2. **machen Sie eine legale Bewegung, ohne die kleinste Scheibe zu bewegen (nur eine Bewegung ist immer möglich).**

Hanoi-Turm

Loop



Quelle Ziel

$H_5(A, B, C)$

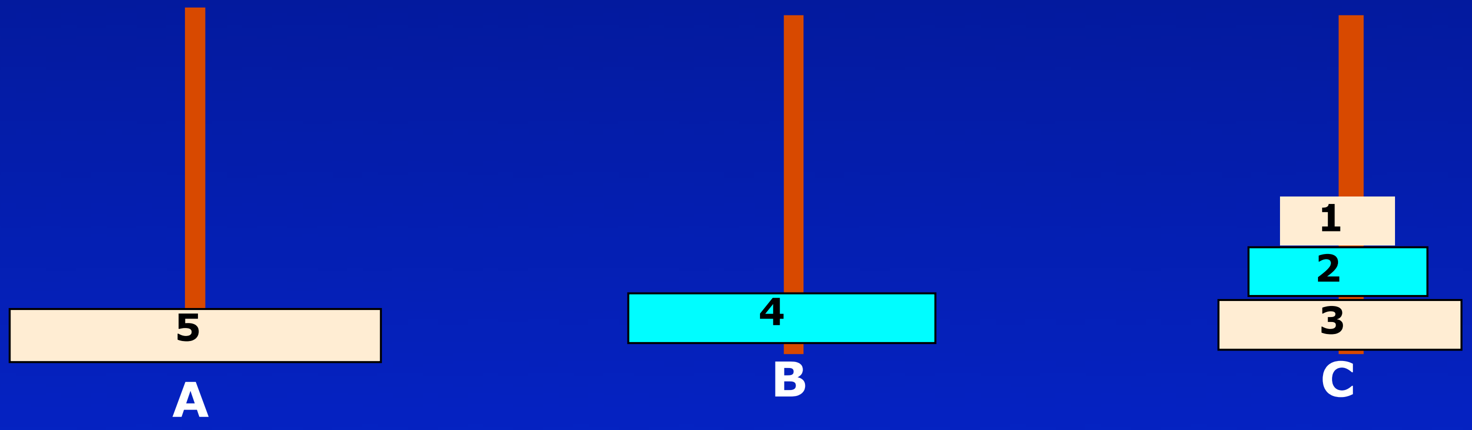
So lange nicht alle Scheiben am Ziel sind:

1. **bewegen Sie die kleinste Scheibe zu der Stange, wo diese als letztes nicht gewesen ist.**

2. machen Sie eine legale Bewegung, ohne die kleinste Scheibe zu bewegen (nur eine Bewegung ist immer möglich).

Hanoi-Turm

Loop



Quelle Ziel

$H_5(A, B, C)$

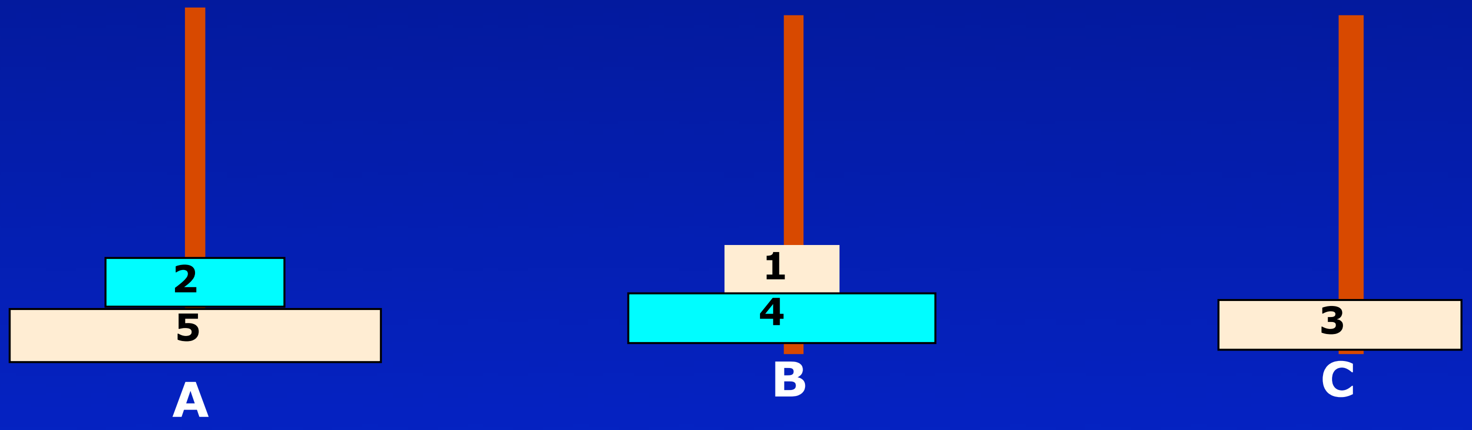
So lange nicht alle Scheiben am Ziel sind:

1. bewegen Sie die kleinste Scheibe zu der Stange, wo diese als letztes nicht gewesen ist.

2. **machen Sie eine legale Bewegung, ohne die kleinste Scheibe zu bewegen (nur eine Bewegung ist immer möglich).**

Hanoi-Turm

Loop



Quelle Ziel

$H_5(A, B, C)$

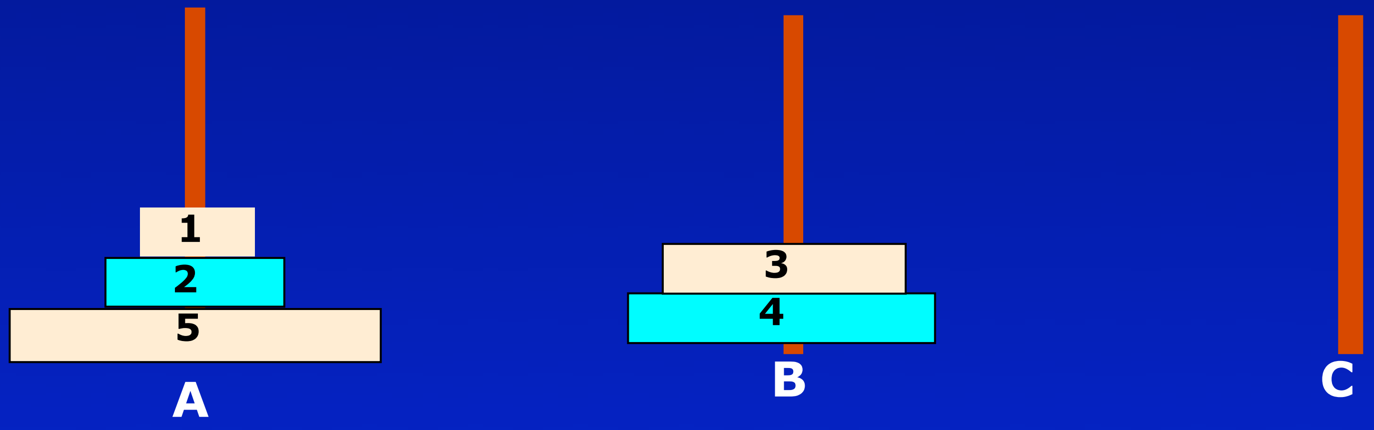
So lange nicht alle Scheiben am Ziel sind:

1. **bewegen Sie die kleinste Scheibe zu der Stange, wo diese als letztes nicht gewesen ist.**

2. **machen Sie eine legale Bewegung, ohne die kleinste Scheibe zu bewegen (nur eine Bewegung ist immer möglich).**

Hanoi-Turm

Loop



Quelle Ziel

$H_5(A, B, C)$

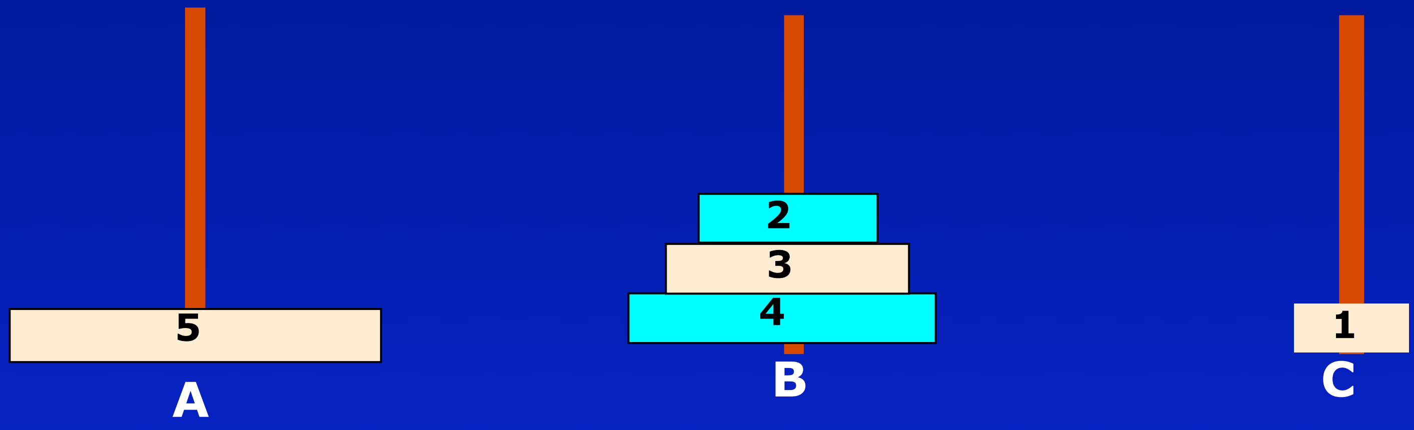
So lange nicht alle Scheiben am Ziel sind:

1. **bewegen Sie die kleinste Scheibe zu der Stange, wo diese als letztes nicht gewesen ist.**

2. **machen Sie eine legale Bewegung, ohne die kleinste Scheibe zu bewegen (nur eine Bewegung ist immer möglich).**

Hanoi-Turm

Loop



Quelle Ziel

$H_5(A, B, C)$

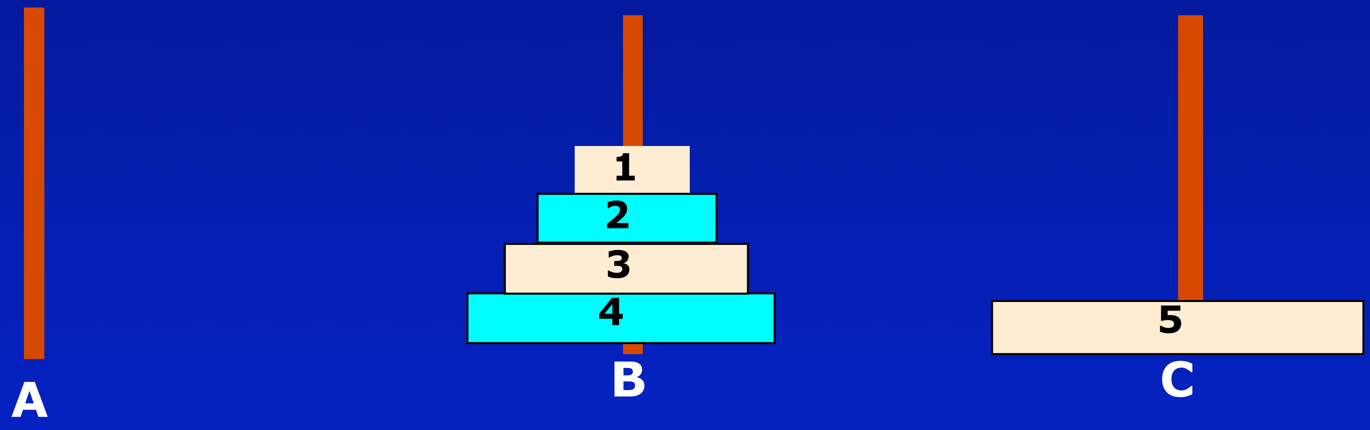
So lange nicht alle Scheiben am Ziel sind:

1. **bewegen Sie die kleinste Scheibe zu der Stange, wo diese als letztes nicht gewesen ist.**

2. **machen Sie eine legale Bewegung, ohne die kleinste Scheibe zu bewegen (nur eine Bewegung ist immer möglich).**

Hanoi-Turm

Loop



usw.... bis alle Scheiben am Ziel sind

So lange nicht alle Scheiben am Ziel sind:

1. **bewegen Sie die kleinste Scheibe zu der Stange, wo diese als letztes nicht gewesen ist.**

2. **machen Sie eine legale Bewegung, ohne die kleinste Scheibe zu bewegen (nur eine Bewegung ist immer möglich).**