Programozás I.

Rekurzió

Sergyán Szabolcs
sergyan.szabolcs@nik.uni-obuda.hu

Óbudai Egyetem
Neumann János Informatikai Kar

2013. december 29.

Rekurzív algoritmusok

Faktoriális

Matematikai definíció

Az $n \in \mathbb{N}$ faktoriálisa (jelölés: $n!$) alatt az első n darab pozitív természetes szám szorzatát értjük, $n = 0$ esetén pedig 1-et.

$$n! = \begin{cases} 1, & \text{ha } n = 0, \\ \prod_{i=1}^{n} i, & \text{ha } n \geq 1. \end{cases}$$

Megvalósítás a *Sorozatszámítás* tétellel

Eljárás fakt(N, R)
    $R \leftarrow 1$
    Ciklus $i \leftarrow 1$-**től** N-**ig**
        $R \leftarrow R * i$
    Ciklus vége
Eljárás vége

Rekurzív faktoriális

Matematikai definíció

A faktoriális rekurzív módon is definiálható:

$$n! = \begin{cases} 1, & \text{ha } n = 0, \\ n \cdot (n-1)!, & \text{ha } n \geq 1. \end{cases}$$

Függvénnyel leírva

$$\text{fakt}(n) = \begin{cases} 1, & \text{ha } n = 0, \\ n \cdot \text{fakt}(n-1), & \text{ha } n \geq 1. \end{cases}$$

Összetevők

A rekurzív definíciónak két része van:

- Alapeset (itt most az $n = 0$ eset)
- Indukciós lépés: visszavezetjük a problémát kisebb számok esetére.

Rekurzív faktoriális

Rekurzív algoritmus

Függvény fakt(N)
    Ha $N = 0$, **akkor**
        return(1)
    Különben
        return($N \cdot$ fakt($N - 1$))
    Elágazás vége
Függvény vége

Kiértékelés

$$
\begin{aligned}
\text{fakt}(5) = \quad & 5 \cdot \text{fakt}(4) \\
& 5 \cdot 4 \cdot \text{fakt}(3) \\
& 5 \cdot 4 \cdot 3 \cdot \text{fakt}(2) \\
& 5 \cdot 4 \cdot 3 \cdot 2 \cdot \text{fakt}(1) \\
& 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 \cdot \text{fakt}(0) \\
& 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 \cdot 1 \\
& 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 \\
& 5 \cdot 4 \cdot 3 \cdot 2 \\
& 5 \cdot 4 \cdot 6 \\
& 5 \cdot 24 \\
& 120
\end{aligned}
$$

Rekurzió megvalósítása

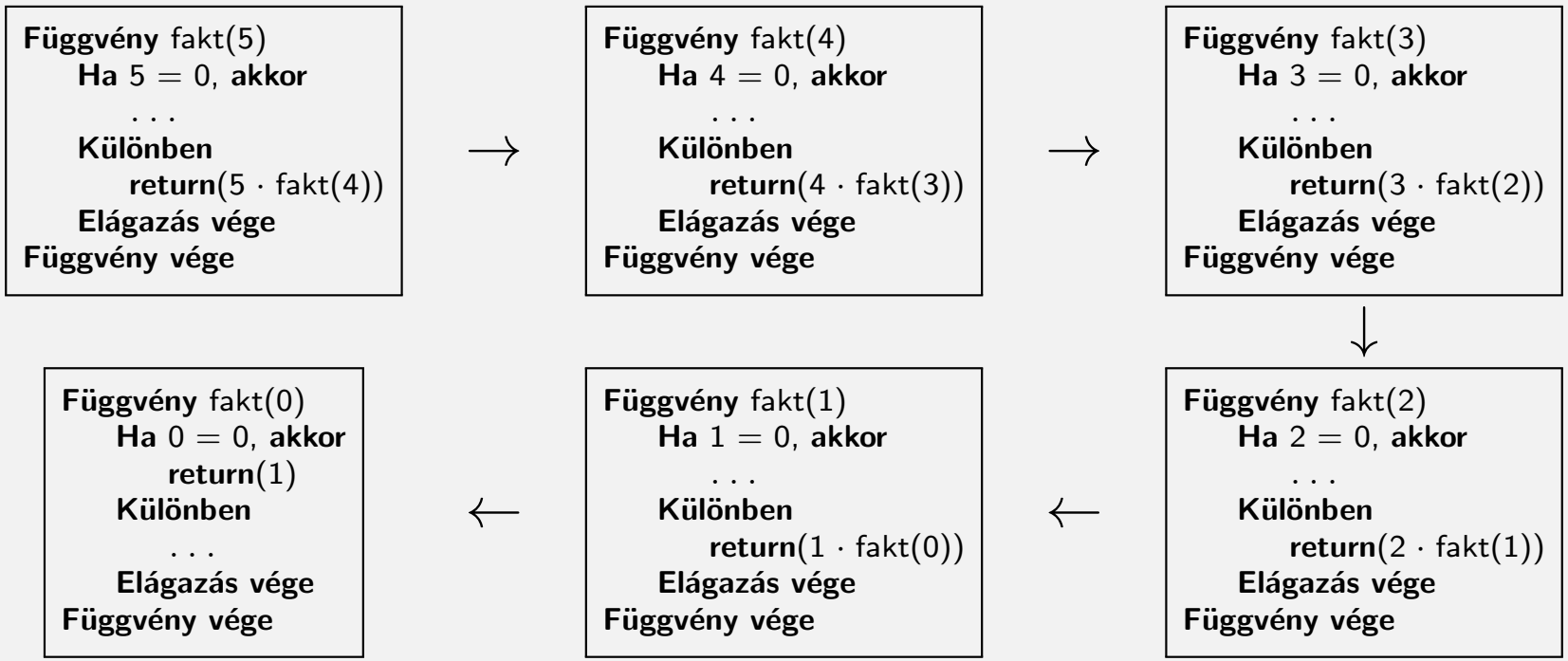
Problémák

- Ugyanabból a függvényből egyszerre több is fut?
  - Előző példában a fakt függvényből egyszerre 6 is futott.
- A függvényen belüli lokális változók értéke felülíródik?
  - Előző példában honnan lehet tudni, hogy N értéke éppen mennyi?

Megoldás

- Minden meghívott függvény más, csak a nevük azonos.
- A memóriában eltároljuk, hogy honnan hívtunk egy függvény, majd ide lehet visszatérni később.
- A lokális változók minden egyes függvényben külön-külön létrejönnek.
- Azonos nevű, de különböző változókról van szó, hiszen a függvények is különbözőek.
- Az ezen célra fenntartott memória túlcsordulhat túl sok függvényhívás esetén.

Rekurzió megvalósítása



Rekurzív algoritmusok

Fibonacci számok

Nyulak szaporodásának vizsgálata

Fibonacci a nyulak szaporodását vizsgálta, és a következőket tapasztalta:

- Minden nyúlpár - amikor szaporodik -, akkor két utóddal járul hozzá a népességhez.
- A nyulak a születésüket követően két egymás utáni időpontban szaporodnak.
- Vizsgáljuk, hogy adott időpontban hány új nyúlpár születik.

Matematikai leírás

Jelölje $Fib(n)$, hogy az n-edik szaporodáskor hány új nyúlpár születik.

- Kezdetben van egy nyúlpár: $Fib(0) = 1$
- Következő szaporodáskor ez a pár szaporodik: $Fib(1) = 1$
- Minden más szaporodáskor az előző és az azt megelőző szaporodáskor született nyulak járulnak hozzá a szaporodáshoz (mégpedig egy-egy nyúlpárral). Tehát: $Fib(n) = Fib(n-1) + Fib(n-2)$

Fibonacci számok

Rekurzív algoritmus

Függvény Fib(N)
    Ha $N \leq 1$ **akkor**
        return(1)
    Különben
        return(Fib($N-1$) + Fib($N-2$))
    Elágazás vége
Függvény vége

Fibonacci számok

Az algoritmus megvalósítható rekurzió nélkül is.

Iteratív algoritmus

Függvény Fib(N)
    $a \leftarrow 1$
    $e \leftarrow 1$
    Ciklus $i \leftarrow 1$-**től** $N - 1$-**ig**
        $temp \leftarrow a + e$
        $e \leftarrow a$
        $a \leftarrow temp$
    Ciklus vége
    return(a)
Függvény vége

Rekurzív algoritmusok

Binomiális együtthatók

Matematikai definíció

$$\binom{n}{k} = \begin{cases} 1, & \text{ha } k = 0 \\ \binom{n-1}{k} + \binom{n-1}{k-1}, & \text{ha } 0 < k < n \\ 1, & \text{ha } k = n \end{cases}$$

Algoritmus

Függvény Bin(N, K)
    Ha ($K = 0$) **vagy** ($K = N$) **akkor**
        return(1)
    Különben
        return(Bin($N-1, K$) + Bin($N-1, K-1$))
    Elágazás vége
Függvény vége

Rekurzív algoritmusok

Szöveg megfordítása

Feladat

- Az X változóban adott egy N hosszúságú szöveg.
- Feladat, hogy a karakterek sorrendjét megfordítsuk.
- Például: `abcdef` $\rightarrow$ `fedcba`

Rekurzív megvalósítás

Függvény Fordítás(X, N)
    Ha $N > 1$ **akkor**
        return(Fordítás($X[2..N], N-1) + X[1]$)
    Különben
        return($X[N]$)
    Elágazás vége
Függvény vége

Palindrom

Feladat

- Egy N-jegyű szám palindrom, ha az i-edik számjegye megegyezik az $(N + 1 - i)$-edik számjegyével. Például: 987654356789.
- Egy szám számjegyeit tároljuk az N elemű X tömbben.
- Egy egyjegyű szám palindrom.
- Ha a két szélső számjegy azonos, akkor vizsgáljuk a szélsők nélküli számot.

Palindrom

Rekurzív megvalósítás

Függvény Palindrom_e(X, N)
    Ha $N <= 1$ **akkor**
        return(*Igaz*)
    Különben
        Ha $X[1] = X[N]$ **akkor**
            return(Palindrom_e($X[2..N-1]$, $N-2$))
        Különben
            return(*Hamis*)
        Elágazás vége
    Elágazás vége
Függvény vége

Hatványozás

Az a pozitív egész szám n-edik hatványát kiszámíthatjuk az alábbi módon:

Algoritmus

Függvény Hatvány(a, n)
    $temp \leftarrow 1$
    Ciklus $i \leftarrow 1$-**től** n-**ig**
        $temp \leftarrow temp * a$
    Ciklus vége
    return($temp$)
Függvény vége

Hatványozás

Ötlet

$$a^n = \begin{cases} a^{n/2} \cdot a^{n/2}, & \text{ha n páros} \\ a^{(n-1)/2} \cdot a^{(n-1)/2} \cdot a, & \text{ha } n \text{ páratlan} \end{cases}$$

Rekurzív megvalósítás

Függvény Hatvány(a, n)
    Ha $n = 0$ **akkor**
        return(1)
    Különben
        Ha n páros, **akkor**
            return(Hatvány(a, $n/2$)·Hatvány(a, $n/2$))
        Különben
            return(Hatvány(a, $(n-1)/2$)·Hatvány(a, $(n-1)/2$)·a)
        Elágazás vége
    Elágazás vége
Függvény vége

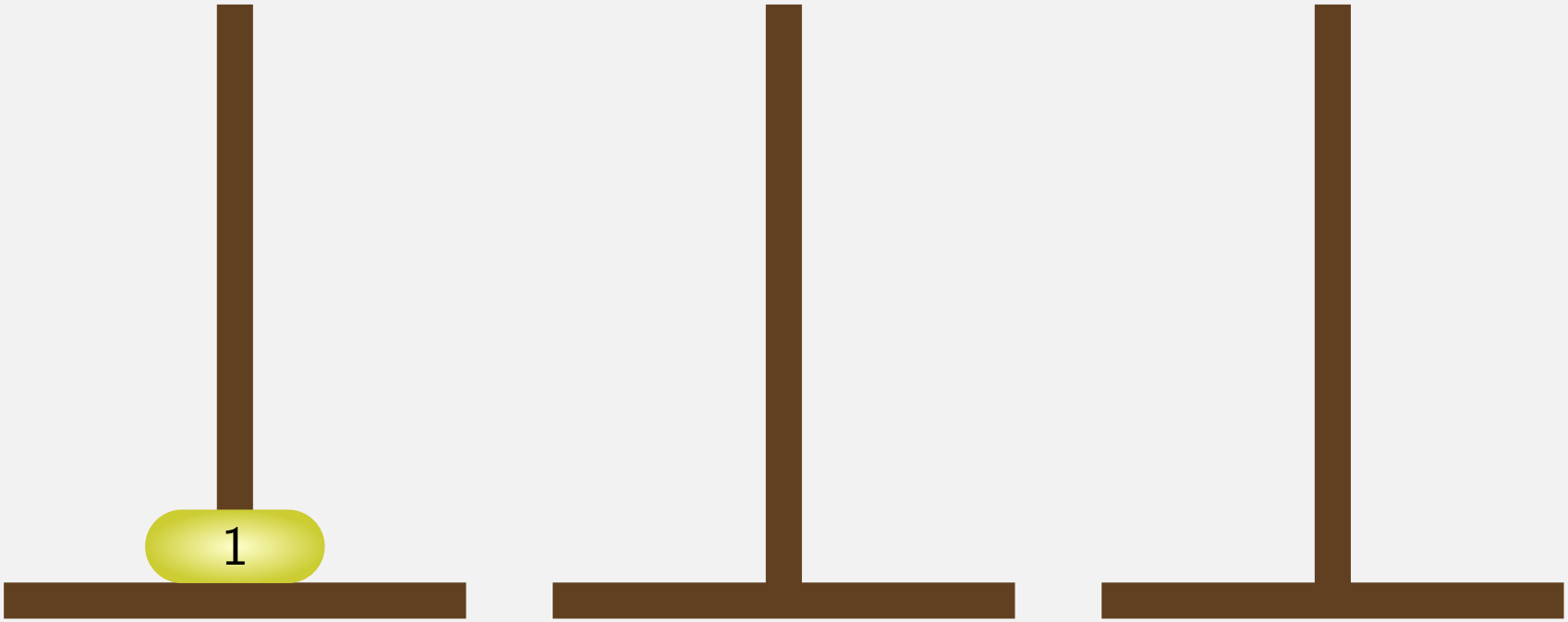
Rekurzív algoritmusok

Hanoi tornyai

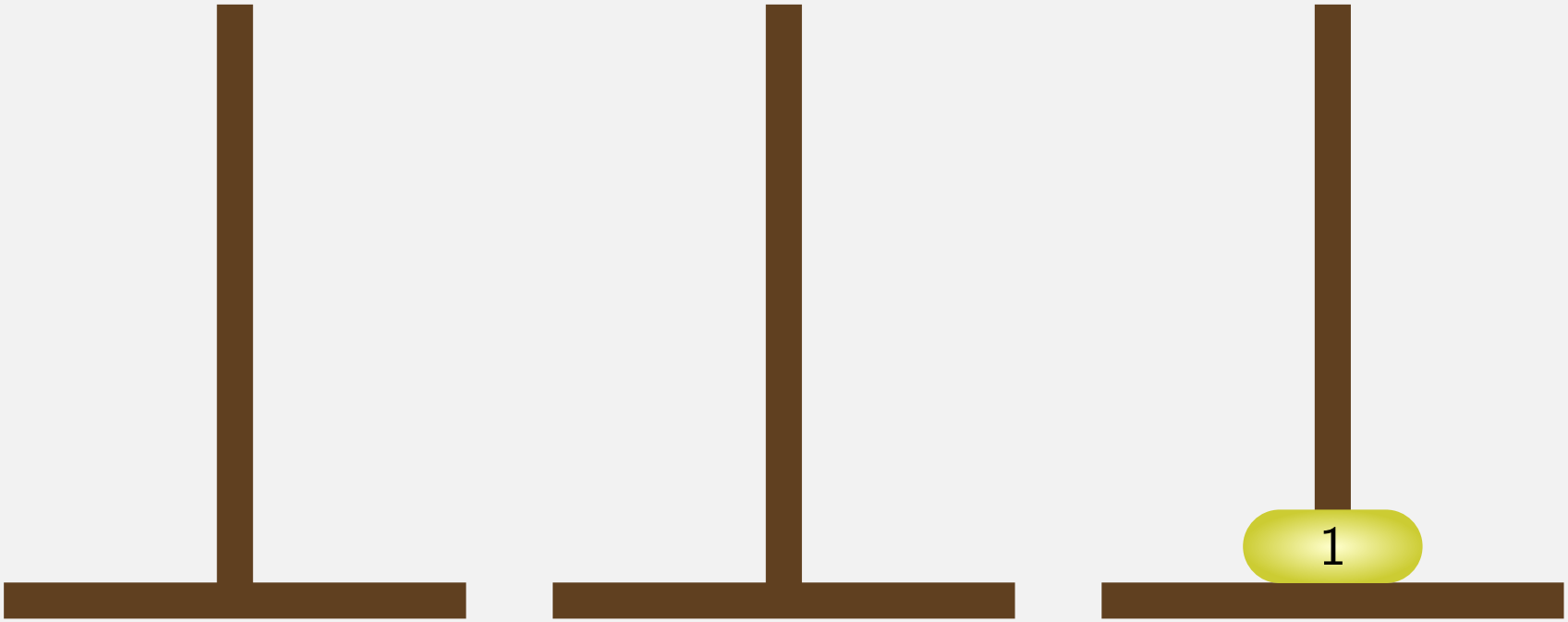
Feladat

- Az egyik szélső rúdról át kell pakolni a korongokat a másik szélső rúdra.
- Segítségül a középső rúd is használható.
- Egyszerre egyetlen korong mozgatható.
- Csak felül lévő korongot lehet mozgatni egyik rúdról a másikra.
- Egy korong nem rakható nála kisebb méretű korongra.

Hanoi tornyai – 1 Korong



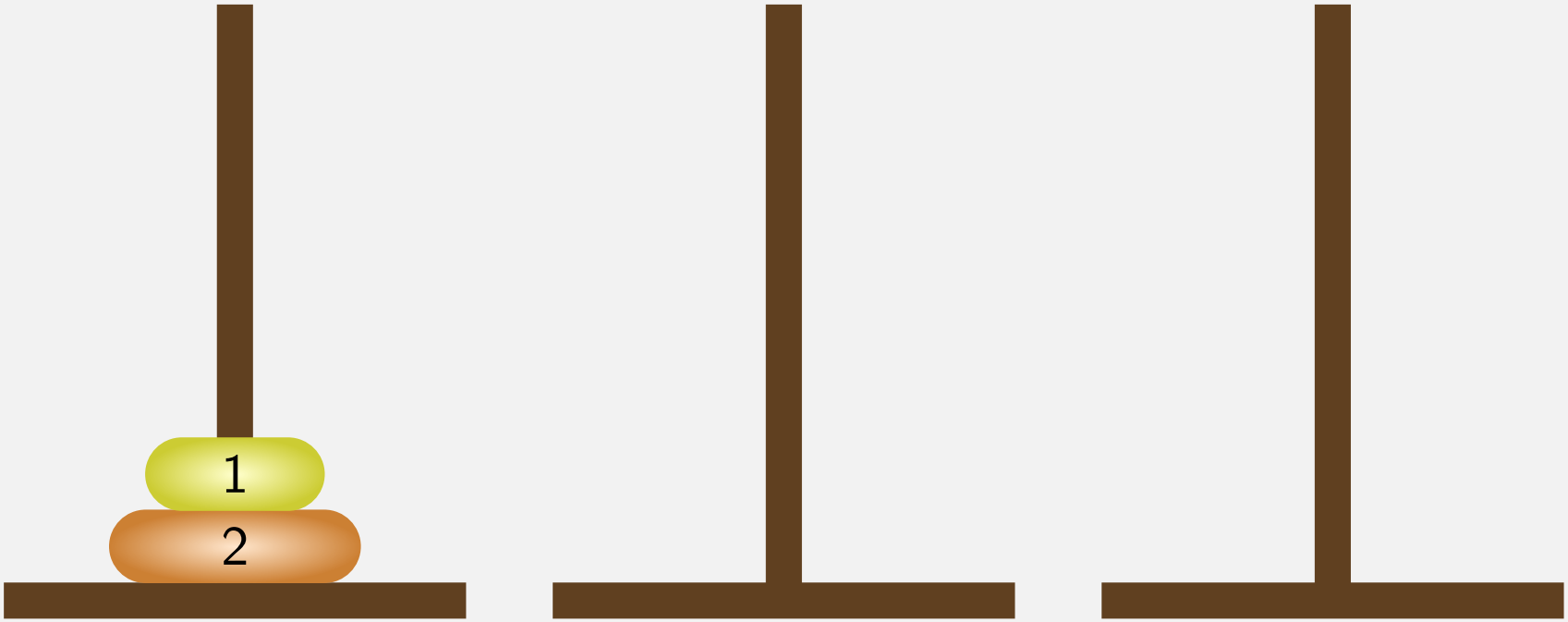
Hanoi tornyai – 1 Korong



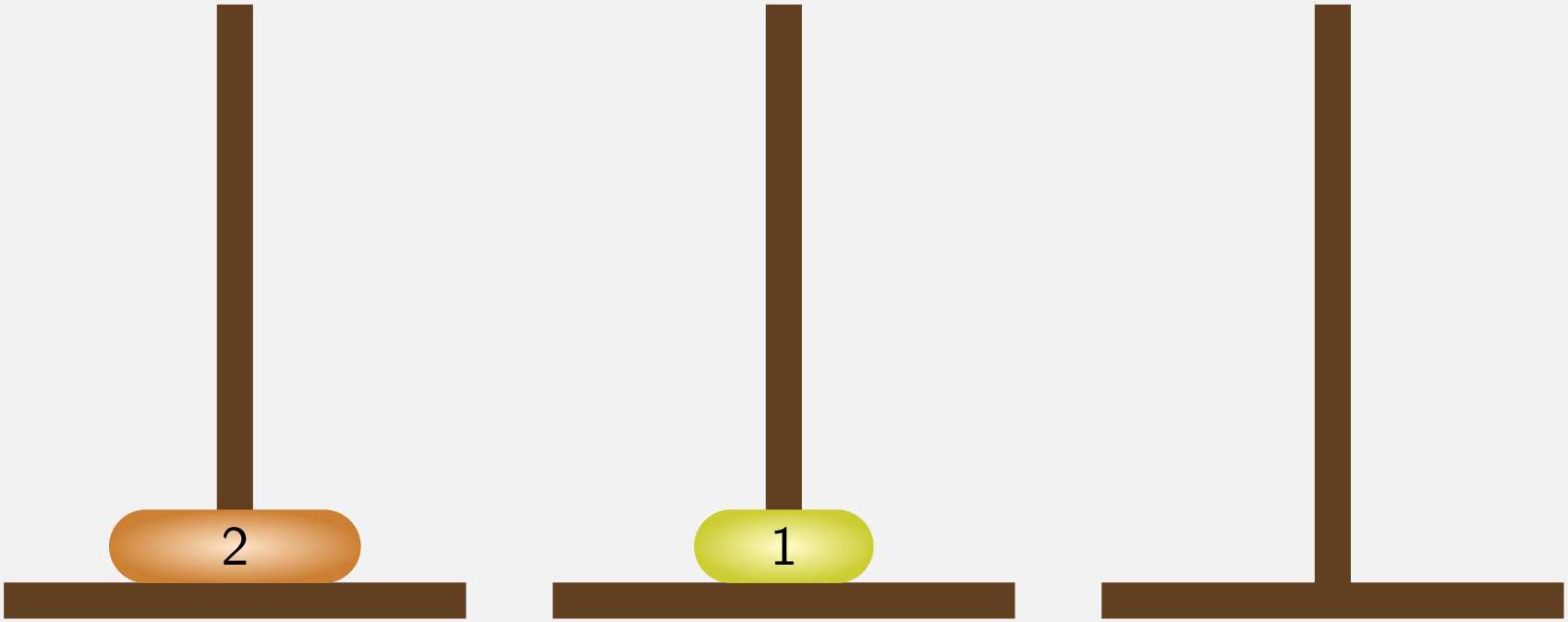
Mozgatott korong **1** rúdról **3** rúdra.

Hanoi tornyai – 1 Korong

Hanoi tornyai – 2 Korong

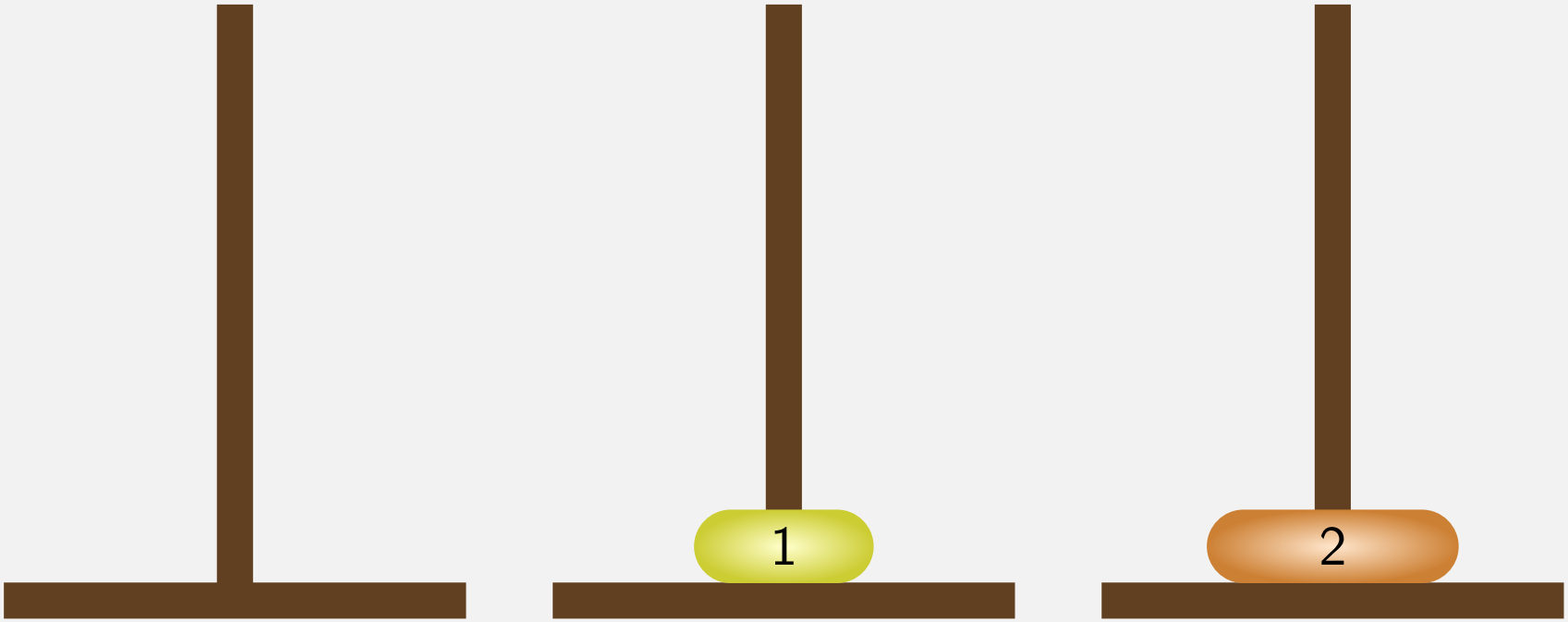


Hanoi tornyai – 2 Korong



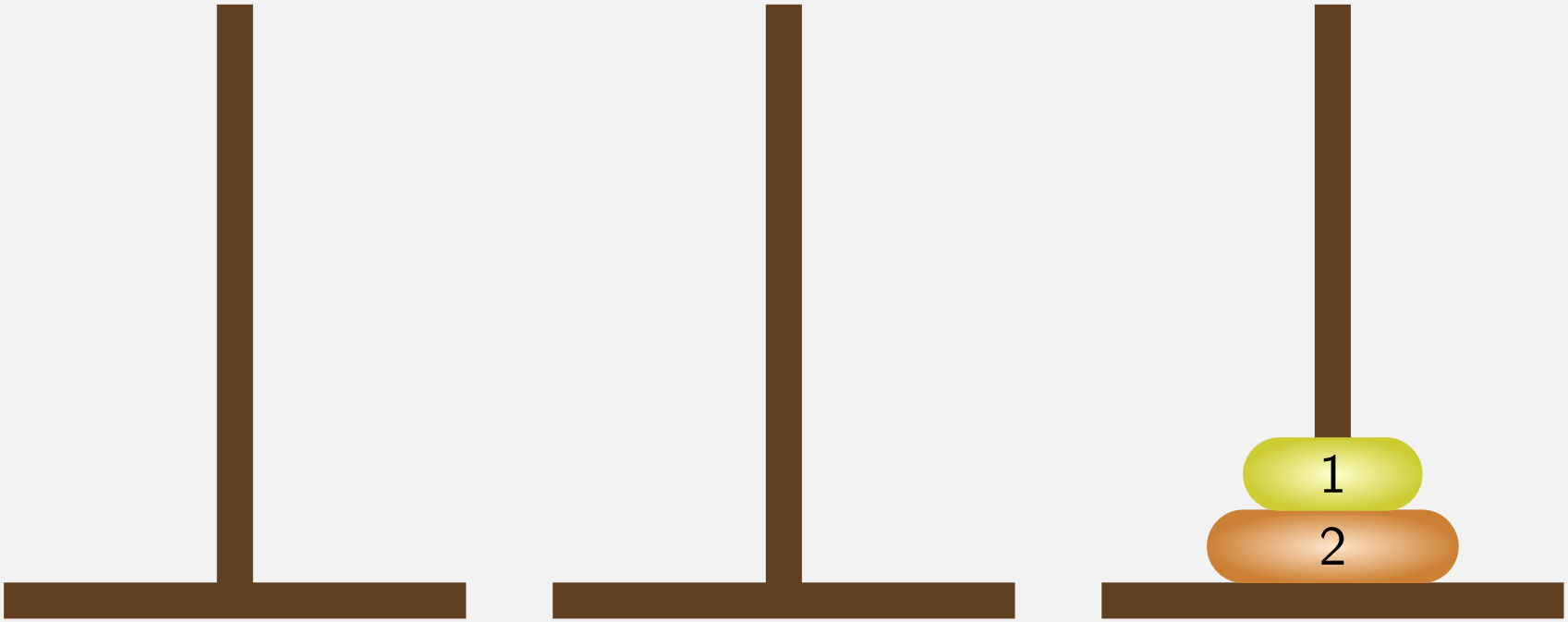
Mozgatott korong **1** rúdról **2** rúdra.

Hanoi tornyai – 2 Korong



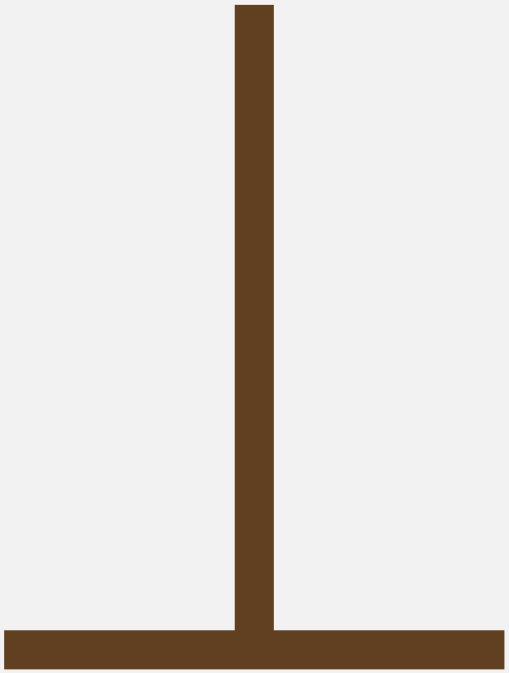
Mozgatott korong **1** rúdról **3** rúdra.

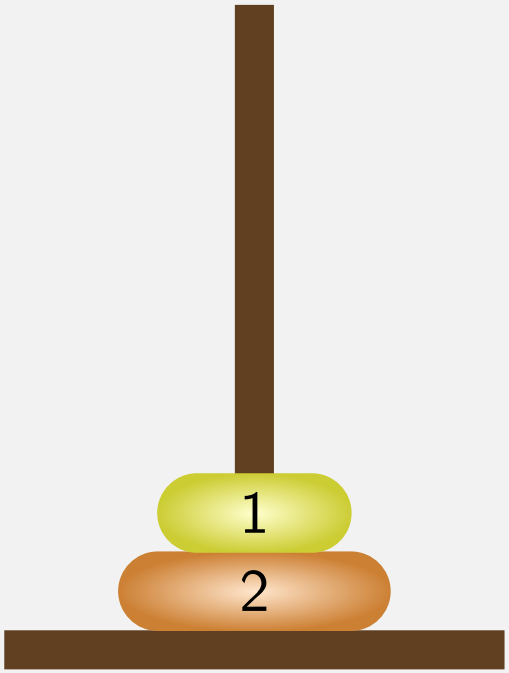
Hanoi tornyai – 2 Korong



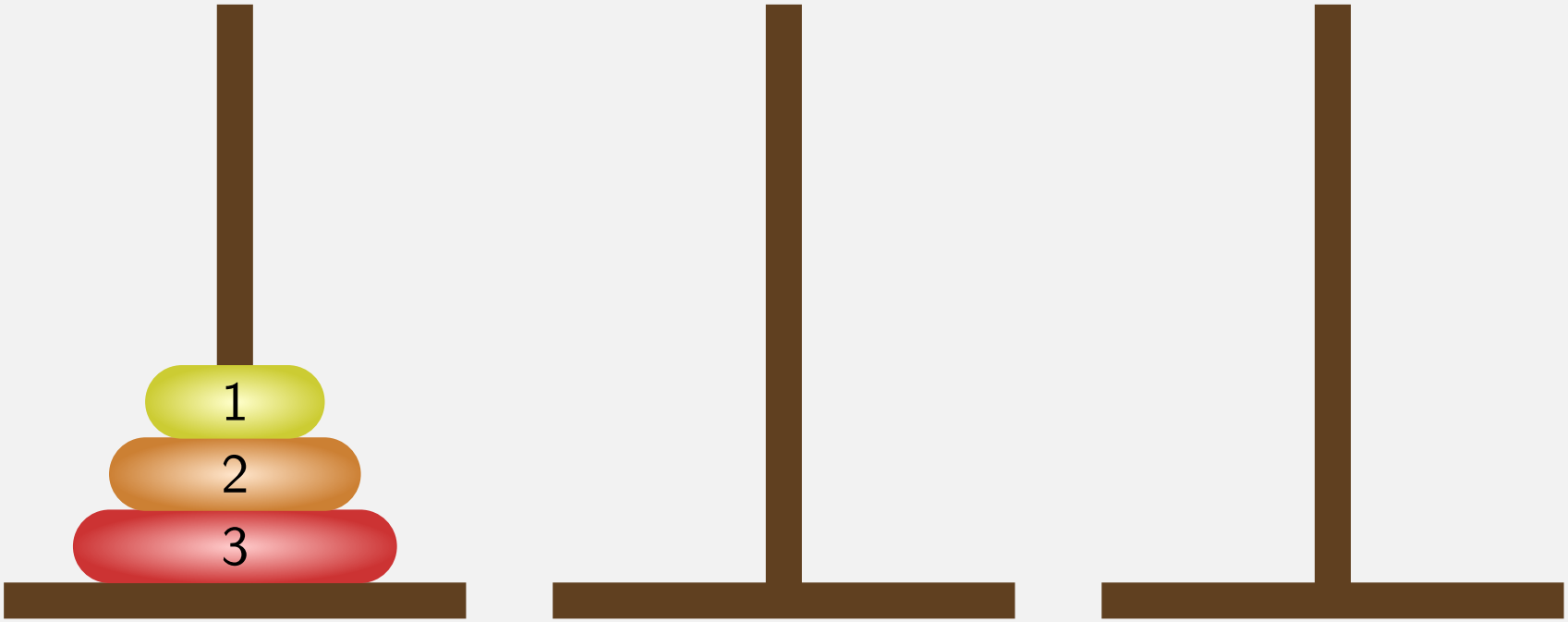
Mozgatott korong **2** rúdról **3** rúdra.

Hanoi tornyai – 2 Korong

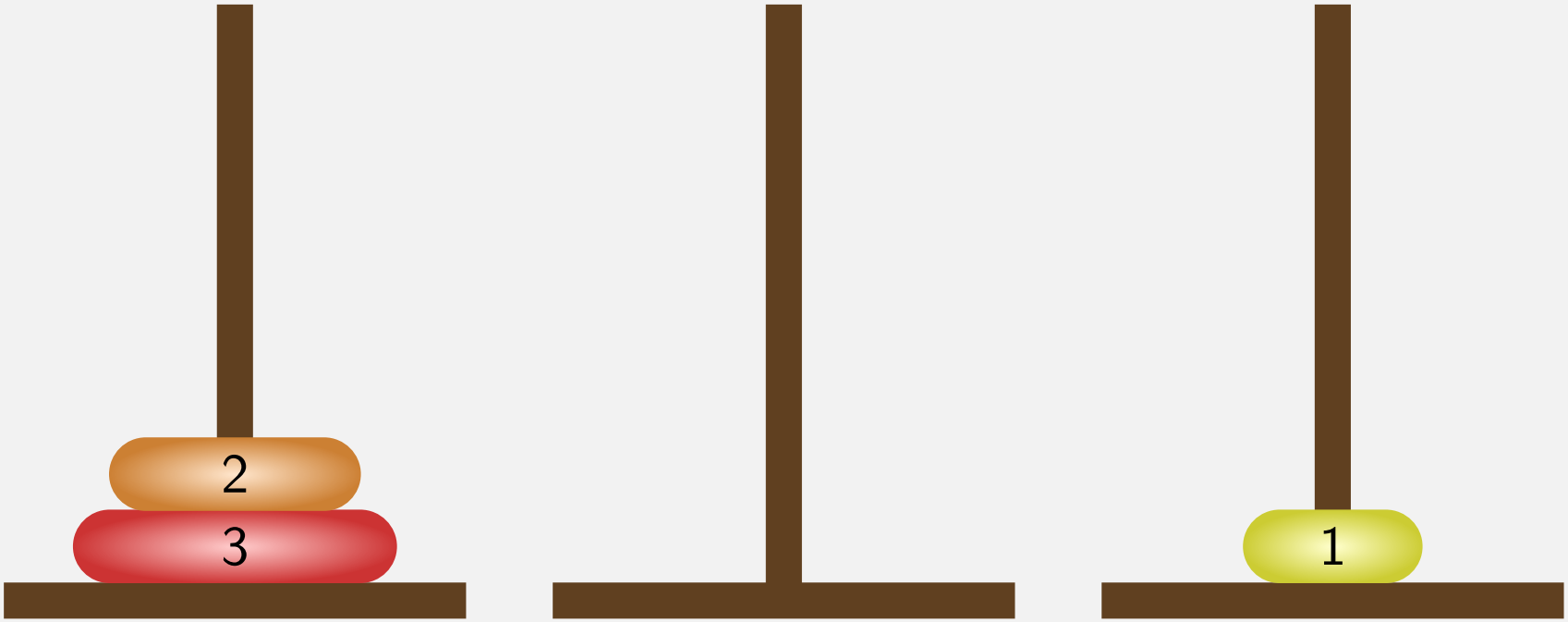




Hanoi tornyai – 3 Korong

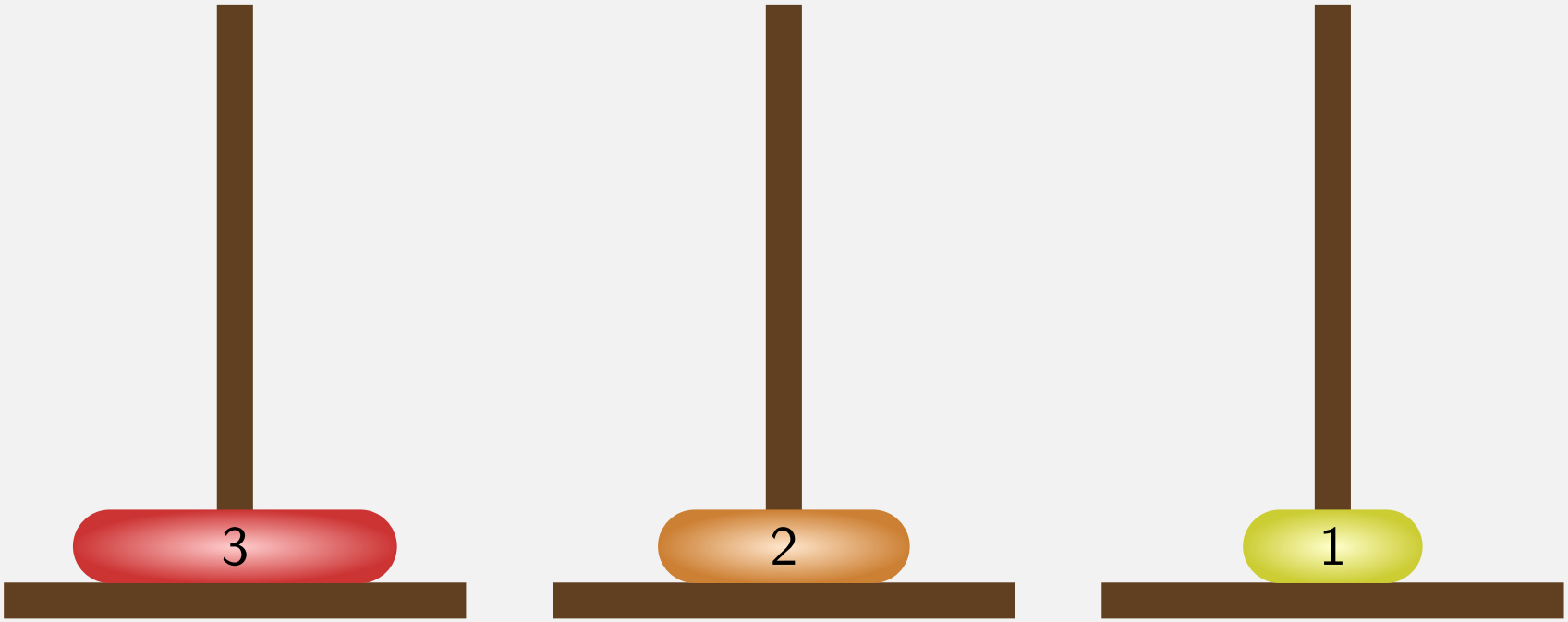


Hanoi tornyai – 3 Korong



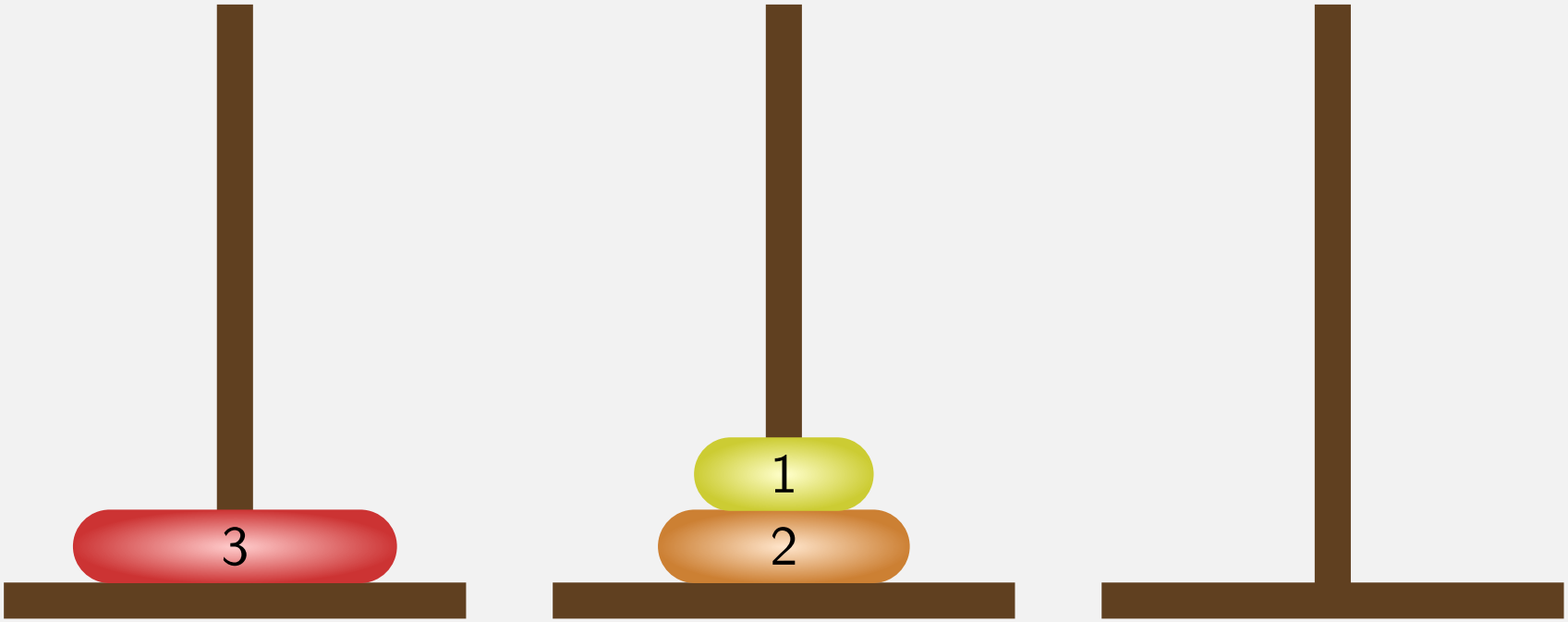
Mozgatott korong **1** rúdról **3** rúdra.

Hanoi tornyai – 3 Korong



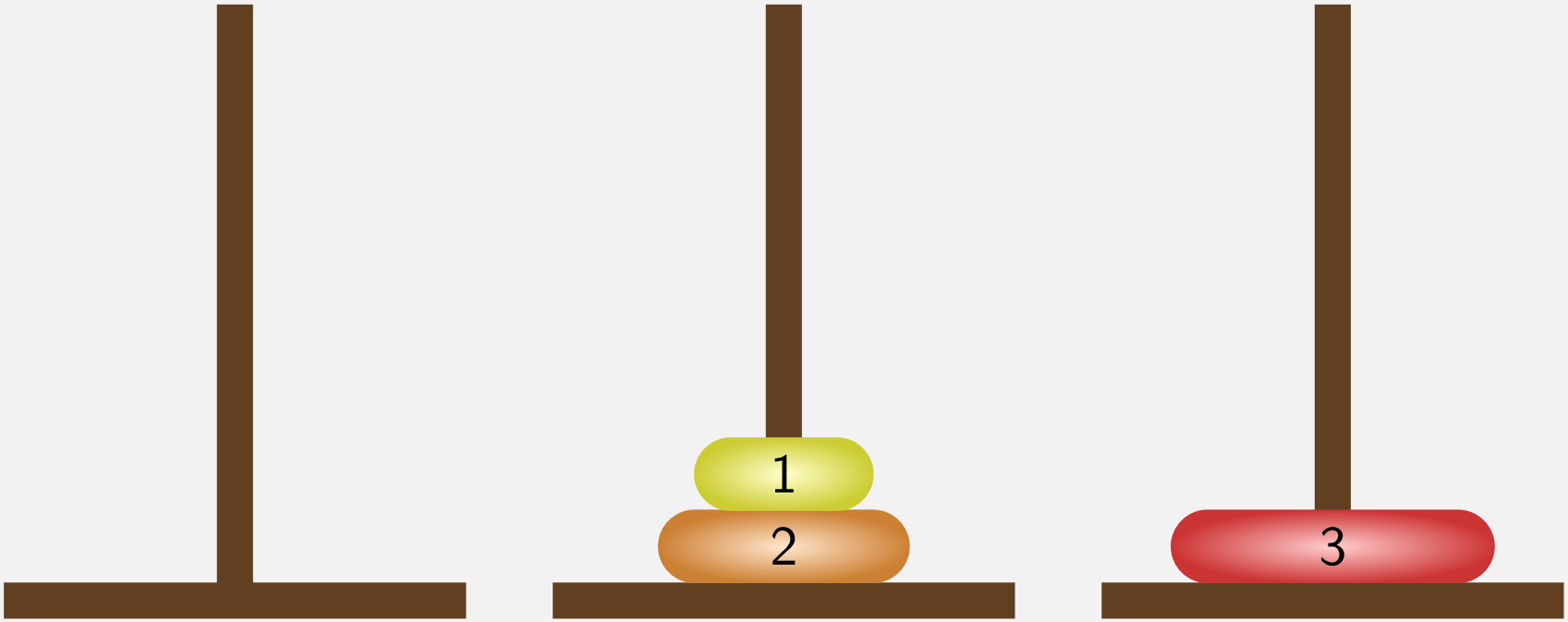
Mozgatott korong **1** rúdról **2** rúdra.

Hanoi tornyai – 3 Korong



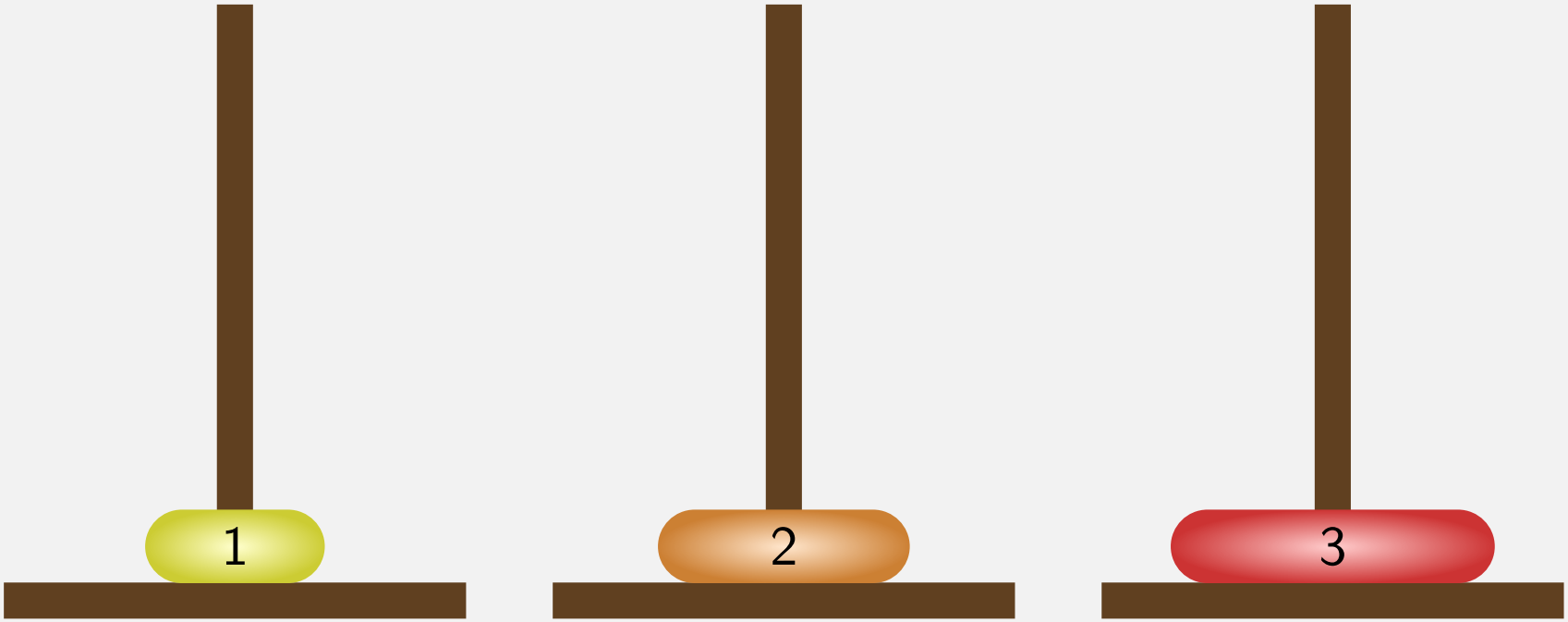
Mozgatott korong **3** rúdról **2** rúdra.

Hanoi tornyai – 3 Korong



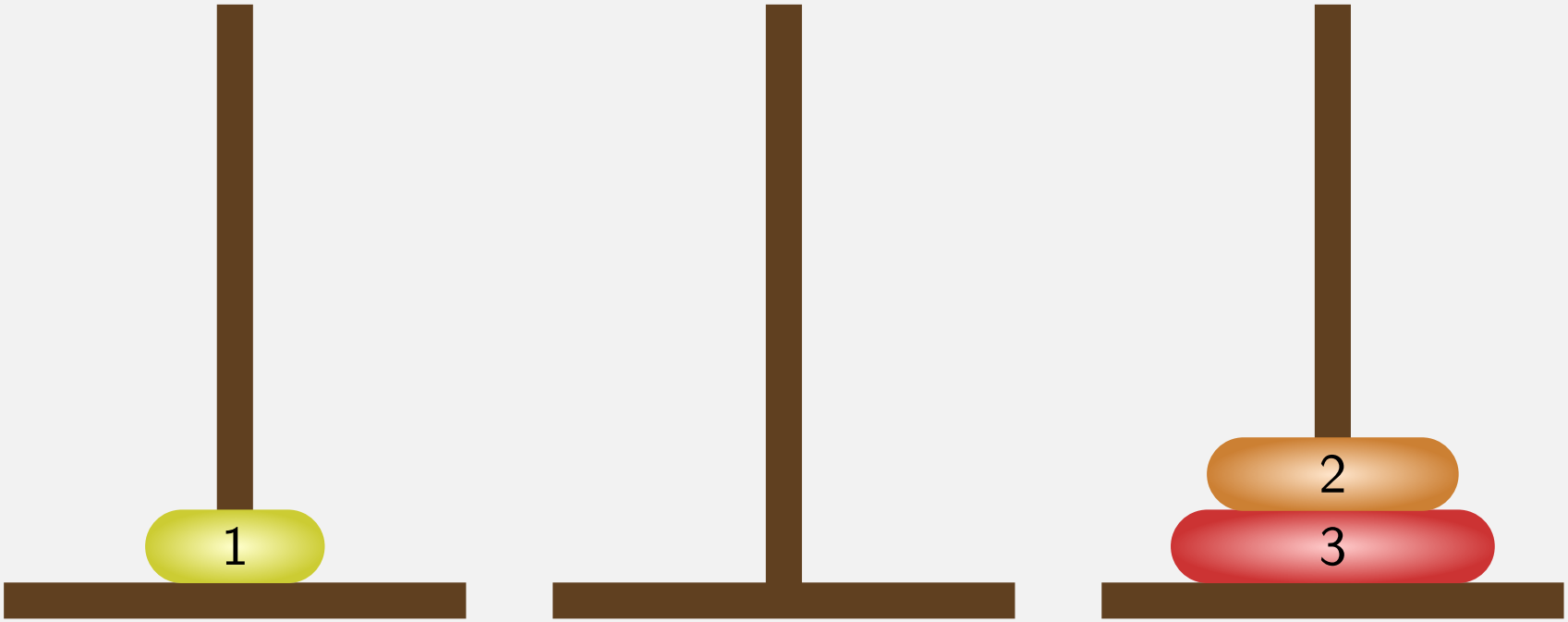
Mozgatott korong **1** rúdról **3** rúdra.

Hanoi tornyai – 3 Korong



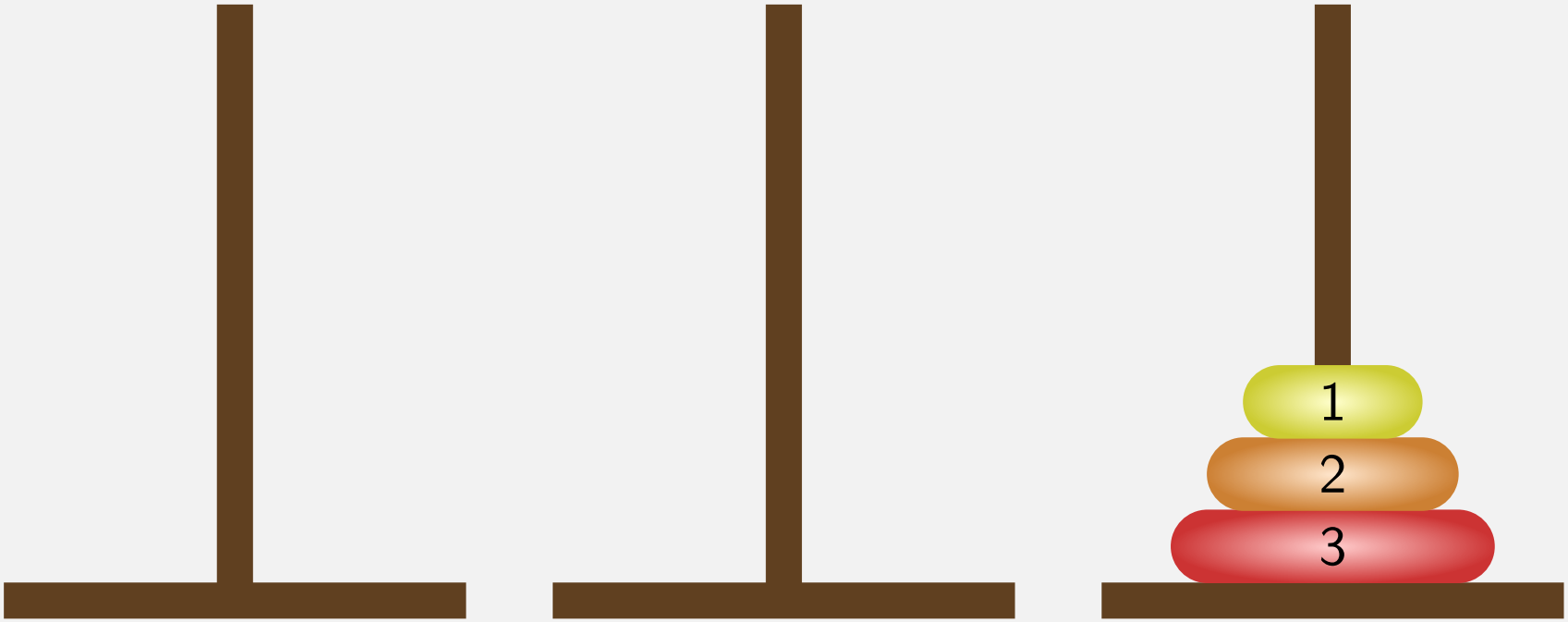
Mozgatott korong **2** rúdról **1** rúdra.

Hanoi tornyai – 3 Korong



Mozgatott korong **2** rúdról **3** rúdra.

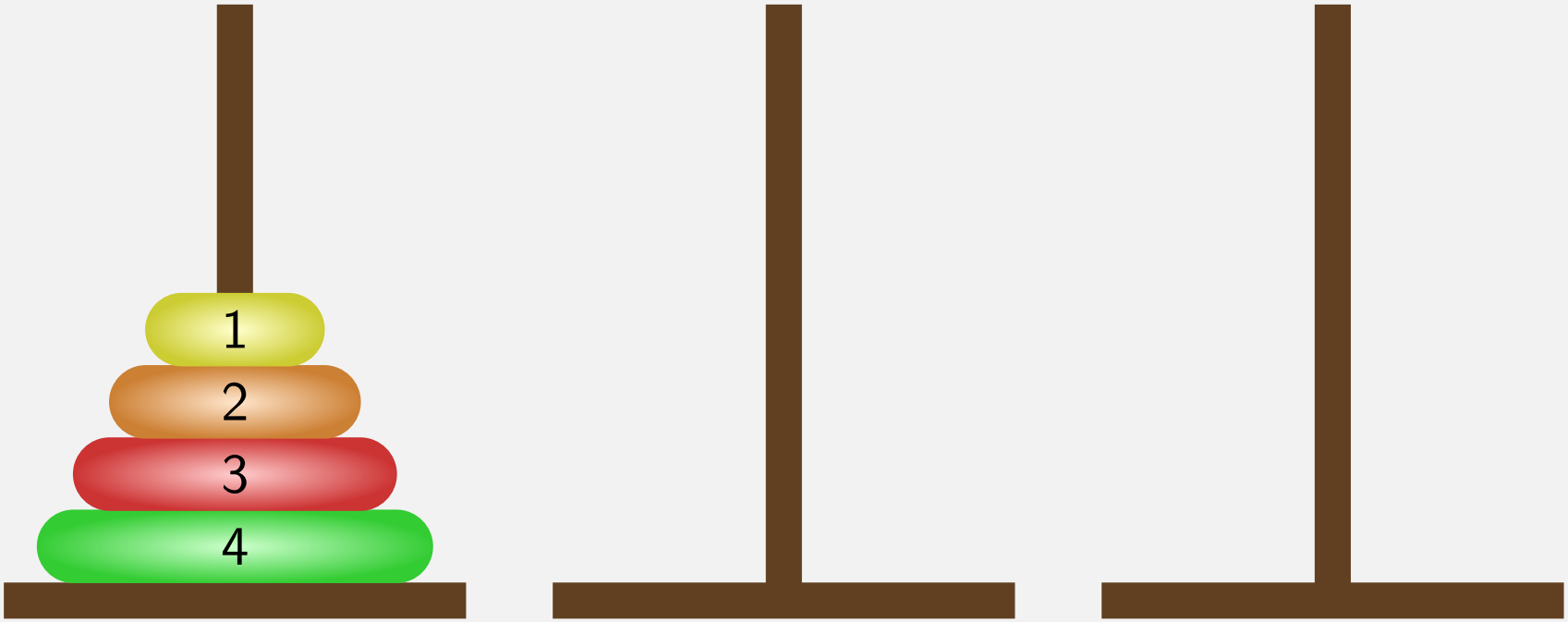
Hanoi tornyai – 3 Korong



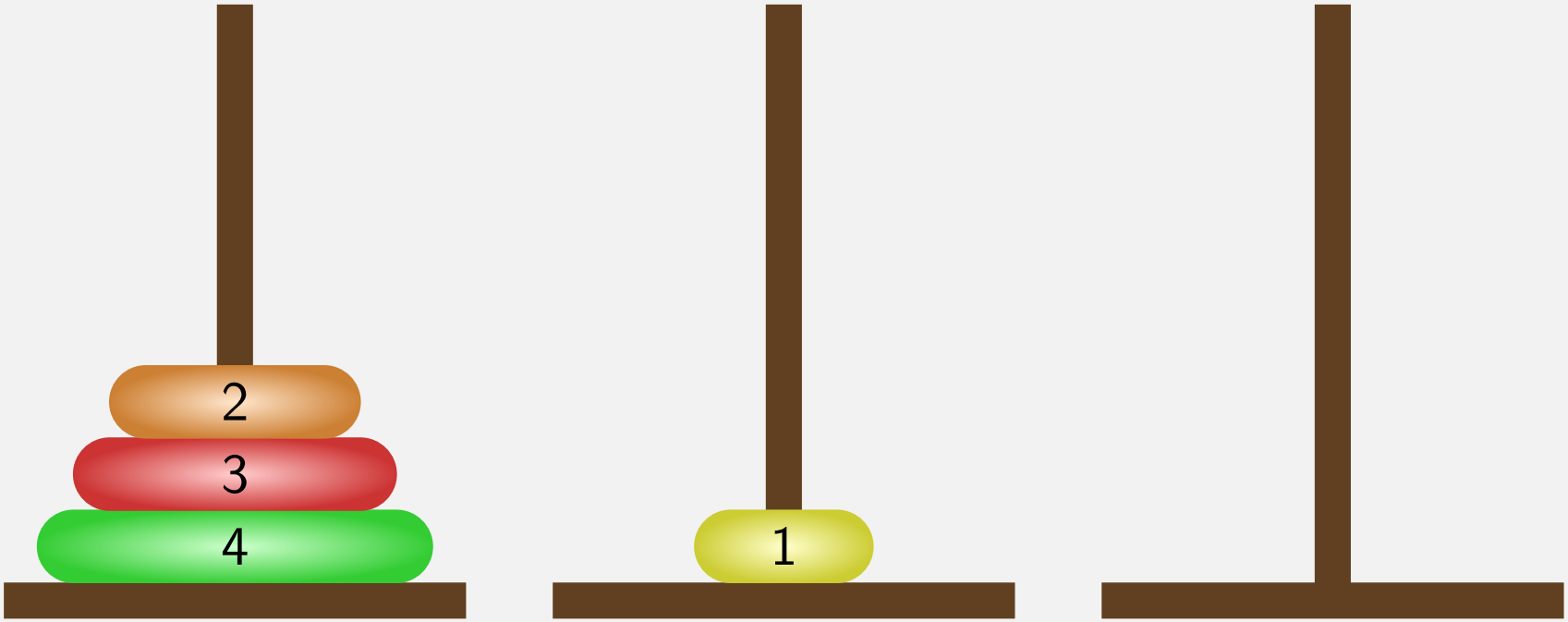
Mozgatott korong **1** rúdról **3** rúdra.

Hanoi tornyai – 3 Korong

Hanoi tornyai – 4 Korong

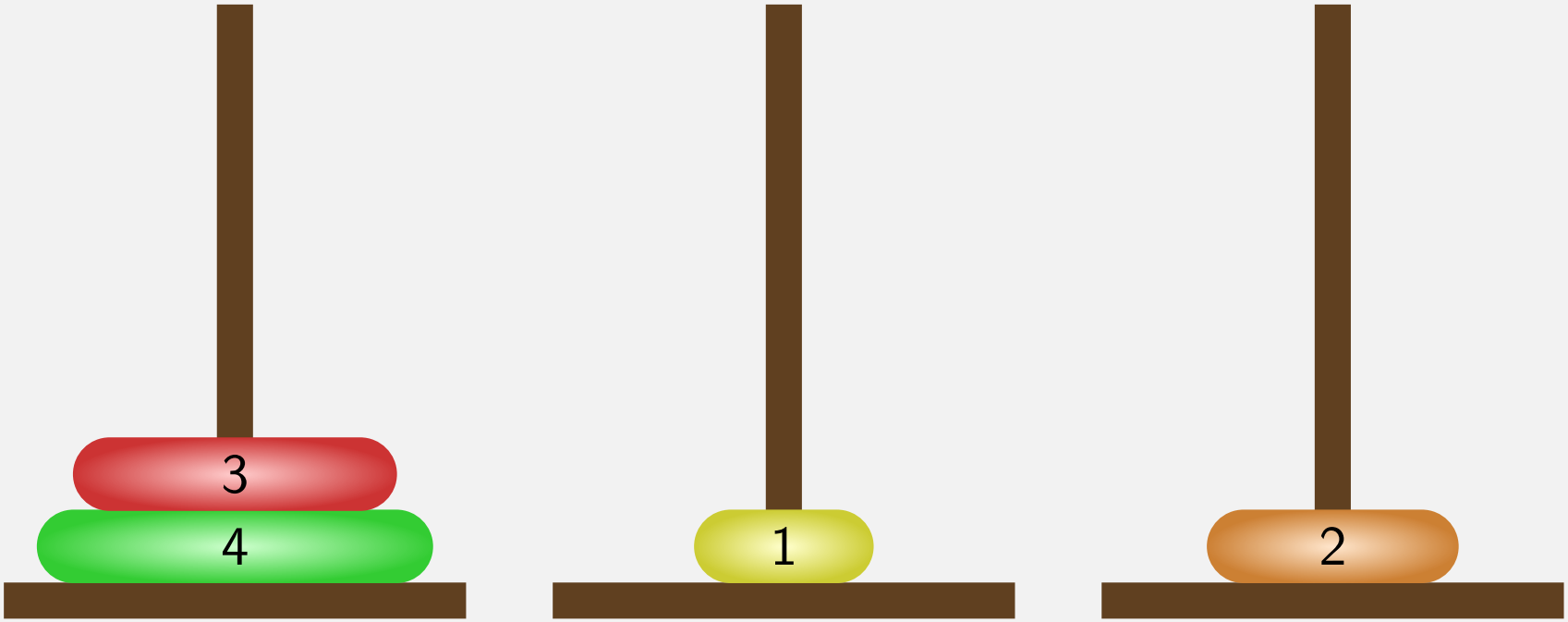


Hanoi tornyai – 4 Korong



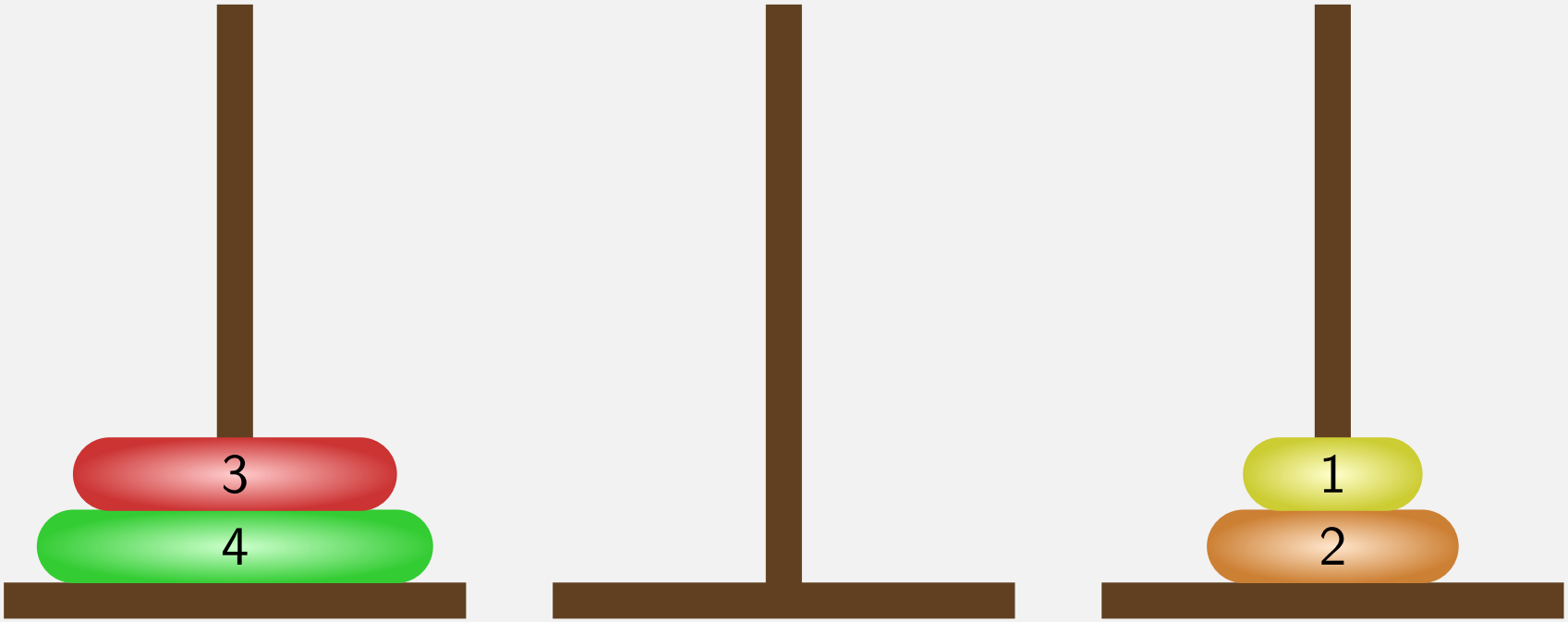
Mozgatott korong **1** rúdról **2** rúdra.

Hanoi tornyai – 4 Korong



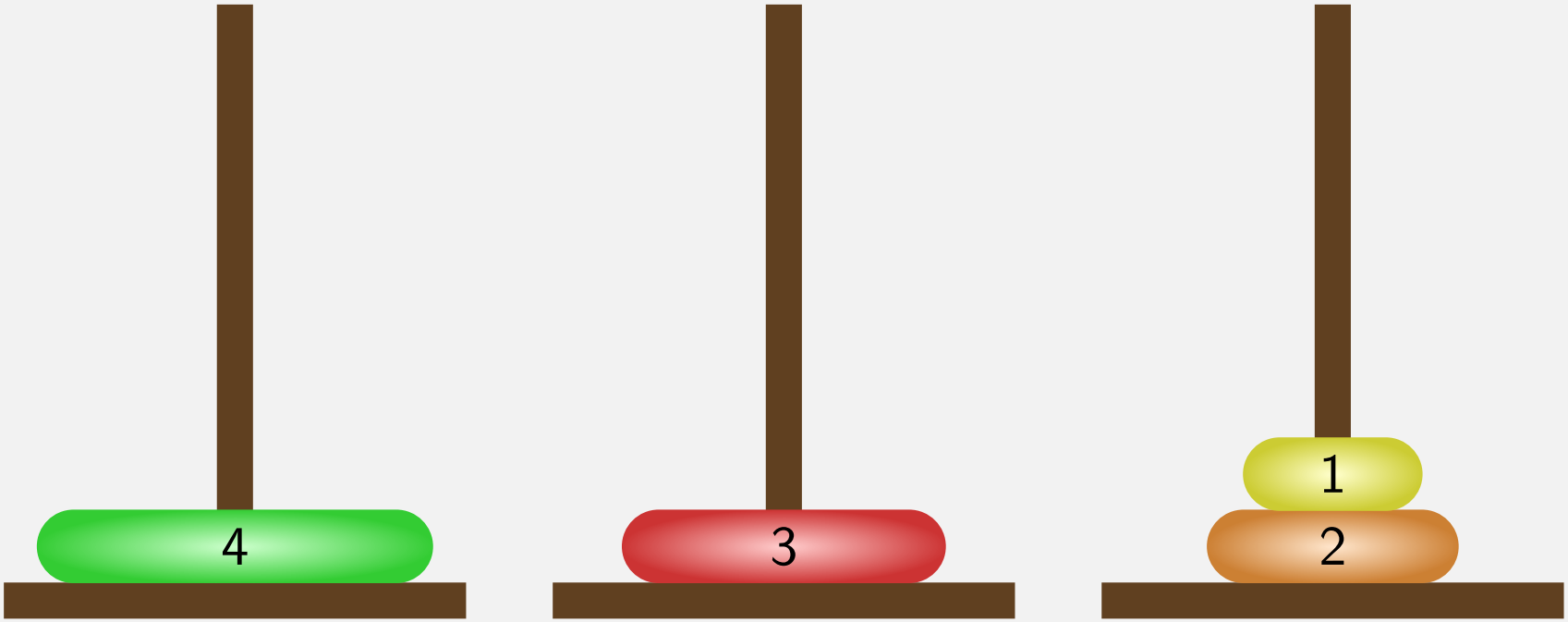
Mozgatott korong **1** rúdról **3** rúdra.

Hanoi tornyai – 4 Korong



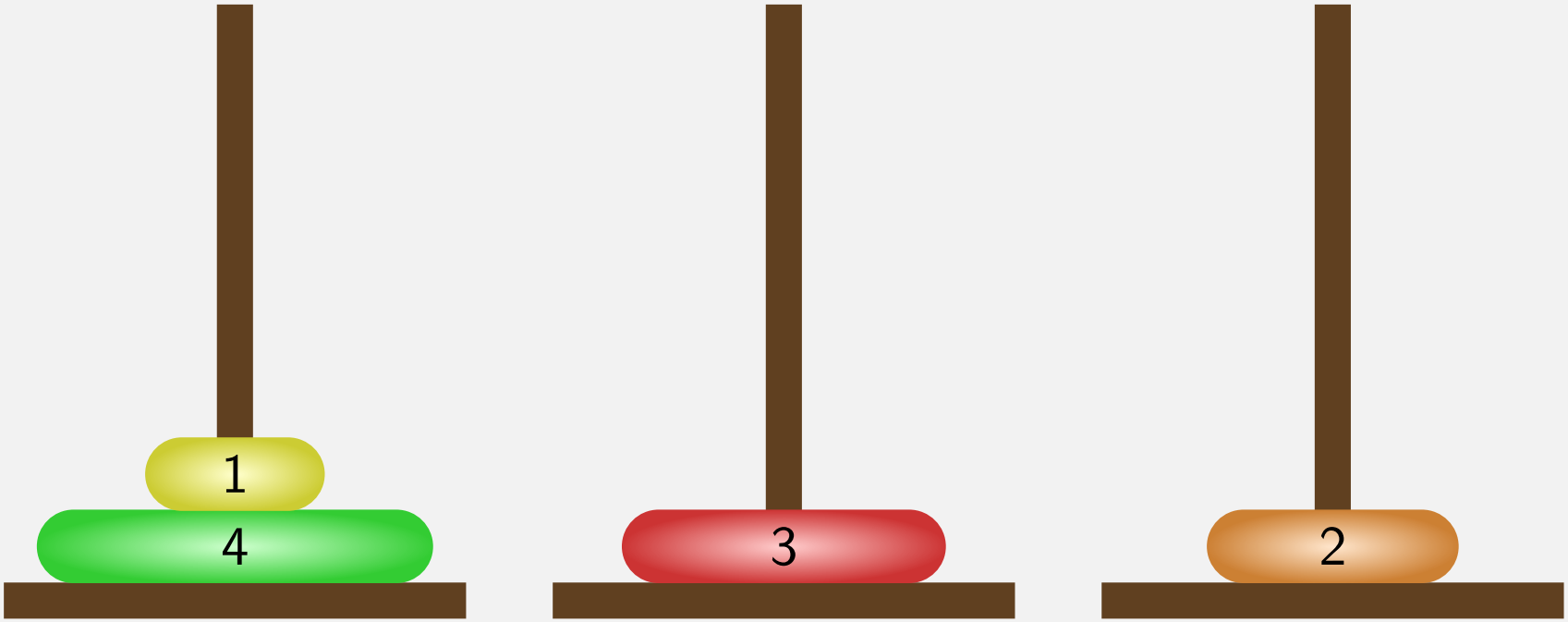
Mozgatott korong **2** rúdról **3** rúdra.

Hanoi tornyai – 4 Korong



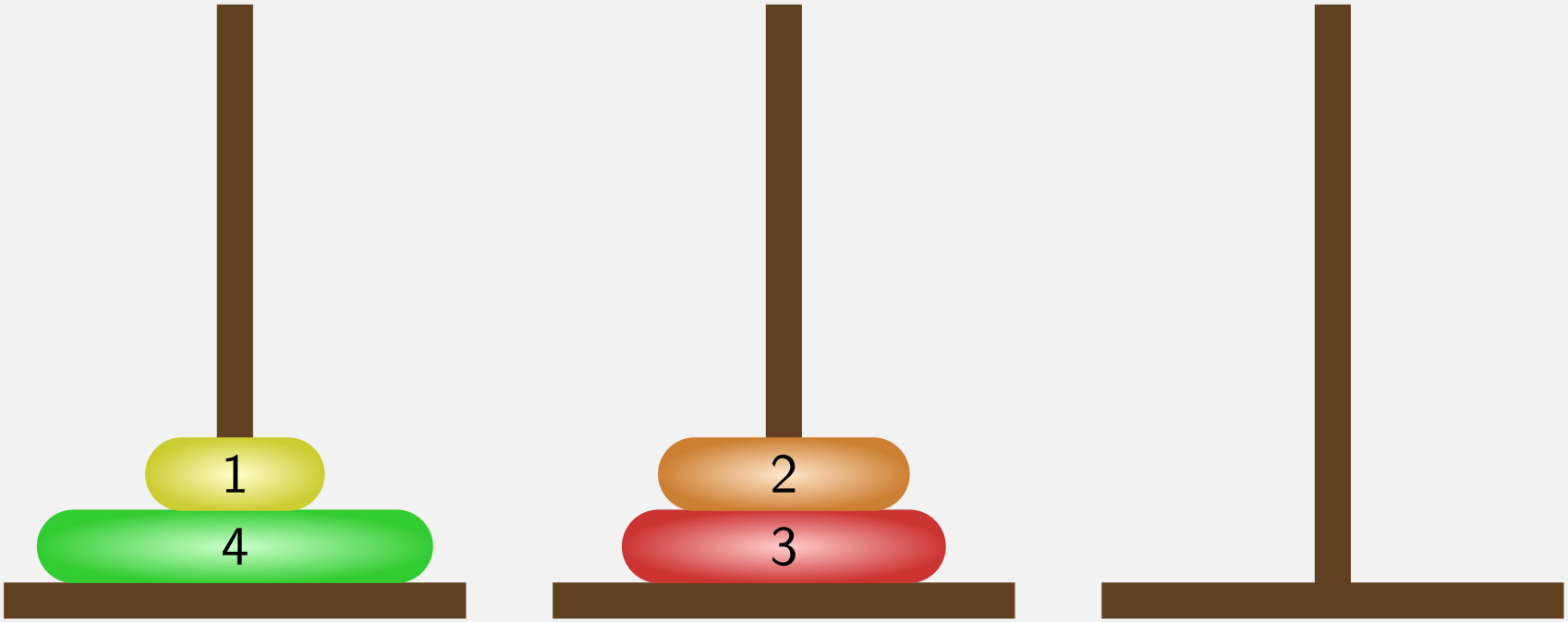
Mozgatott korong **1** rúdról **2** rúdra.

Hanoi tornyai – 4 Korong



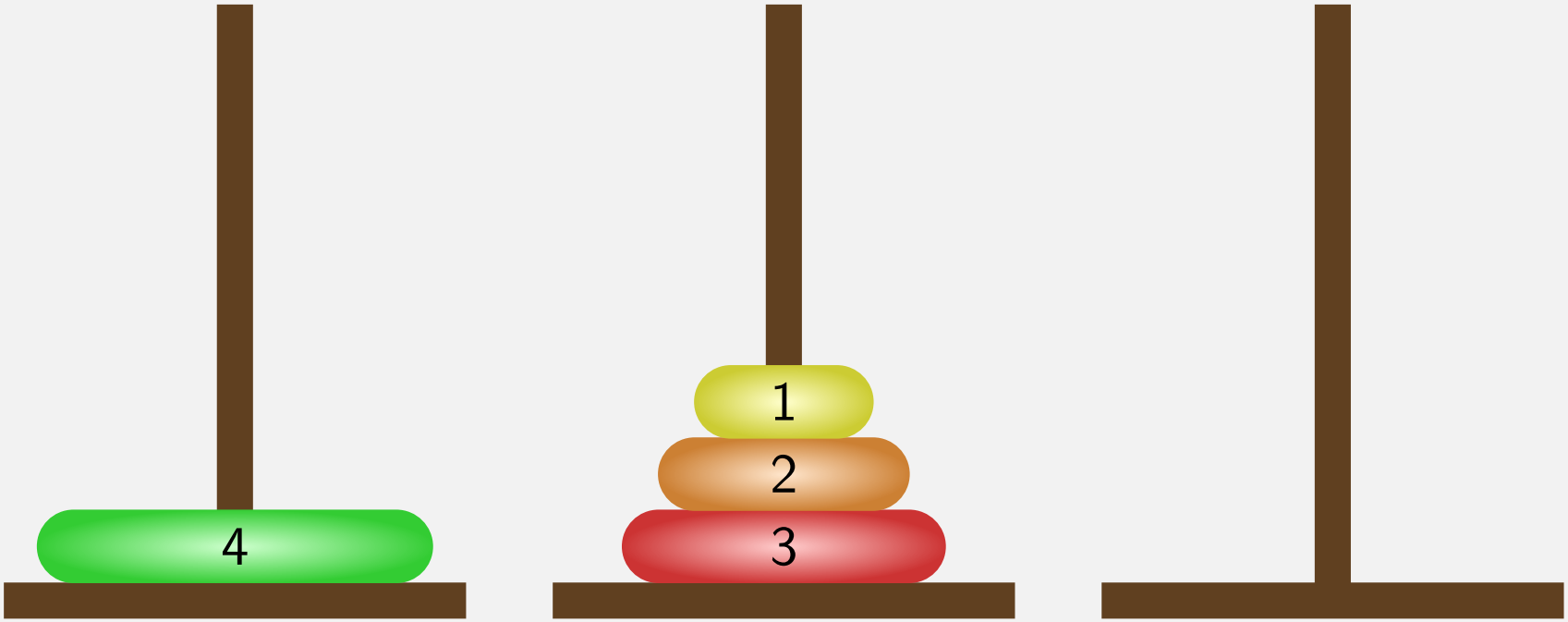
Mozgatott korong **3** rúdról **1** rúdra.

Hanoi tornyai – 4 Korong



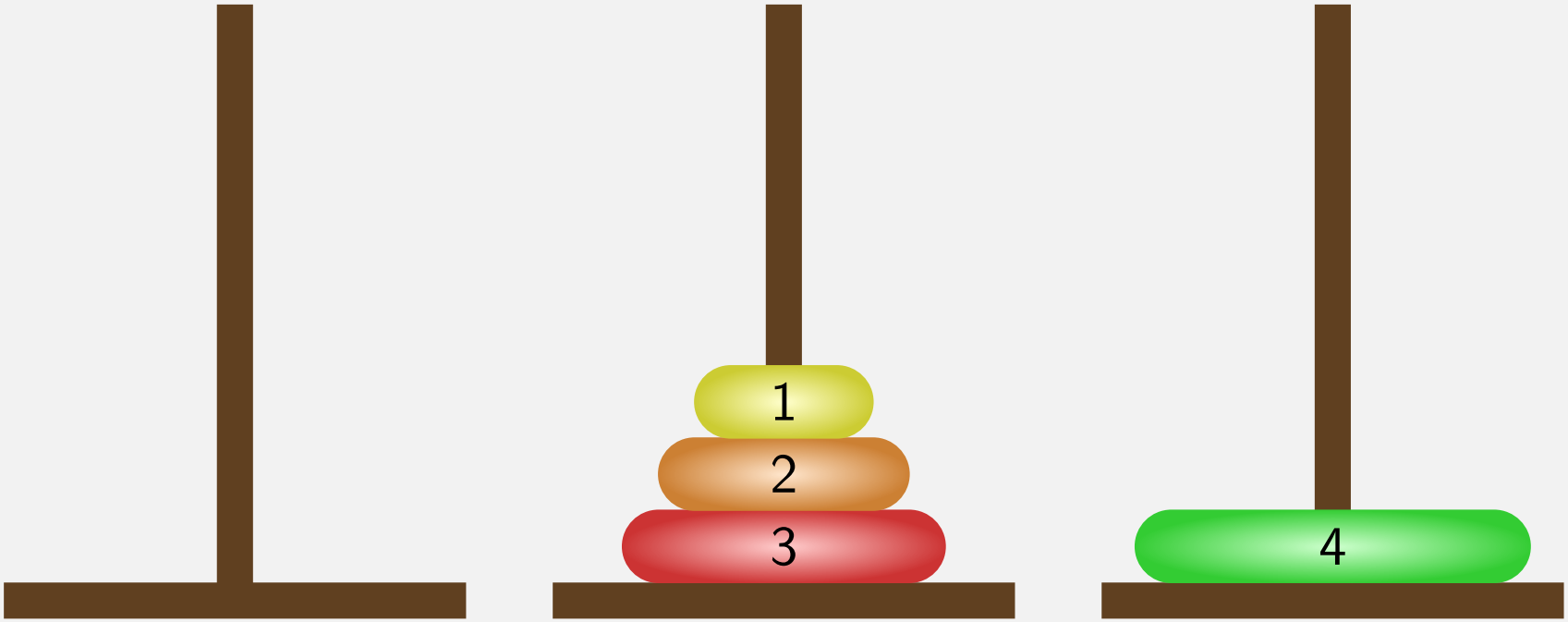
Mozgatott korong **3** rúdról **2** rúdra.

Hanoi tornyai – 4 Korong



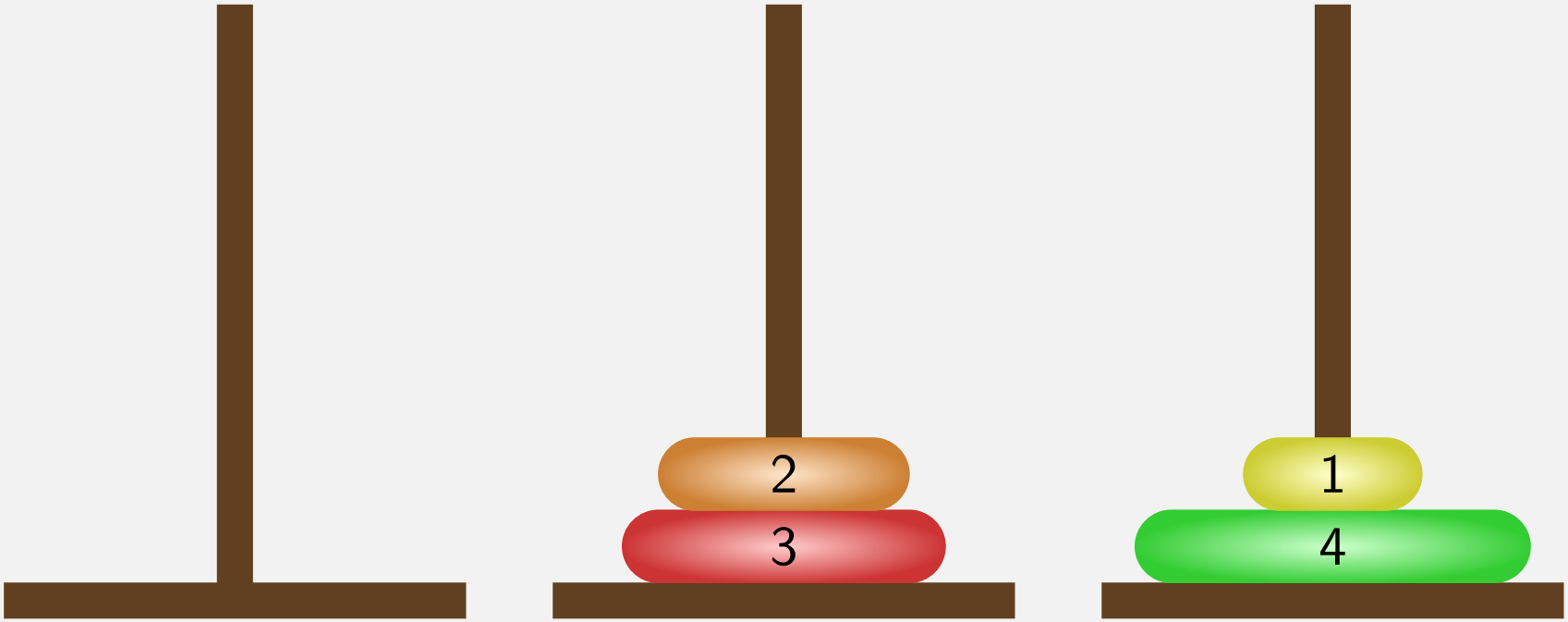
Mozgatott korong **1** rúdról **2** rúdra.

Hanoi tornyai – 4 Korong



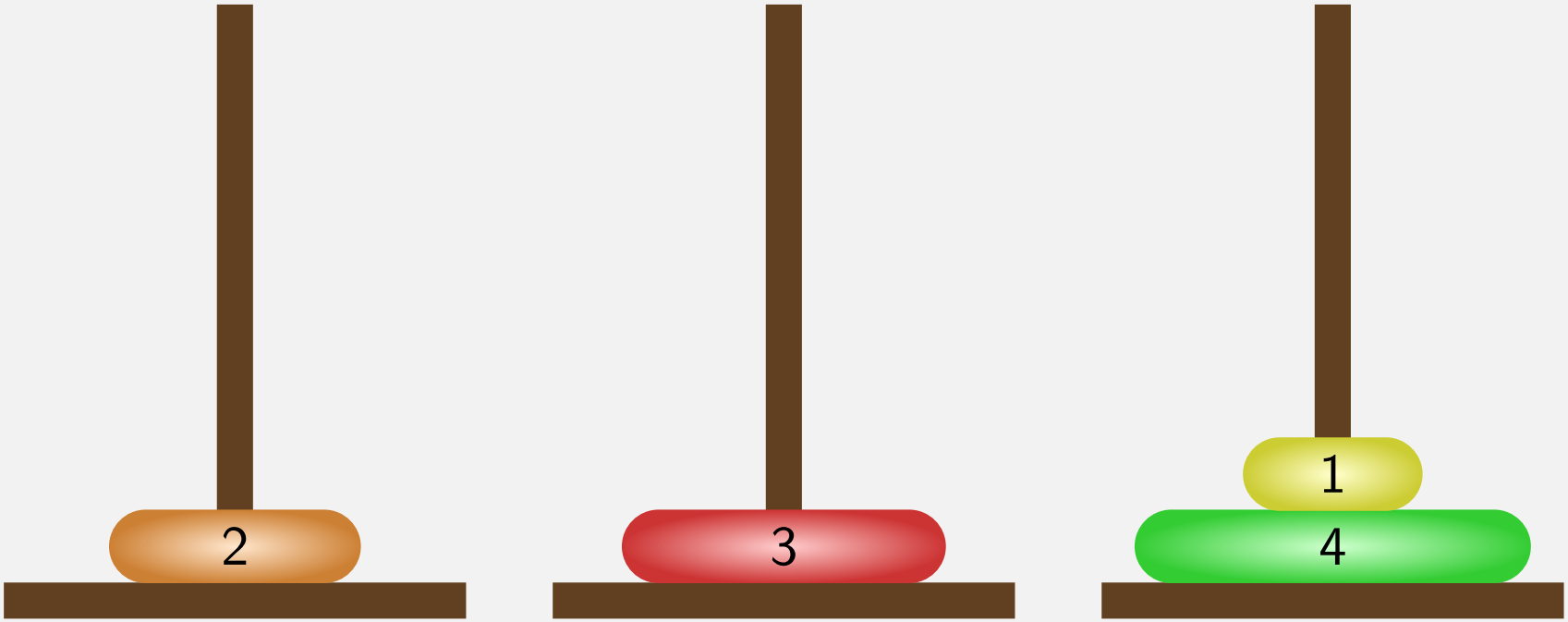
Mozgatott korong **1** rúdról **3** rúdra.

Hanoi tornyai – 4 Korong



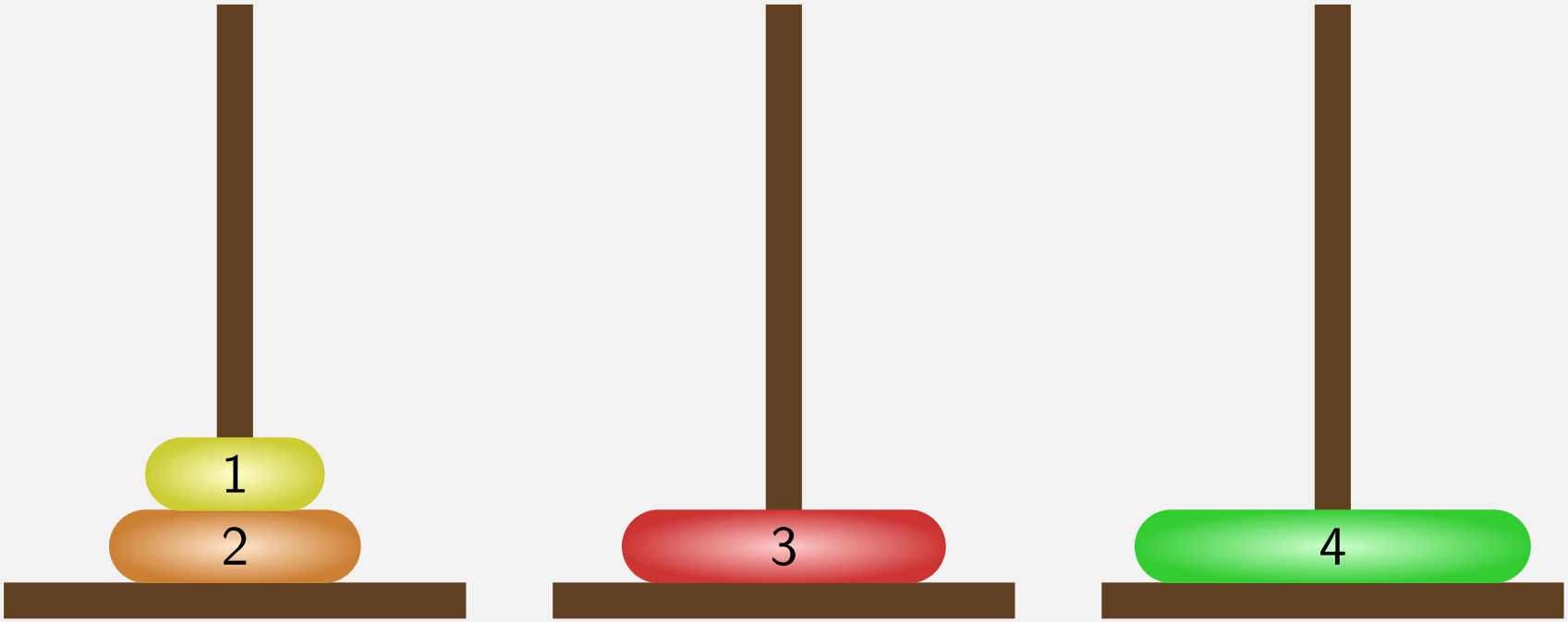
Mozgatott korong **2** rúdról **3** rúdra.

Hanoi tornyai – 4 Korong



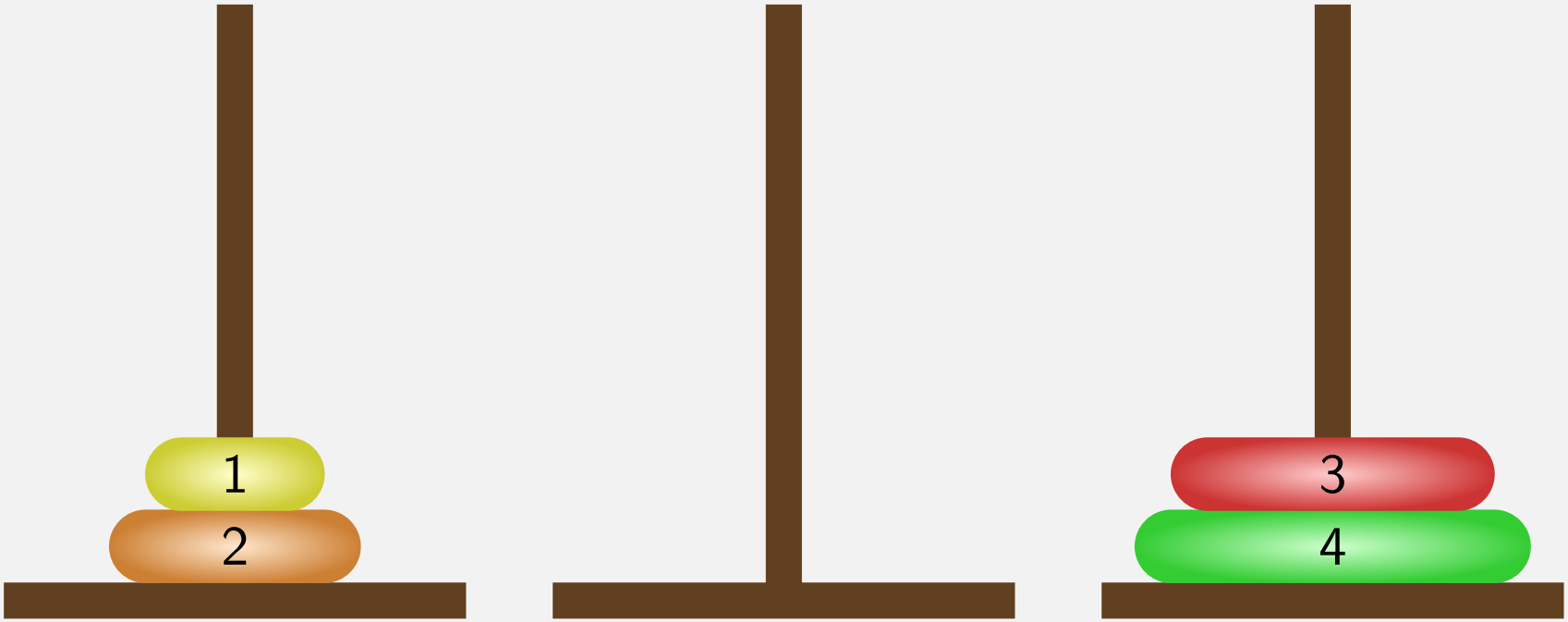
Mozgatott korong **2** rúdról **1** rúdra.

Hanoi tornyai – 4 Korong



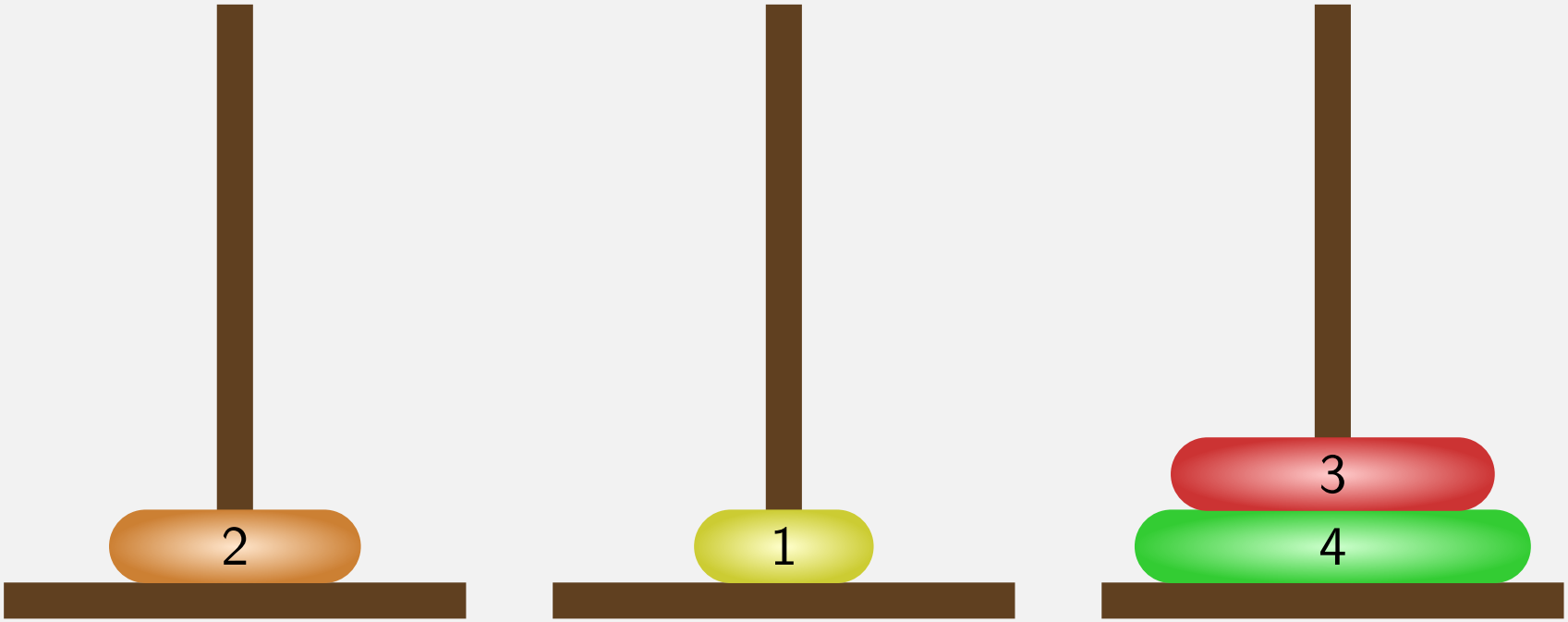
Mozgatott korong **3** rúdról **1** rúdra.

Hanoi tornyai – 4 Korong



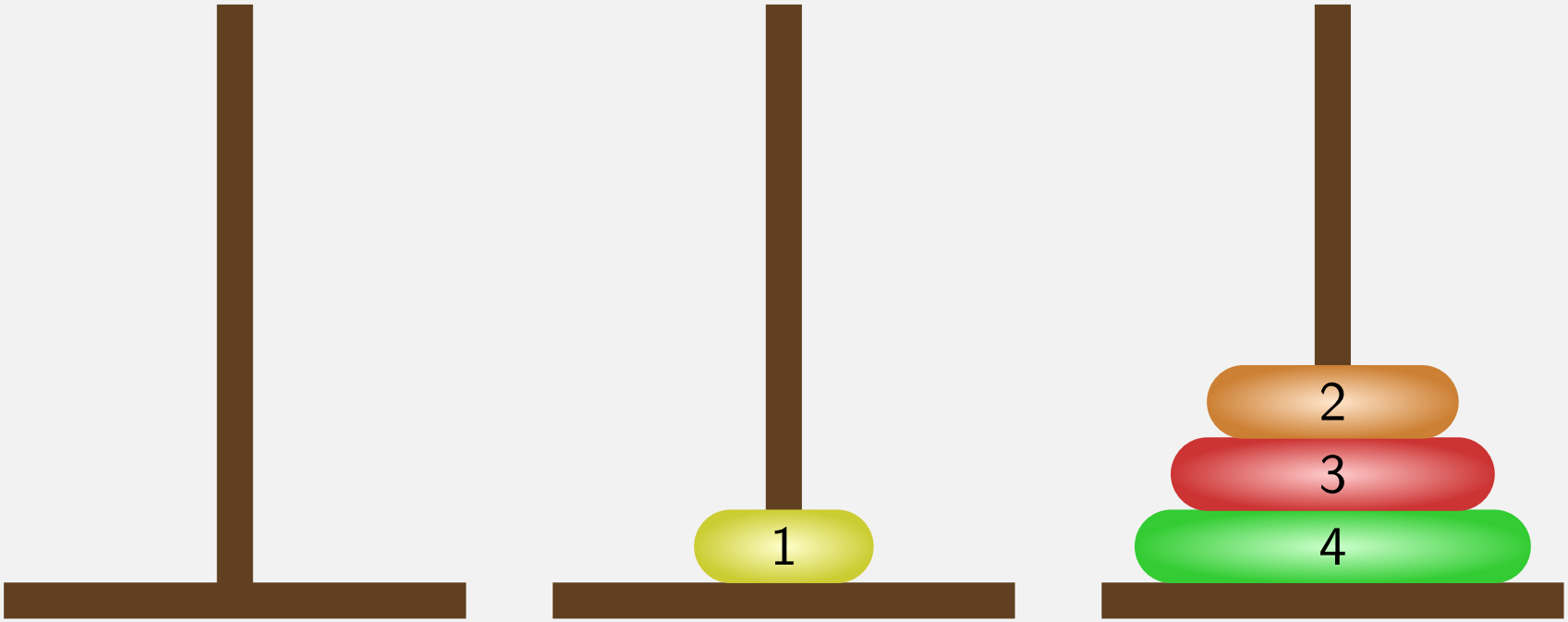
Mozgatott korong **2** rúdról **3** rúdra.

Hanoi tornyai – 4 Korong



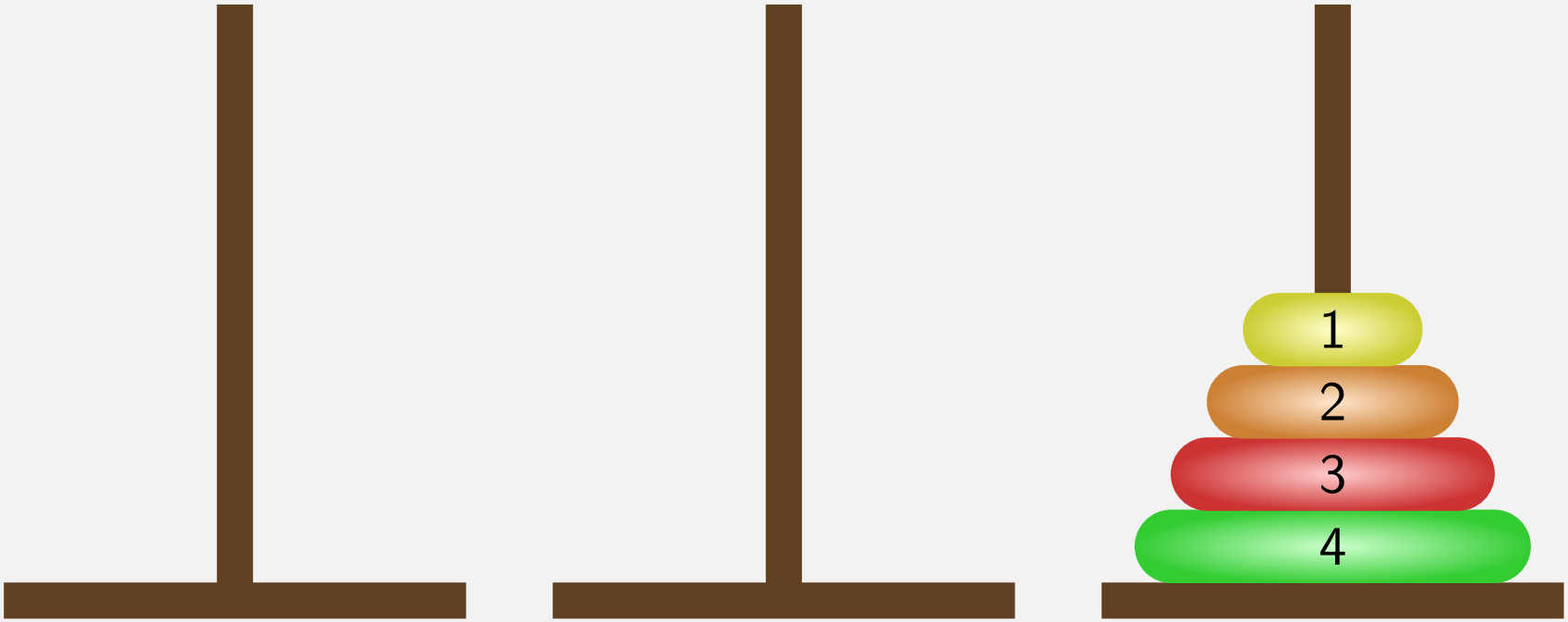
Mozgatott korong **1** rúdról **2** rúdra.

Hanoi tornyai – 4 Korong



Mozgatott korong **1** rúdról **3** rúdra.

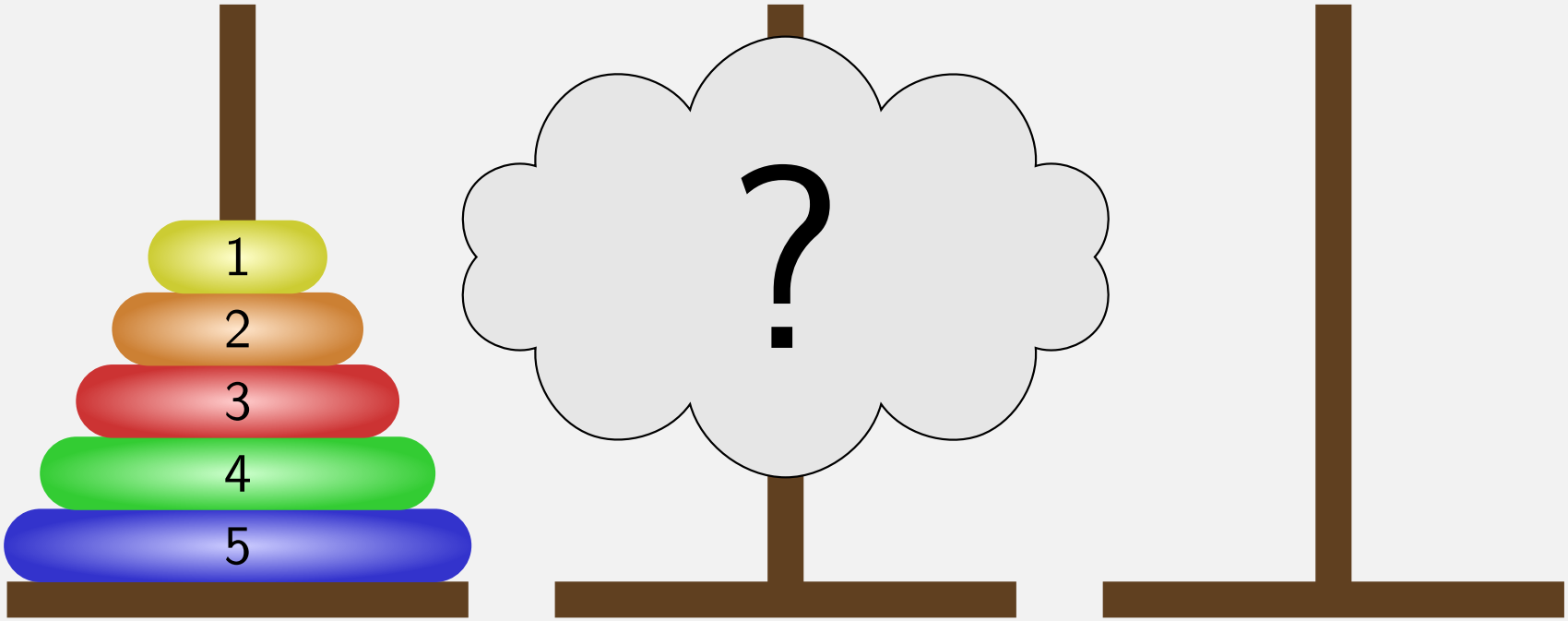
Hanoi tornyai – 4 Korong



Mozgatott korong **2** rúdról **3** rúdra.

Hanoi tornyai – 4 Korong

Hanoi tornyai – 5 Korong



Hanoi tornyai

Megoldási ötlet

- Ha $N-1$ korongot át tudnánk mozgatni a középső rúdra, akkor az N-edik korong ezután már áttehető a jobbszélső rúdra, majd a középső rúdról kell az $N-1$ korongot a jobbszélsőre tenni.
- Ez egy rekurzív gondolat!
- A rekurzió biztos leáll, mert a legkisebb korong mindig felül van, és bárhonnan bárhova mozgatható.

Algoritmus

Eljárás Hanoi(N, *Forras*, *Cel*, *Seged*)
  Ha $N > 0$ **akkor**
    Hanoi($N-1$, *Forras*, *Seged*, *Cel*)
    Ki: N, *Forras*, *Cel*
    Hanoi($N-1$, *Seged*, *Cel*, *Forras*)
  Elágazás vége
Eljárás vége

Rekurzív algoritmusok

Ötlet a rekurzív megvalósításhoz (összegzés példáján)

- Vegyük az A sorozat utolsó (N-edik) elemét. Ezt az elemet kell hozzáadni az összes előtte lévő összegéhez.
- Az $N - 1$ darab elem összegét a rekurzívan meghívott függvény majd szolgáltatja.
- A rekurzió leállása: Ha $N = 0$, akkor adjon vissza a függvény 0-t.

Rekurzív megvalósítás

Függvény Sorozatszámítás(A, N)
  Ha $N = 0$ **akkor**
    return(R_0)
  Különben
    return($A[N]$ művelet Sorozatszámítás($A[1..N-1]$,$N-1$))
  Elágazás vége
Függvény vége

Eljárás Sorozatszámítás(A, N, R)
  $R \leftarrow R_0$
  Ciklus $i \leftarrow$ 1-**től** N-**ig**
    $R \leftarrow R$ művelet $A[i]$
  Ciklus vége
Eljárás vége

Megszámlálás

Rekurzív megvalósítás

Függvény Megszámlálás(A, N, T)
  Ha $N = 1$ **akkor**
    Ha $T(A[N])$ **akkor**
      return(1)
    Különben
      return(0)
    Elágazás vége
  Különben
    Ha $T(A[N])$ **akkor**
      return(1 + Megszámlálás(A, $N - 1$, T))
    Különben
      return(Megszámlálás(A, $N - 1$, T))
    Elágazás vége
  Elágazás vége
Függvény vége

Eljárás Megszámlálás(A, N, T, DB)
  $DB \leftarrow 0$
  Ciklus $i \leftarrow$ 1-**től** N-**ig**
    Ha $T(A[i])$ **akkor**
      $DB \leftarrow DB + 1$
    Elágazás vége
  Ciklus vége
Eljárás vége

Maximumkiválasztás

Rekurzív megvalósítás

Függvény Maximumkiválasztás(A, N)
    Ha $N = 1$ **akkor**
        return(N)
    Különben
        $eddigiMax \leftarrow$ Maximumkiválasztás(A, $N - 1$)
        Ha $A[N] > A[eddigiMax]$ **akkor**
            return(N)
        Különben
            return($eddigiMax$)
        Elágazás vége
    Elágazás vége
Függvény vége

Eljárás Maximumiválasztás(A, N, MAX)
  $MAX \leftarrow 1$
  Ciklus $i \leftarrow$ 2-**től** N-**ig**
    Ha $A[i] > A[MAX]$ **akkor**
      $MAX \leftarrow i$
    Elágazás vége
  Ciklus vége
Eljárás vége

Lineáris keresés

Ötlet a rekurzív megvalósításhoz

- Tegyük fel, hogy nem rendezett a sorozatunk
- Ha az X sorozat első eleme nem egyezik meg a keresett Y-nal, akkor hívjuk meg ismét a függvényt, de már csak a másodiktól az N-edik elemig terjedő részsorozattal.
- E jelölje a vizsgálandó részsorozat első elemének indexét, U pedig az utolsó elem indexét
- A függvény visszatérési értéke legyen 0, ha Y nincs benne X-ben, egyéb esetben pedig az X-beli indexe annak az elemnek, amely értéke egyenlő Y-nal

Lineáris keresés

Rekurzív megvalósítás

Függvény Keresés(X, E, U, Y)
    Ha $E > U$ **akkor**
        return(0)
    Különben
        Ha $X[E] = Y$ **akkor**
            return(E)
        Különben
            return(Keresés(X, $E + 1$, U, Y)
        Elágazás vége
    Elágazás vége
Függvény vége