Maths for Computer Science
Solving recurrences

Denis TRYSTRAM
Lecture notes MoSIG1

sept. 2024

Objective

- Present **induction**, which is the basic principle used for solving recurrences.
- Show its application on several examples.

This lecture is devoted to **studving a variety of types of recurrences** (linear recurrence, multiple steps recurrences, etc.).

We will show how to use various ways for solving problems by recurrence.

Brief overview of this sequence

- Recall of the induction principle and recurrence proofs.
- A first example (direct application)
- Take care at the first steps!
- Linear and bilinear recurrences:
  Hanoi's towers and The token game

An example of induction: Factorial

The first classical example of the *recurrent* mode of computing involves the *factorial function* $\textsc{Fact}$ (of a nonnegative integer).

- The "direct" mode of computing $\textsc{Fact}$ at an argument n is:

$$\textsc{Fact}(n) \;=\; 1 \times 2 \times \cdots \times n$$

- The *recurrent* mode of computing $\textsc{Fact}(n)$ is more compact—and it better exposes the inherent structure of the function.

$$\textsc{Fact}(n) \;=\; \begin{cases} n \times \textsc{Fact}(n-1) & \text{if } \; n > 1 \\ 1 & \text{if } \; n = 1 \end{cases}$$

Proof by recurrence

Goal:

proving that a statement $P(n)$ involving integer n is true using the induction principle.

- **Basis.** Solve the statement for the small value(s) of n and verify $P(1)$
- **Induction step.** Prove the statement for $n+1$ assuming it is correct for $k \leq n$.

Principle: start by writing the expression at rank $n+1$ and apply an external definition, where you exhibit previous ranks $k \leq n$. Then, replace $P(n)$ by its expression and deduce $P(n+1)$ by algebraic manipulations.

Example

Proposition *P(n)*.

$\forall n$, the n-th perfect square is the sum of the first n odd integers.

$$n^2 = \sum_{k=1}^{n} (2k-1)$$

Example

Proposition *P(n)*.

$\forall n$, the n-th perfect square is the sum of the first n odd integers.

$$n^2 = \sum_{k=1}^{n} (2k-1)$$

Proof.

For every positive integer k, let $\mathbf{P}(k)$ denote the assertion

$$k^2 = 1+3+5+\cdots+(2k-1)$$

Let us proceed by the standard format of an inductive argument:

- **Basis.** Because $1^2 = 1$, proposition $\mathbf{P}(1)$ is true.
- **Induction step.** Let us assume, for the sake of induction, that assertion $\mathbf{P}(k)$ is true for all positive integers smaller than n.

Consider now the summation at rank $n+1$

$$1+3+5+\cdots+(2n-1)+(2n+1)$$

Because $\mathbf{P}(n)$ is true, we know that

$$\begin{aligned} 1+3+\cdots+(2n+1) &= \big(1+3+\cdots+(2n-1)\big)+(2n+1) \\ &= n^2+(2n+1) \\ &= (n+1)^2 \end{aligned}$$

The Principle of (Finite) Induction tells us that $\mathbf{P}(n)$ is true for all integer n.

When to use recurrence?

- We need to have a precise idea of what has to be proven!
- The analysis of the first ranks[1] helps in finding recurrence patterns.

[1] and not only the first one

Starting an induction

Let us (mis)use the method of Finite Induction to craft a fallacious proof of the following absurd "fact".

Proposition.

All horses are the same color.

- **The base case**. If there is only a single horse in the set, then obviously **P** is true (*all* horses in a singleton are the same color).
- **Inductive hypothesis.** in every set of n horses, all horses in the set are the same color.

Extension to $n+1$ horses.

- Let us be given a set of $n+1$ horses.
- Remove one horse from the set, then the remaining set, call it S has n horses.
- By our inductive hypothesis, all of the horses in S have the same color.
- Now remove one horse from S and replace it with the horse that was removed from the $(n+1)$-horse set.
  We now have a new n-horses set S'.
- Once again we invoke the inductive hypothesis to conclude that all horses in S' have the same color. If we now reunite all of the horses, the *transitivity* of the relation "have the same color" guarantees that all of the horses in the $(n+1)$-horses set have the same color.

What's wrong?

We all know that all horses do *not* share the same color.

What's wrong?

We all know that all horses do *not* share the same color.
The base case was not adequate for the "proof"

- When we remove first one horse from the set of $n+1$ horses and then another horse from that set, and we still have a horse left to compare those two horses to, *we must have started with at least* **three** *horses!*
- This means that $n+1$ *must be no smaller than* 3*, so n must be no smaller than* 2*.* The base of the induction must, therefore, be sets that contain 2 horses—and the same-color "proposition" is, of course, absurd for such sets!

This brings us to the critical issue of how to select the "small" cases that comprise the base of our induction.

Content

- Fibonacci sequence
- Preliminary: Hanoi's towers
- Beyond standard proofs
  The Token Game.
  Non linear recurrences.
  in particular, bilinear recurrences.

  $u_{n+1} = \alpha.u_n + \beta.u_{n-1} + \gamma$ where u_0 and u_1 are given.

Applications

Fibonacci numbers

The simplest possible bilinear recurrence ($\alpha = \beta = 1$ and $\gamma = 0$).
$F(n+1) = F(n) + F(n-1)$ with $F(0) = 1$ and $F(1) = 1$

Same with a different seed.
$L(n+1) = L(n) + L(n-1)$ with $L(0) = 1$ and $L(1) = 3$

Hanoi's towers

$T(n) = 2.T(n-1) + 1$ with $T(0) = 1$ and $T(1) = 1$

Token Game

$T(n+1) = T(n) + 2.T(n-1) + 1$ with $T(0) = 1$ and $T(1) = 2$

Stern sequence

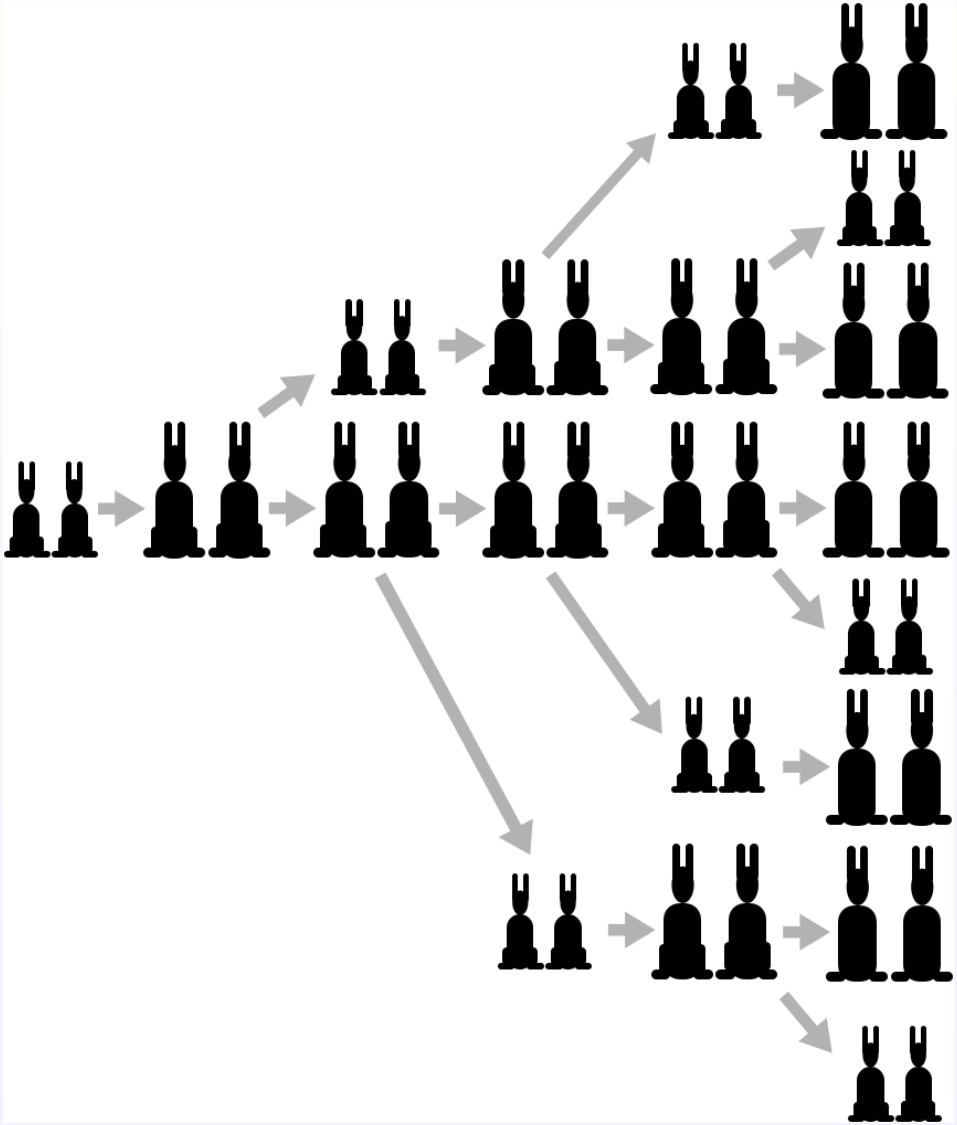
$s(0) = 0$ and $s(1) = 1$
$s(2n) = s(n)$ and $s(2n+1) = s(n) + s(n+1)$

Definition of Fibonacci numbers

The original problem has been introduced by Leonardo of Pisa (Fibonacci) in the middle age.

- Fibonacci numbers are the number of pairs of rabbits that can be produced at the successive generations.
- Starting by a single pair of rabbits and assuming that each pair produces a new pair of rabbits at each generation during only two generations.

Definition (pictorially)



Definition (more formally)

Definition:

Given the two numbers $F(0) = 1$ and $F(1) = 1$
the Fibonacci numbers are obtained by the following expression:
$F(n+1) = F(n) + F(n-1)$

Definition (more formally)

Definition:

Given the two numbers $F(0) = 1$ and $F(1) = 1$
the Fibonacci numbers are obtained by the following expression:
$F(n+1) = F(n) + F(n-1)$

The first ranks:

n	0	1	2	3	4	5	6	7	8	9	10	...
F(n)	1	1	2	3	5	8	13	21	34	55	89	...

Combinatorial interpretation

Proposition

The Fibonacci number $F(n)$ can be interpreted as the number of length-n binary strings in which each occurence of a 1 is directly preceded by a 0.

Let S_n be the set of such strings of length n.

Proof

By the previous definition, every binary string ω_n ends either with 0 or with 01.

- If ω_n ends with 0, then, it has the form $x0$ where the prefix x is a binary string of length $n-1$.
  Moreover, x must belongs to S_{n-1} in order ω_n belongs to S_n.
  Therefore S_n contains $|S_{n-1}|$ strings of this form.
- If ω_n ends with 01, then it has the form $\omega_n = y01$, where the prefix y is a binary string of length $n-2$.
  Moreover, y must belong to S_{n-2} in order for ω_n to belong to S_n, that contains $|S_{n-2}|$ strings of this form.

$$F(n) = |S_n| = F(n-1) + F(n-2)$$

Studying a first property

Proposition:

$\sum_{k=0}^{n} F(k)$?

Studying a first property

Proposition:

$\sum_{k=0}^{n} F(k)$?

Let compute the expression on the first ranks:

- $n = 1$, $F(1) + F(0) = 1 + 1 = 2$
- $n = 2$, $F(2) + F(1) + F(0) = 2 + 1 + 1 = 4$
- $n = 3$,
  $F(3) + F(2) + F(1) + F(0) = 3 + 2 + 1 + 1 = 7$
- $n = 4$,
  $F(4) + F(3) + F(2) + F(1) + F(0) = 5 + 3 + 2 + 1 + 1 = 12$

Any idea of the expression?

Guess

Let compute the expression on the first ranks:

- $n = 1$,
  $1 + 1 = 2 = 3 - 1$
- $n = 2$,
  $2 + 1 + 1 = 4 = 5 - 1$
- $n = 3$,
  $3 + 2 + 1 + 1 = 7 = 8 - 1$
- $n = 4$,
  $5 + 3 + 2 + 1 + 1 = 12 = 13 - 1$

Proposition:

$\sum_{k=0}^{n} F(k) = F(n+2) - 1$

Proof $\sum_{k=0}^{n} F(k) + 1 = F(n+2)$

By induction

- The **basis case** (for $n = 0$) is true since $F(2) = 1 + F(0)$.
- **Induction step:** Let assume the property holds at rank n for $F(n+2)$ and compute $F(n+3)$:
  Apply the definition of Fibonacci numbers:
  $F(n+3) = F(n+1) + F(n+2)$
  Replace the last term by the recurrence hypothesis:
  $F(n+2) = 1 + \sum_{k=0}^{n} F(k)$
  Thus,
  $F(n+3) = F(n+1) + 1 + \sum_{k=0}^{n} F(k) = 1 + \sum_{k=0}^{n+1} F(k)$

Product of two consecutive Fibonacci numbers

Proposition:

$F(n).F(n-1) = \sum_{k=0}^{n-1} F(k)^2$ (for $n \geq 1$)

Let check the expression on the first ranks:

- $n = 2$, $F(2).F(1) = F(1)^2 + F(0)^2 = 1 + 1 = 2$
- $n = 3$, $F(3).F(2) = F(2)^2 + F(1)^2 + F(0)^2 = 4 + 1 + 1 = 6$
- $n = 4$, $F(4).F(3) = F(3)^2 + F(2)^2 + F(1)^2 + F(0)^2 = 15$
- $n = 5$,
  $F(5).F(4) = F(4)^2 + F(3)^2 + F(2)^2 + F(1)^2 + F(0)^2 = 40$

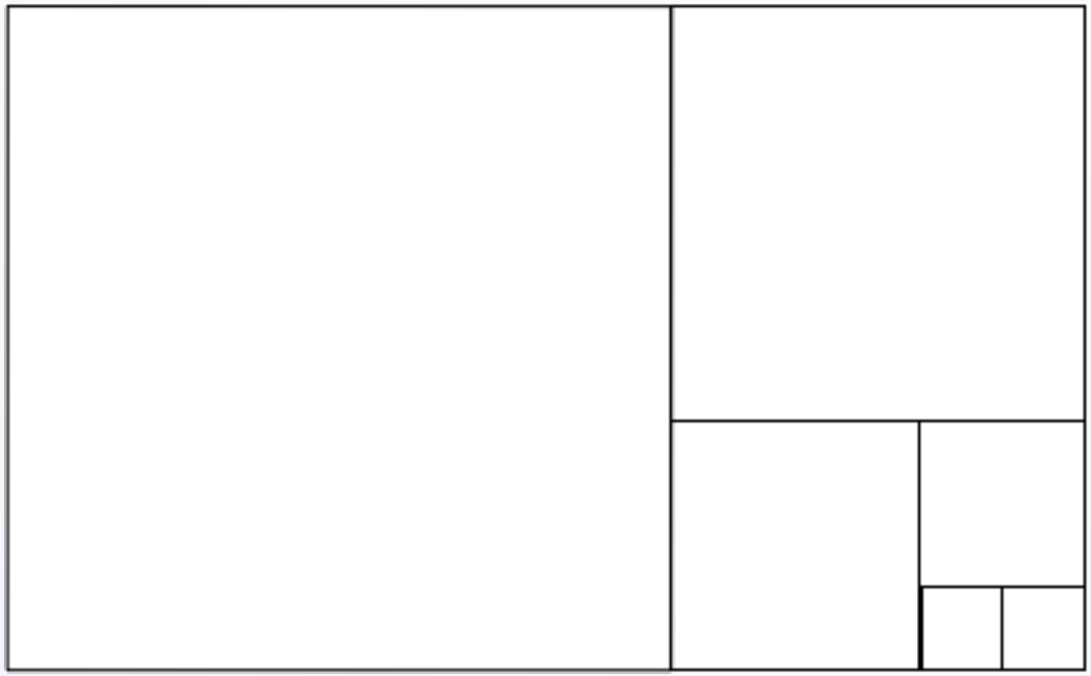
Proof by induction

- The **basis case** (for $n = 1$) is true since $F(1).F(0) = F(0)^2 = 1$.
- **Induction step**[2]**:** Let assume the property holds at rank n and compute $F(n+1).F(n)$:
  Apply the definition of $F(n+1)$:
  $F(n+1).F(n) = (F(n) + F(n-1)).F(n)$
  $= F(n)^2 + F(n).F(n-1)$
  Apply now the induction hypothesis to this last term:
  $F(n+1).F(n) = F(n)^2 + \sum_{k=0}^{n-1} F(k)^2 = \sum_{k=0}^{n} F(k)^2$

[2] exactly the same scheme as before!

An alternative proof (draft)

The relation can be proved very easily by the geometric argument shown below. The basis is a unit square (1 by 1).



Another property dealing with squares

Proposition:

$F(n+2)^2 = 4.F(n).F(n+1) + F(n-1)^2$ for $n \geq 2$.

Let check the expression on the first ranks:

$n = 1,\ F(3)^2 = 3^2 = 4.F(1).F(2) + F(0)^2 = 8 + 1 = 9$

$n = 2,\ F(4)^2 = 5^2 = 4.F(2).F(3) + F(1)^2 = 24 + 1 = 25$

$n = 3,\ F(5)^2 = 8^2 = 4.F(3).F(4) + F(2)^2 = 60 + 4 = 64$

$n = 4,\ F(6)^2 = 13^2 = 4.F(4).F(5) + F(3)^2 = 160 + 9 = 169$

...

Analytic proof

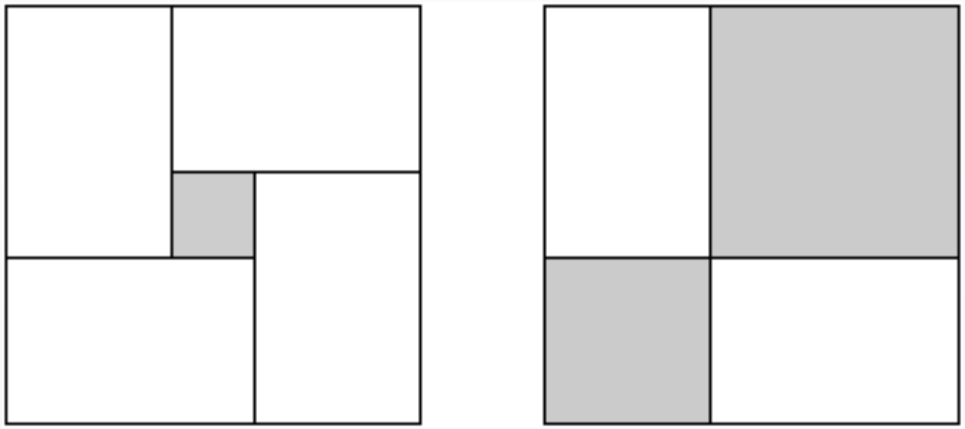
Use the definition of the Fibonacci numbers and expand:

$$F(n+2)^2 = (F(n+1) + F(n))^2$$
$$= F(n+1)^2 + 2.F(n+1).F(n) + F_n^2$$
$$= 4.F(n+1).F(n) - 2.F(n+1).F(n) + F(n+1)^2 + F(n)^2$$
$$= 4.F(n+1).F(n) + (F(n+1) - F(n))^2$$

Again, using the definition of $F(n+1)$ into the square, we get the expected result:

$$F(n+2)^2 = 4.F(n+1).F(n) + F(n-1)^2$$

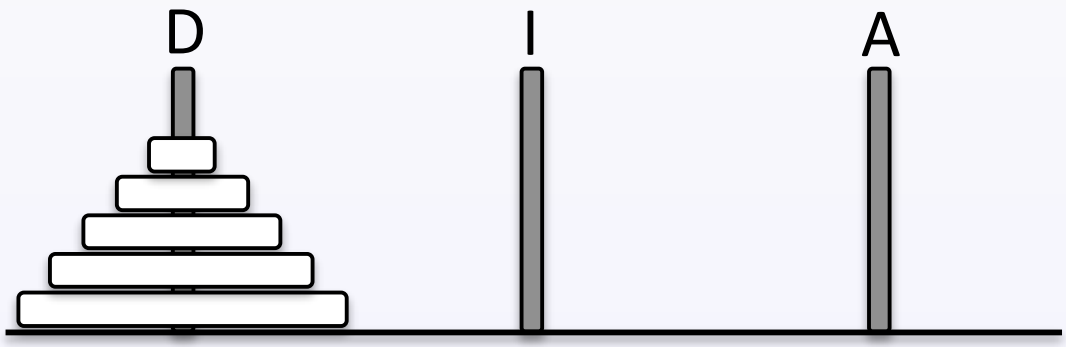
Graphical proof



Take care: a figure alone is not a proof!
The argument should be a generic one.
Here, it is enough to say that the small square in the left if of size $F(n-1))$...

Hanoi's Towers

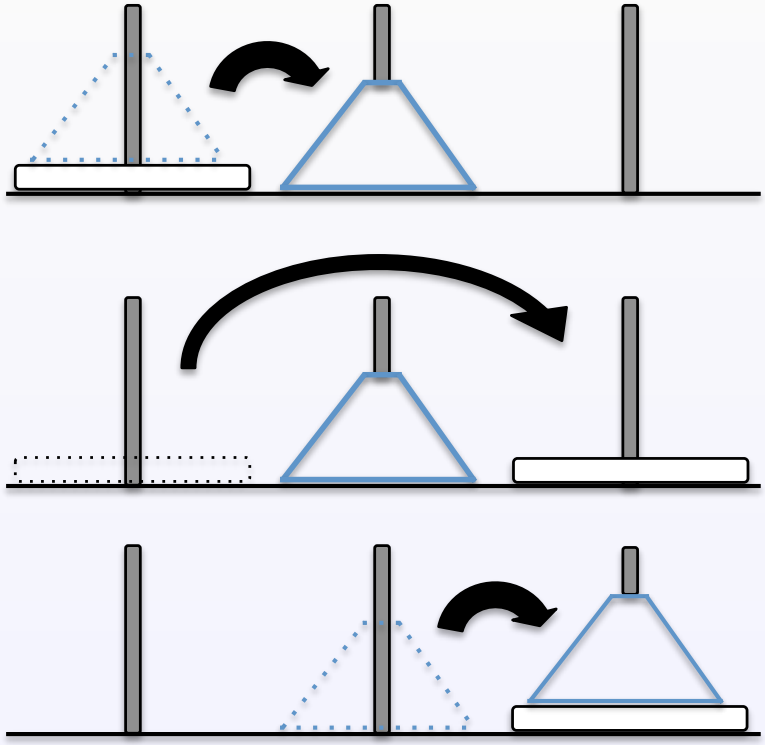
It is a well-known problem: A set of n disks of decreasing diameters initially stacked on one of three pegs.
Problem definition. The goal is to transfer the entire tower from a given peg (D) to another fixed one (A), moving only one disk at a time and never moving a larger disk on top of a smaller one.



The main question is to determine the *best* way to realize this operation (which means in a minimum number of moves).

The classical (recursive) solution

The natural method is recursive. It consists in moving the $n-1$ top disks on the intermediate peg, then, put the largest one on the target peg, and moving again the $n-1$ disks on top of it.



Compute the cost

- **Basis** is straightforward since $H_1 = 2^1 - 1 = 1$.
- **Induction step**
  $H_n = 2H_{n-1} + 1$ where $H_{n-1} = 2^{n-1} - 1$,
  thus, $H_n = 2(2^{n-1} - 1) + 1 = 2^n - 1$ and we are done.

Notice that this expression can also be obtained directly as the sum of a geometric series:

$$\begin{aligned} H_n &= 2H_{n-1} + 1 \\ &= 2(2H_{n-2} + 1) + 1 \\ &= \dots \\ &= \textstyle\sum_{k=0}^{n-1} 2^k = 2^n - 1 \end{aligned}$$

formal algorithm

input: an integer n, three pegs indexed as D (departure), A (arrival) and I (intermediate). All the disks are stacked on peg D.
output: The disks are stacked on peg A.

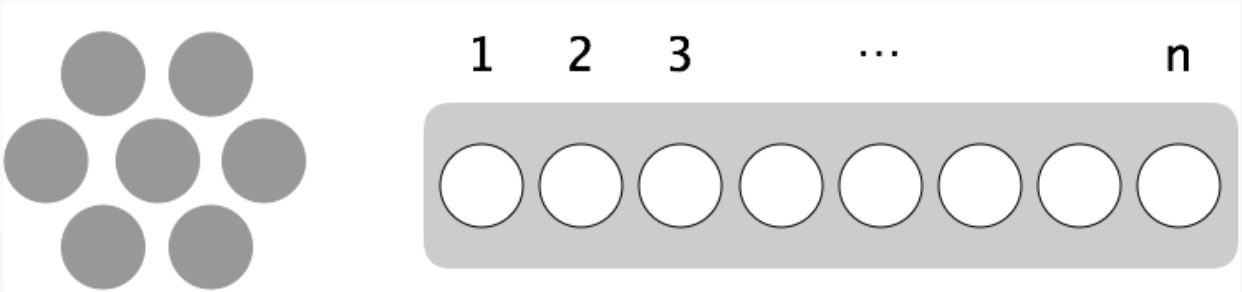
If $(n \neq 0)$ then

- Hanoi(n-1,D,I,A)
- move disk from D to A
- Hanoi(n-1,I,A,D)

Analysis. It is easy to compute the number of moves, using the same recurrence equation as for the lower bound:
$H_n = 2^n - 1$.

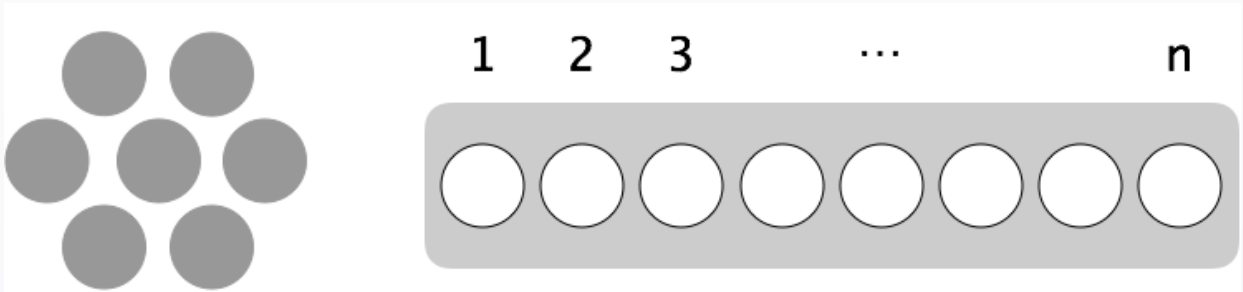
Extension to bilinear equations: The Token Game

Consider a bank with n circle positions numbered from 1 to n and n tokens. Initially, the bank is empty.



Extension to bilinear equations: The Token Game

Consider a bank with n circle positions numbered from 1 to n and n tokens. Initially, the bank is empty.



The game consists in determining the process to fill the bank with the n tokens, putting or removing one token at a time according to one of the two following constraints.

- **Rule 1.** Position 1: Put a token if it is empty or remove it.
- **Rule 2.** Position next to the first empty position (i.e. on the right): Put a token if the position is empty or remove it.

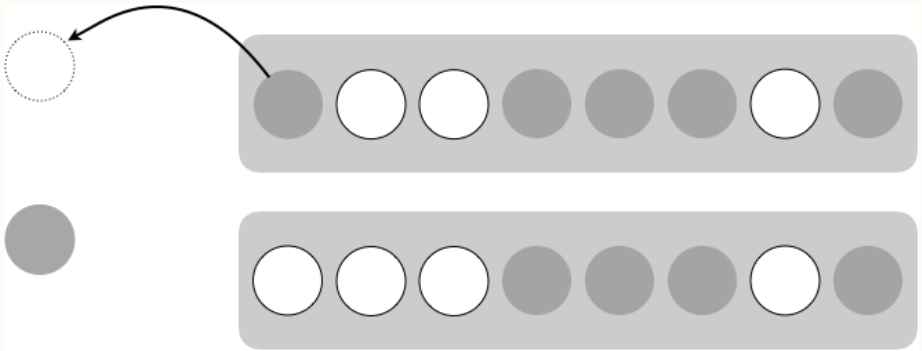


Figure: Rule 1: Position 1 contains a token, thus, remove it.

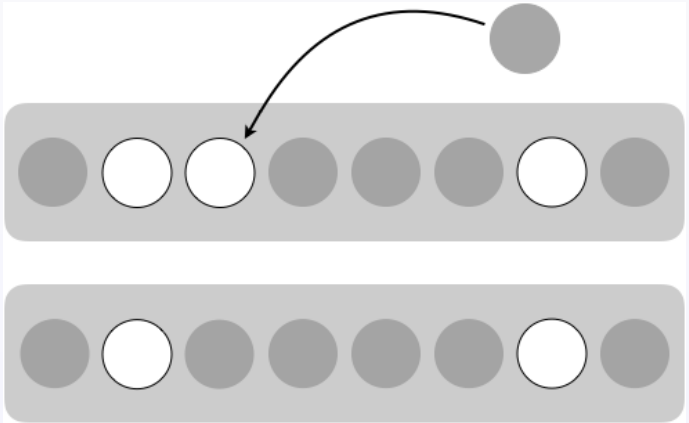


Figure: Rule 2: The position next to the first idle position (i.e. position 3 here) is idle, thus, put a remaining token.

Intuition of the solution

How the player should choose successive moves, with the goal of filling the bank as quickly as possible?

One can garner some strategic observations about how to play the Game by looking at small instances.

- When $n = 1$, the player should simply fill the slot using a Type-1 move.
- When $n = 2$, the player must first play a Type-2, then a Type-1 move.
- As the Game proceeds, observation suggests that the player should begin with a Type-1 move when n is odd and with a Type-2 move when n is even.
- Another easy observation is:
  The player should not play two successive moves of the same type, because the second one just undoes the first.

Analysis (2)

A strategy is beginning to emerge:

1. Choose the initial move based on the parity of n.
2. Subsequently, alternate between the two types of moves.

This strategy is pleasingly simple, but:
(a) Does it lead us to the required terminal state?
(b) What is the cost of a (successful) play using this strategy?

Analysis (3)

A token can be placed into the *last* bank-slot (i.e., the nth) via the Type-2 move

In order for this move to be eligible, the bank must be in the following configuration:

$$\big[\text{tokens in slots } \ 1, 2, \ldots, n-2\big], \big[\text{no token in } n-1 \text{ and } n\big]$$

This configuration requires that the first $n-2$ slots have been filled.[3]

Once the player has achieved this configuration and executed the mandated move, the player henceforth ignores the token in bank-slot n. **Here comes the recursion!**

[3] Remark that $n-2$ has the same parity as n.

Analysis

The Game can be played by recursively executing the *super-steps* depicted below, on successively smaller banks and piles.

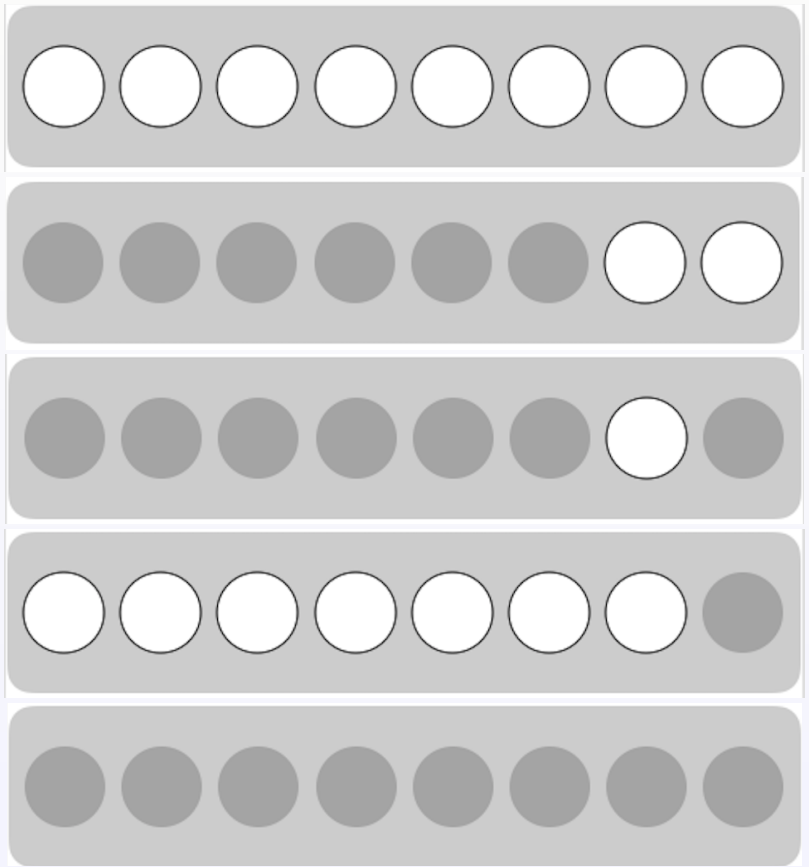


Figure: A schematic of the recursive play of the Game.

Analysis

The cost of the recursive solution is:

$$f(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2 & \text{if } n = 2 \\ f(n-2) + 1 + f^{-1}(n-2) + f(n-1) & \text{if } n > 2 \end{cases} \quad (1)$$

Analysis

The cost of the recursive solution is:

$$f(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2 & \text{if } n = 2 \\ f(n-2) + 1 + f^{-1}(n-2) + f(n-1) & \text{if } n > 2 \end{cases} \quad (1)$$

Let f^{-1} denote the bank-emptying operation.
If one thinks in terms of mirror-image operations, then one discovers a recursive solution for emptying the bank:

$$f^{-1}(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2 & \text{if } n = 2 \\ f^{-1}(n-1) + f(n-2) + 1 + f^{-1}(n-2) & \text{if } n > 2 \end{cases} \quad (2)$$

Analysis (6)

Both systems of equations are coupled!

Analysis (6)

Both systems of equations are coupled!

- Taking the difference between expressions (1) and (2), we find that the costs of $f(n)$ and $f^{-1}(n)$ are equal!

$$\begin{aligned} f(n) - f^{-1}(n) &= f(n-2) + 1 + f^{-1}(n-2) + f(n-1) \\ &\quad - f^{-1}(n-1) - f(n-2) - 1 - f^{-1}(n-2) \\ &= f(n-1) - f^{-1}(n-1) \\ &\quad \cdots \\ &= f(1) - f^{-1}(1) \\ &= 0 \end{aligned}$$

Analysis (7)

The final equation is:

$$f(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2 & \text{if } n = 2 \\ f(n-1) + 2f(n-2) + 1 & \text{if } n > 2 \end{cases} \quad (3)$$

Solving the recurrence

We can dramatically simplify the previous recurrence by focusing on the function[4]

$$g(n) \;=\; f(n) \;+\; f(n-1) \quad \text{for} \quad n \geq 2$$

instead of on f.

[4] the intuition behind this proposal is to remark that the expression simplifies if we add $f(n-1)$ in each side.

Solving the recurrence

We can dramatically simplify the previous recurrence by focusing on the function[4]

$$g(n) \;=\; f(n) \;+\; f(n-1) \quad \text{for} \quad n \geq 2$$

instead of on f.

Elementary calculation shows that $g(n)$ satisfies the recurrence

$$g(n) \;=\; \begin{cases} 3 & \text{if} \quad n = 2 \\ 2g(n-1)+1 & \text{if} \quad n > 2 \end{cases} \tag{4}$$

[4] the intuition behind this proposal is to remark that the expression simplifies if we add $f(n-1)$ in each side.

We have, thereby, replaced the initial *bilinear* recurrence by the *(singly) linear* recurrence (4).

We know how to solve geometric summations.

$$g(n) \;=\; 2^{n-1} + 2^{n-2} + \cdots + 2^2 + 2 + 1 \;=\; 2^n - 1 \quad (5)$$

We can now return to evaluating $f(n)$ in the light of our analysis of $g(n)$.

$$f(n) = \begin{cases} 1 & \text{if } n = 1 \\ g(n) - f(n-1) = (2^n - 1) - f(n-1) & \text{if } n > 1 \end{cases} \tag{6}$$

- We begin to solve the *singly* linear recurrence (6) for $f(n)$ using the strategy we developed for simple geometric sums.
- We expand the recurrence in order to discern its pattern and then analyze the summation that the pattern leads to:

$$\begin{aligned} f(n) &= 2^n - 2^{n-1} + f(n-2) + 1 - 1 \\ &= 2^n - 2^{n-1} + 2^{n-2} - f(n-3) - 1 \\ &= 2^n - 2^{n-1} + 2^{n-2} - 2^{n-3} + f(n-4) + 1 - 1 \\ &\vdots \end{aligned}$$

We have now reached the *penultimate* step in finding the value of $f(n)$; specifically, we have derived the following parity-specified summations.

$$\text{For even values of } n: \quad f(n) = \sum_{k=1}^{n} (-1)^k \, 2^k \tag{7}$$

$$\text{For odd values of } n: \quad f(n) = \sum_{k=0}^{n} (-1)^{k+1} \, 2^k \tag{8}$$

We first gather the positive and negative terms in summation We thereby find that, for odd values of n:

$$\begin{aligned}
f(n) &= \left(2^n + 2^{n-2} + \cdots + 2\right) - \left(2^{n-1} + 2^{n-3} + \cdots + 1\right) \\
&= 2^{n-1} + 2^{n-3} + \cdots + 1 \\
&= 2^{n-1} \cdot \left(1 + \frac{1}{4} + \frac{1}{16} + \cdots + \frac{1}{2^{n-1}}\right) \\
&= 2^{n-1} \cdot \left(1 + \frac{1}{4} + \frac{1}{16} + \cdots + \frac{1}{4^{(n-1)/2}}\right) \\
&= 2^{n-1} \cdot \frac{4}{3} \cdot \left(1 - \left(\frac{1}{4}\right)^{(n+1)/2}\right) \\
&= \frac{2^{n+1}}{3} \cdot \left(1 - \frac{1}{2^{n+1}}\right) \\
&= \frac{1}{3}\left(2^{n+1} - 1\right)
\end{aligned}$$

The final touch

For each n, $f(n)$ is given by:

$$f(n) = \begin{cases} \left(2^{n+1} - 1\right)/3 & \text{if } n \text{ is odd} \\ \left(2^{n+1} - 2\right)/3 & \text{if } n \text{ is even} \end{cases} \tag{9}$$

A geometric approach for the case of odd n

.

Remark that 2^{n-1} is a perfect square whenever n is odd, we set out to represent $f(n)$ as the aggregated area of a shrinking sequence of squares, of successive dimensions

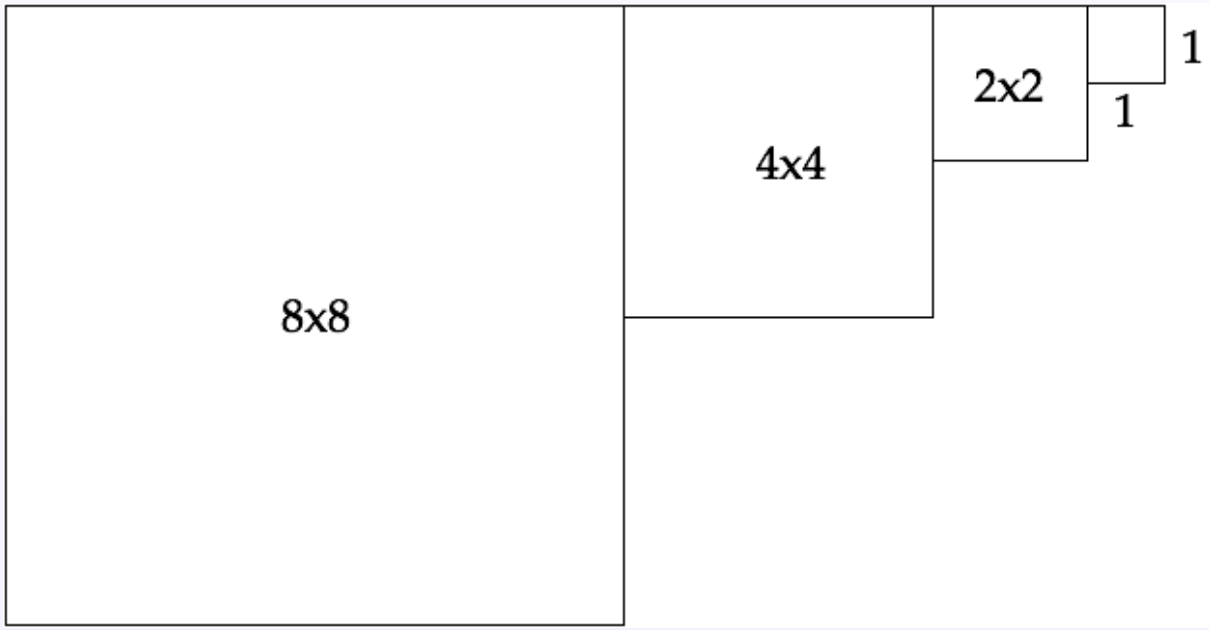
$2^{(n-1)/2} \times 2^{(n-1)/2}$, $2^{(n-3)/2} \times 2^{(n-3)/2}$, $2^{(n-5)/2} \times 2^{(n-5)/2}$, $\ldots$, 1×1

A geometric approach for the case of odd n

.

Remark that 2^{n-1} is a perfect square whenever n is odd, we set out to represent $f(n)$ as the aggregated area of a shrinking sequence of squares, of successive dimensions

$2^{(n-1)/2} \times 2^{(n-1)/2}$, $2^{(n-3)/2} \times 2^{(n-3)/2}$, $2^{(n-5)/2} \times 2^{(n-5)/2}$, $\ldots$, 1×1



We can now use a geometric construction to evaluate $f(n)$ for arbitrary odd n. We take three copies of the cascade.