Základy algoritmizace

6. Rekurze

doc. Ing. Jiří Vokřínek, Ph.D.
Katedra počítačů
Fakulta elektrotechnická
České vysoké učení technické v Praze

Základy algoritmizace

- Dnes:
  - Rekurze
    - Faktoriál
    - Obrácený výpis posloupnosti
    - Hanojské věže
    - Fibonacciho posloupnost

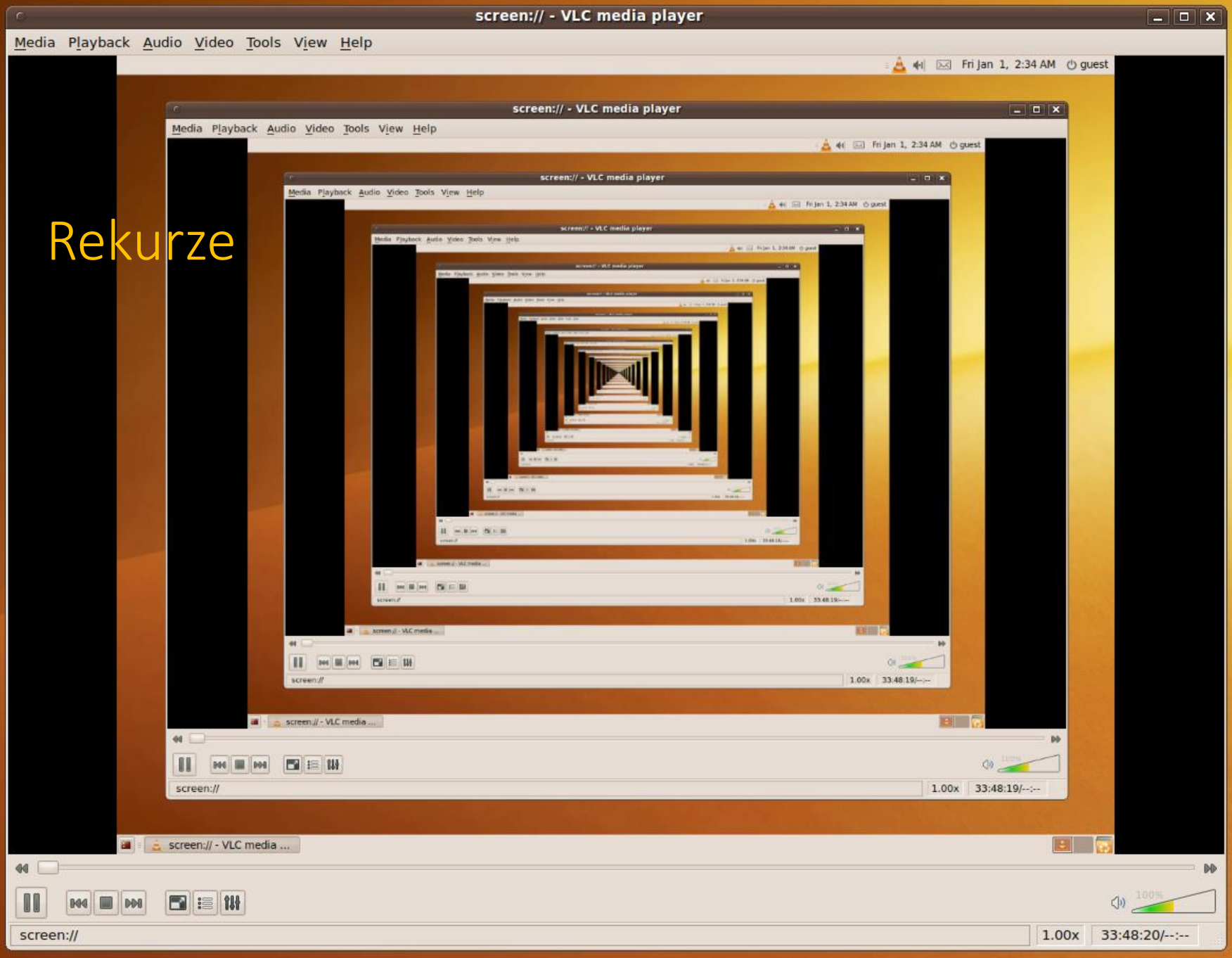
Rekurze

„To iterate is human, to recurse divine“

L. Peter Deutsch

http://www.devtopics.com/101-great-computer-programming-quotes/

Rekurze



Source: http://algosaur.us/recursion/

Výpočet faktoriálu

- Iterace

$n! = n*(n-1)*(n-2)* \dots *2*1$

```
def factorialI(n):
    f = 1
    for i in range(n,1,-1):
        f*=i
    return f
```

- Rekurze

$n! = 1$ pro $n \leq 1$

$n! = n*(n-1)!$ pro $n > 1$

```
def factorialR(n):
    if n>1:
        return n * factorialR(n-1)
    else:
        return 1
```

Pozor, Python omezuje počet vnoření (default recursion limit) na 1000

Příklad – výpis posloupnosti

- Úloha

  Vytvořte program, který přečte posloupnost čísel a vypíše ji v opačném pořadí

- Rozklad problému
  - Zavedeme abstraktní příkaz *„obrať posloupnost“*
  - Příkaz rozložíme do tří kroků:
    1. Přečti číslo

       Číslo uložíme pro pozdější „obrácený“ výpis
    2. Pokud není detekován konec, „obrať posloupnost“

       Pokračujeme ve čtení čísel
    3. Vypiš číslo

       Vypíšeme uložené číslo

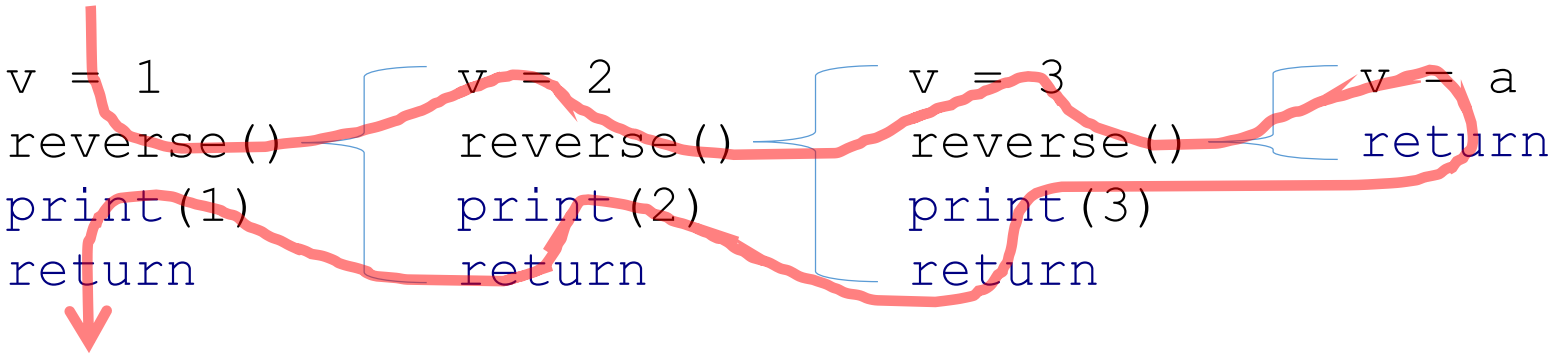
Příklad – výpis posloupnosti

- Řešení

```
def reverse():
    v = input()
    if v.isdigit():
        reverse()
        print(v)
    return

reverse()
```

```
1
2
3
a
3
2
1
```



Příklad – hanojské věže

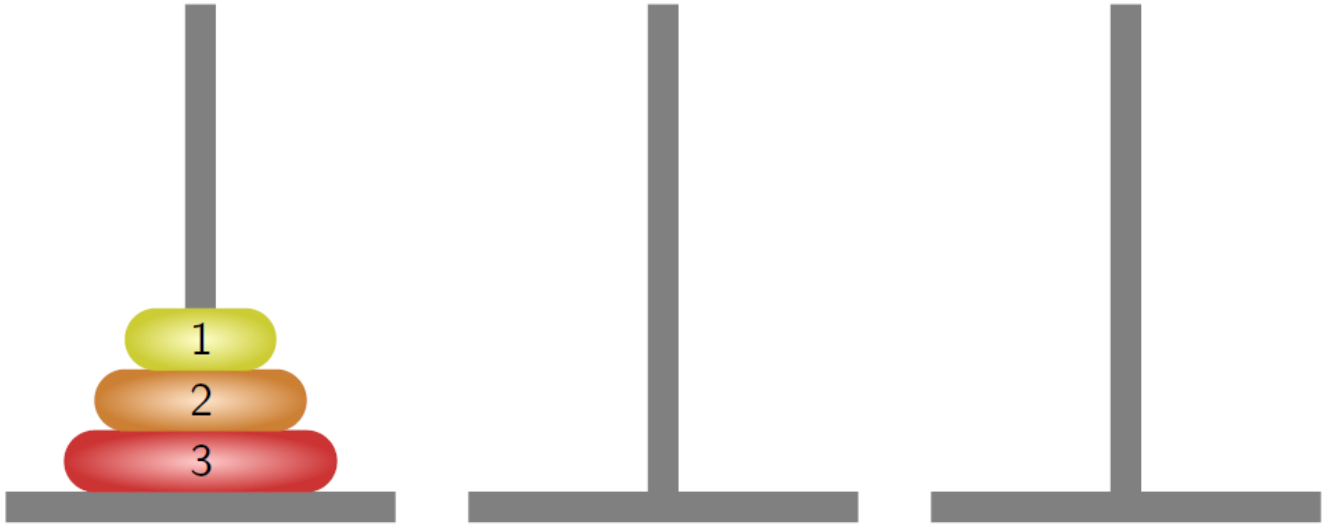
- Úloha

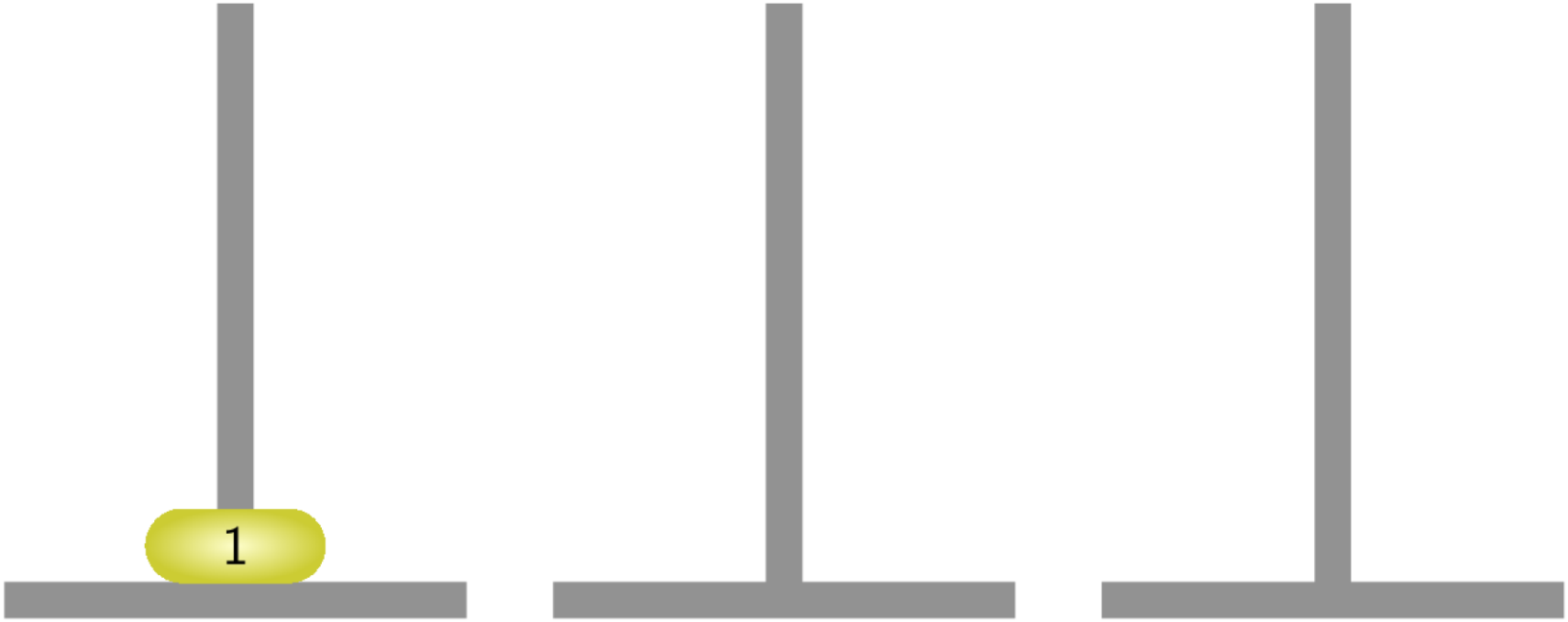
  Přemístit disky na druhou jehlu s použitím třetí (pomocné) za dodržení pravidel:

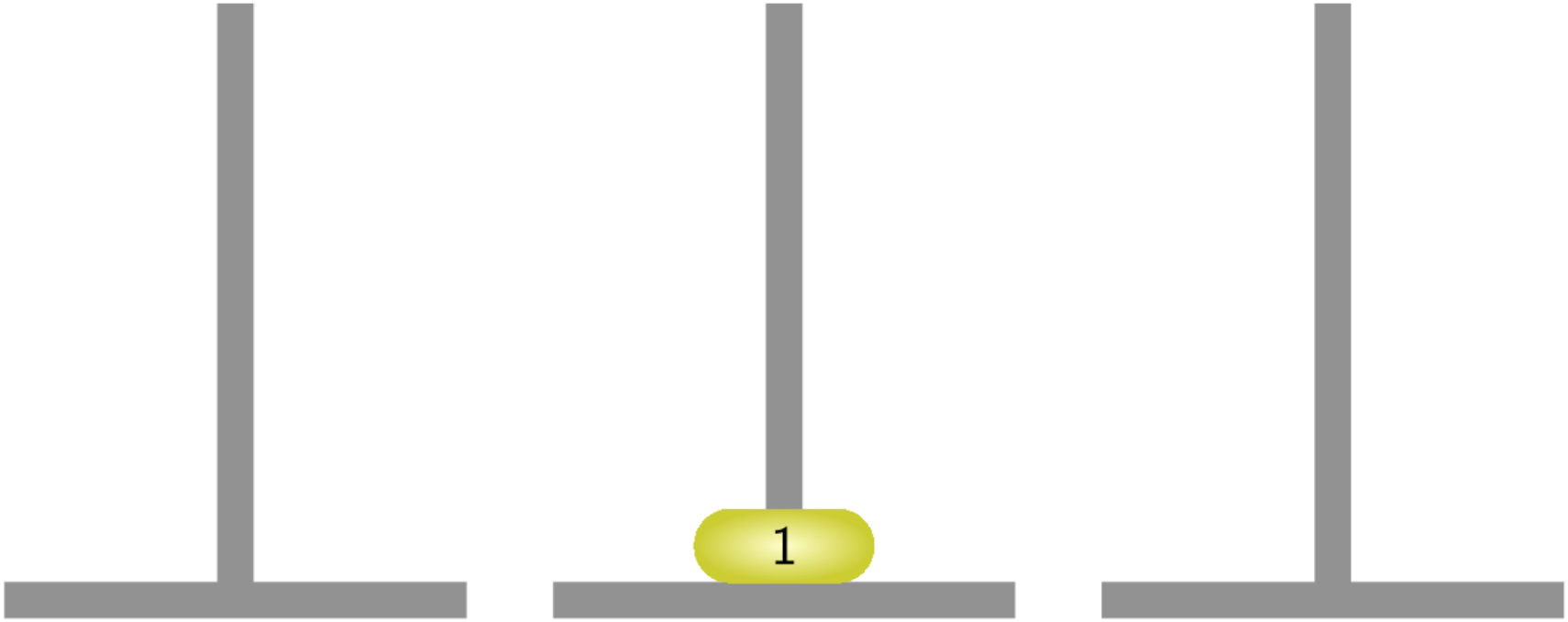
  1. V každém kroku můžeme přemístit pouze jeden disk a to vždy z jehly na jehlu

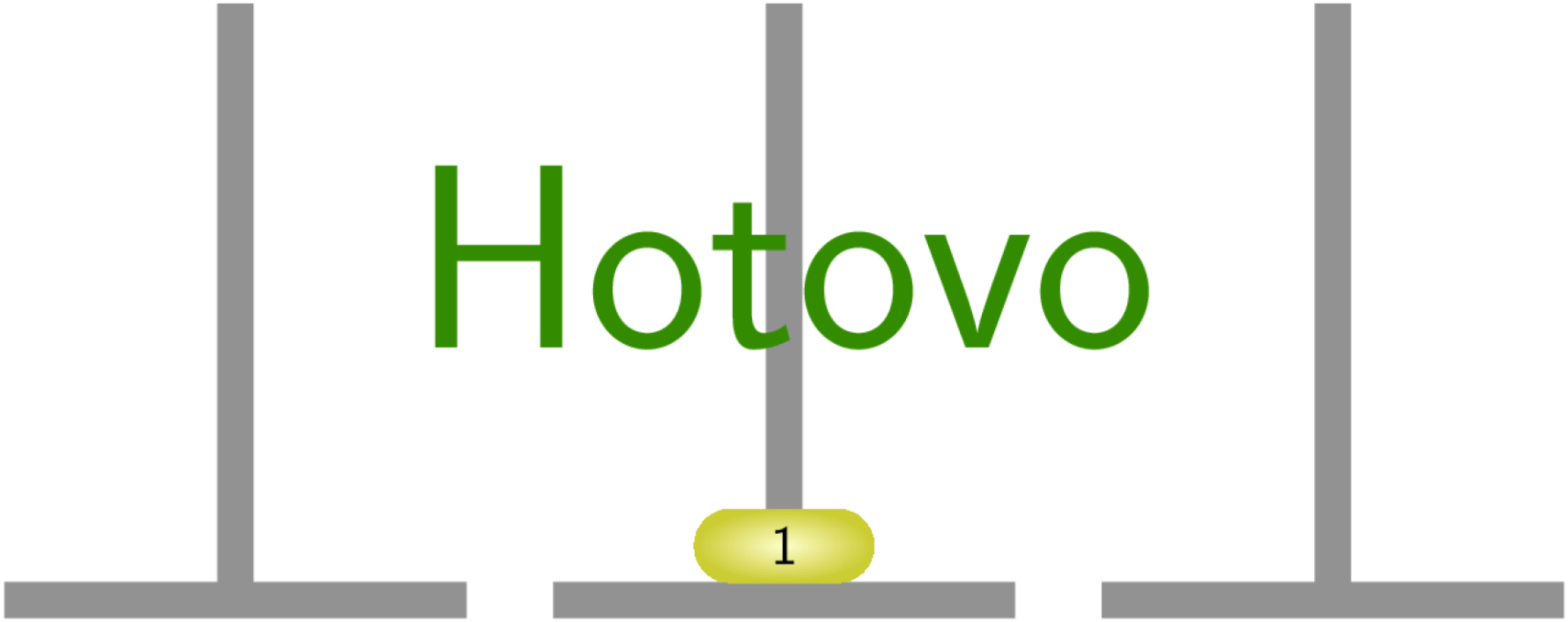
     Disky nelze odkládat mimo jehly

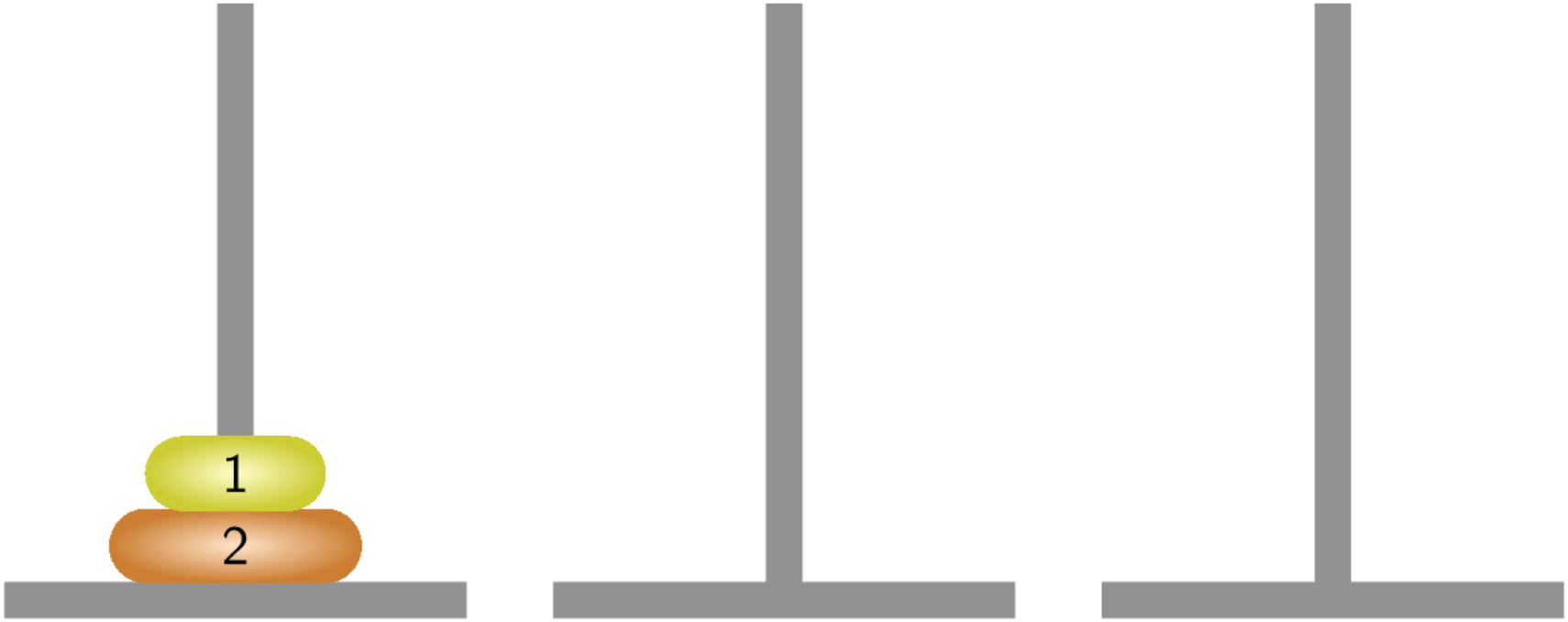
  2. Položit větší disk na menší není povoleno

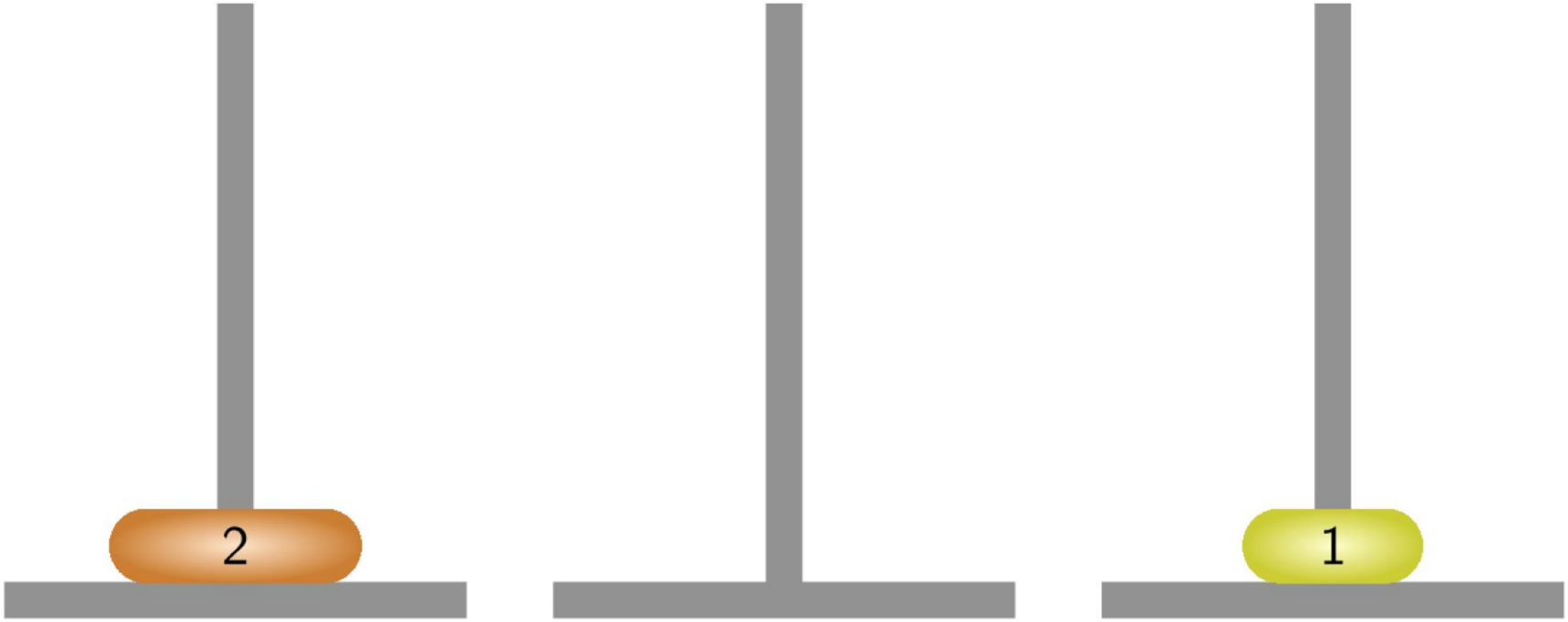


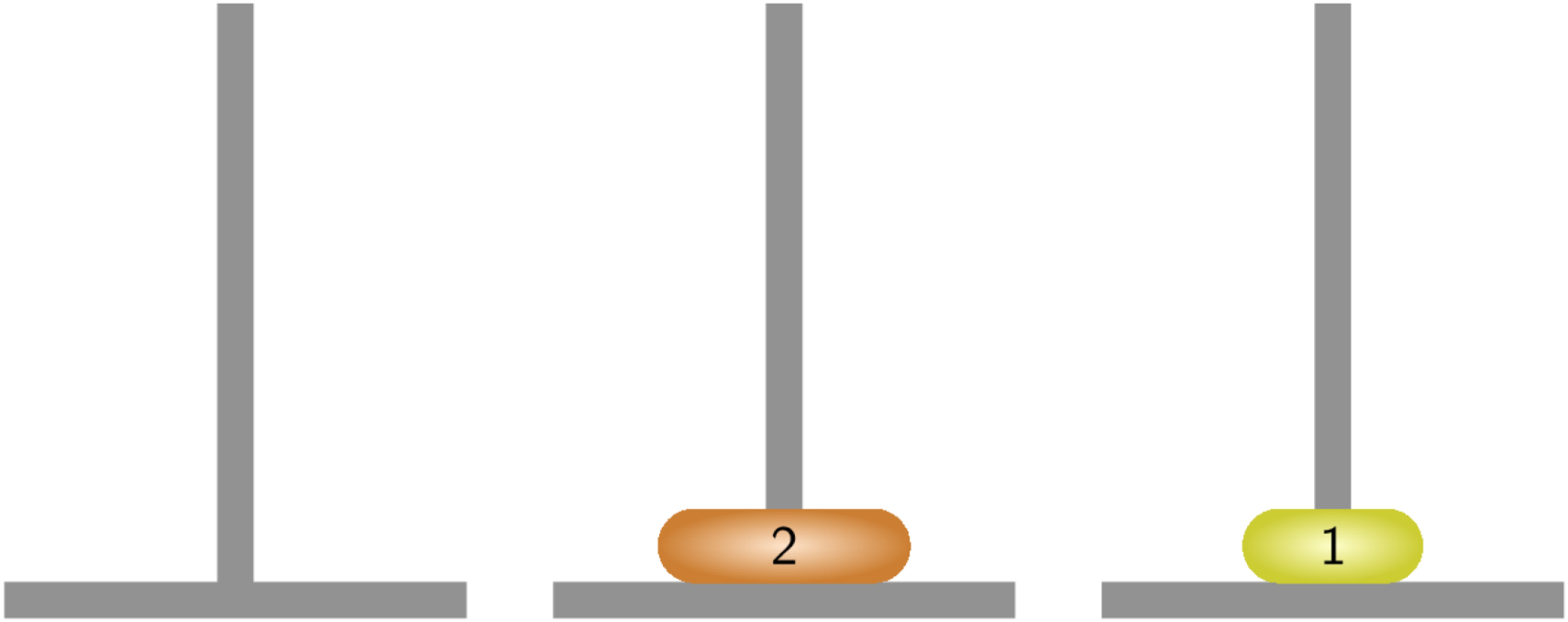

1

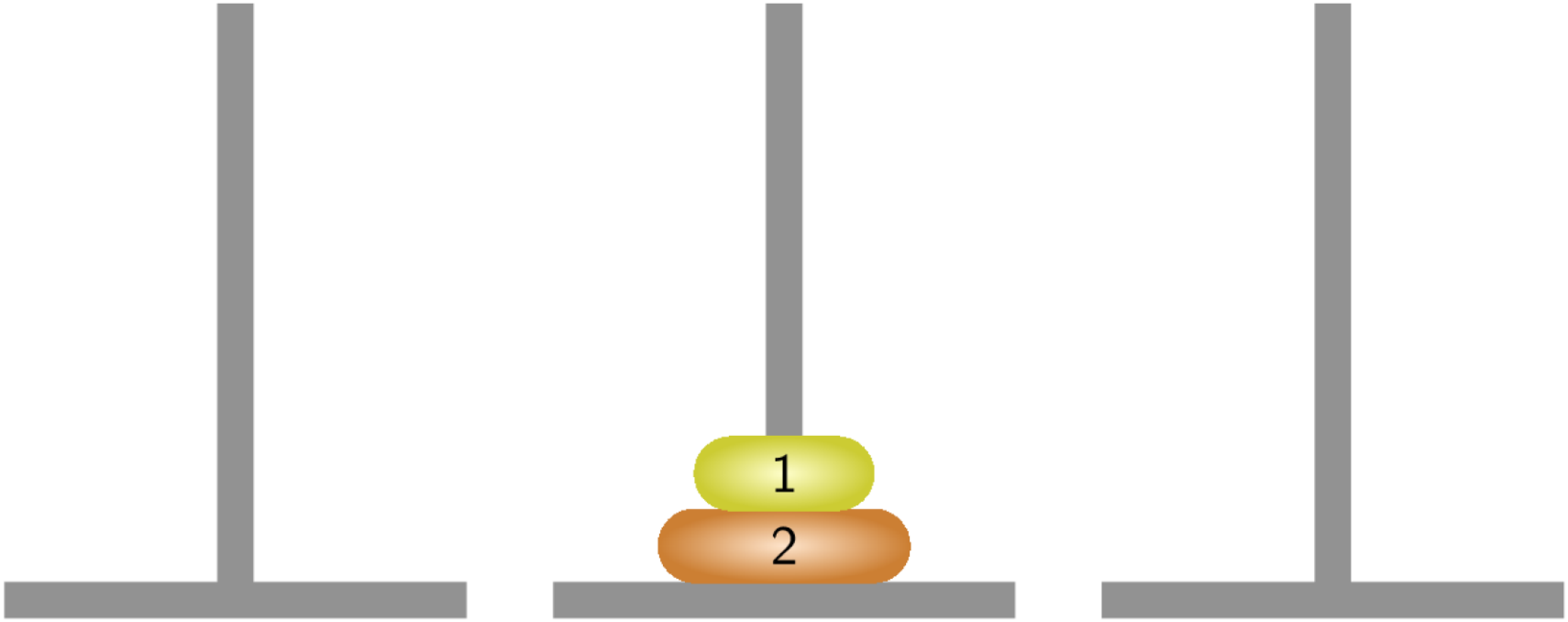

1

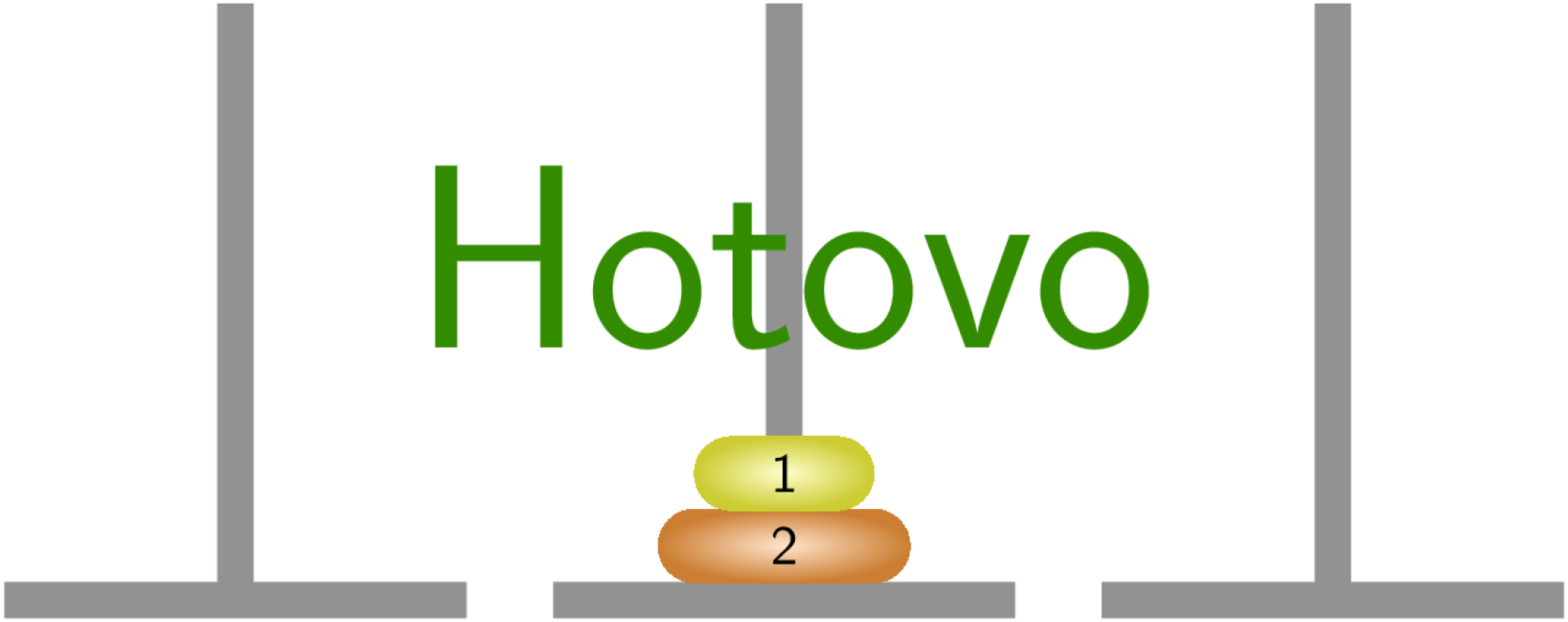

Hotovo
1

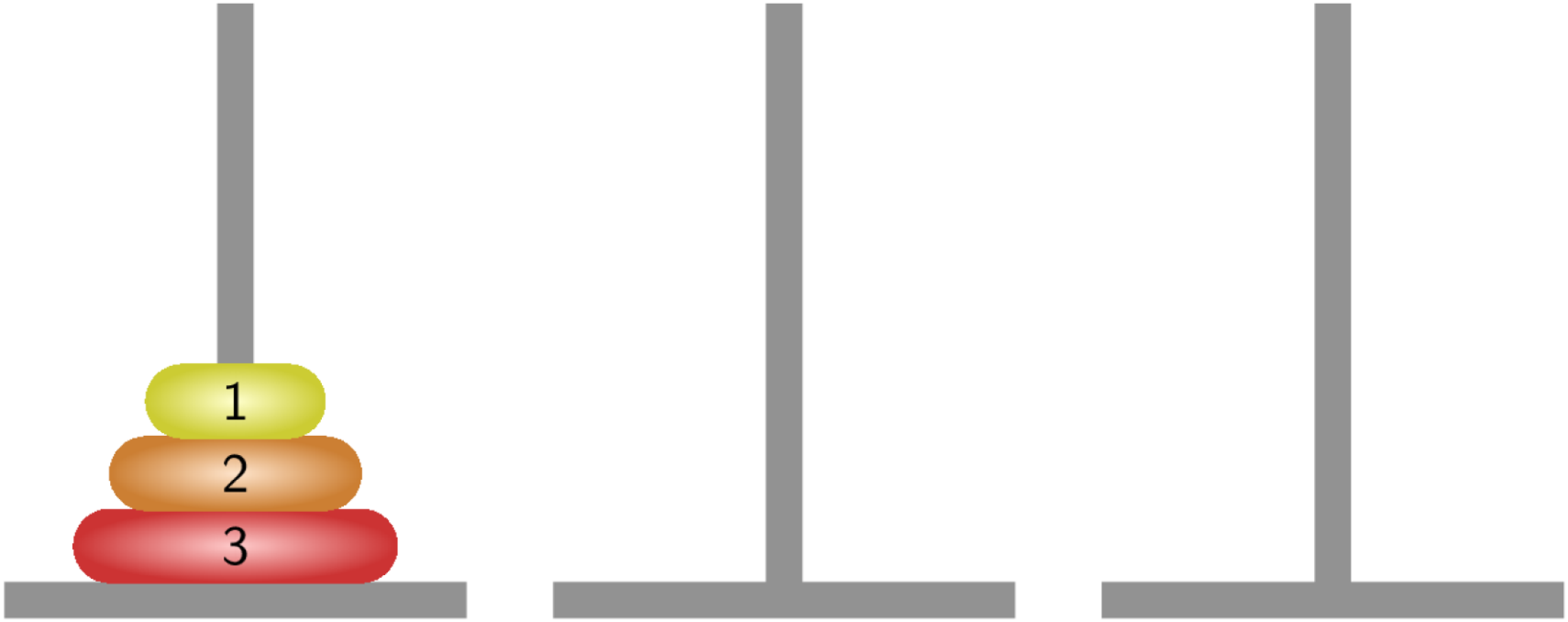

1
2

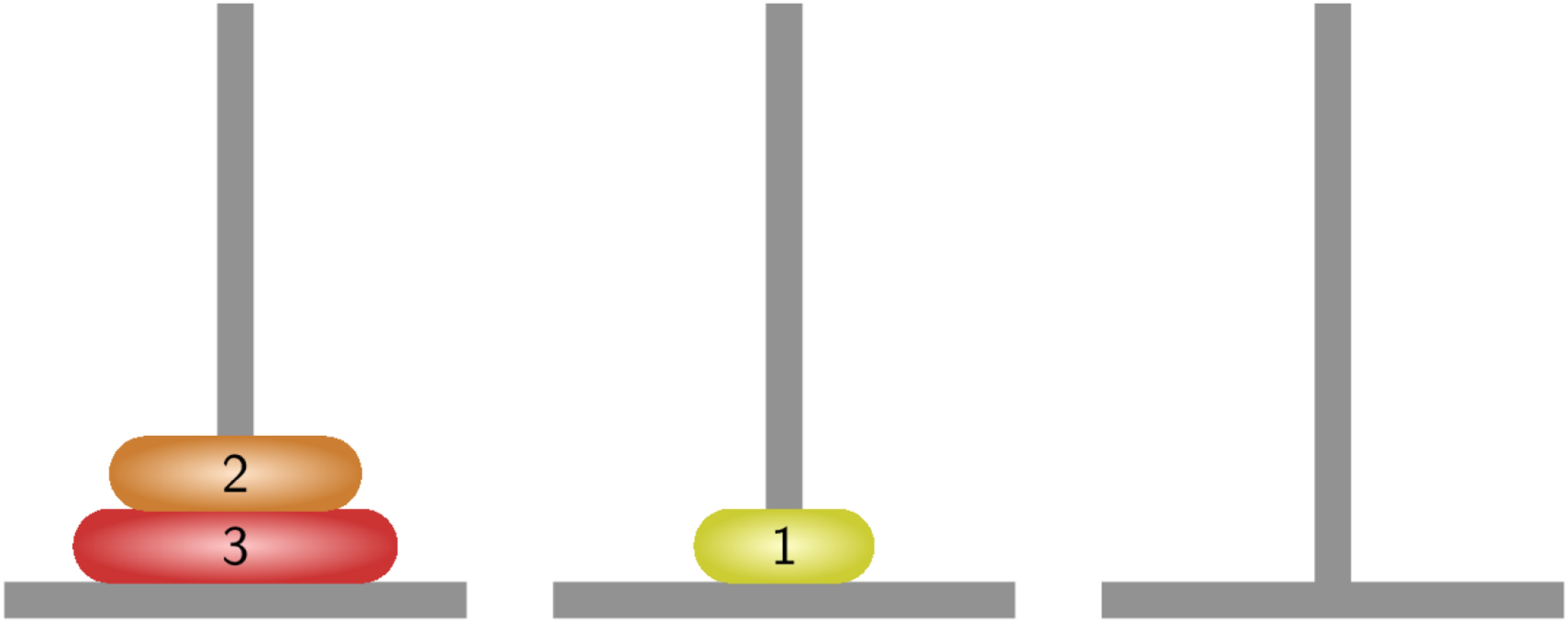

2
1

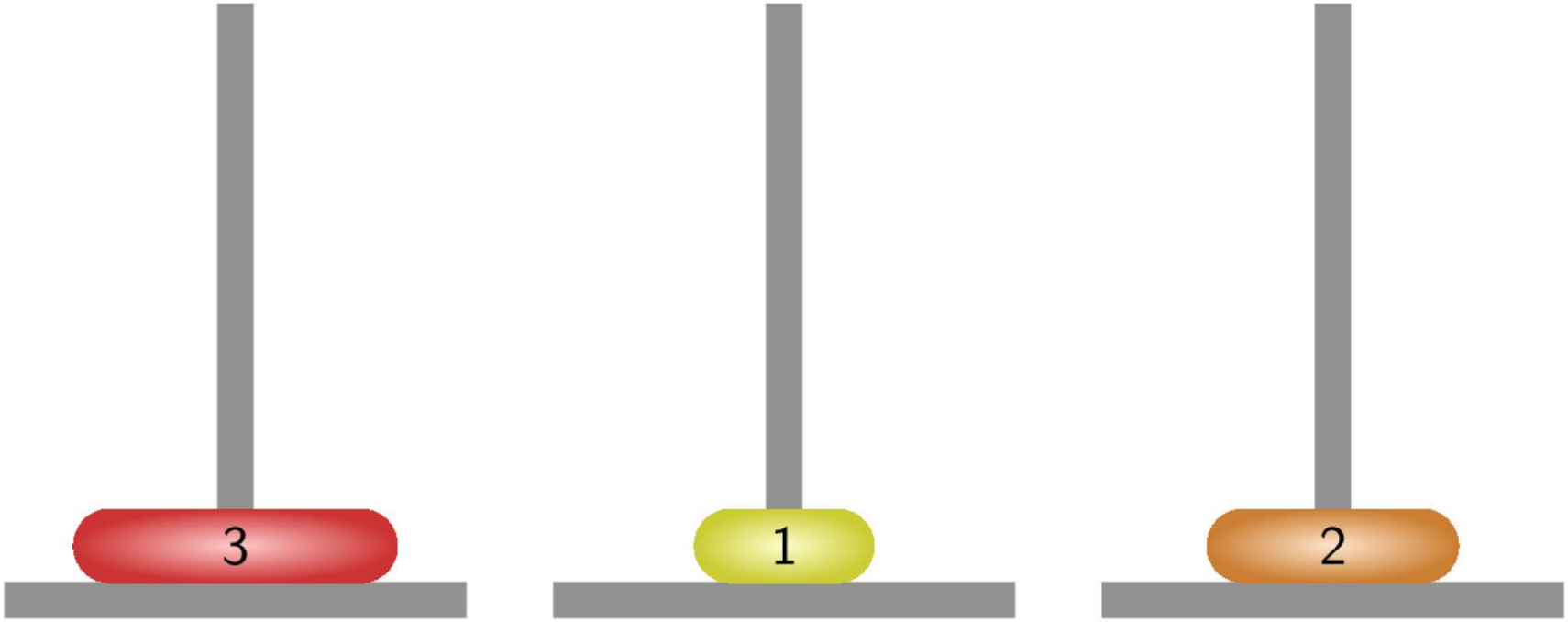

2
1

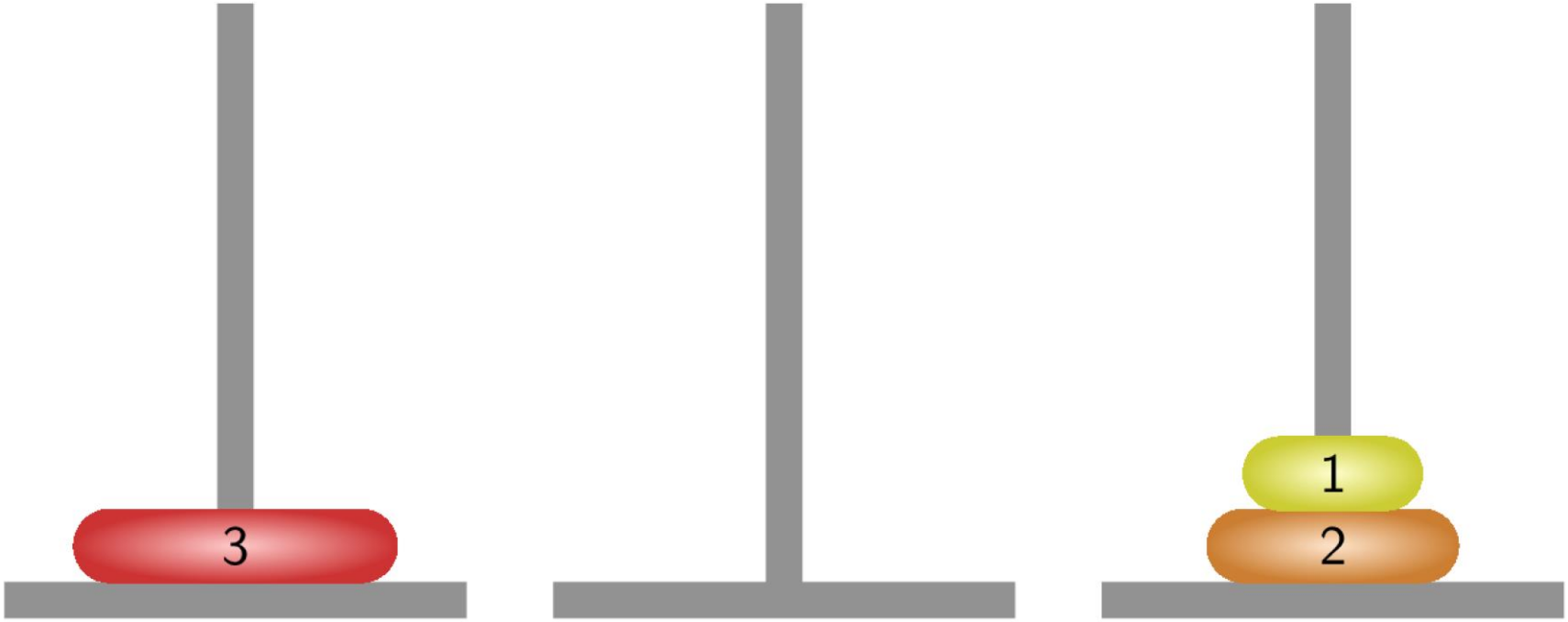

1
2

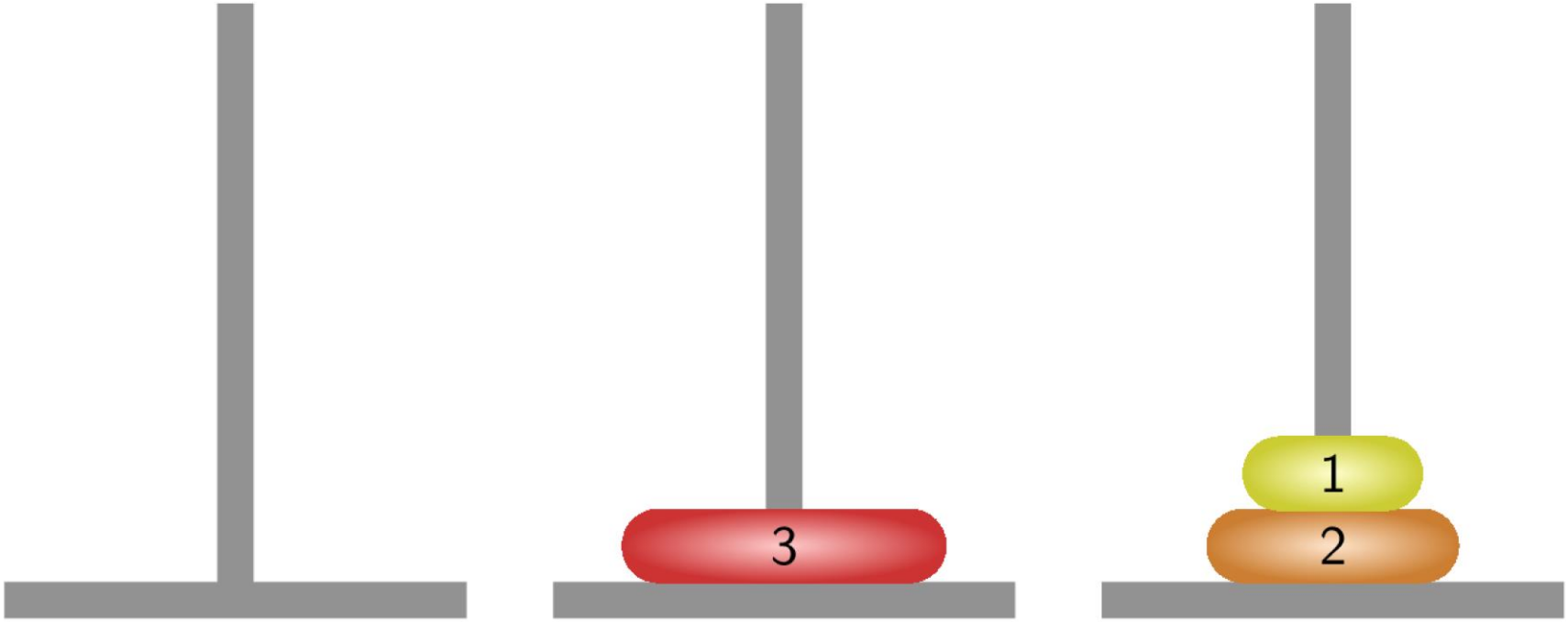

Hotovo
1
2

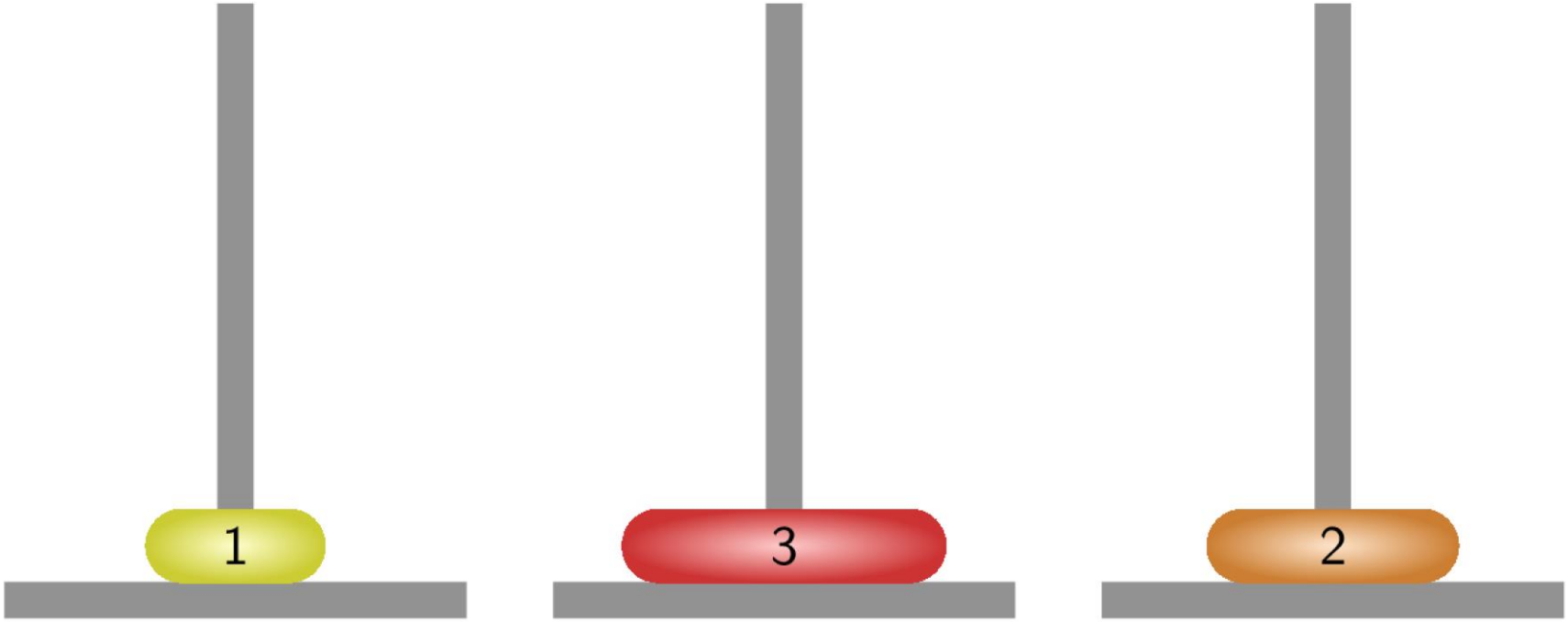

1
2
3

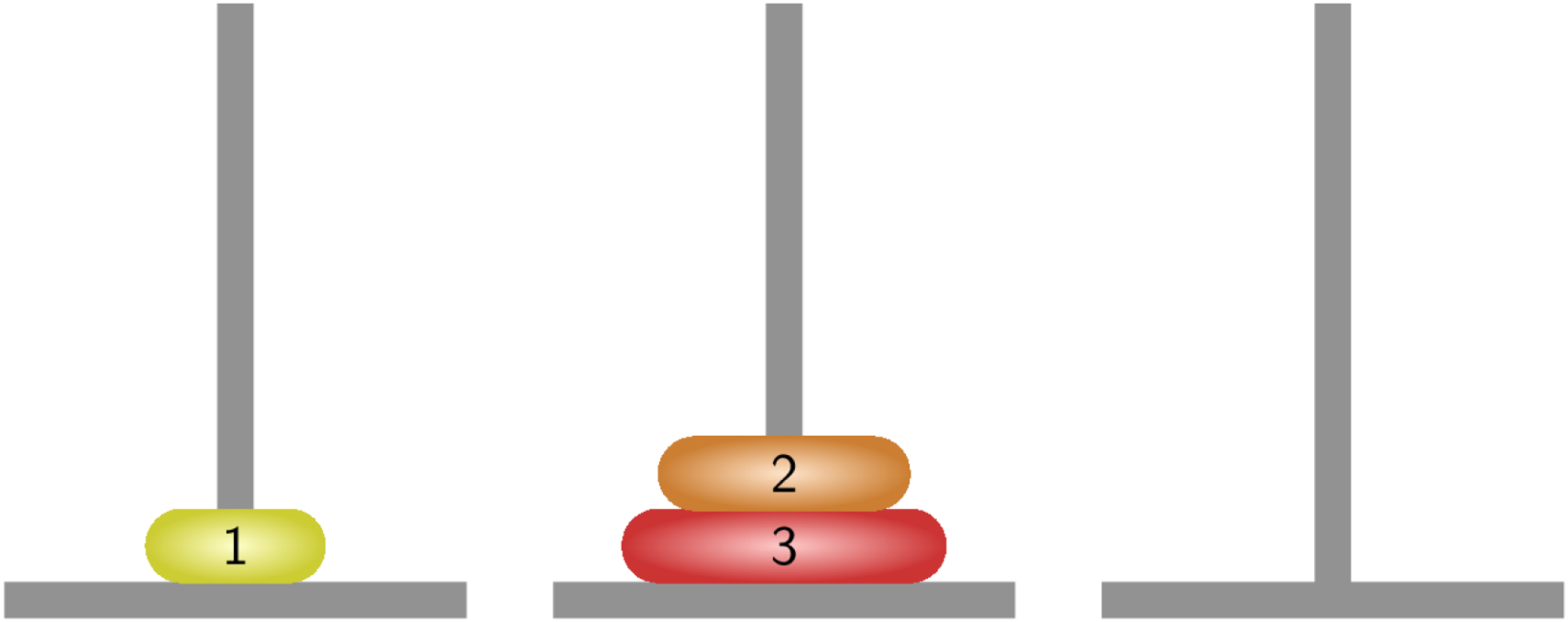

2
3
1

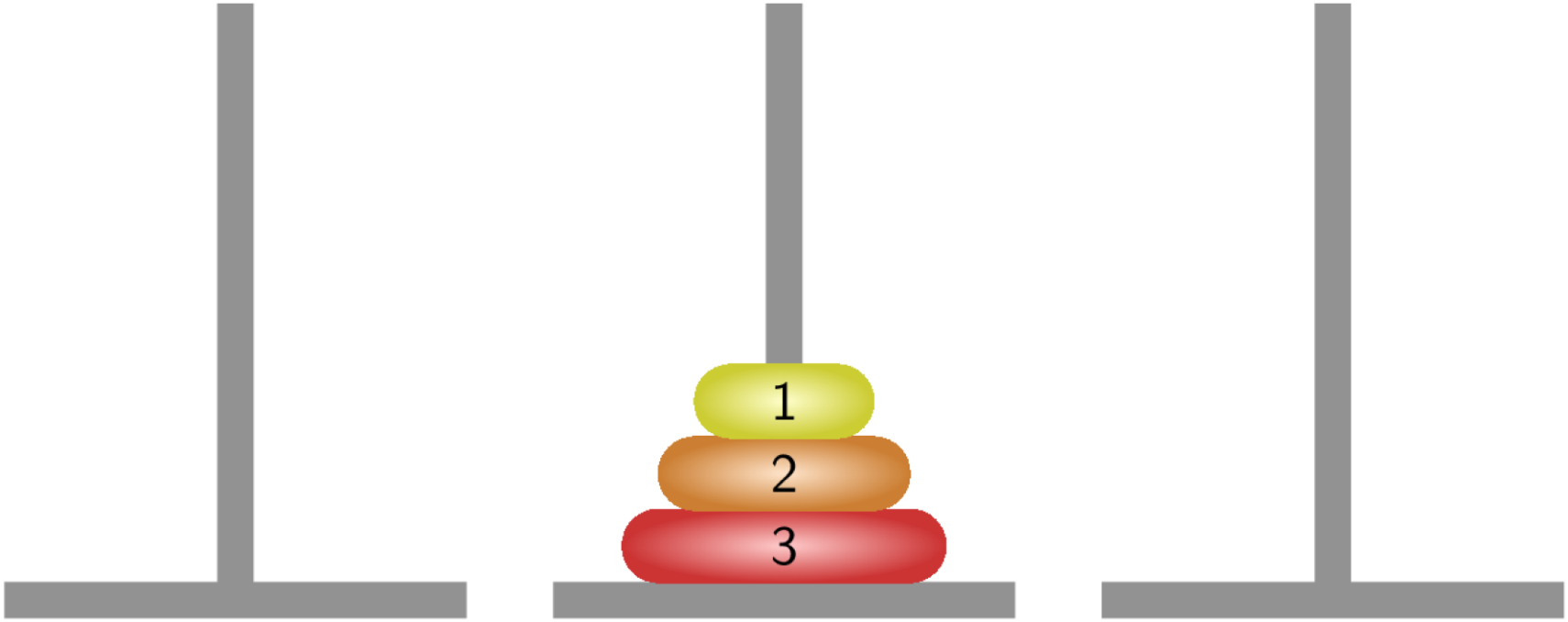

3
1
2

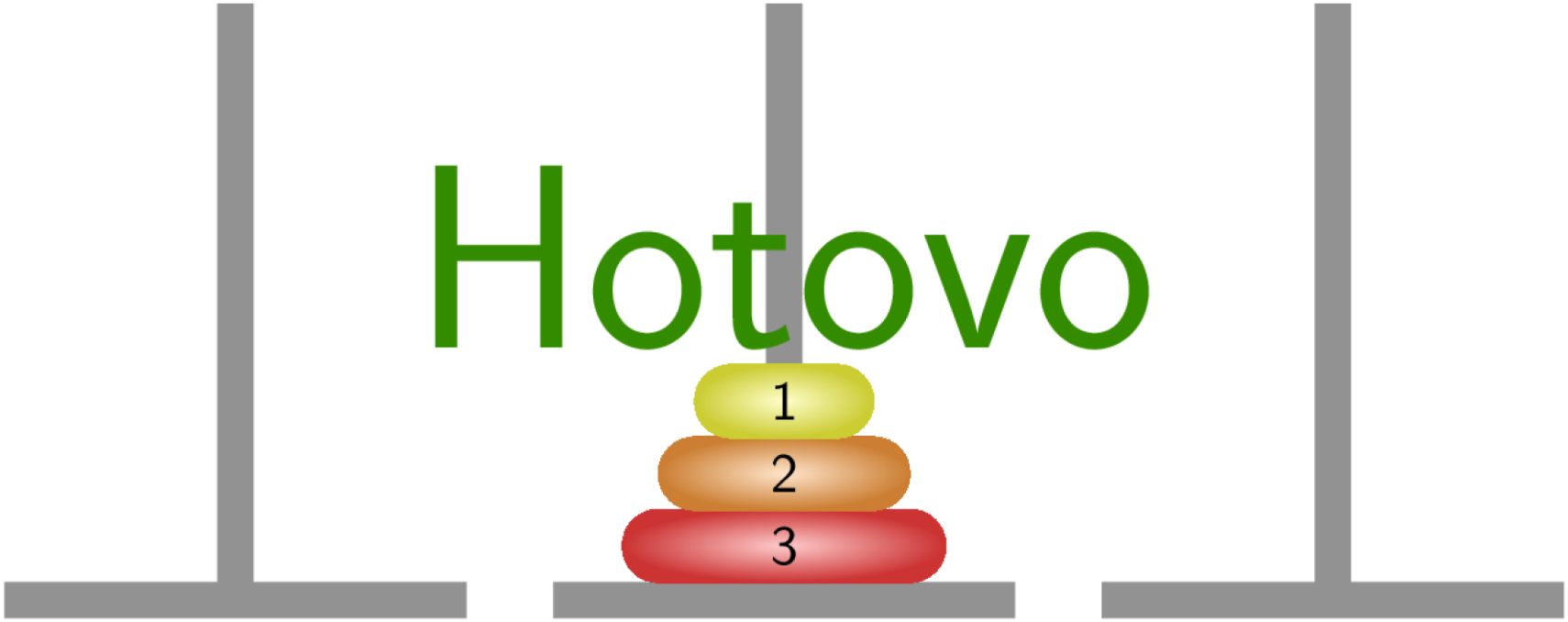

3
1
2

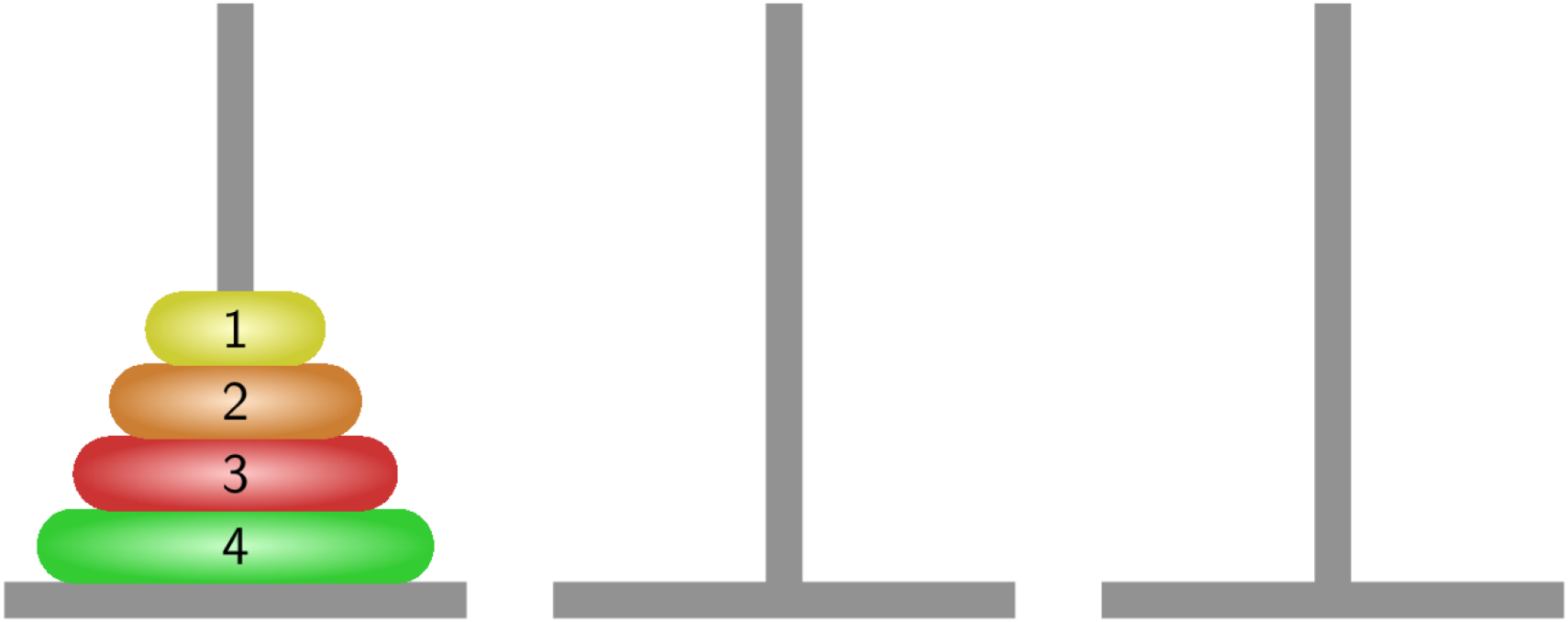

3
1
2

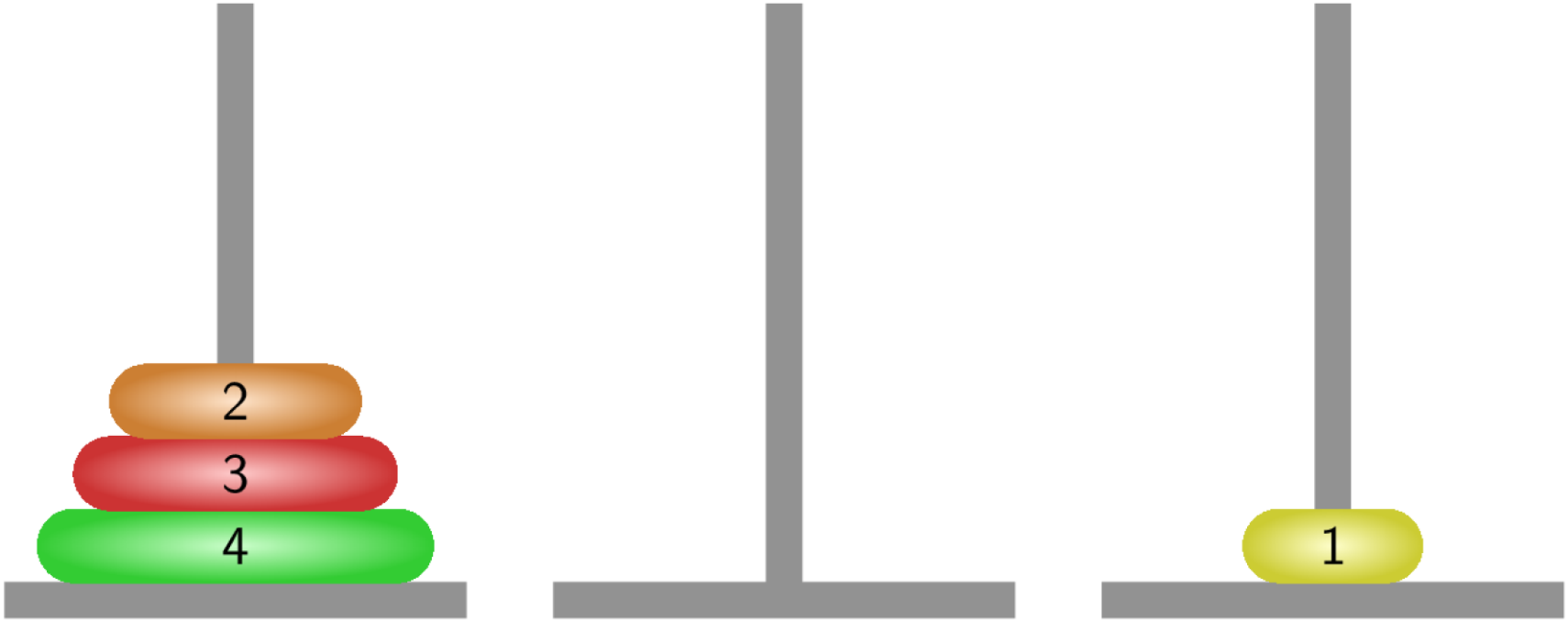

1
3
2

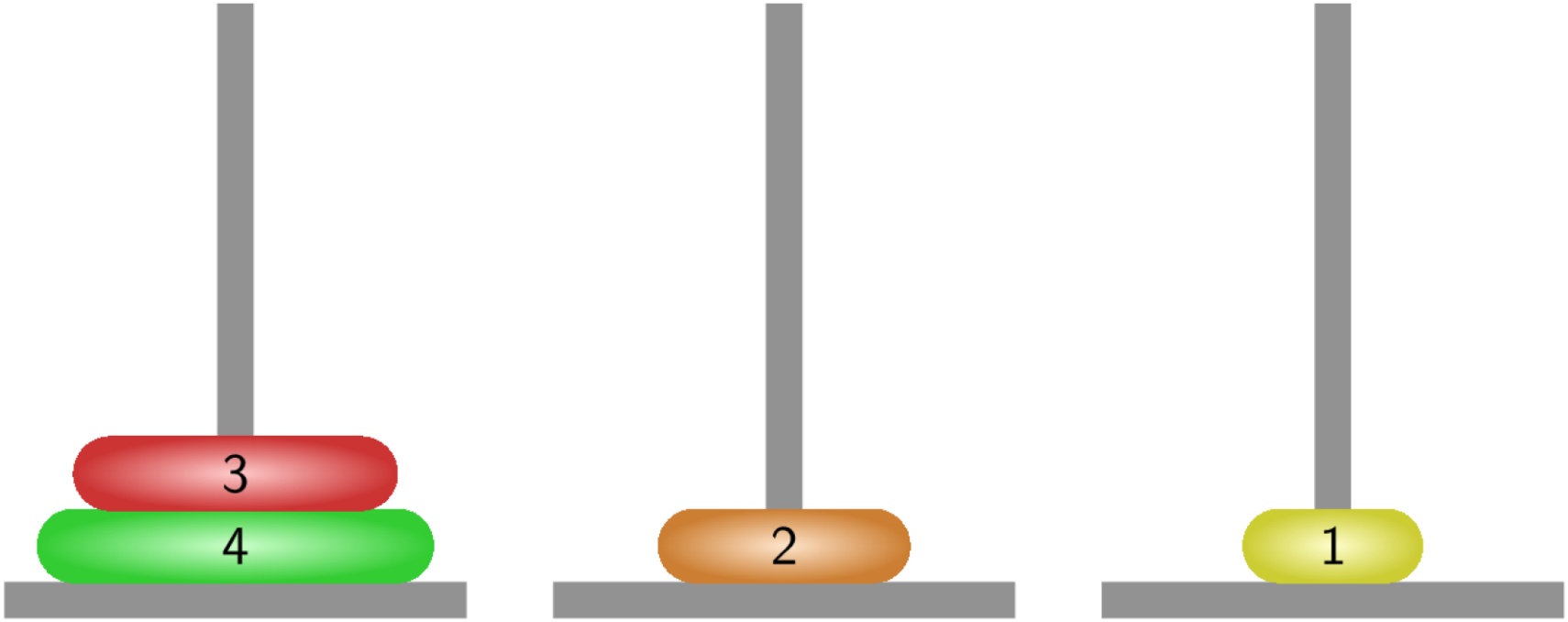

1
2
3

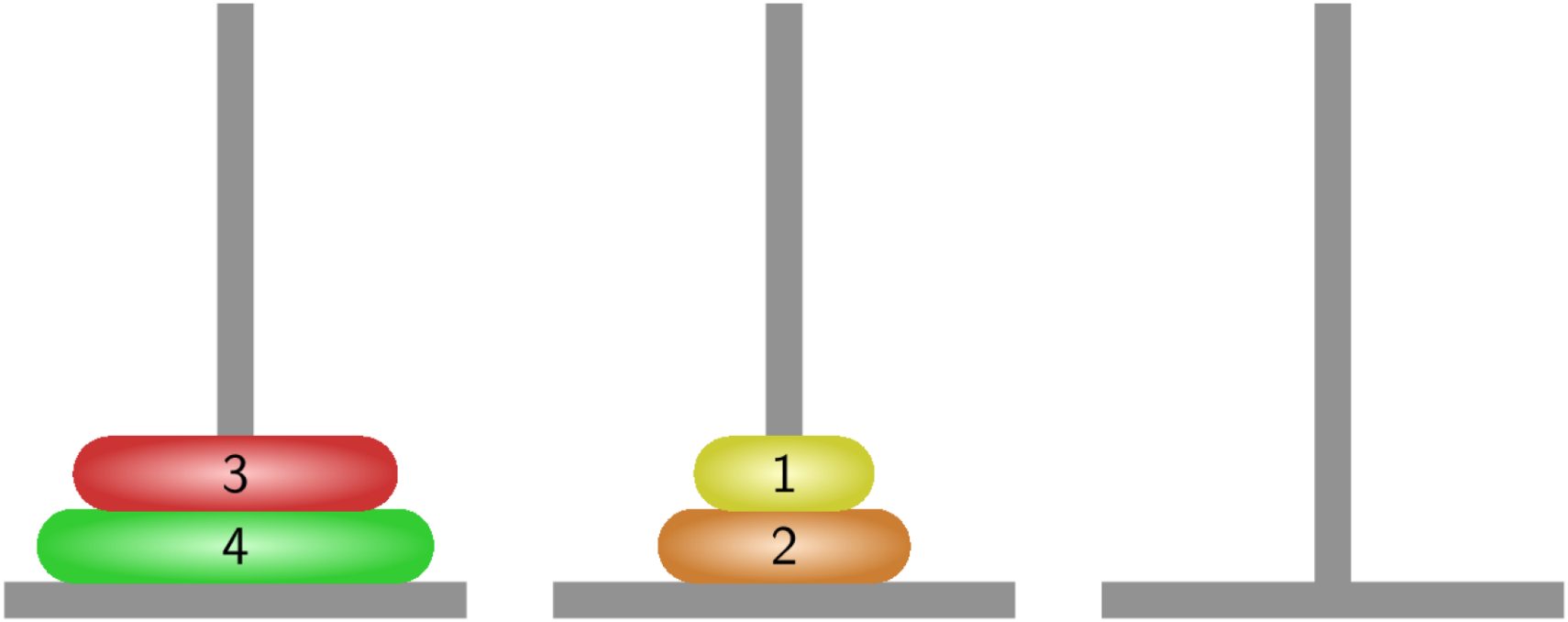

1
2
3

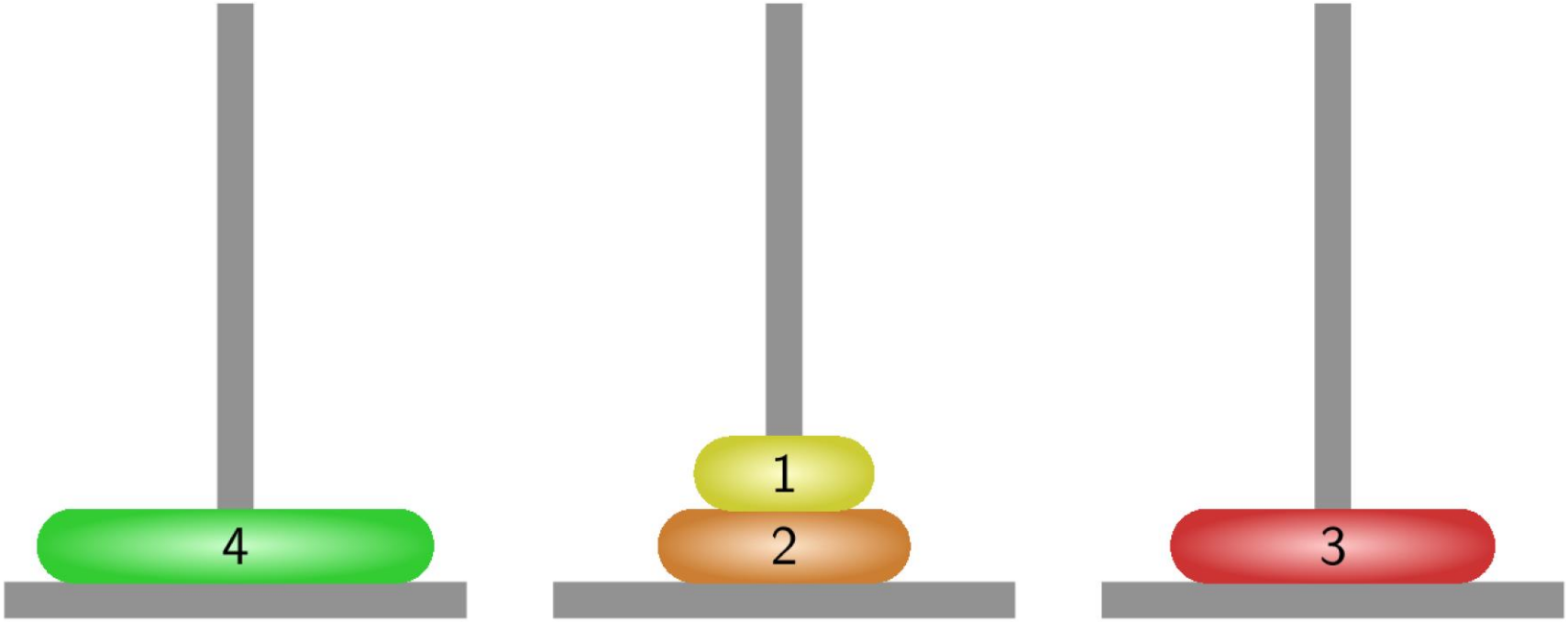

Hotovo
1
2
3

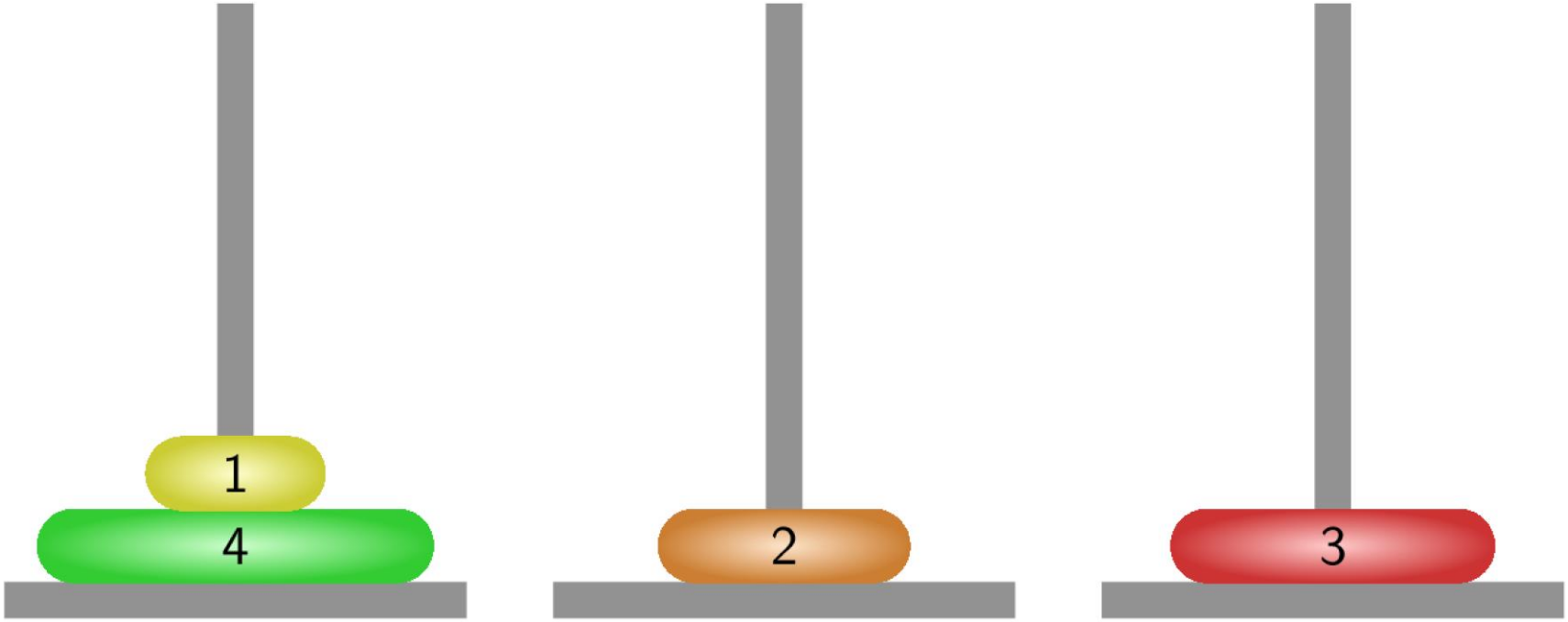

1
2
3
4

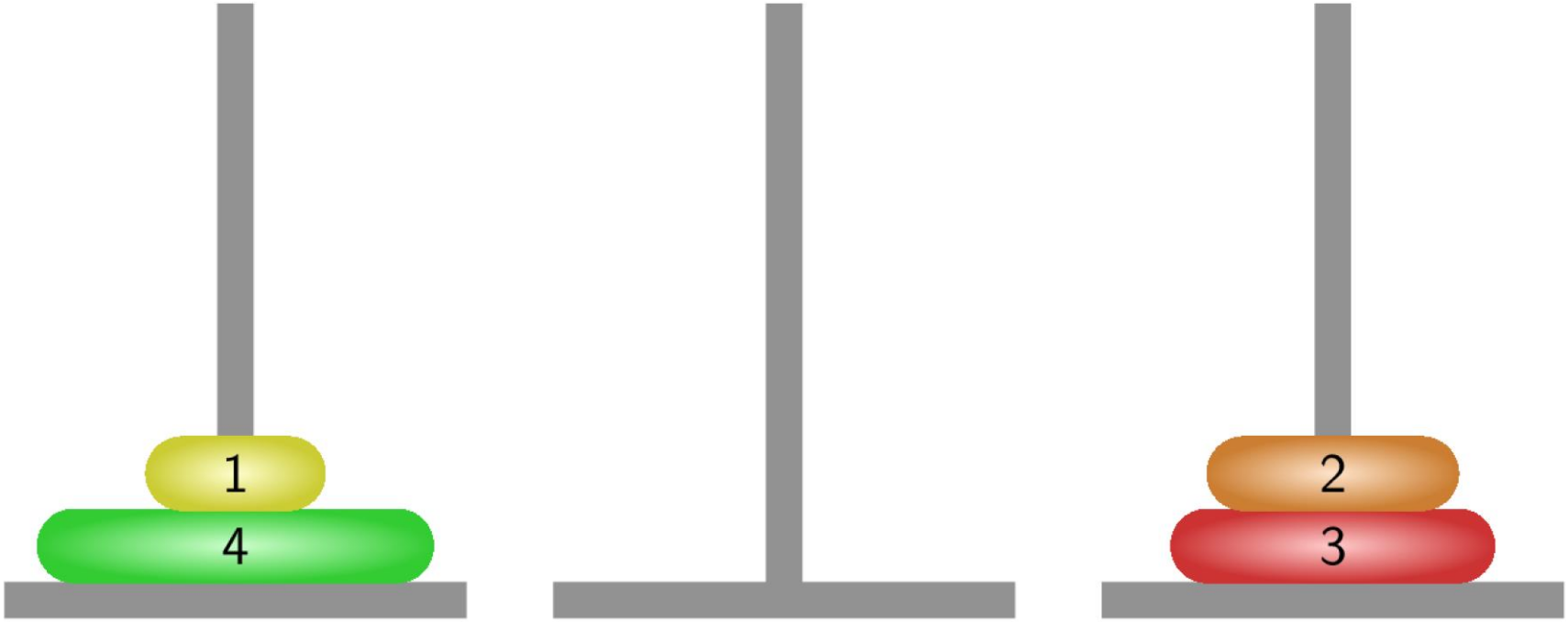

2
3
4
1

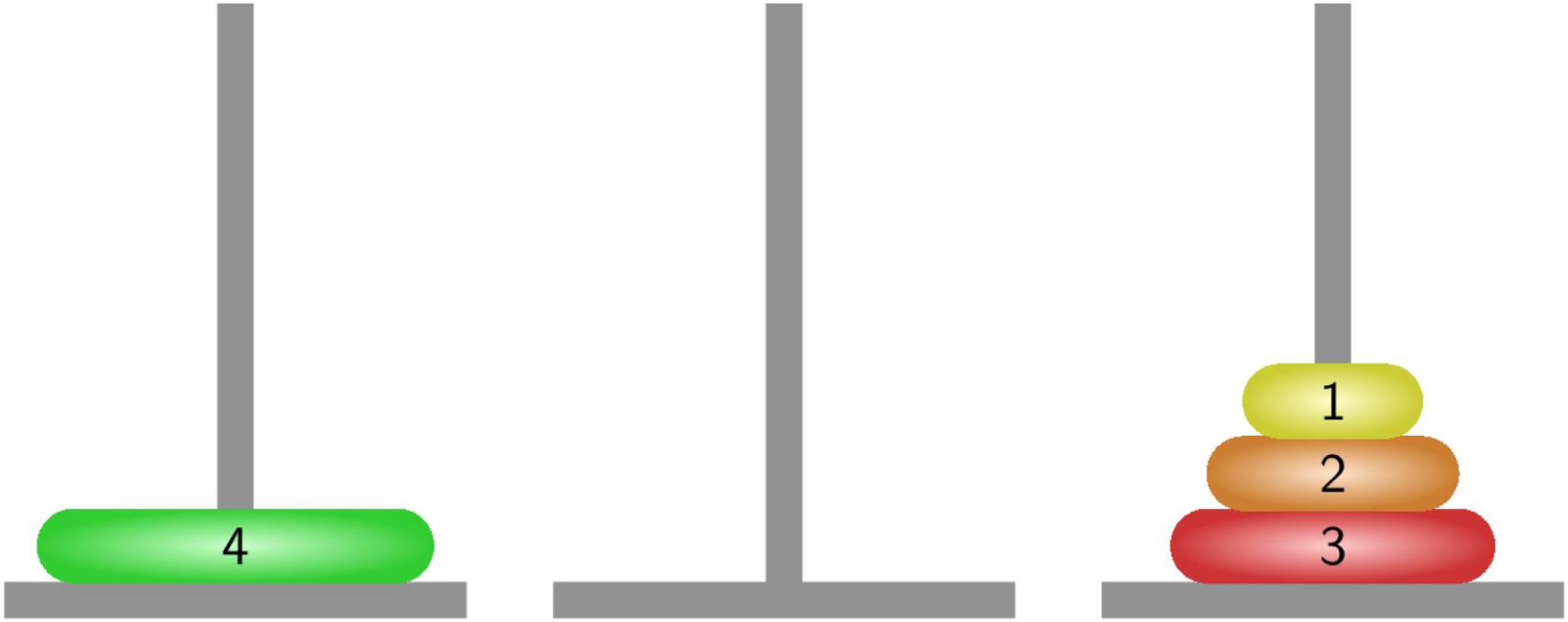

3
4
2
1

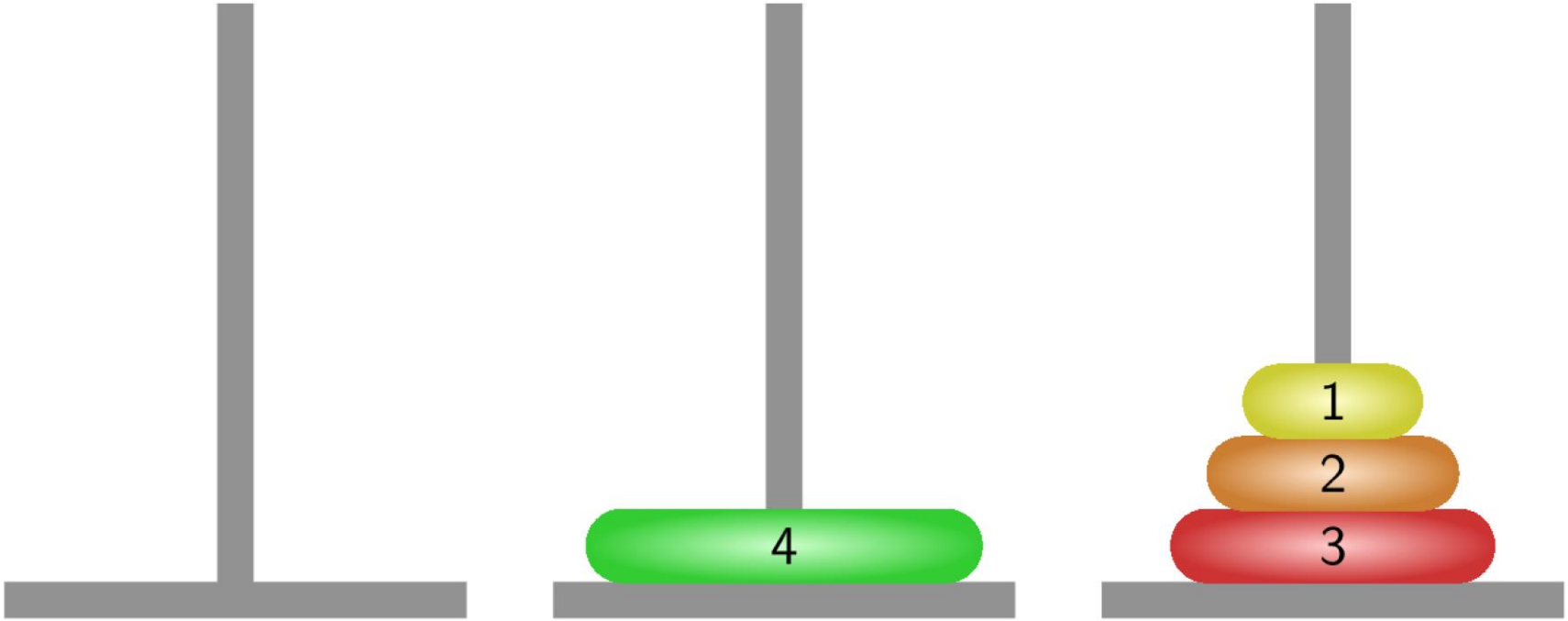

3
4
1
2

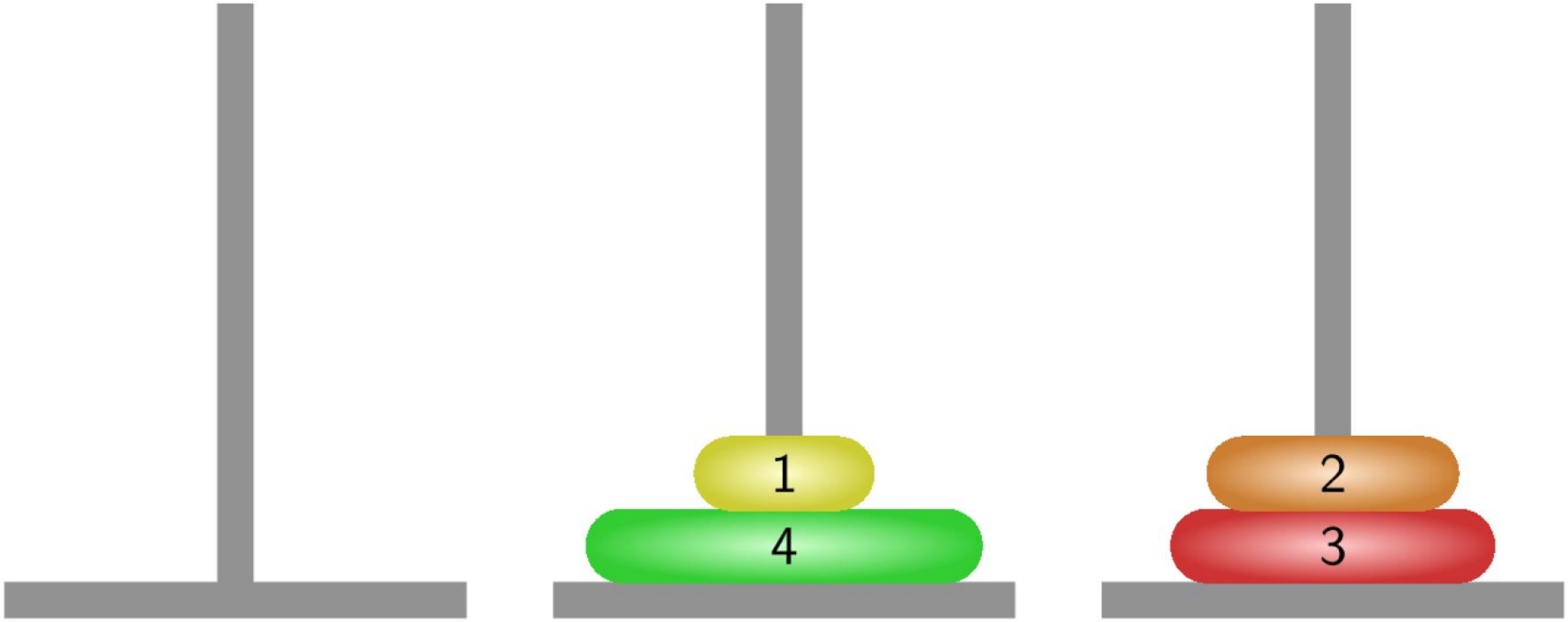

4
1
2
3

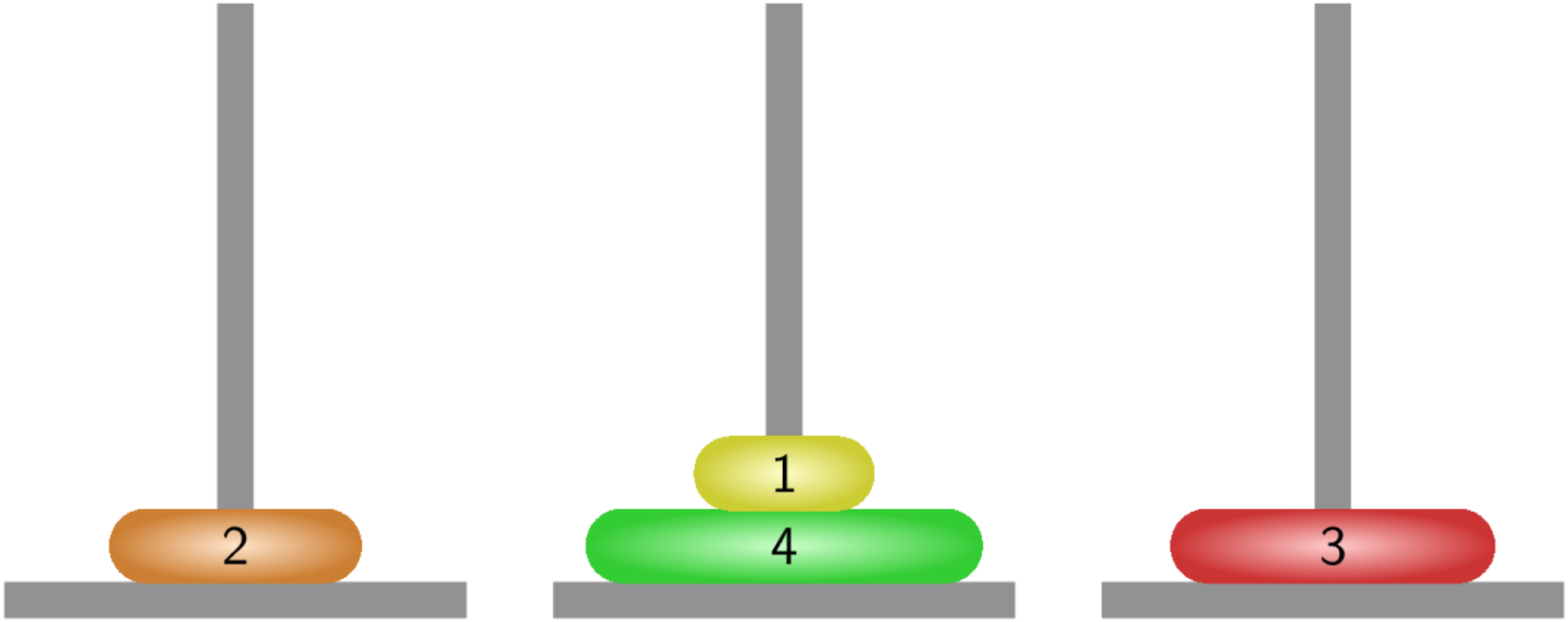

1
4
2
3

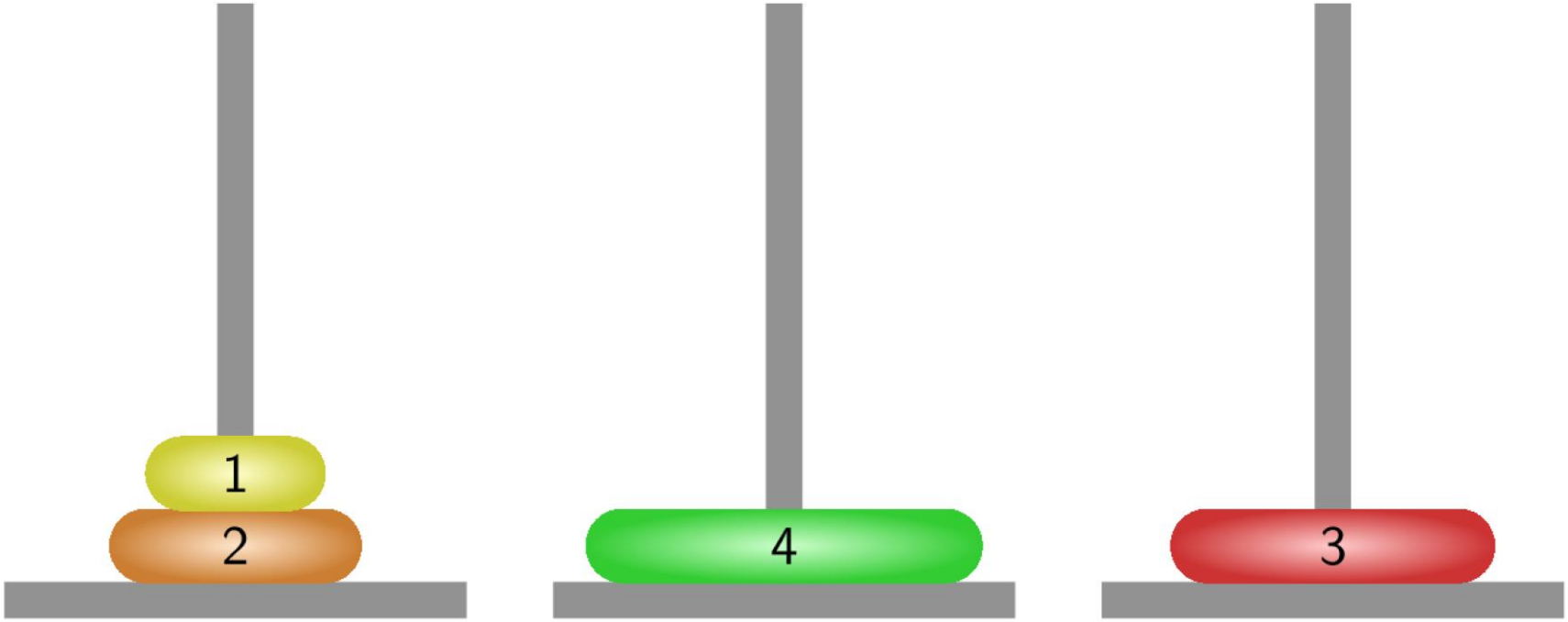

1
4
2
3

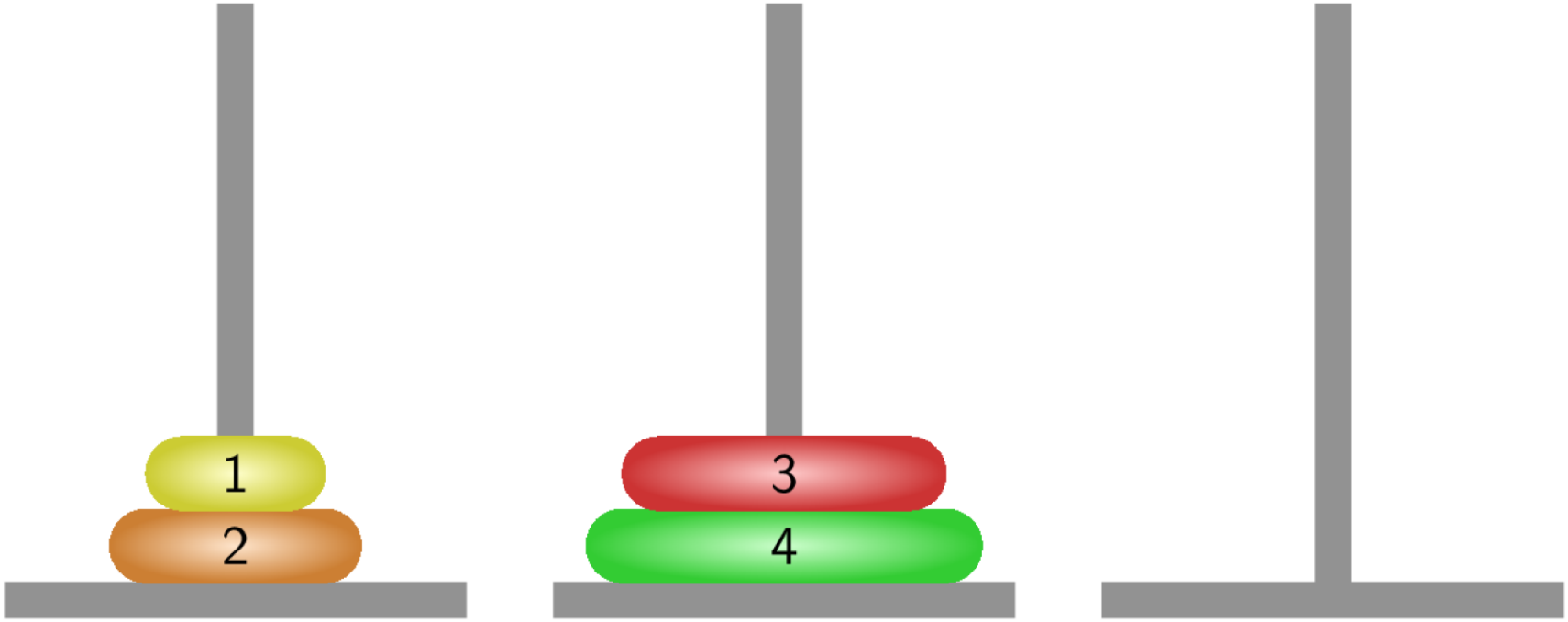

4
1
2
3

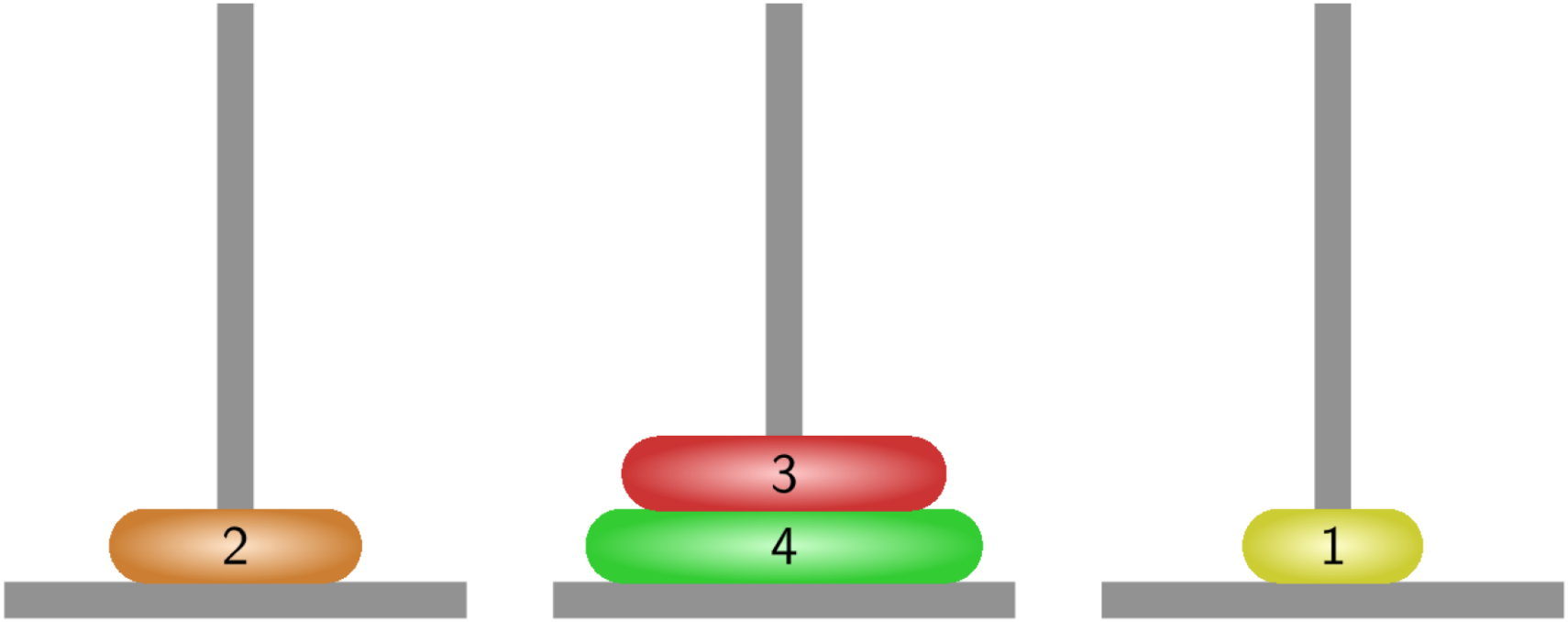

4
1
2
3

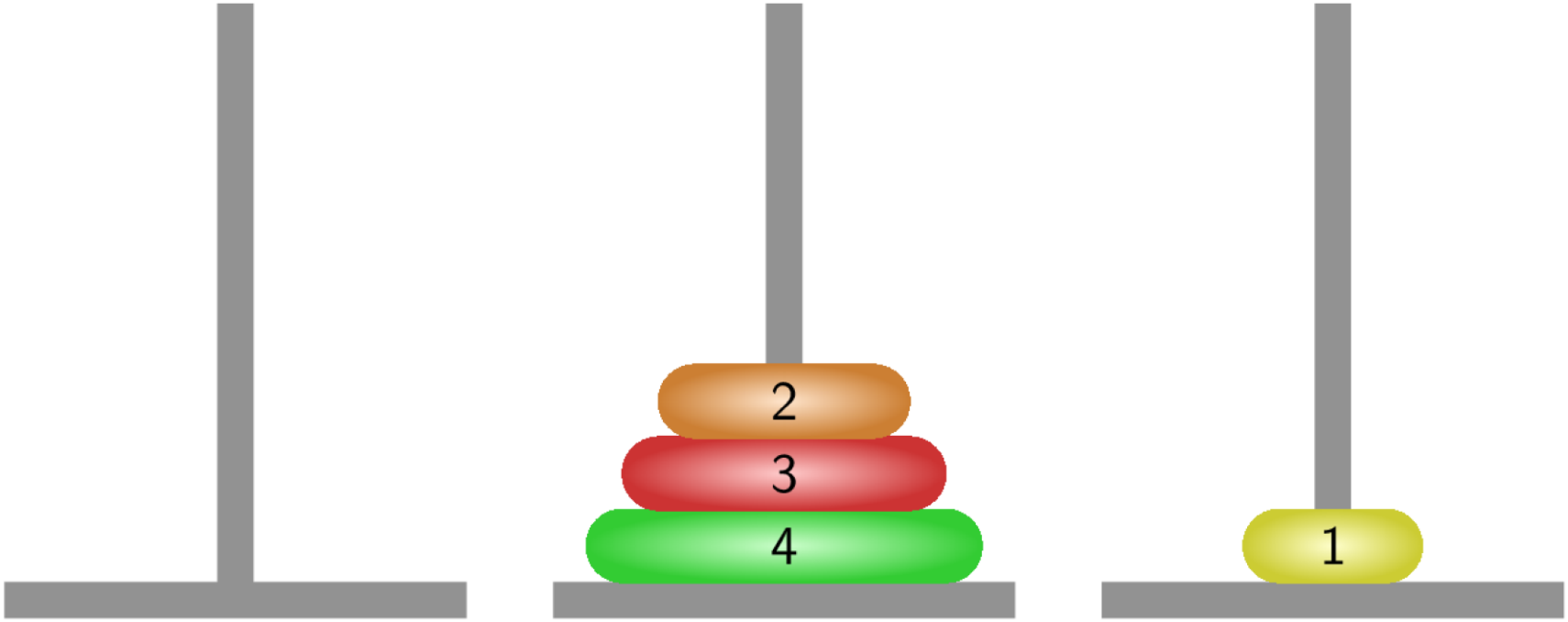

1
4
2
3

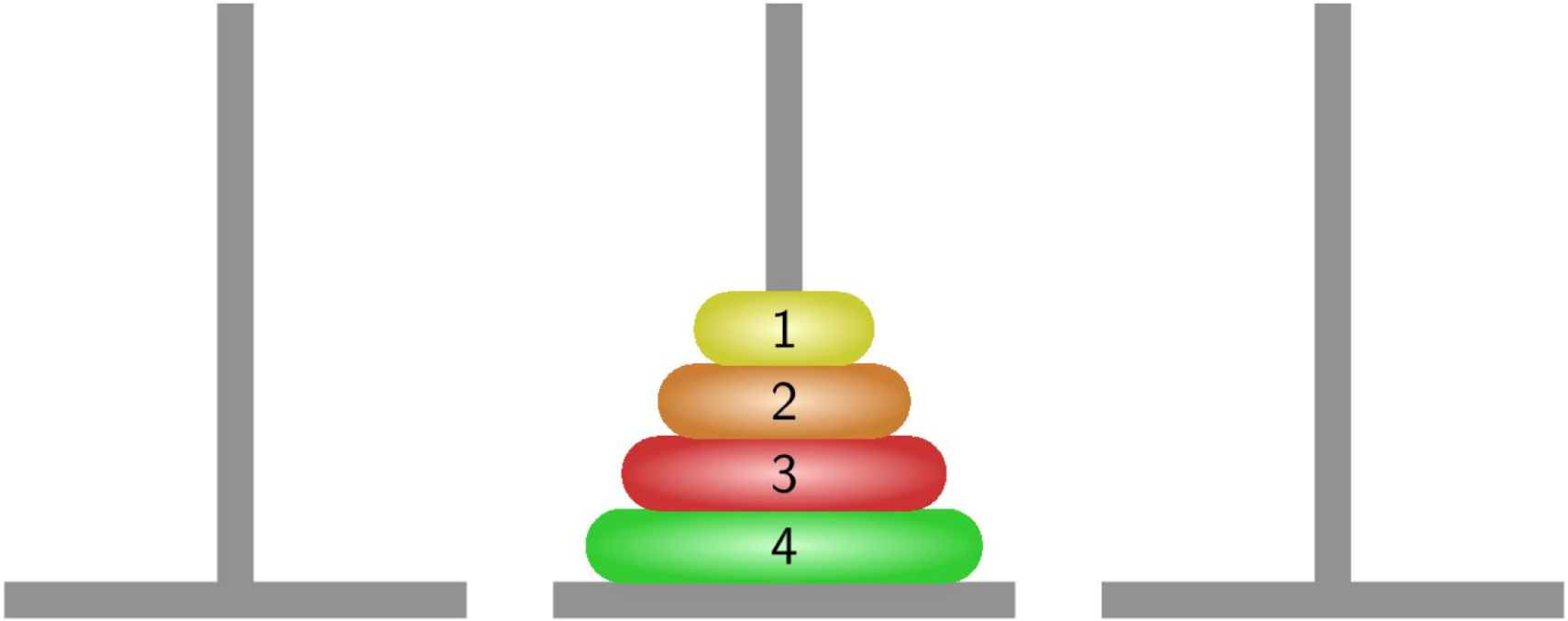

2
1
4
3

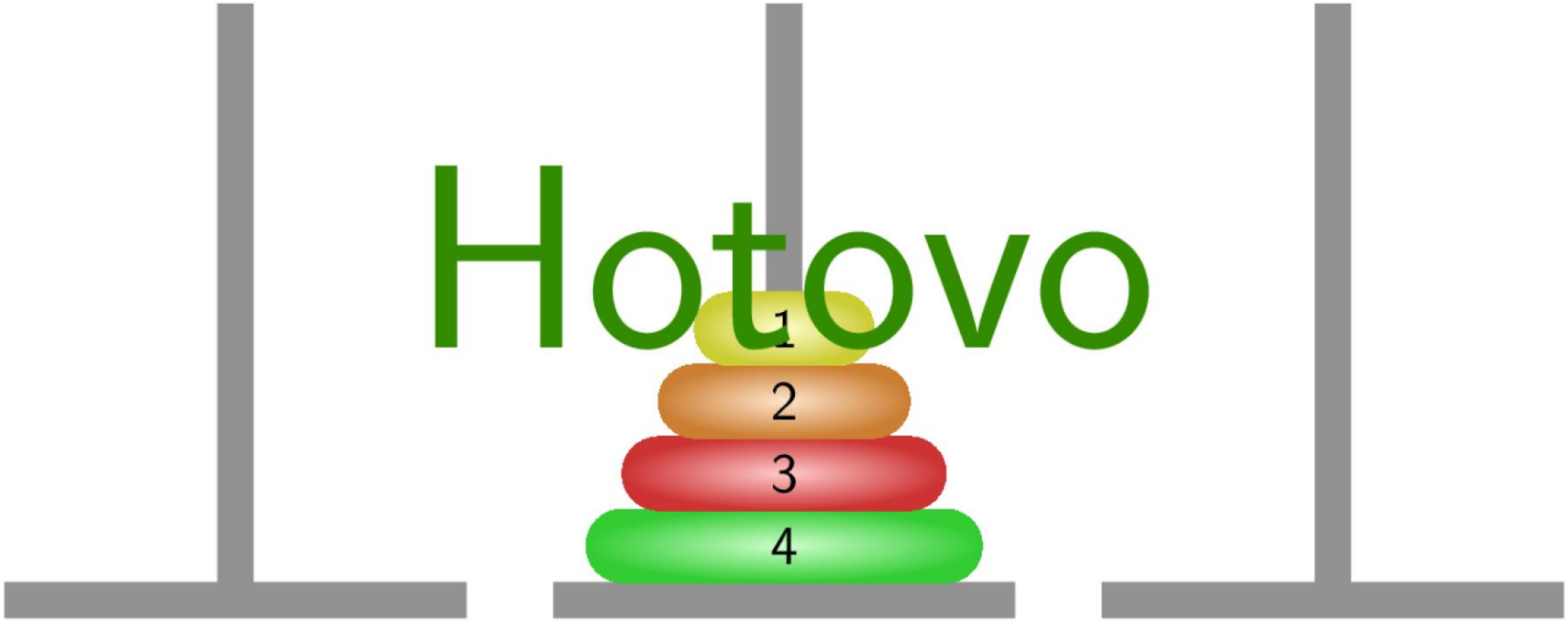

1
2
4
3

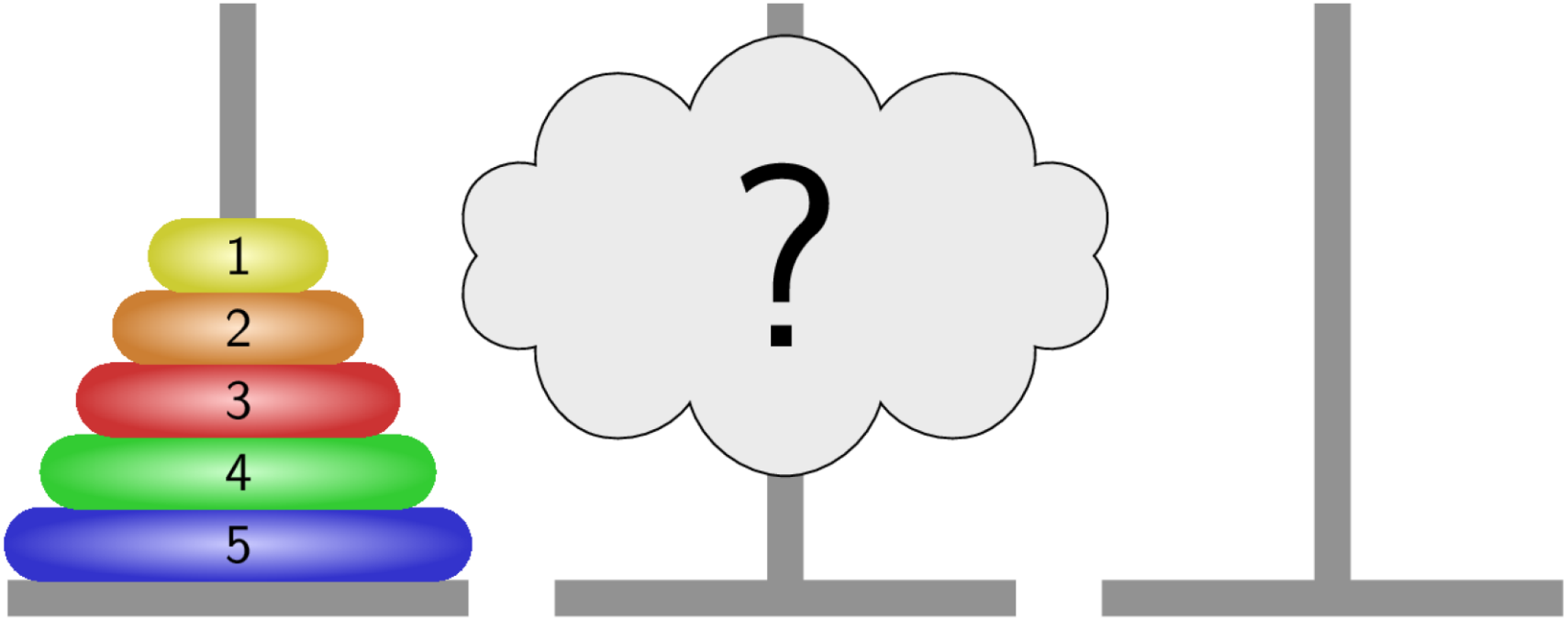

1
2
3
4


2
3
4
1


2
3
4
1


1
2
3
4


Hotovo
1
2
3
4


1
2
3
4
5
?

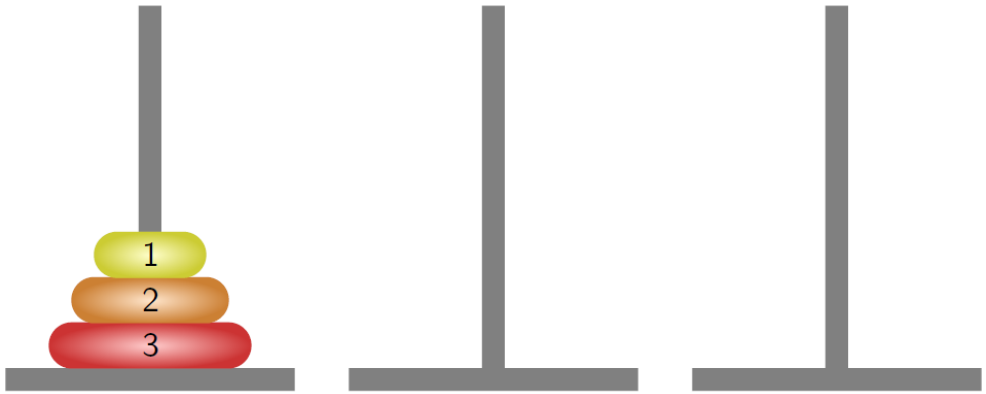
Příklad – hanojské věže

- Řešení
  - Zavedeme abstraktní příkaz moveTower(n, 1, 2, 3) realizující přesun *n* disků z jehly 1 na jehlu 2 s použitím jehly 3.
  - Pro *n* > 0 můžeme příkaz rozložit na tři jednodušší příkazy
    1. moveTower(n-1, 1, 3, 2)

       Přesun n-1 disků z jehly 1 na jehlu 3
    2. „přenes disk z jehly 1 na jehlu 2“

       Přesun největšího disku na cílovou pozici (abstraktní)
    3. moveTower(n-1, 3, 2, 1)

       Přesun n-1 disků na cílovou pozici

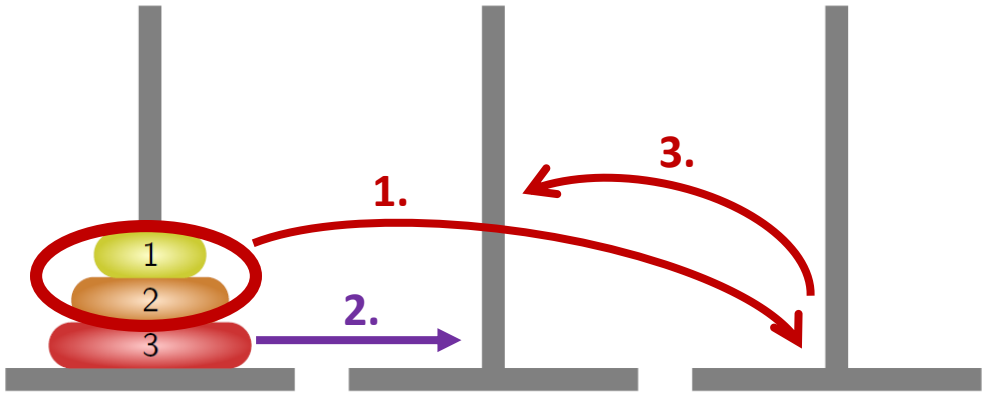


Příklad – hanojské věže

- Řešení

```
def moveTower(n, tFrom, tTo, tmp):
    if n > 0:
        1. moveTower(n-1, tFrom, tmp, tTo) #move to tmp
        2. print("Moving disk from",tFrom,"to",tTo)
        3. moveTower(n-1, tmp, tTo, tFrom) #move from tmp

discs = 4
moveTower( discs, 1, 2, 3)
```



Příklad – hanojské věže

Příklad výpisu

- 1 disk

  Moving disk from 1 to 2

- 2 disky

  Moving disk from 1 to 3
  Moving disk from 1 to 2
  Moving disk from 3 to 2

- 3 disky

  Moving disk from 1 to 2
  Moving disk from 1 to 3
  Moving disk from 2 to 3
  Moving disk from 1 to 2
  Moving disk from 3 to 1
  Moving disk from 3 to 2
  Moving disk from 1 to 2

- 4 disky

  Moving disk from 2 to 1
  Moving disk from 2 to 3
  Moving disk from 1 to 3
  Moving disk from 1 to 2
  Moving disk from 3 to 2
  Moving disk from 3 to 1
  Moving disk from 2 to 1
  Moving disk from 3 to 2
  Moving disk from 1 to 3
  Moving disk from 1 to 2
  Moving disk from 3 to 2

Rekurzivní algoritmy

- Rekurzivní funkce jsou přímou realizací rekurzivních algoritmů
- Rekurzivní algoritmus předepisuje výpočet *„shora dolů“* v závislosti na velikosti vstupních dat
  - Pro nejmenší (nejjednodušší) vstup je výpočet určen přímo
  - Pro obecný vstup je výpočet předepsán s využitím **téhož** algoritmu pro **menší vstup**
- Výhodou rekurzivních funkcí je jednoduchost a přehlednost

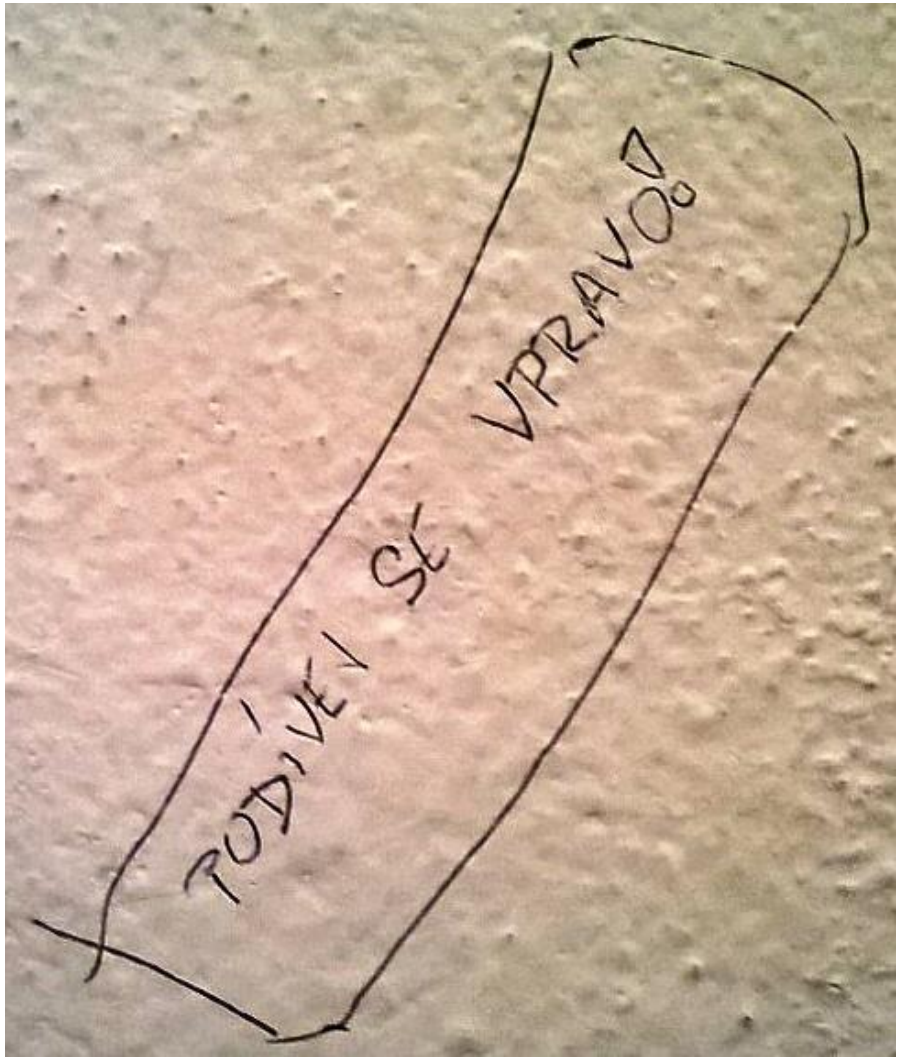
Rekurzivní algoritmy

- Příklad (pseudokód)

```
def recursion(n):
    # do something before recursion
    if n is "trivial":
        return "solution" # or just do nothing
    else:
        sol = recursion(n - 1)
    # do something after recursion
    return n + sol
```

Rekurzivní algoritmy

- Příklad (studentská grafika)



Je toto rekurze?

Rekurzivní algoritmy

- Nevýhodou rekurzivních algoritmů může být časová náročnost způsobená např. zbytečným opakováním výpočtu
- Řadu rekurzivních algoritmů lze nahradit iteračními, které počítají výsledek *„zdola nahoru“*, tj. od menších (jednodušších) vstupních dat k větším (složitějším)
- Pokud algoritmus *„zdola nahoru“* nenajdeme, lze rekurzivitu odstranit pomocí zásobníku

Např. zásobník využijeme pro uložení stavu řešení problému

- Př. – binární řešení hanojských věží

```
def hanoiBinary(n):
    for x in range(1,1<<n):
        print( "move from",(x&x-1)%3,"to",((x|x-1)+1)%3 )
```

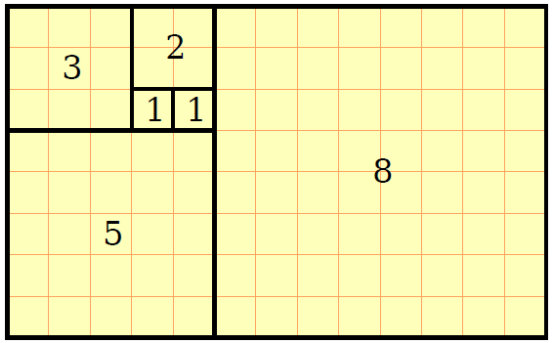
Příklad – Fibonacciho posloupnost

- 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, . . .

  Nebo 0, 1, 1, 2, 3, 5, . . .

- $F_n = F_{n-1} + F_{n-2}$ pro $F_1 = 1, F_2 = 1$

  nebo $F_1 = 0, F_2 = 1$



- Nekonečná posloupnost přirozených čísel, kde každé číslo je součtem dvou předchozích
- Limita poměru dvou následujících čísel Fibonacciho posloupnosti je rovna zlatému řezu
  - Sectio aurea – ideální poměr mezi různými délkami
  - Rozdělení úsečky na dvě části tak, že poměr vetší časti ku menší je stejný jako poměr celé úsečky k vetší části
  - $\varphi = \frac{1+\sqrt{5}}{2} \approx 1{,}618\,033\,988\,749\,848\ldots$

Příklad – Fibonacciho posloupnost

- Historie
  - Indičtí matematici (450 nebo 200BC)
  - Leonardo Pisano (1175 – 1250) – popis růstu populace králíků

    Italský matematik známý také jako Fibonacci
    - F_n – velikost populace po n měsících za předpokladu, že
      - První měsíc se narodí jediný pár
      - Narozené páry jsou produktivní od 2. měsíce svého života
      - Každý měsíc zplodí každý produktivní pár jeden další pár
      - Králíci nikdy neumírají, nejsou nemocní atp.
  - Henry E. Dudeney (1857 – 1930) – popis populace krav
    - „Jestliže každá kráva vyprodukuje své první tele (jalovici) za rok a poté každý rok jednu další jalovici, kolik budete mít krav za 12 let, jestliže žádná nezemře a na počátku budete mít jednu krávu?“

      Po 12 let je k dispozici jeden či více býků

Příklad – Fibonacciho posloupnost

- Řešení
- Platí:
  - $f_0 = 1$
  - $f_1 = 1$
  - $f_n = f_{n-1} + f_{n-2}$, pro $n > 1$

```
def fibonacci(n):
    if n<2: return 1
    return fibonacci(n-1)+fibonacci(n-2)
```

Zápis je elegantní, ale je takový výpočet efektivní?

Příklad – Fibonacciho posloupnost

- Počet operací při výpočtu Fibonacciho čísla

```
def fibonacciR(n):
    global counter
    counter +=1
    if n<2: return 1
    return fibonacciR(n-1)+fibonacciR(n-2)

def fibonacciI(n):
    global counter
    fib = fibM1 = fibM2 = 1
    for i in range(2,n+1):
        fibM2 = fibM1
        fibM1 = fib
        fib = fibM1 + fibM2
        counter +=3
    return fib
counter = 0
print(fibonacciR(30), counter)
counter = 0
print(fibonacciI(30), counter)
```

1346269 **2692537**

1346269 **87**

Příklad – Fibonacciho posloupnost

- Rekurzivní výpočet
  - Počet operací roste exponenciálně s $n \sim 2^n$
- Iterační algoritmus
  - Počet operací je proporcionální k $n \sim 3n$

- Skutečný počet operací závisí na konkrétní implementaci, programovacím jazyku (překladači) a hardware

O efektivitě a složitosti algoritmů budeme hovořit v jedné z příštích přednášek

Základy algoritmizace

- Dnes:
  - Rekurze
    - Faktoriál
    - Obrácený výpis posloupnosti
    - Hanojské věže
    - Fibonacciho posloupnost
  - Více najdete např. na
    - http://algosaur.us/recursion/
    - https://mitpress.mit.edu/sicp/full-text/book/book-Z-H-11.html

Příště rekurzivní řazení