

Data Structures I: Recursion



Mauricio Toro
Department of Systems and Informatics
Universidad EAFIT

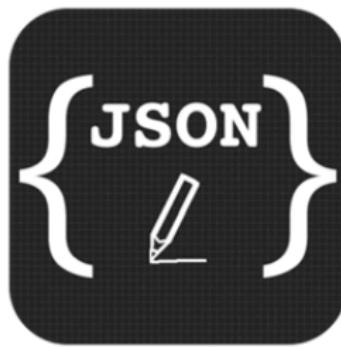
Cocktail of the day: Grasshopper



Disclaimer: Keep alcohol out of the hands of minors.

- 30 ml Crème de menthe
- 30 ml Crème de cacao
- 30 ml Fresh cream





<https://www.youtube.com/watch?v=vrPBtQAxw7c>

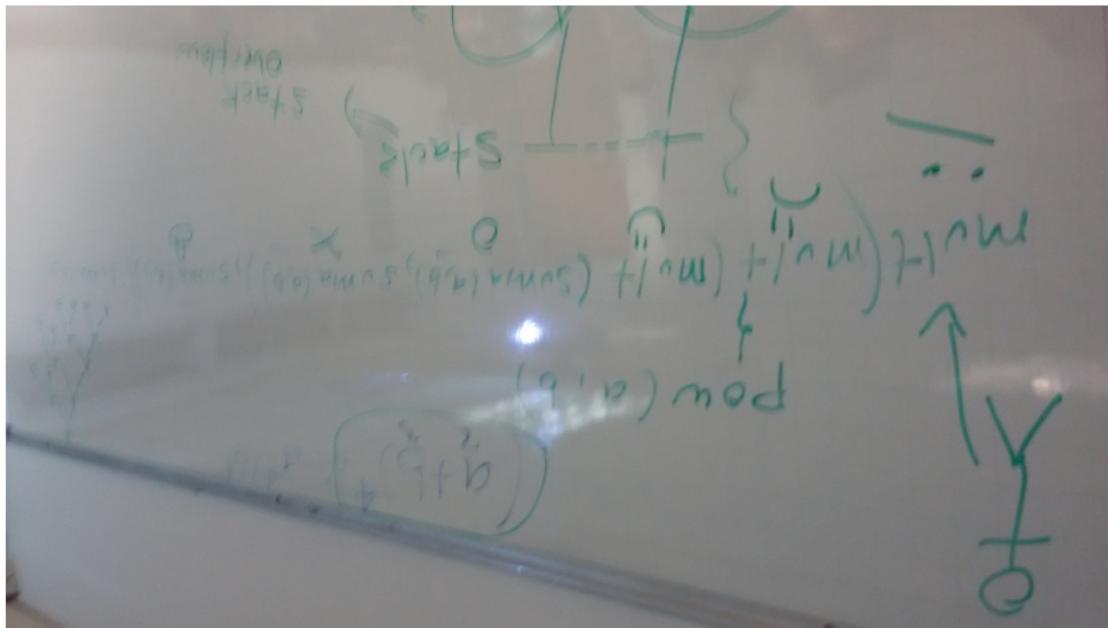


Figure: Function call to perform $(a + b)^4$

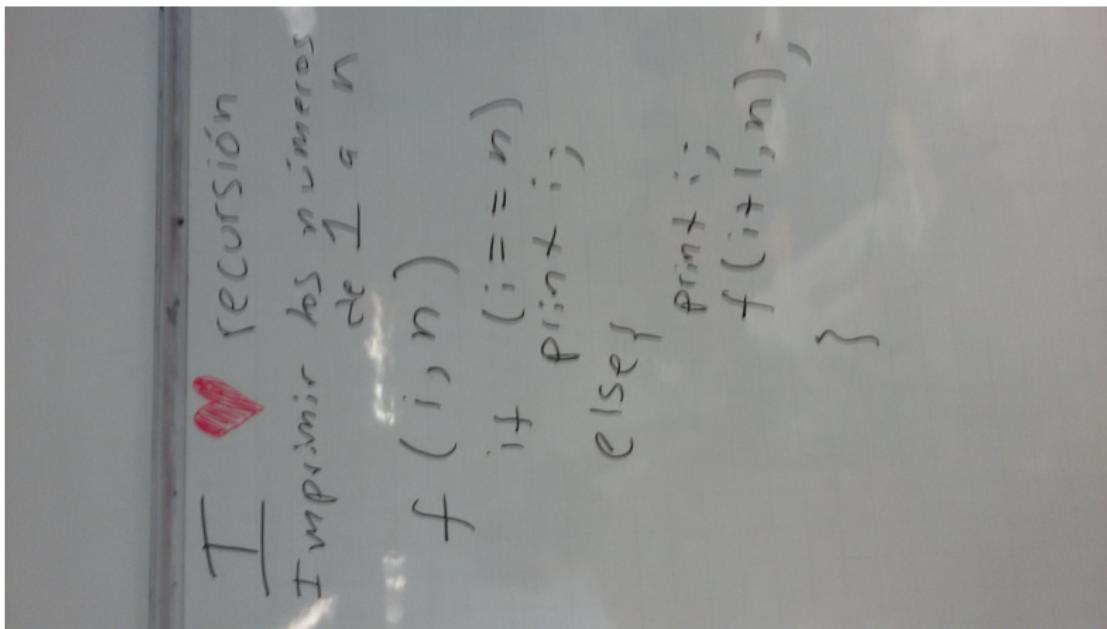


Figure: Print numbers recursively

Sum numbers recursively

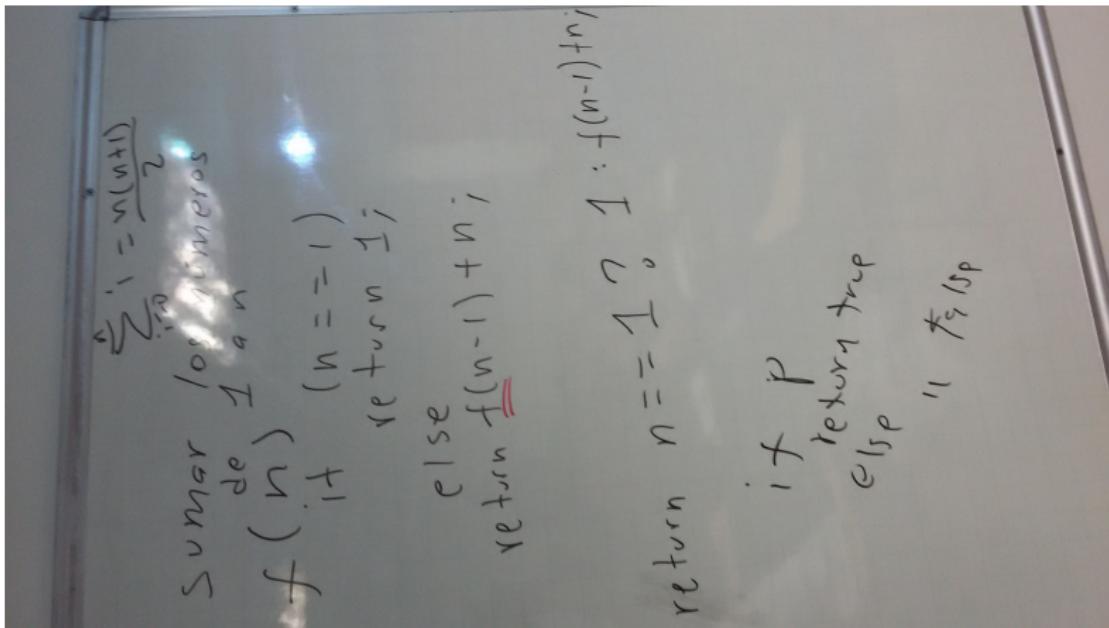
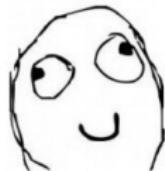


Figure: Sum numbers recursively

$$\frac{7!}{4! \cdot 3!}$$

How other people read it:



7 factorial,
divided by 4 factorial,
divided by 3 factorial.

How I read it:



7!!!
4!!!!
3!!!!

Figure: Comic about factorial

```
factorial(5)
    factorial(4)
        factorial(3)
            factorial(2)
                factorial(1)
                    return 1
                return 2*1 = 2
            return 3*2 = 6
        return 4*6 = 24
    return 5*24 = 120
```

Simulation: <http://visualgo.net/recursion.html>

```
public static int factorial(int N) {  
    if (N == 1) return 1;  
    return N * factorial(N-1);  
}
```



Figure: Recursion tree of factorial function

- $T(n) = C + T(n - 1)$, $n > 1$
- $T(n)$ is $O(n)$



Figure: Euclid, the inventor of Euclid's algorithm

- The greatest common divisor (gcd) of two positive integers is the largest integer that divides evenly into both of them.
- The gcd of 102 and 68 is 34 since both 102 and 68 are multiples of 34.
- We can compute the gcd using the following property, which holds for positive integers p and q :
 - If $p > q$, the gcd of p and q is the same as the gcd of q and $p \% q$.

Taken from

<http://introcs.cs.princeton.edu/java/23recursion/>

- The greatest common divisor (gcd) of two positive integers is the largest integer that divides evenly into both of them.
- The gcd of 102 and 68 is 34 since both 102 and 68 are multiples of 34.
- We can compute the gcd using the following property, which holds for positive integers p and q :
 - If $p > q$, the gcd of p and q is the same as the gcd of q and $p \% q$.

Taken from

<http://introcs.cs.princeton.edu/java/23recursion/>

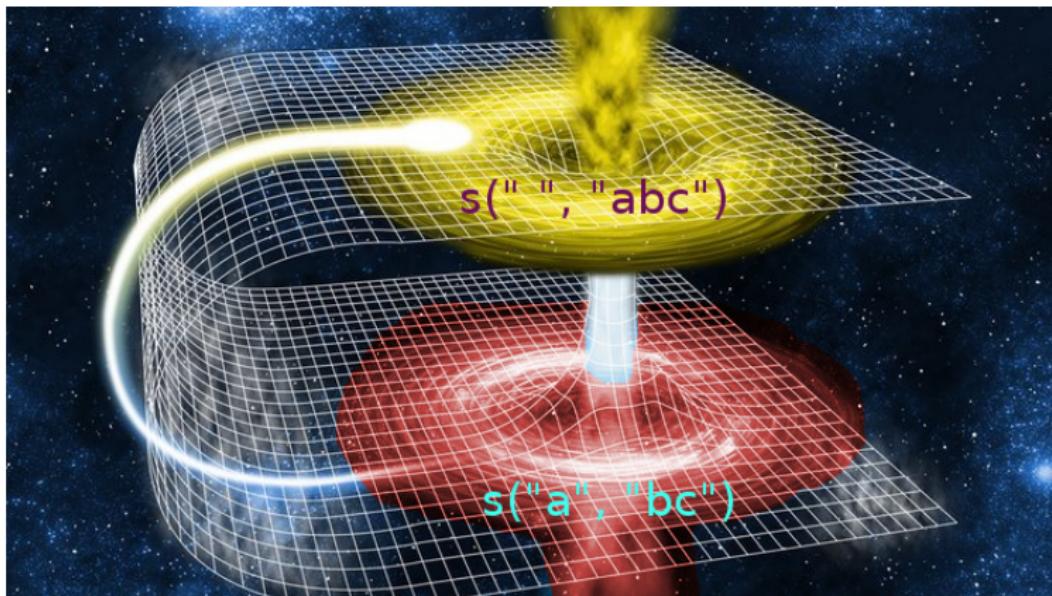


Figure: A universe where “a” belongs to the solution.

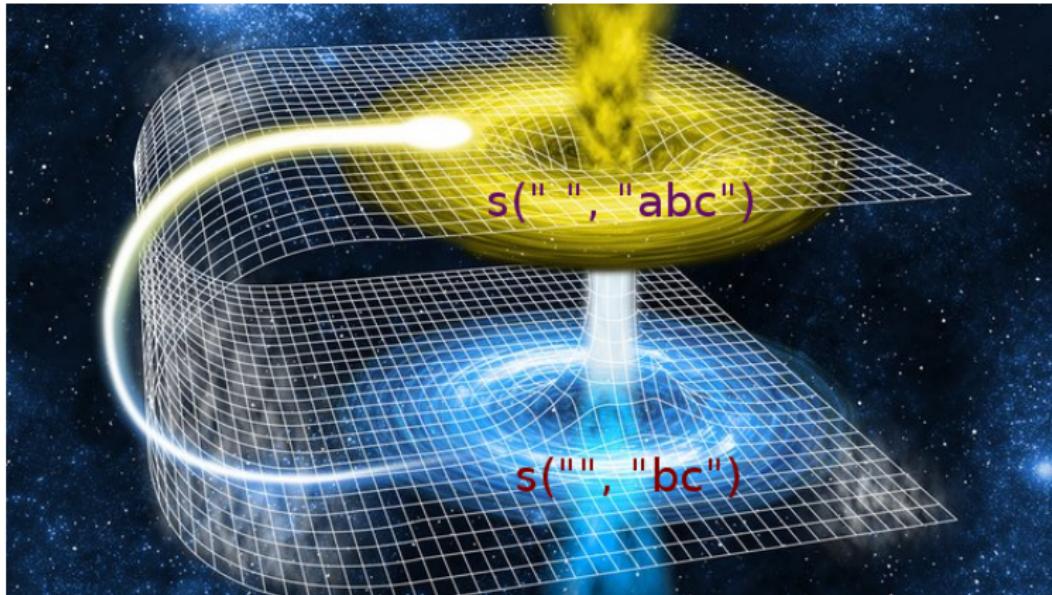


Figure: A universe where “a” does NOT belong to the solution.

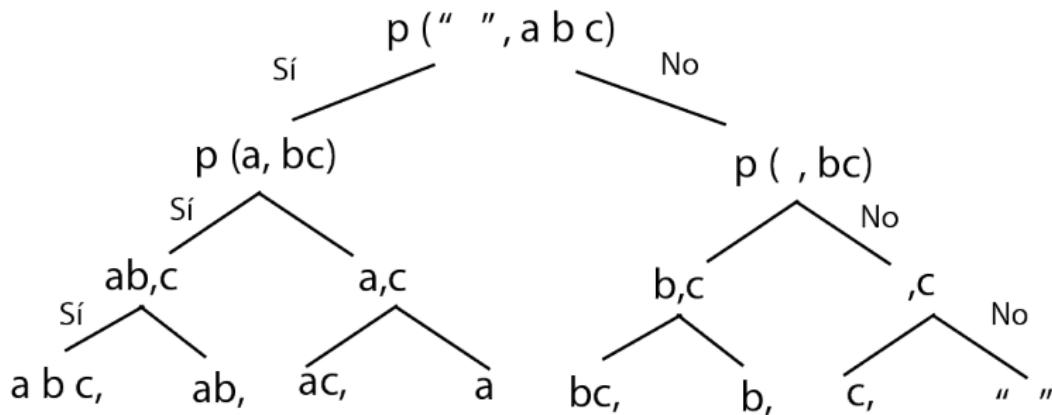


Figure: Recursion tree for subset generation

```
public static int gcd(int p, int q) {  
    if (q == 0) return p;  
    else return gcd(q, p % q);  
}
```

Simulation: <http://visualgo.net/recursion.html>



Figure: Towers of Hanoi

- We have 3 poles and n discs that fit onto the poles.
- The discs differ in size and are initially arranged on one of the poles, in order from largest (disc n) at the bottom to smallest (disc 1) at the top.
- The task is to move the stack of discs to another pole, while obeying the following rules:
 - 1 Move only one disc at a time.
 - 2 Never place a disc on a smaller one.

- One legend says that the world will end when a certain group of monks solved the Towers of Hanoi problem in a temple with 64 golden discs on three diamond needles.

Taken from

<http://introcs.cs.princeton.edu/java/23recursion/>

```
hanoi(topN, A, B, C)
if (topN == 1)
    Move Disk 1 from A to C
else
    hanoi(topN - 1, A, C, B)
    Move Disk topN from A to C
    hanoi(topN - 1, B, A, C)
```



Taken from

<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Moved disc from pole 1 to pole 3.

Taken from

<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



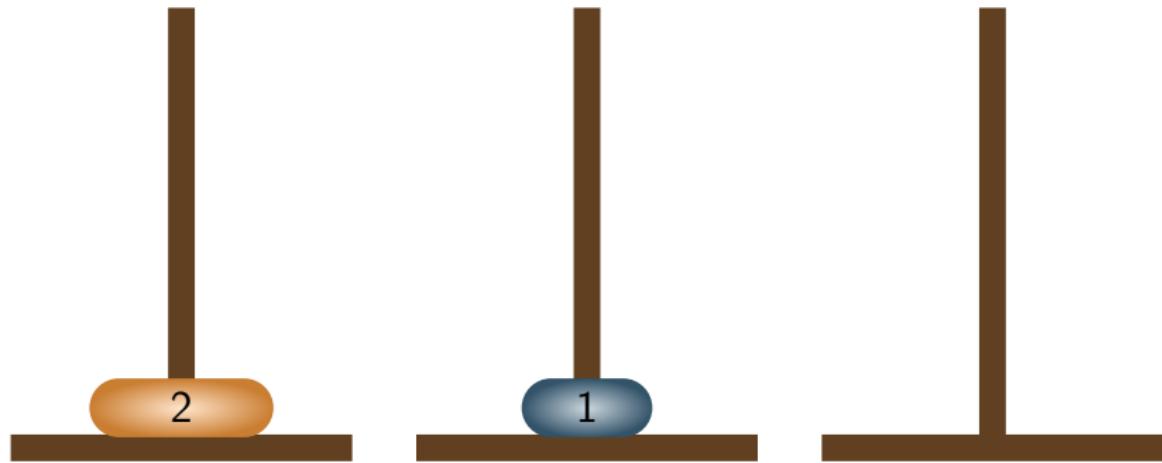
Taken from

<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Taken from

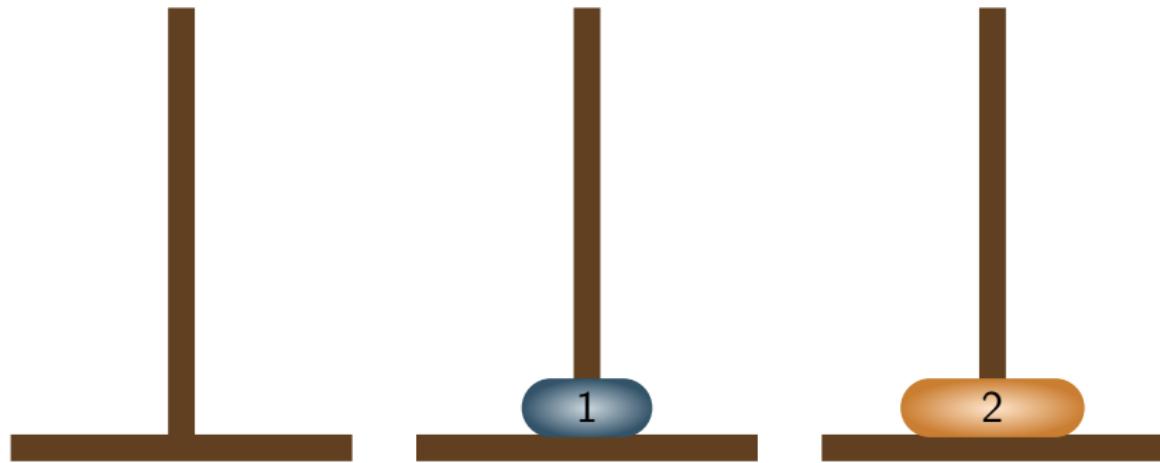
<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Moved disc from pole 1 to pole 2.

Taken from

<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Moved disc from pole 1 to pole 3.

Taken from

<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Moved disc from pole 2 to pole 3.

Taken from

<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Taken from

<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Taken from

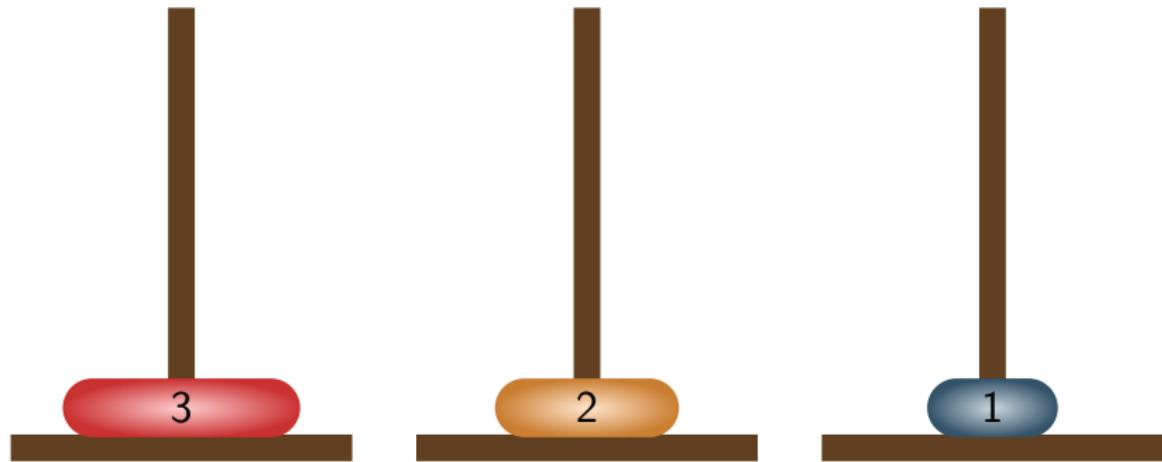
<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Moved disc from pole 1 to pole 3.

Taken from

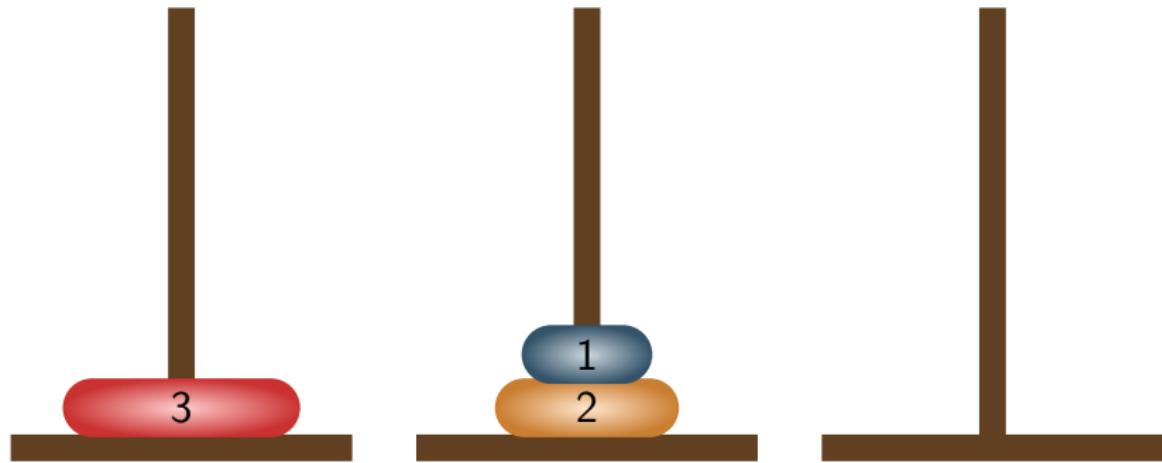
<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Moved disc from pole 1 to pole 2.

Taken from

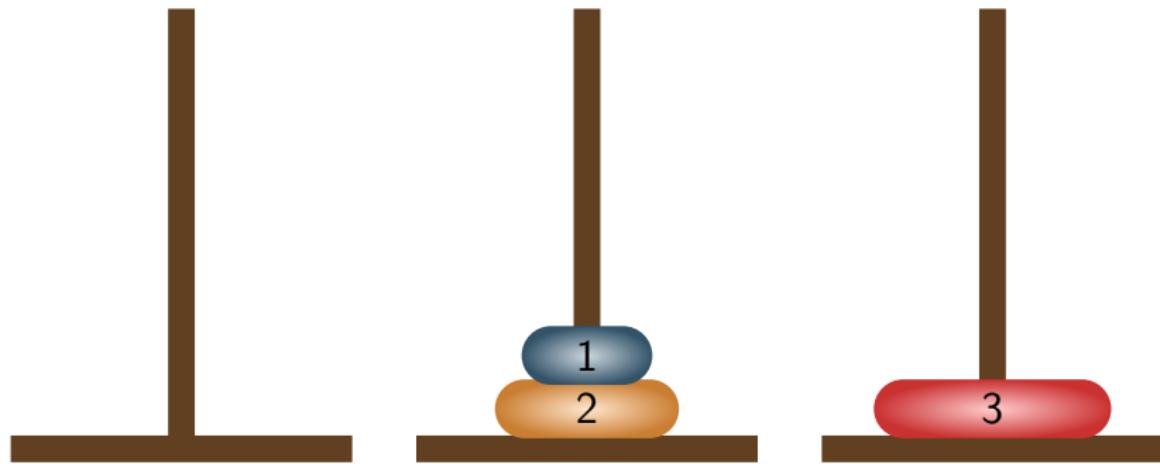
<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Moved disc from pole 3 to pole 2.

Taken from

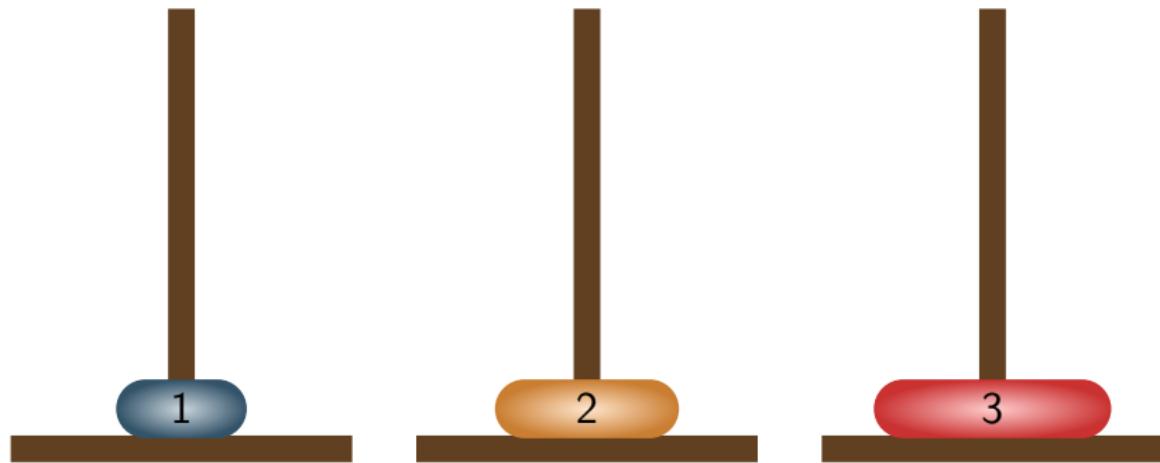
<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Moved disc from pole 1 to pole 3.

Taken from

<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Moved disc from pole 2 to pole 1.

Taken from

<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Moved disc from pole 2 to pole 3.

Taken from

<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Moved disc from pole 1 to pole 3.

Taken from

<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



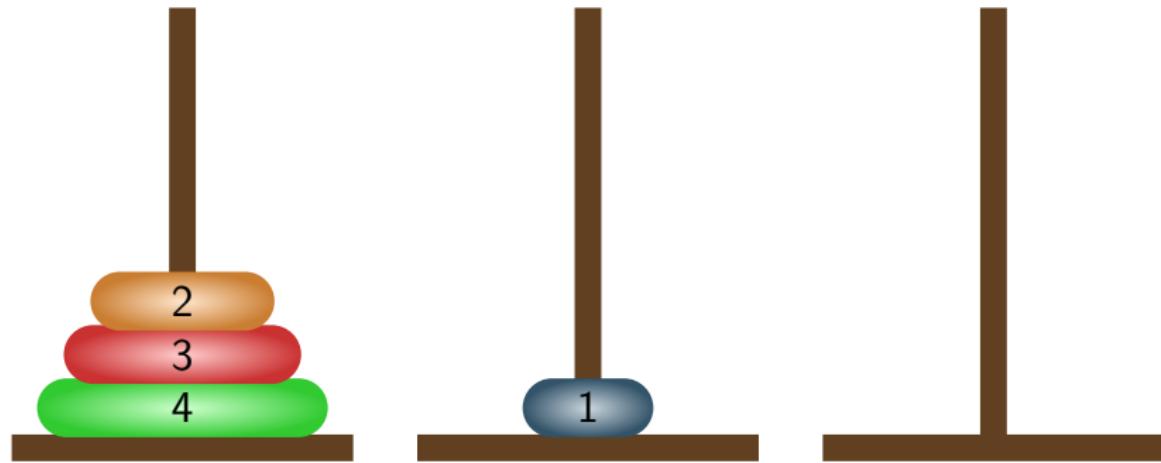
Taken from

<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Taken from

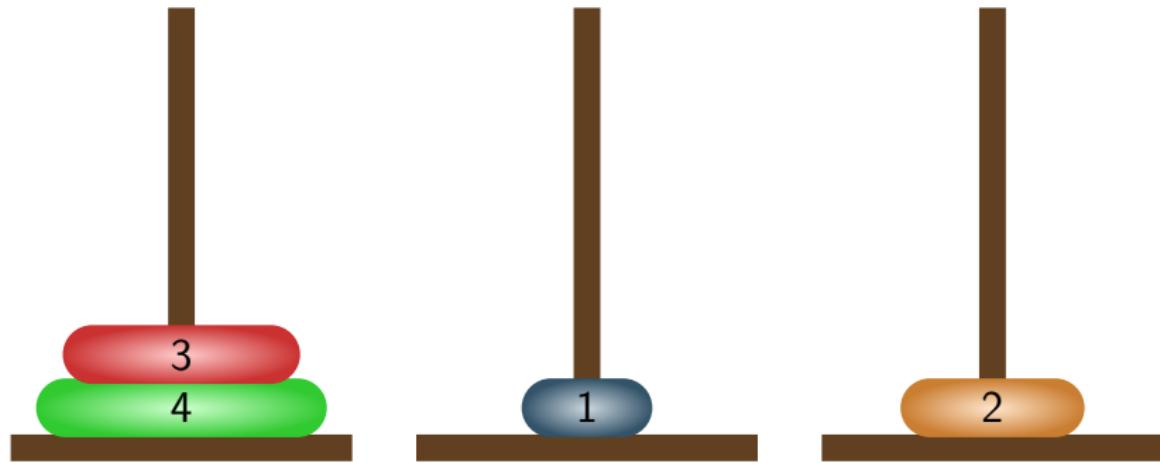
<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Moved disc from pole 1 to pole 2.

Taken from

<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Moved disc from pole 1 to pole 3.

Taken from

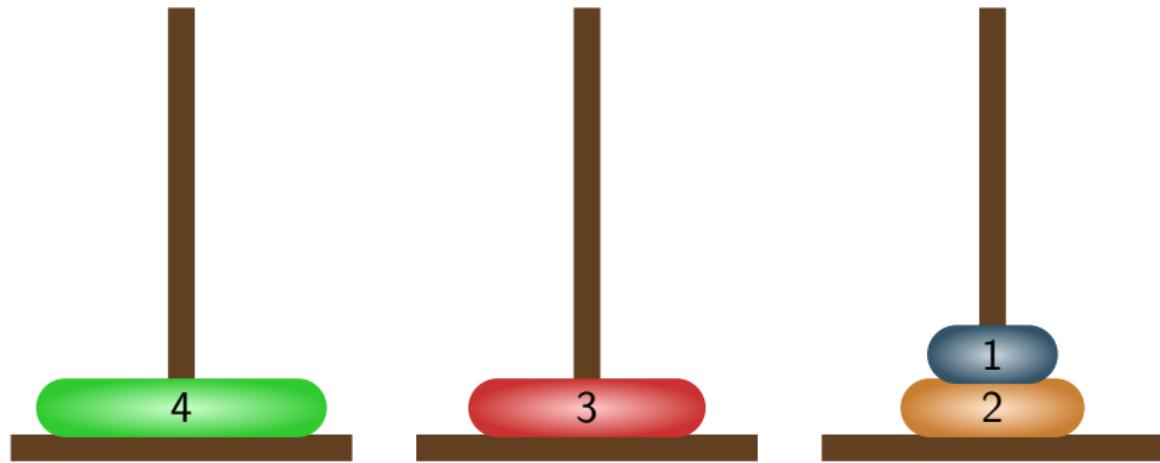
<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Moved disc from pole 2 to pole 3.

Taken from

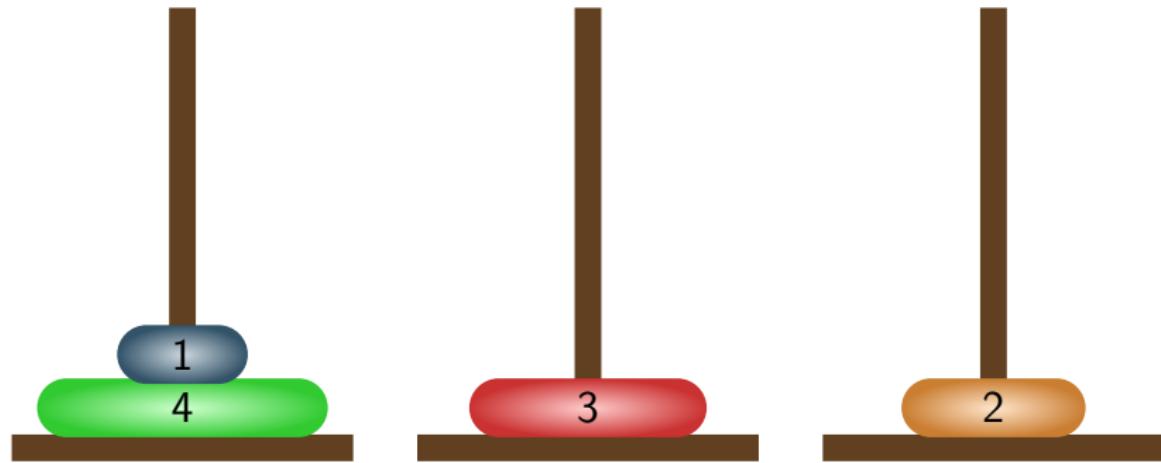
<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Moved disc from pole 1 to pole 2.

Taken from

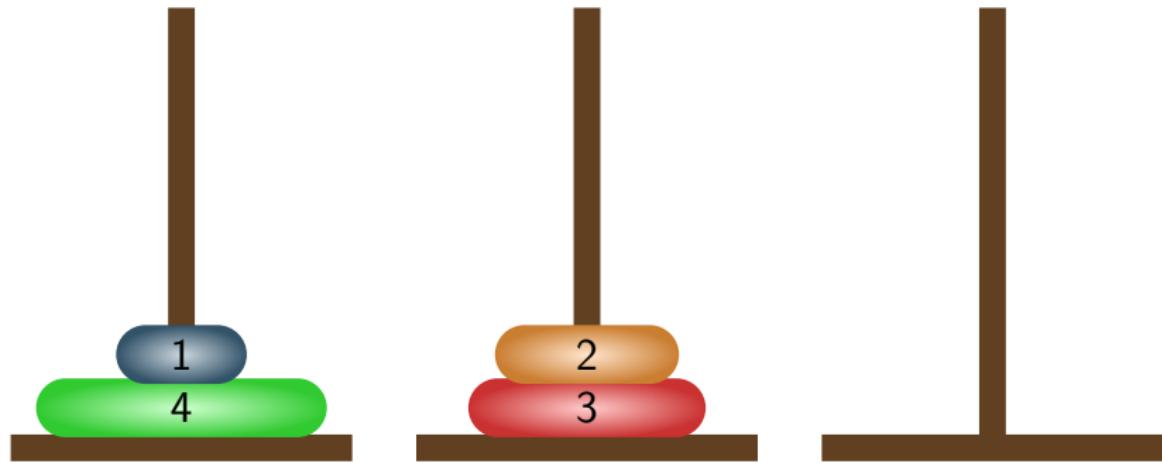
<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Moved disc from pole 3 to pole 1.

Taken from

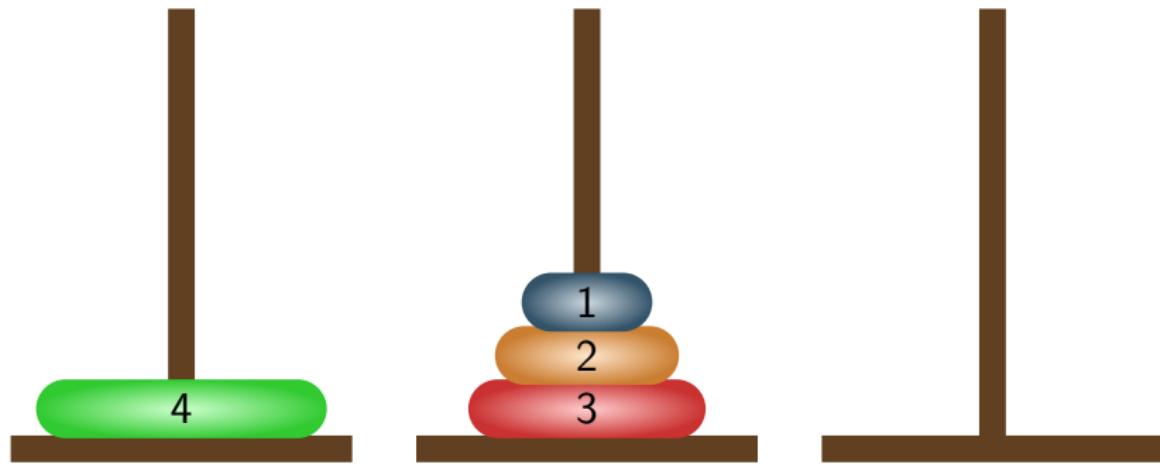
<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Moved disc from pole 3 to pole 2.

Taken from

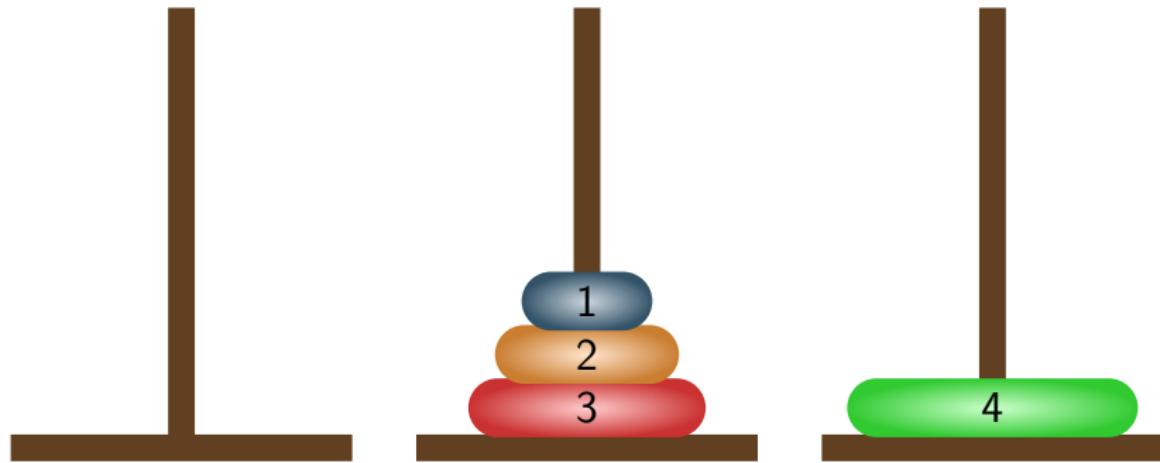
<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Moved disc from pole 1 to pole 2.

Taken from

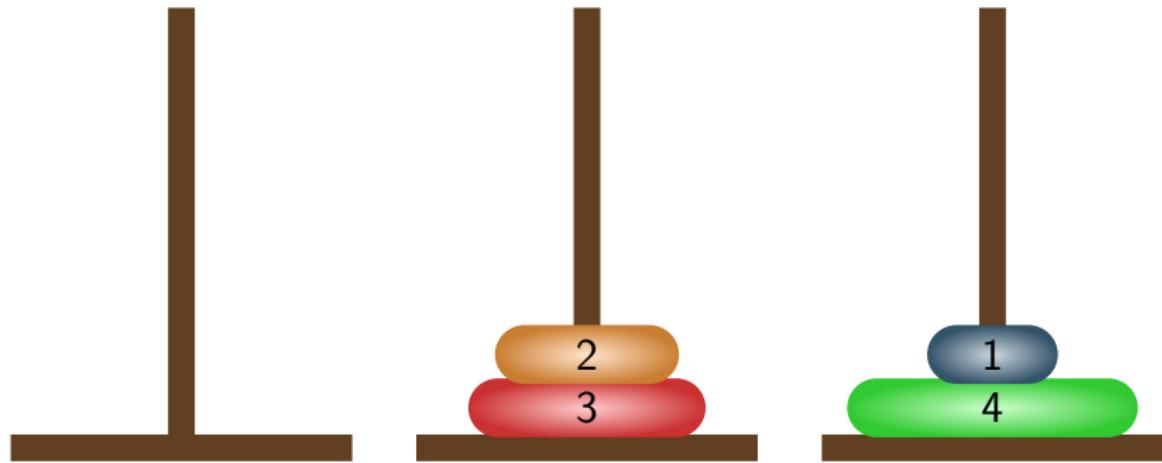
<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Moved disc from pole 1 to pole 3.

Taken from

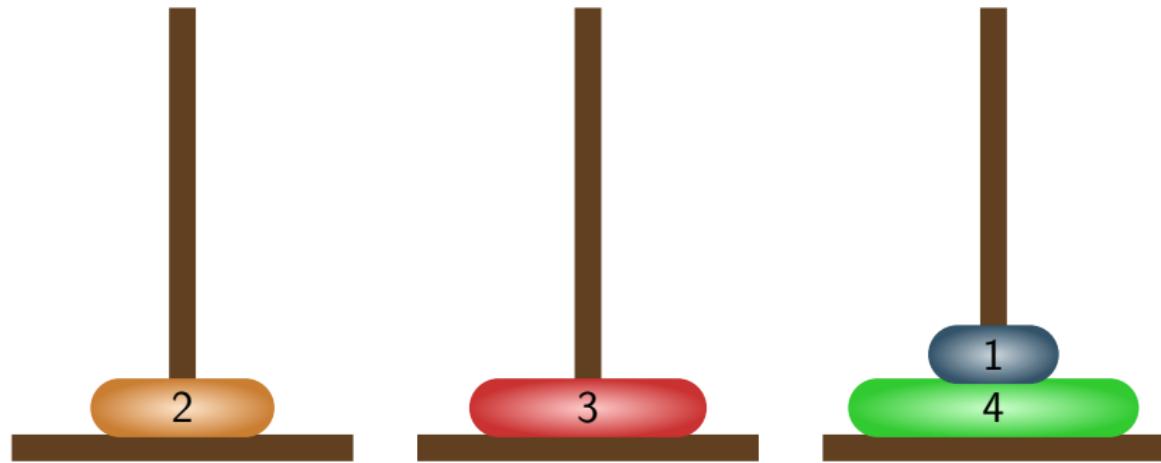
<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Moved disc from pole 2 to pole 3.

Taken from

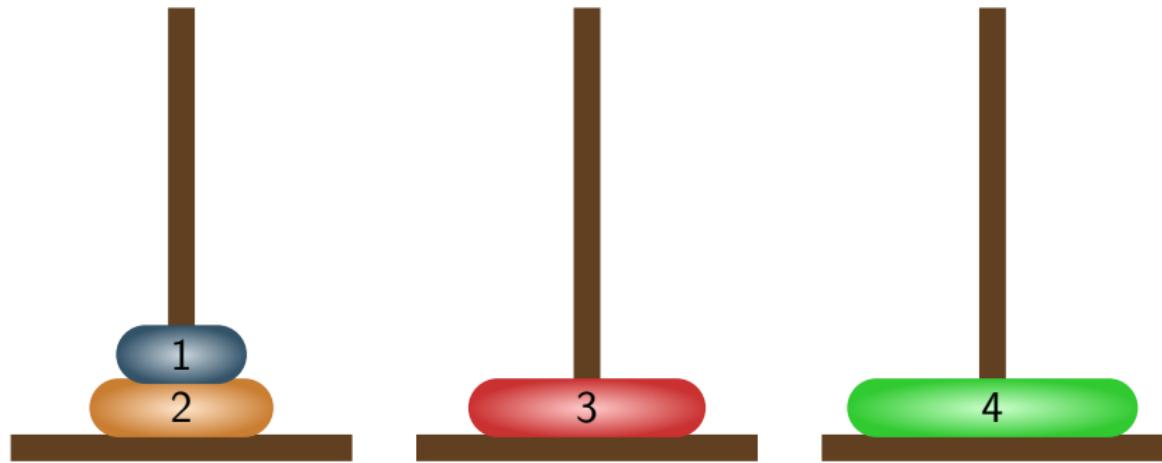
<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Moved disc from pole 2 to pole 1.

Taken from

<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Moved disc from pole 3 to pole 1.

Taken from

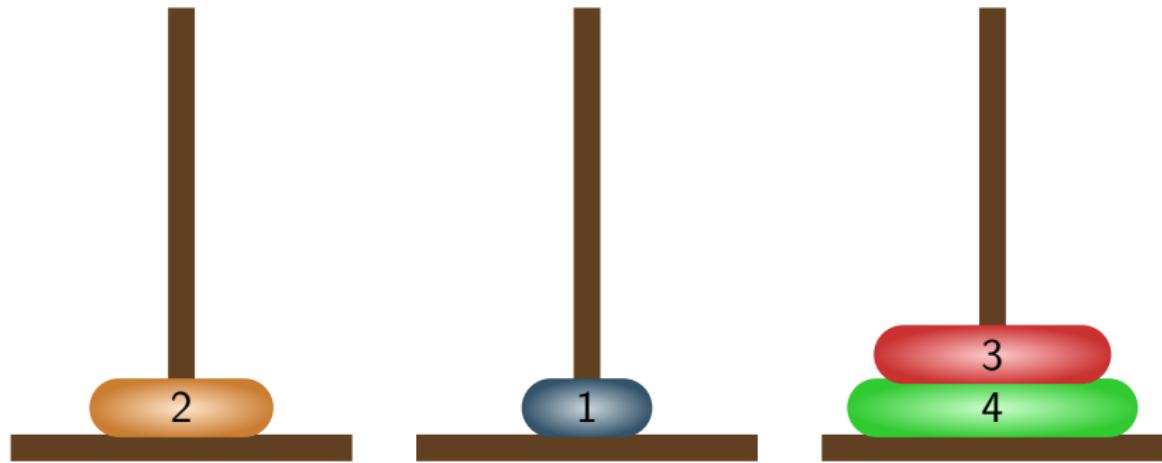
<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Moved disc from pole 2 to pole 3.

Taken from

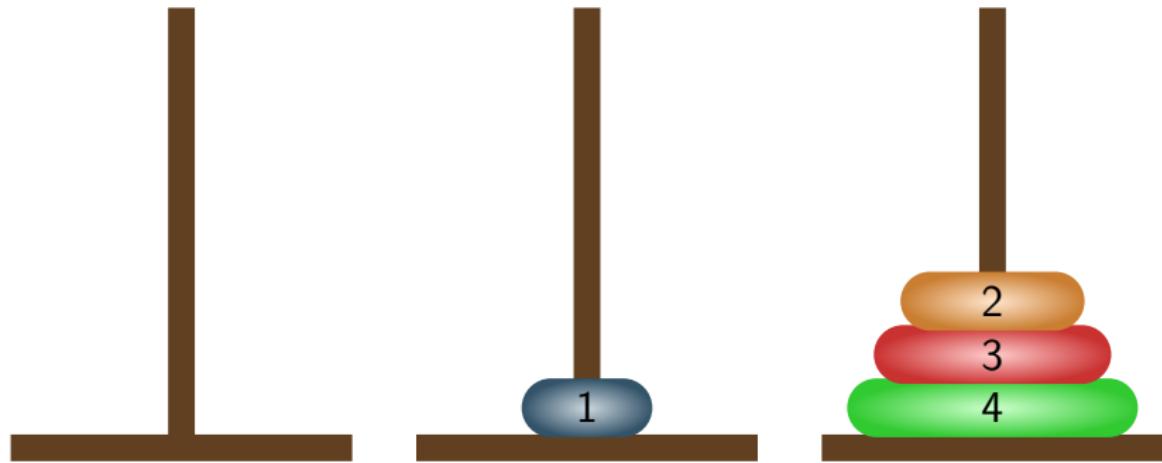
<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Moved disc from pole 1 to pole 2.

Taken from

<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Moved disc from pole 1 to pole 3.

Taken from

<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Moved disc from pole 2 to pole 3.

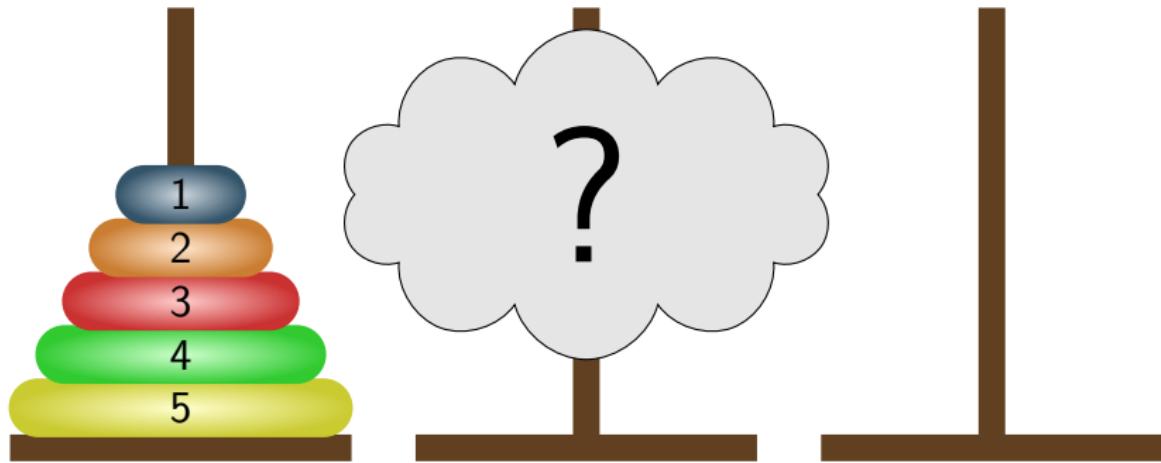
Taken from

<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Taken from

<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Taken from

<http://www.texexample.net/tikz/examples/towers-of-hanoi/>



Figure: Algorithm

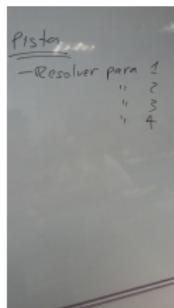


Figure: Algorithm

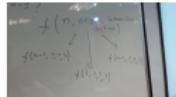


Figure: Algorithm

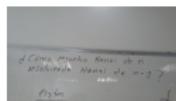


Figure: Algorithm

```
public static void t(int topN, char from,
                     char inter, char to) {
    if (topN == 1)
        print("Disk\u20221\u2022from\u2022" + from + "\u2022to\u2022" + to);
    else {
        t(topN - 1, from, to, inter);
        print("Disk\u2022" + topN + "\u2022from\u2022" + from +
              "\u2022to\u2022" + to);
        t(topN - 1, inter, from, to);
    }
}
```

- $T(1) = 1$
- $T(n) = 2T(n - 1)$, $n > 1$
- $T(n) = 2^n - 1$
- $T(n)$ is $O(2^n)$

- If the monks move discs at the rate of one per second, it would take
 - 1 more than 31 years to finish a 30-disc problem
 - 2 more than 348 centuries for them to finish a 40-disc problem
 - 3 more than 5.8 billion centuries to solve the 64-disc problem

Taken from

<http://introcs.cs.princeton.edu/java/23recursion/>

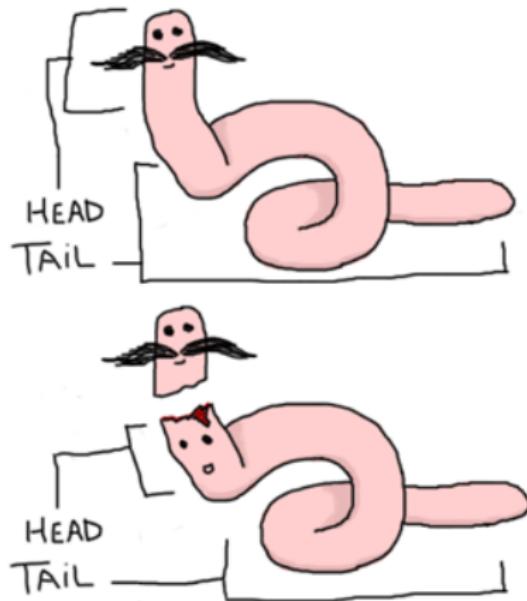


Figure: Tail/Head analogy

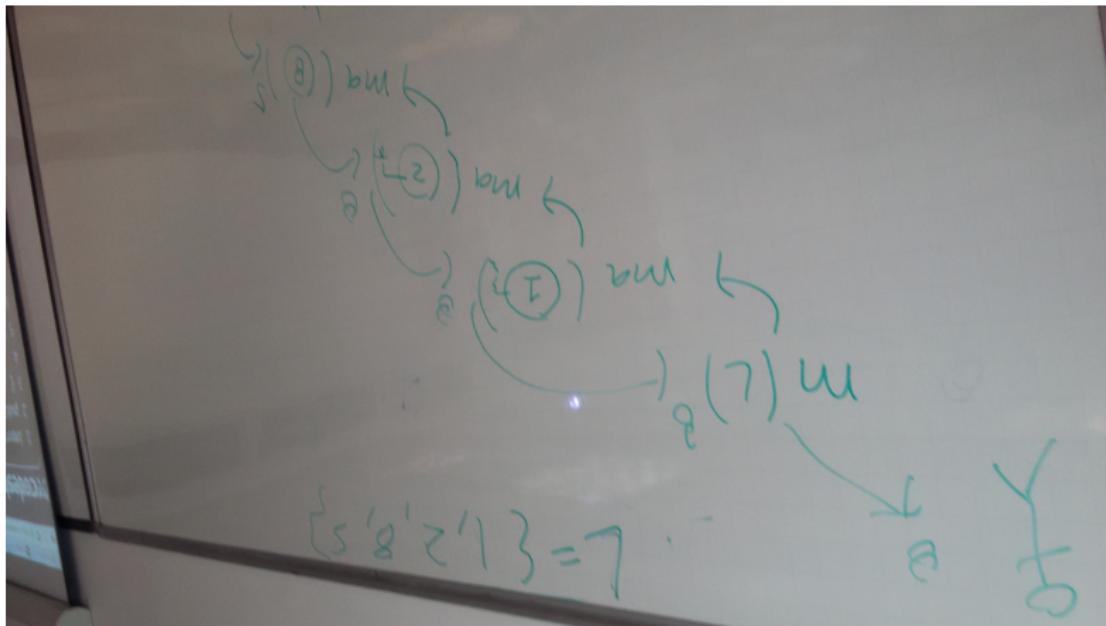


Figure: Example

- A recursive method is a method that call itself.
- The complexity of recursive factorial is $O(n)$.
- The complexity of solving the towers of Hanoi is $O(2^n)$.

- Please learn how to reference images, trademarks, videos and fragments of code.
- Avoid plagiarism

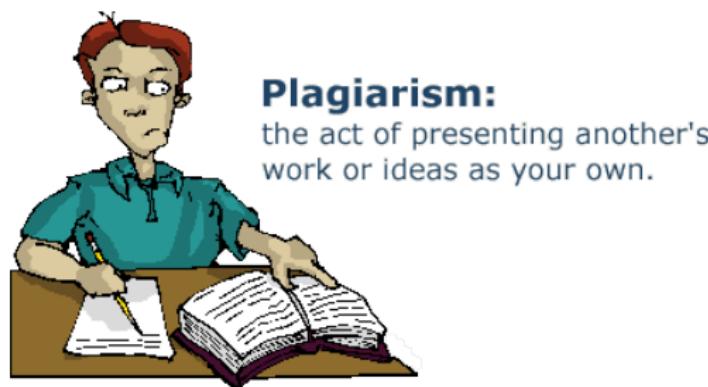


Figure: Figure about plagiarism, University of Malta [Uni09]



University of Malta.

Plagiarism — The act of presenting another's work or ideas as your own, 2009.

[Online; accessed 29-November-2013].

■ Recursion

- <http://introcs.cs.princeton.edu/java/23recursion/>