Recurrent Problems

ITT9131 Konkreetne Matemaatika

Contents

Next section

The Tower of Hanoi

The Tower of Hanoi puzzle was invented by the French mathematician Edouard Lucas in 1883.

Using mathematical induction one can prove that

For the Tower of Hanoi puzzle with $n \geqslant 0$ (and 3 pegs), the minimum number of moves needed

$$T_n = 2^n - 1.$$

Let's look at the example borrowed from Martin Hofmann and Berteun Damman.

The Tower of Hanoi

The Tower of Hanoi puzzle was invented by the French mathematician Edouard Lucas in 1883.

Using mathematical induction one can prove that

For the Tower of Hanoi puzzle with $n \geqslant 0$ (and 3 pegs), the minimum number of moves needed

$$T_n = 2^n - 1.$$

Let's look at the example borrowed from Martin Hofmann and Berteun Damman.

The Tower of Hanoi

The Tower of Hanoi puzzle was invented by the French mathematician Edouard Lucas in 1883.

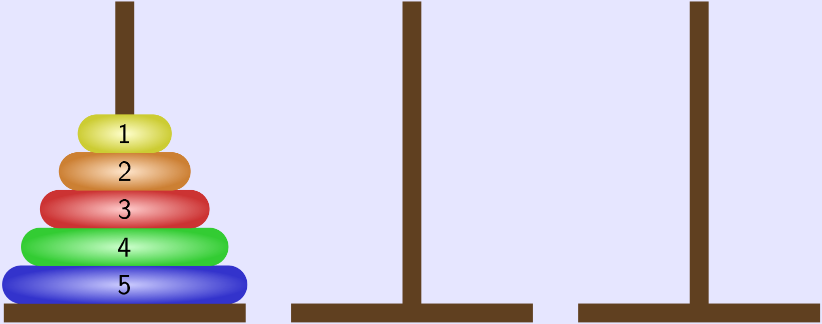
Using mathematical induction one can prove that

For the Tower of Hanoi puzzle with $n \geqslant 0$ (and 3 pegs), the minimum number of moves needed

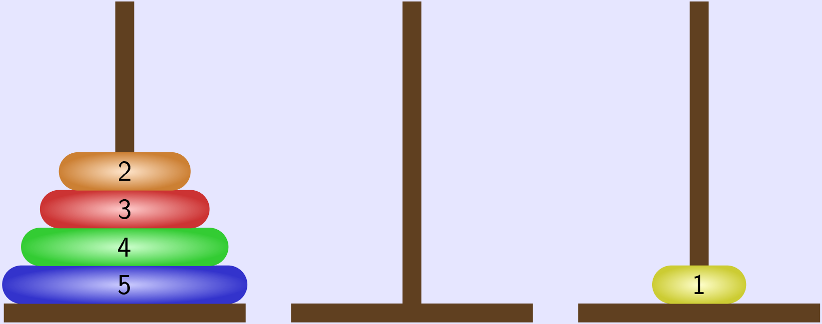
$$T_n = 2^n - 1.$$

Let's look at the example borrowed from Martin Hofmann and Berteun Damman.

Tower of Hanoi – 5 Discs

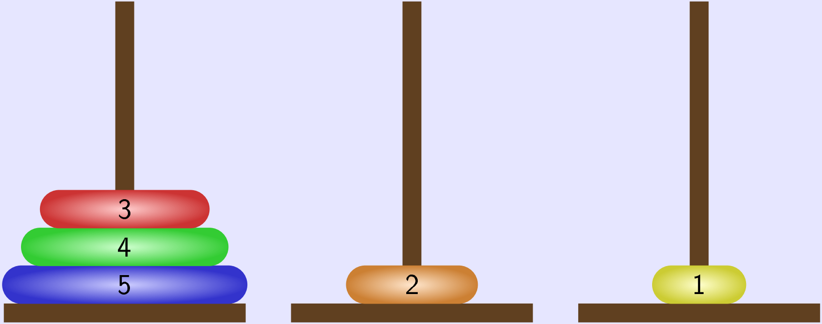


Tower of Hanoi – 5 Discs



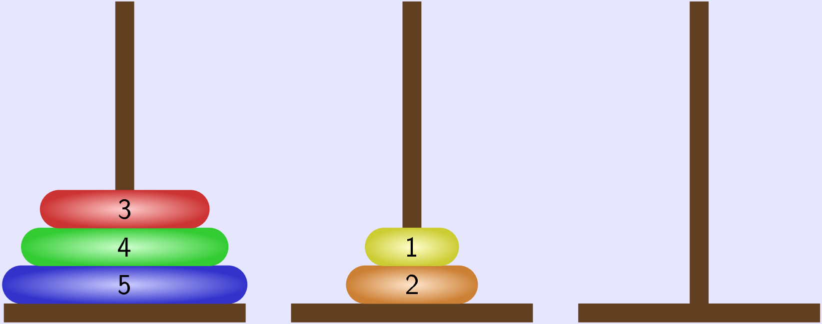
Moved disc from pole 1 to pole 3.

Tower of Hanoi – 5 Discs



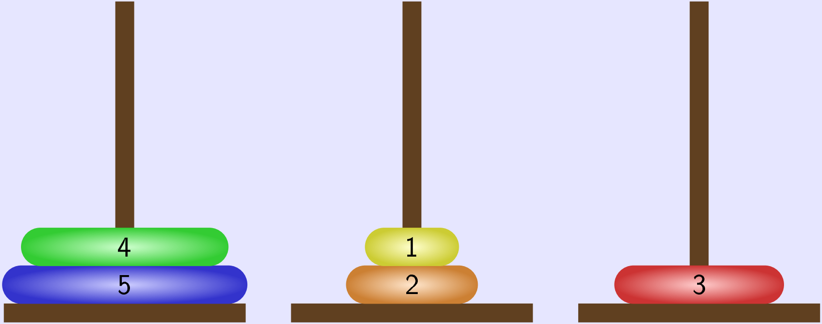
Moved disc from pole 1 to pole 2.

Tower of Hanoi – 5 Discs



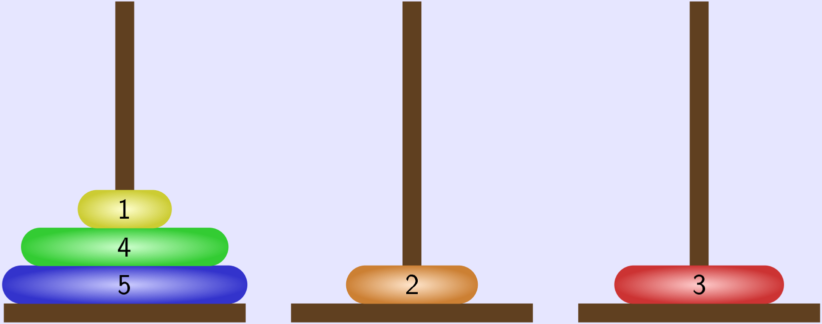
Moved disc from pole 3 to pole 2.

Tower of Hanoi – 5 Discs



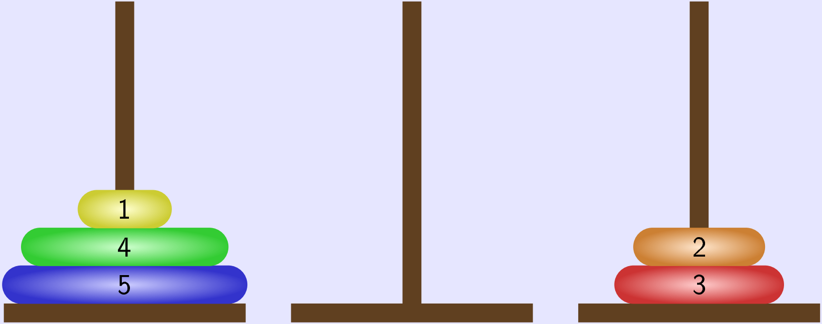
Moved disc from pole 1 to pole 3.

Tower of Hanoi – 5 Discs



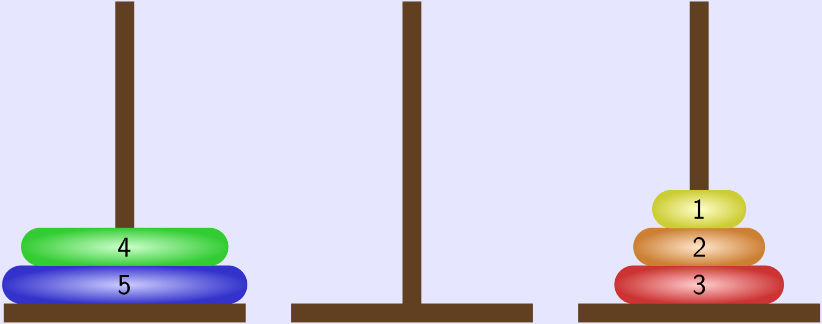
Moved disc from pole 2 to pole 1.

Tower of Hanoi – 5 Discs



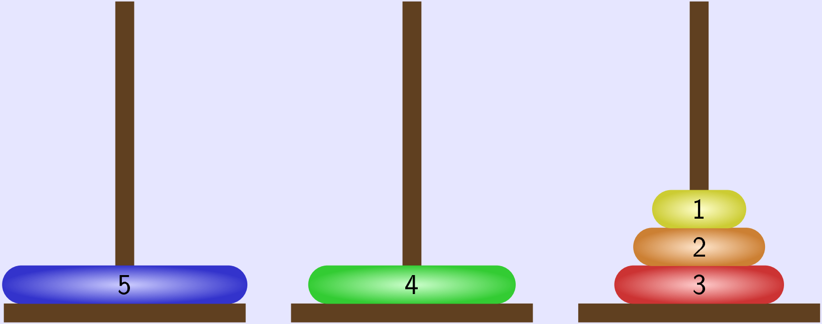
Moved disc from pole 2 to pole 3.

Tower of Hanoi – 5 Discs



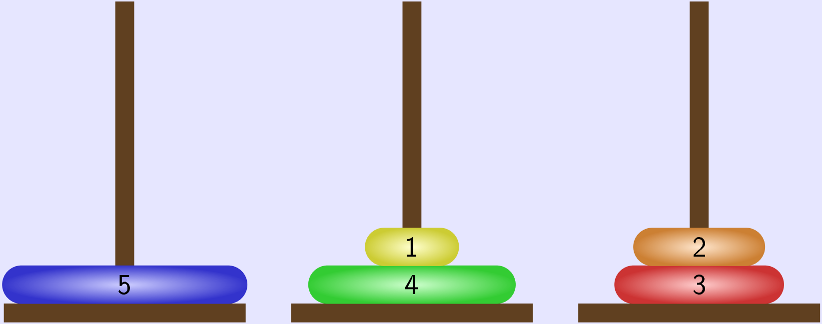
Moved disc from pole 1 to pole 3.

Tower of Hanoi – 5 Discs



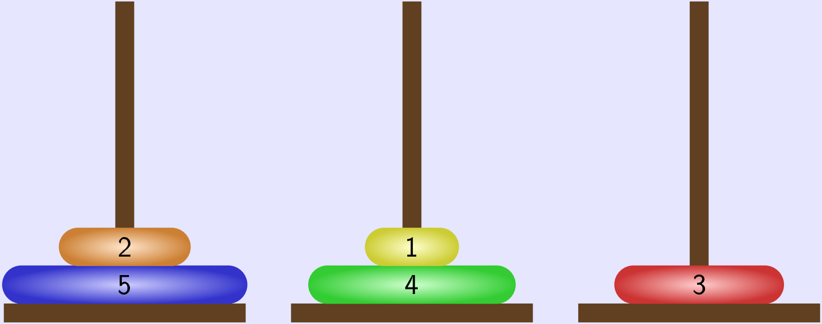
Moved disc from pole 1 to pole 2.

Tower of Hanoi – 5 Discs



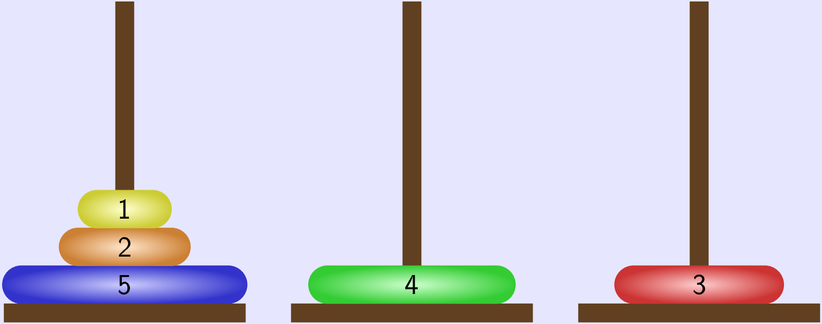
Moved disc from pole 3 to pole 2.

Tower of Hanoi – 5 Discs



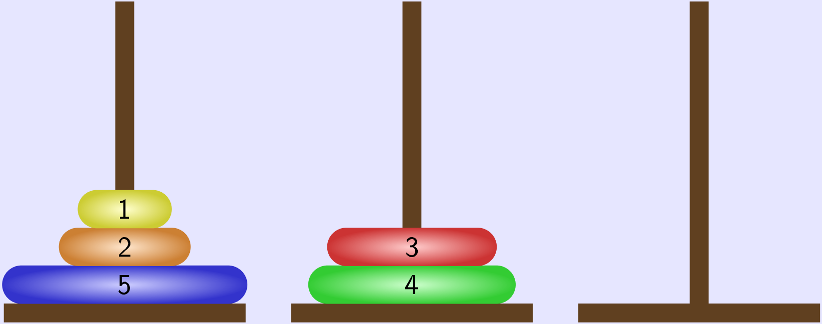
Moved disc from pole 3 to pole 1.

Tower of Hanoi – 5 Discs



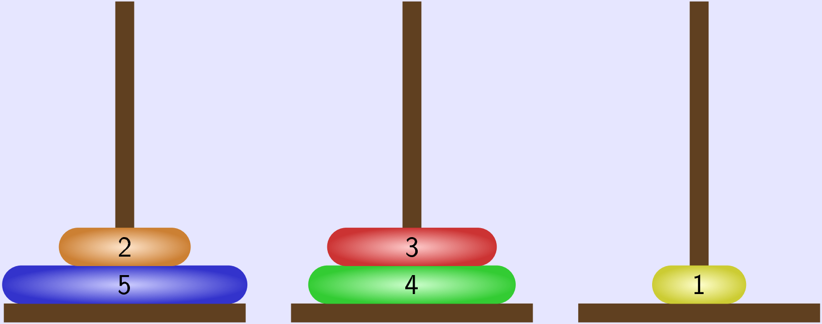
Moved disc from pole 2 to pole 1.

Tower of Hanoi – 5 Discs



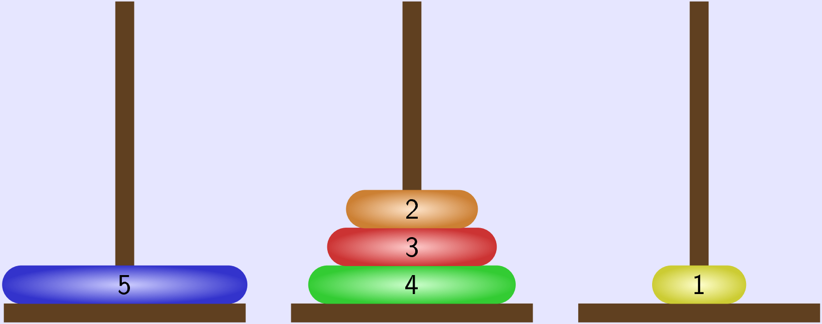
Moved disc from pole 3 to pole 2.

Tower of Hanoi – 5 Discs



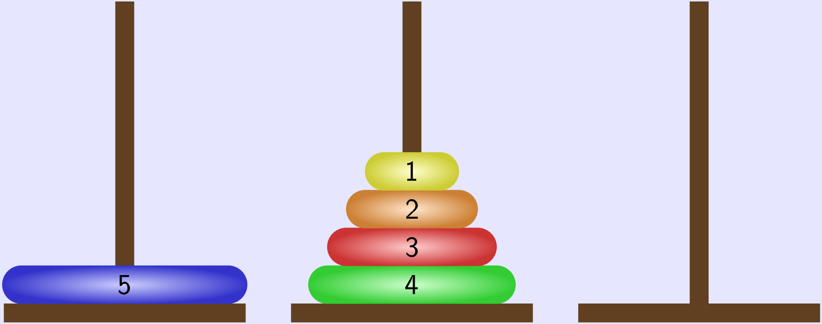
Moved disc from pole 1 to pole 3.

Tower of Hanoi – 5 Discs



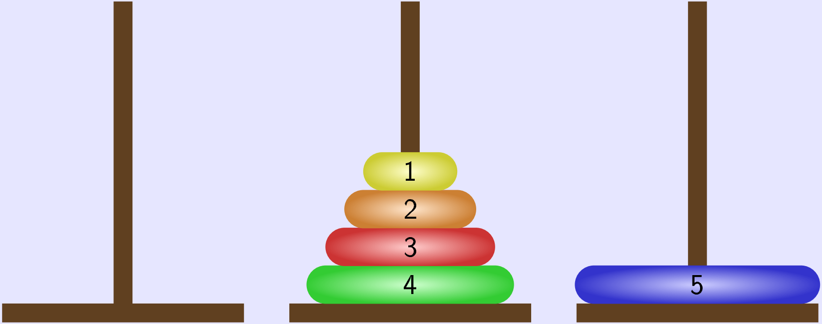
Moved disc from pole 1 to pole 2.

Tower of Hanoi – 5 Discs



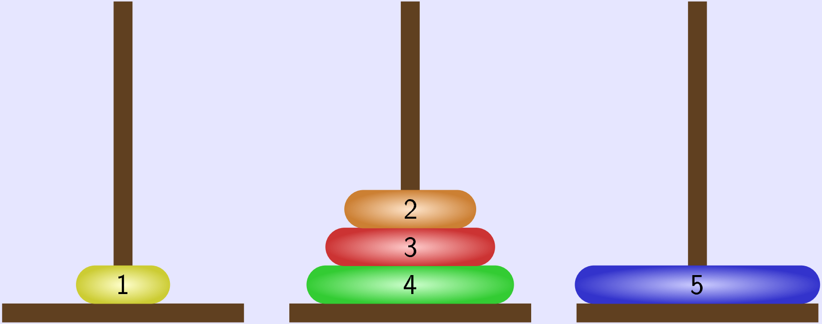
Moved disc from pole 3 to pole 2.

Tower of Hanoi – 5 Discs



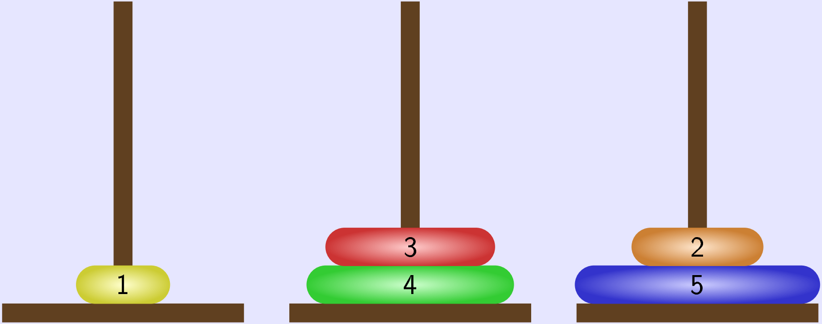
Moved disc from pole 1 to pole 3.

Tower of Hanoi – 5 Discs



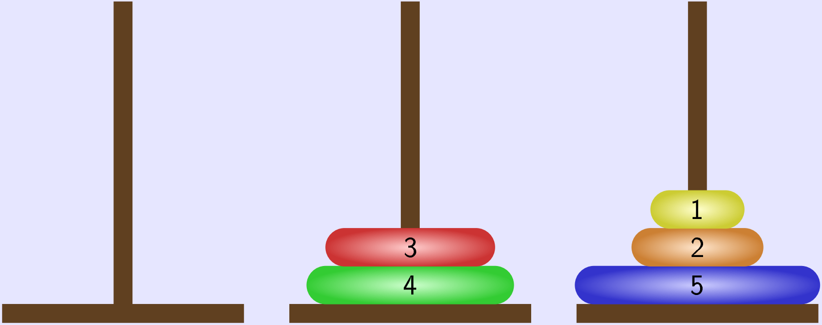
Moved disc from pole 2 to pole 1.

Tower of Hanoi – 5 Discs



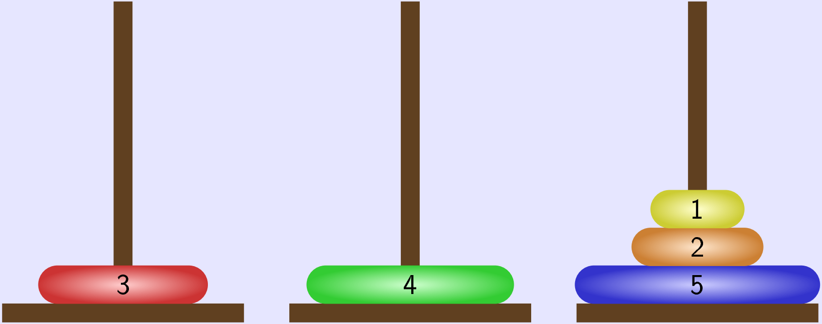
Moved disc from pole 2 to pole 3.

Tower of Hanoi – 5 Discs



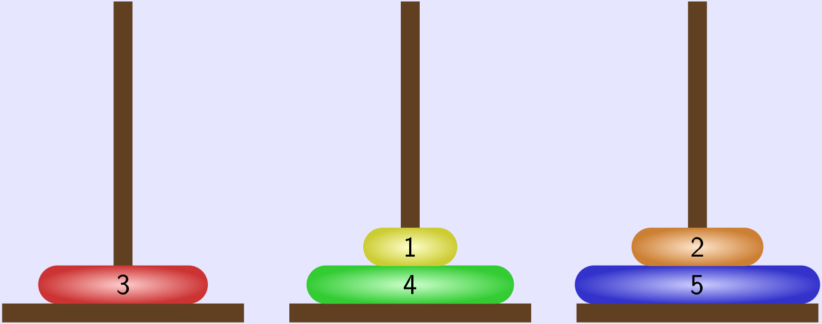
Moved disc from pole 1 to pole 3.

Tower of Hanoi – 5 Discs



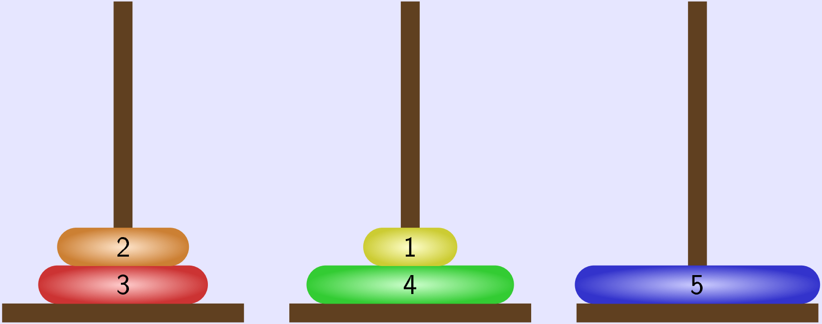
Moved disc from pole 2 to pole 1.

Tower of Hanoi – 5 Discs



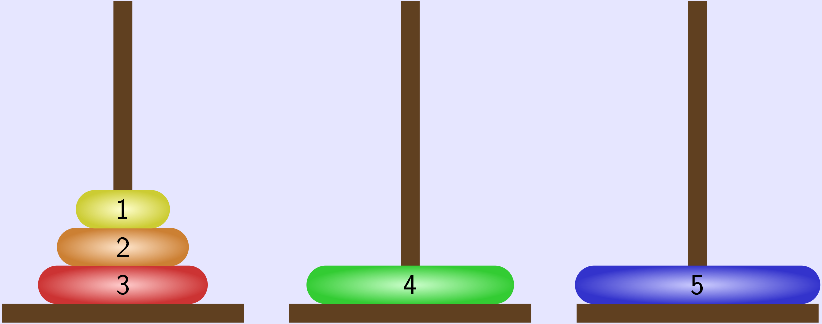
Moved disc from pole 3 to pole 2.

Tower of Hanoi – 5 Discs



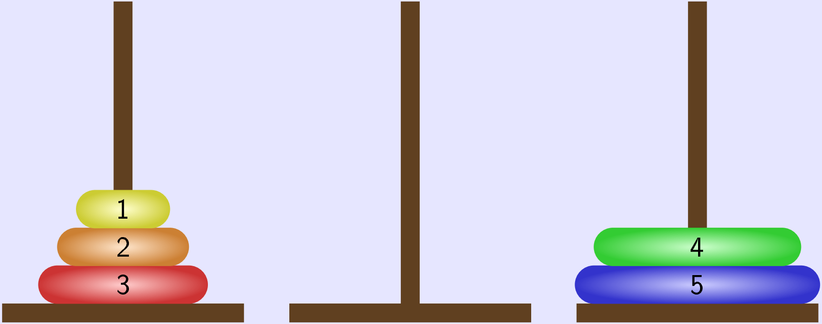
Moved disc from pole 3 to pole 1.

Tower of Hanoi – 5 Discs



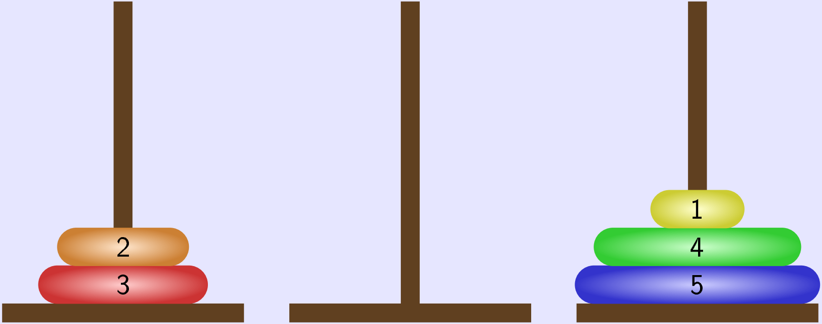
Moved disc from pole 2 to pole 1.

Tower of Hanoi – 5 Discs



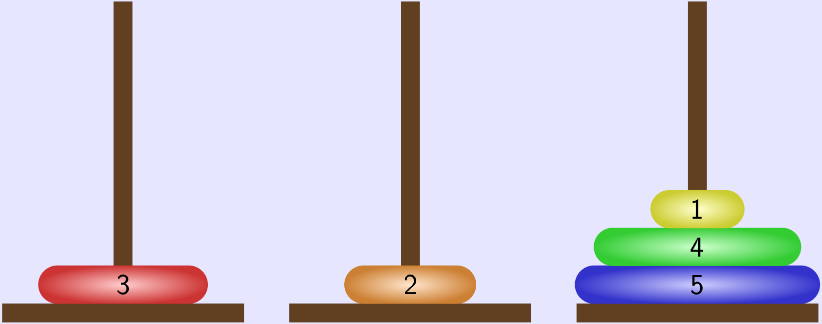
Moved disc from pole 2 to pole 3.

Tower of Hanoi – 5 Discs



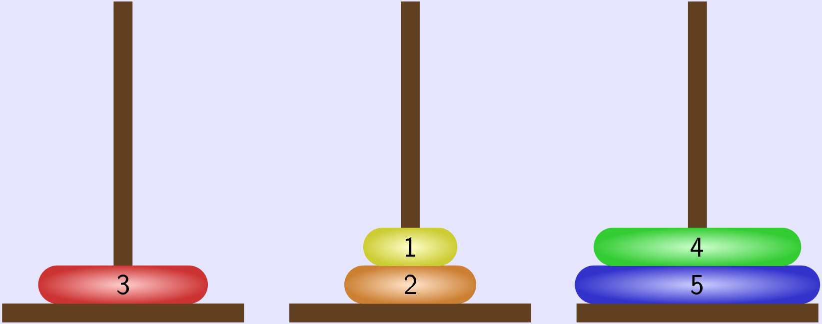
Moved disc from pole 1 to pole 3.

Tower of Hanoi – 5 Discs



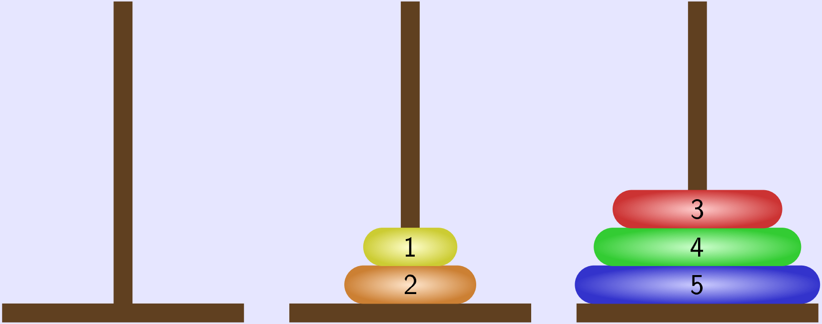
Moved disc from pole 1 to pole 2.

Tower of Hanoi – 5 Discs



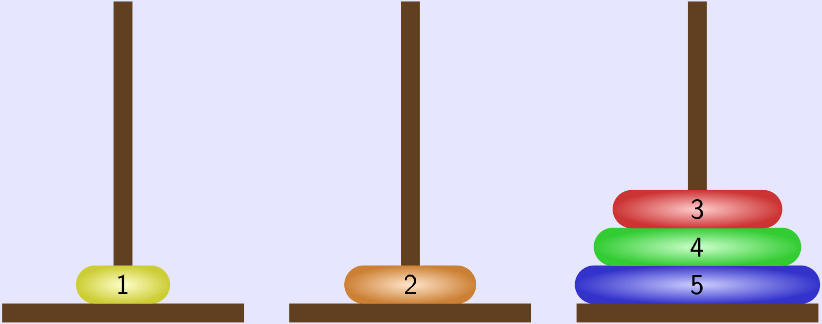
Moved disc from pole 3 to pole 2.

Tower of Hanoi – 5 Discs



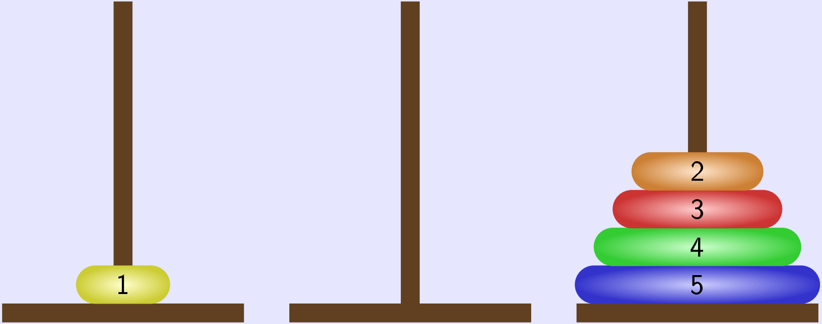
Moved disc from pole 1 to pole 3.

Tower of Hanoi – 5 Discs



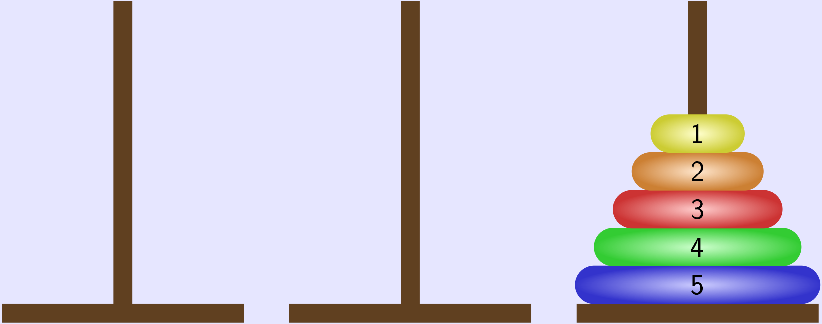
Moved disc from pole 2 to pole 1.

Tower of Hanoi – 5 Discs



Moved disc from pole 2 to pole 3.

Tower of Hanoi – 5 Discs



Moved disc from pole 1 to pole 3.

Tower of Hanoi – 5 Discs

A recursive solution in Python

```
#!/usr/bin/env python3

import os

def hanoi(n, start='1', step='2', stop='3'):
    '''Solve the Hanoi tower with n disks, from start
    peg to stop peg, using step peg as a spool'''
    if n > 0:
        hanoi(n-1, start, stop, step)
        move(n, start, stop)
        hanoi(n-1, step, start, stop)

def move(n, start, stop):
    '''Display move of disk n from start to stop'''
    print("Disk %d: %s -> %s" % (n, start, stop))

if __name__ == '__main__':
    n = int(input('How many disks? '))
    hanoi(n)
```

Question: why does this program show that $T_n = 2^n - 1$?

A recursive solution in Python

```python
#!/usr/bin/env python3

import os

def hanoi(n, start='1', step='2', stop='3'):
    '''Solve the Hanoi tower with n disks, from start
    peg to stop peg, using step peg as a spool'''
    if n > 0:
        hanoi(n-1, start, stop, step)
        move(n, start, stop)
        hanoi(n-1, step, start, stop)

def move(n, start, stop):
    '''Display move of disk n from start to stop'''
    print("Disk %d: %s -> %s" % (n, start, stop))

if __name__ == '__main__':
    n = int(input('How many disks? '))
    hanoi(n)
```

Question: why does this program show that $T_n = 2^n - 1$?

Warmup: What is wrong with this “proof”?

Theorem

All horses are the same color.

“Proof”

The thesis is clearly true for $n = 1$, so let $n > 1$.

- Put the n horses on a line.
- By inductive hypothesis, the $n - 1$ *leftmost* horses are the same color, and so do the $n - 1$ *rightmost* horses.
- Then the $n - 2$ horses *in the middle* are the same color.
- The first and last horse must then be *that* color.

Solution: For $n = 2$ there are no “$n - 2$ horses in the middle”.

Warmup: What is wrong with this "proof"?

Theorem

All horses are the same color.

"Proof"

The thesis is clearly true for $n = 1$, so let $n > 1$.

- Put the n horses on a line.
- By inductive hypothesis, the $n-1$ *leftmost* horses are the same color, and so do the $n-1$ *rightmost* horses.
- Then the $n-2$ horses *in the middle* are the same color.
- The first and last horse must then be *that* color.

Solution: For $n = 2$ there are no "$n-2$ horses in the middle".

Next section

Lines in the Plane

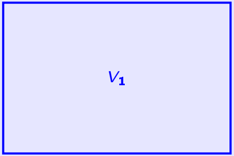
Problem

Popularly: How many slices of pizza can a person obtain by making n straight cuts with a pizza knife?

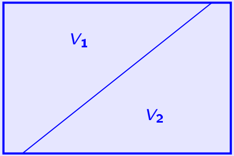
Academically: What is the maximum number L_n of regions defined by n lines in the plane?

Solved first in 1826, by the Swiss mathematician Jacob Steiner .

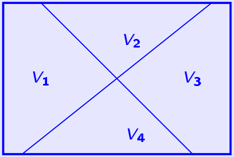
Lines in the Plane – small cases



$L_0 = 1$



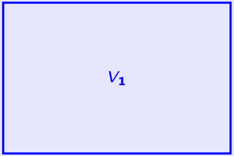
$L_1 = 2$



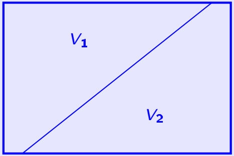
$L_2 = 4$

$L_3 = L_2 + 3 = 7$

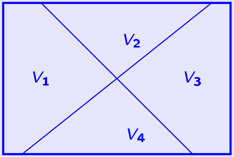
Lines in the Plane – small cases



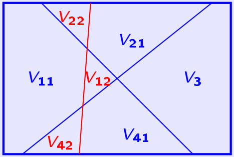
$L_0 = 1$



$L_1 = 2$



$L_2 = 4$



$L_3 = L_2 + 3 = 7$

Lines in the Plane – generalization

Observation:

The n-th line (for $n > 0$) increases the number of regions by k

iff it splits k of the "old regions"

iff it hits the previous lines in $k-1$ different places.

Lines in the Plane – generalization

Observation:

The n-th line (for $n > 0$) increases the number of regions by k

iff it splits k of the "old regions"

iff it hits the previous lines in $k - 1$ different places.

Lines in the Plane – generalization

Observation:

The n-th line (for $n > 0$) increases the number of regions by k

iff it splits k of the "old regions"

iff it hits the previous lines in $k-1$ different places.

Lines in the Plane – generalization

Observation:

The n-th line (for $n > 0$) increases the number of regions by k

iff it splits k of the "old regions"

iff it hits the previous lines in $k-1$ different places.

k must be less or equal to n. – Why?

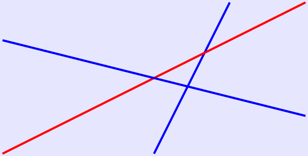
Lines in the Plane – generalization

Observation:

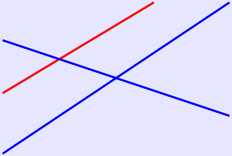
The n-th line (for $n > 0$) increases the number of regions by k

iff it splits k of the "old regions"

iff it hits the previous lines in $k-1$ different places.



$k = 3$; 2 places



$k = 2$; 1 places

Lines in the Plane – generalization (2)

Therefore the new line can intersect the $n-1$ "old" lines in at most "n- 1" different points, we have established the upper bound

$$\boxed{L_n \leqslant L_{n-1} + n \qquad \text{for } n > 0.}$$

If n-th line is not parallel to any of the others (hence it intersects them all), and doesn't go through any of the existing intersection points (hence it intersects them all in different places) then we get the recurrent equation:

$$L_0 = 1;$$
$$L_n = L_{n-1} + n \qquad \text{for } n > 0.$$

n	0	1	2	3	4	5	6	7	8	9	$\cdots$
L_n	1	2	4	7	11	16	22	29	37	46	$\cdots$

Lines in the Plane – generalization (2)

Therefore the new line can intersect the $n-1$ "old" lines in at most "n- 1" different points, we have established the upper bound

$$\boxed{L_n \leqslant L_{n-1} + n \qquad \text{for } n > 0.}$$

If n-th line is not parallel to any of the others (hence it intersects them all), and doesn't go through any of the existing intersection points (hence it intersects them all in different places) then we get the recurrent equation:

$$L_0 = 1;$$
$$L_n = L_{n-1} + n \qquad \text{for } n > 0.$$

n	0	1	2	3	4	5	6	7	8	9	$\cdots$
L_n	1	2	4	7	11	16	22	29	37	46	$\cdots$

Lines in the Plane – solving recurrence

Observation:

$$
\begin{aligned}
L_n &= L_{n-1} + n \\
&= L_{n-2} + (n-1) + n \\
&= L_{n-3} + (n-2) + (n-1) + n \\
&\ldots\ldots\ldots \\
&= L_0 + 1 + 2 + \ldots + (n-2) + (n-1) + n \\
&= 1 + S_n, \qquad \text{where } S_n = 1 + 2 + 3 + \ldots + (n-1) + n
\end{aligned}
$$

Lines in the Plane – solving recurrence (2)

Evaluation of $S_n = 1+2+\cdots+(n-1)+n$.

Recurrent equation:

$$S_0 = 0;$$
$$S_n = S_{n-1} + n \qquad \text{for } n > 0.$$

Solution (Gauss, 1786):

$$\begin{array}{rcccccccccc} S_n & = & 1 & + & 2 & + & \cdots & + & (n-1) & + & n \\ +S_n & = & n & + & (n-1) & + & \cdots & + & 2 & + & 1 \\ \hline 2S_n & = & (n+1) & + & (n+1) & + & \cdots & + & (n+1) & + & (n+1) \end{array}$$

$$2S_n = n(n+1)$$

$$S_n = \frac{n(n+1)}{2}$$

Lines in the Plane – solving recurrence (2)

Evaluation of $S_n = 1 + 2 + \cdots + (n-1) + n$.

Recurrent equation:

$$S_0 = 0;$$
$$S_n = S_{n-1} + n \qquad \text{for } n > 0.$$

Solution (Gauss, 1786):

$$\begin{array}{cccccccccccc} S_n & = & 1 & + & 2 & + & \cdots & + & (n-1) & + & n \\ +S_n & = & n & + & (n-1) & + & \cdots & + & 2 & + & 1 \\ \hline 2S_n & = & (n+1) & + & (n+1) & + & \cdots & + & (n+1) & + & (n+1) \end{array}$$

$$2S_n = n(n+1)$$

$$S_n = \frac{n(n+1)}{2}$$

Lines in the Plane – solving recurrence (2)

Evaluation of $S_n = 1 + 2 + \cdots + (n-1) + n$.

Recurrent equation:

$$S_0 = 0;$$
$$S_n = S_{n-1} + n \qquad \text{for } n > 0.$$

Solution (Gauss, 1786):

$$\begin{array}{rcccccccccc} S_n & = & 1 & + & 2 & + & \cdots & + & (n-1) & + & n \\ +S_n & = & n & + & (n-1) & + & \cdots & + & 2 & + & 1 \\ \hline 2S_n & = & (n+1) & + & (n+1) & + & \cdots & + & (n+1) & + & (n+1) \end{array}$$

$$2S_n = n(n+1)$$

$$S_n = \frac{n(n+1)}{2}$$

Lines in the Plane – solving recurrence (3)

Theorem: Closed formula for L_n

$$L_n = \frac{n(n+1)}{2} + 1, \qquad \text{for } n > 0.$$

Proof (by induction).

Basis: $L_0 = \frac{0(0+1)}{2} + 1 = 1$.

Step: Let assume $L_n = \frac{n(n+1)}{2} + 1$ and evaluate

$$\begin{aligned} L_{n+1} &= L_n + n + 1 \\ &= \frac{n(n+1)}{2} + 1 + n + 1 \\ &= \frac{n(n+1) + 2 + 2n}{2} + 1 \\ &= \frac{n(n+1) + 2(n+1)}{2} + 1 \\ &= \frac{(n+1)(n+2)}{2} + 1. \end{aligned}$$

Q.E.D.

Next section

The Josephus Problem

Legend:

During the Jewish-Roman war, Flavius Josephus, a famous historian of the first century, was among a band of 41 Jewish rebels trapped in a cave by the Romans. Preferring suicide to capture, the rebels decided to form a circle and, proceeding around it, to kill every third remaining person until no one was left. But Josephus, together his friend, wanted to avoid being killed. So he quickly calculated where he and his friend should stand in the vicious circle

The Josephus Problem

Our variation of the problem:

- we start with n people numbered 1 to n around a circle
- we eliminate every second remaining person
  until only one survives

Task is to compute the survivor's number, $J(n)$

Example, $n = 10$.

The elimination order is
2, 4, 6, 8, 10, 3, 7, 1, 9.

So, we have $J(10) = 5$

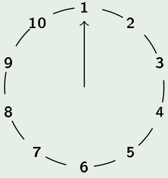
The Josephus Problem

Our variation of the problem:

- we start with n people numbered 1 to n around a circle
- we eliminate every second remaining person
  until only one survives

Task is to compute the survivor's number, $J(n)$

Example, $n = 10$.



The elimination order is
2, 4, 6, 8, 10, 3, 7, 1, 9 .

So, we have $J(10) = 5$

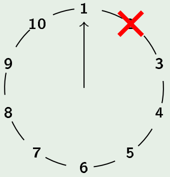
The Josephus Problem

Our variation of the problem:

- we start with n people numbered 1 to n around a circle
- we eliminate every second remaining person
  until only one survives

Task is to compute the survivor's number, $J(n)$

Example, $n = 10$.



The elimination order is
2, 4, 6, 8, 10, 3, 7, 1, 9.

So, we have $J(10) = 5$

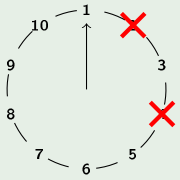
The Josephus Problem

Our variation of the problem:

- we start with n people numbered 1 to n around a circle
- we eliminate every second remaining person until only one survives

Task is to compute the survivor's number, $J(n)$

Example, $n = 10$.



The elimination order is

2, 4, 6, 8, 10, 3, 7, 1, 9.

So, we have $J(10) = 5$

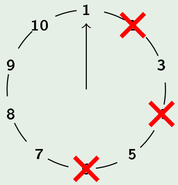
The Josephus Problem

Our variation of the problem:

- we start with n people numbered 1 to n around a circle
- we eliminate every second remaining person
  until only one survives

Task is to compute the survivor's number, $J(n)$

Example, $n = 10$.



The elimination order is
2, 4, 6, 8, 10, 3, 7, 1, 9.

So, we have $J(10) = 5$

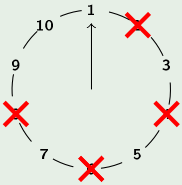
The Josephus Problem

Our variation of the problem:

- we start with n people numbered 1 to n around a circle
- we eliminate every second remaining person until only one survives

Task is to compute the survivor's number, $J(n)$

Example, $n = 10$.



The elimination order is
2, 4, 6, 8, 10, 3, 7, 1, 9.

So, we have $J(10) = 5$

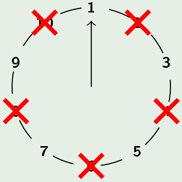
The Josephus Problem

Our variation of the problem:

- we start with n people numbered 1 to n around a circle
- we eliminate every second remaining person
  until only one survives

Task is to compute the survivor's number, $J(n)$

Example, $n = 10$.



The elimination order is
2, 4, 6, 8, 10, 3, 7, 1, 9.

So, we have $J(10) = 5$

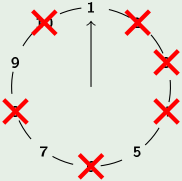
The Josephus Problem

Our variation of the problem:

- we start with n people numbered 1 to n around a circle
- we eliminate every second remaining person
  until only one survives

Task is to compute the survivor's number, $J(n)$

Example, $n = 10$.



The elimination order is
2, 4, 6, 8, 10, 3, 7, 1, 9 .

So, we have $J(10) = 5$

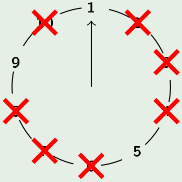
The Josephus Problem

Our variation of the problem:

- we start with n people numbered 1 to n around a circle
- we eliminate every second remaining person
  until only one survives

Task is to compute the survivor's number, $J(n)$

Example, $n = 10$.



The elimination order is
2, 4, 6, 8, 10, 3, 7, 1, 9.

So, we have $J(10) = 5$

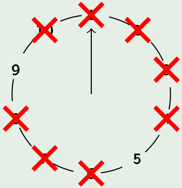
The Josephus Problem

Our variation of the problem:

- we start with n people numbered 1 to n around a circle
- we eliminate every second remaining person
  until only one survives

Task is to compute the survivor's number, $J(n)$

Example, $n = 10$.



The elimination order is
2, 4, 6, 8, 10, 3, 7, 1, 9 .

So, we have $J(10) = 5$

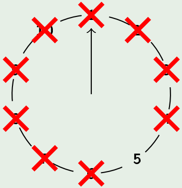
The Josephus Problem

Our variation of the problem:

- we start with n people numbered 1 to n around a circle
- we eliminate every second remaining person
  until only one survives

Task is to compute the survivor's number, $J(n)$

Example, $n = 10$.



The elimination order is
2, 4, 6, 8, 10, 3, 7, 1, 9 .

So, we have $J(10) = 5$

The Josephus Problem – small numbers

Evaluate $J(n)$ for small n:

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	$\cdots$
$J(n)$	1	1	3	1	3	5	7	1	3	5	7	9	11	13	15	1	$\cdots$

Properties

1. $J(n)$ is always odd;
2. Recurrent equation

$$\begin{aligned} J(1) &= 1; \\ J(2n) &= 2J(n) - 1, \qquad \text{for } n \geqslant 1; \\ J(2n+1) &= 2J(n) + 1, \qquad \text{for } n \geqslant 1. \end{aligned}$$

3. Closed formula

$$J(2^m + \ell) = 2\ell + 1, \qquad \text{for } m \geqslant 0 \text{ and } 0 \leqslant \ell < 2^m.$$

The Josephus Problem – small numbers

Evaluate $J(n)$ for small n:

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	$\cdots$
$J(n)$	1	1	3	1	3	5	7	1	3	5	7	9	11	13	15	1	$\cdots$

Properties

1. $J(n)$ is always odd;
2. Recurrent equation

$$\begin{aligned} J(1) &= 1; \\ J(2n) &= 2J(n) - 1, \qquad \text{for } n \geqslant 1; \\ J(2n+1) &= 2J(n) + 1, \qquad \text{for } n \geqslant 1. \end{aligned}$$

3. Closed formula

$$J(2^m + \ell) = 2\ell + 1, \qquad \text{for } m \geqslant 0 \text{ and } 0 \leqslant \ell < 2^m.$$

The Josephus Problem – small numbers

Evaluate $J(n)$ for small n:

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	$\cdots$
$J(n)$	1	1	3	1	3	5	7	1	3	5	7	9	11	13	15	1	$\cdots$

Properties

1. $J(n)$ is always odd;
2. Recurrent equation

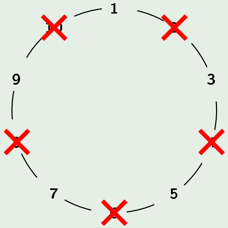
$$
\begin{aligned}
J(1) &= 1; \\
J(2n) &= 2J(n) - 1, \qquad \text{for } n \geqslant 1; \\
J(2n+1) &= 2J(n) + 1, \qquad \text{for } n \geqslant 1.
\end{aligned}
$$

3. Closed formula

$$J(2^m + \ell) = 2\ell + 1, \qquad \text{for } m \geqslant 0 \text{ and } 0 \leqslant \ell < 2^m.$$

The Josephus Problem – recurrent equation (1)

Case $n = 2m$.



First trip eliminates all even numbers. Then we change numbers and repeat:

Old number k	1	3	5	7	9
New number k'	1	2	3	4	5

or

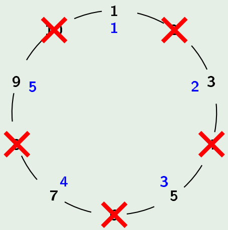
$$k = 2k' - 1$$

That correspondance between "old" and "new numbers" gives us that:

$J(2n) = 2J(n) - 1$

The Josephus Problem – recurrent equation (1)

Case $n = 2m$.



First trip eliminates all even numbers. Then we change numbers and repeat:

Old number k	1	3	5	7	9
New number k'	1	2	3	4	5

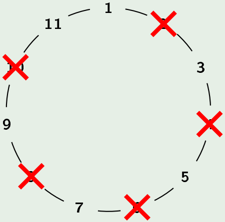
or

$$k = 2k' - 1$$

That correspondance between "old" and "new numbers" gives us that:
$J(2n) = 2J(n) - 1$

The Josephus Problem – recurrent equation (2)

Case $n = 2m + 1$.



First trip eliminates all even numbers. Then we change numbers and repeat:

Old number k	1	3	5	7	9	11
New number k'	0	1	2	3	4	5

or

$$k = 2k' + 1$$

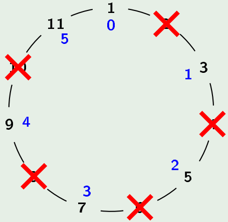
That correspondence between "old" and "new numbers" givs us that:

$$J(2n+1) = 2J(n) + 1$$

.

The Josephus Problem – recurrent equation (2)

Case $n = 2m + 1$.



First trip eliminates all even numbers. Then we change numbers and repeat:

Old number k	1	3	5	7	9	11
New number k'	0	1	2	3	4	5

or

$$k = 2k' + 1$$

That correspondence between "old" and "new numbers" givs us that:

$J(2n+1) = 2J(n) + 1$

.

The Josephus Problem – application of recurrence

The equation

$$\begin{aligned} J(1) &= 1; \\ J(2n) &= 2J(n) - 1, \qquad && \text{for } n \geqslant 1; \\ J(2n+1) &= 2J(n) + 1, \qquad && \text{for } n \geqslant 1. \end{aligned}$$

can be used for computing function for large arguments.

For example

$$\begin{aligned} J(86) &= 2J(43) - 1 = 53 \\ J(43) &= 2J(22) + 1 = 27 \\ J(22) &= 2J(11) - 1 = 13 \\ J(11) &= 7 \end{aligned}$$

The Josephus Problem – closed formula

Teoreem

$$J(2^m + \ell) = 2\ell + 1, \qquad \text{for } m \geqslant 0 \text{ and } 0 \leqslant \ell < 2^m.$$

Proof by induction over m.

Basis If $m = 0$ then also $\ell = 0$, and $J(1) = 1$;

Step

- If $m > 0$ and $2^m + \ell = 2n$, then ℓ is even and

  $$J(2^m + \ell) = 2J(2^{m-1} + \ell/2) - 1 = 2(2\ell/2 + 1) - 1 = 2\ell + 1$$

- If $2^m + \ell = 2n + 1$, then considering that $J(2n + 1) = 2 + J(2n) = 2 + 2(\ell - 1) + 1 = 2\ell + 1$

Q.E.D.

The Josephus Problem – closed formula (2)

Closed formula can be used for computing function $J(n)$:

Example

We have $1030 = 2^{10} + 6$, so $J(1030) = 2 \cdot 6 + 1 = 13$.

Next section

Structural induction

Premises

Let S be a set having the following features:

1. A set S_B of basic cases is contained in S.
2. Finitely many operations $u_i : S^{m_i} \to S$, $i = 1, \ldots, n$, exist such that, if $x_1, \ldots, x_{m_i} \in S$, then $u_i(x_1, \ldots, x_{m_i}) \in S$.
3. Nothing else belongs to S.

Technique

Let P be a property such that:

1. Each base case $x \in S_B$ has property P.
2. For every $i = 1, \ldots, n$ and every $x_1, \ldots, x_{m_i} \in S$, if each value $x_1, \ldots, x_{m_i}$ has property P, then $u_i(x_1, \ldots, x_{m_i})$ has property P.

Then every element of S has property P.

Structural induction

Premises

Let S be a set having the following features:

1. A set S_B of basic cases is contained in S.
2. Finitely many operations $u_i : S^{m_i} \to S$, $i = 1, \ldots, n$, exist such that, if $x_1, \ldots, x_{m_i} \in S$, then $u_i(x_1, \ldots, x_{m_i}) \in S$.
3. Nothing else belongs to S.

Technique

Let P be a property such that:

1. Each base case $x \in S_B$ has property P.
2. For every $i = 1, \ldots, n$ and every $x_1, \ldots, x_{m_i} \in S$, if each value $x_1, \ldots, x_{m_i}$ has property P, then $u_i(x_1, \ldots, x_{m_i})$ has property P.

Then every element of S has property P.

Mathematical induction as a special case of structural induction

Premises

The set $S = \mathbb{N}$ of natural numbers is constructed as follows:

1. A set $S_B = \{0\}$ of basic cases is contained in $\mathbb{N}$.
2. A single operation, the *successor*, $s : \mathbb{N} \to \mathbb{N}$, exists such that, if $n \in \mathbb{N}$, then $s(n) \in \mathbb{N}$.
3. Nothing else belongs to $\mathbb{N}$.

Technique

Let P be a property such that

1. 0 has property P.
2. For every $n \in \mathbb{N}$, if n has property P, then $s(n)$ has property P.

Then every $n \in \mathbb{N}$ has property P.

Mathematical induction as a special case of structural induction

Premises

The set $S = \mathbb{N}$ of natural numbers is constructed as follows:

1. A set $S_B = \{0\}$ of basic cases is contained in $\mathbb{N}$.
2. A single operation, the *successor*, $s : \mathbb{N} \rightarrow \mathbb{N}$, exists such that, if $n \in \mathbb{N}$, then $s(n) \in \mathbb{N}$.
3. Nothing else belongs to $\mathbb{N}$.

Technique

Let P be a property such that

1. 0 has property P.
2. For every $n \in \mathbb{N}$, if n has property P, then $s(n)$ has property P.

Then every $n \in \mathbb{N}$ has property P.