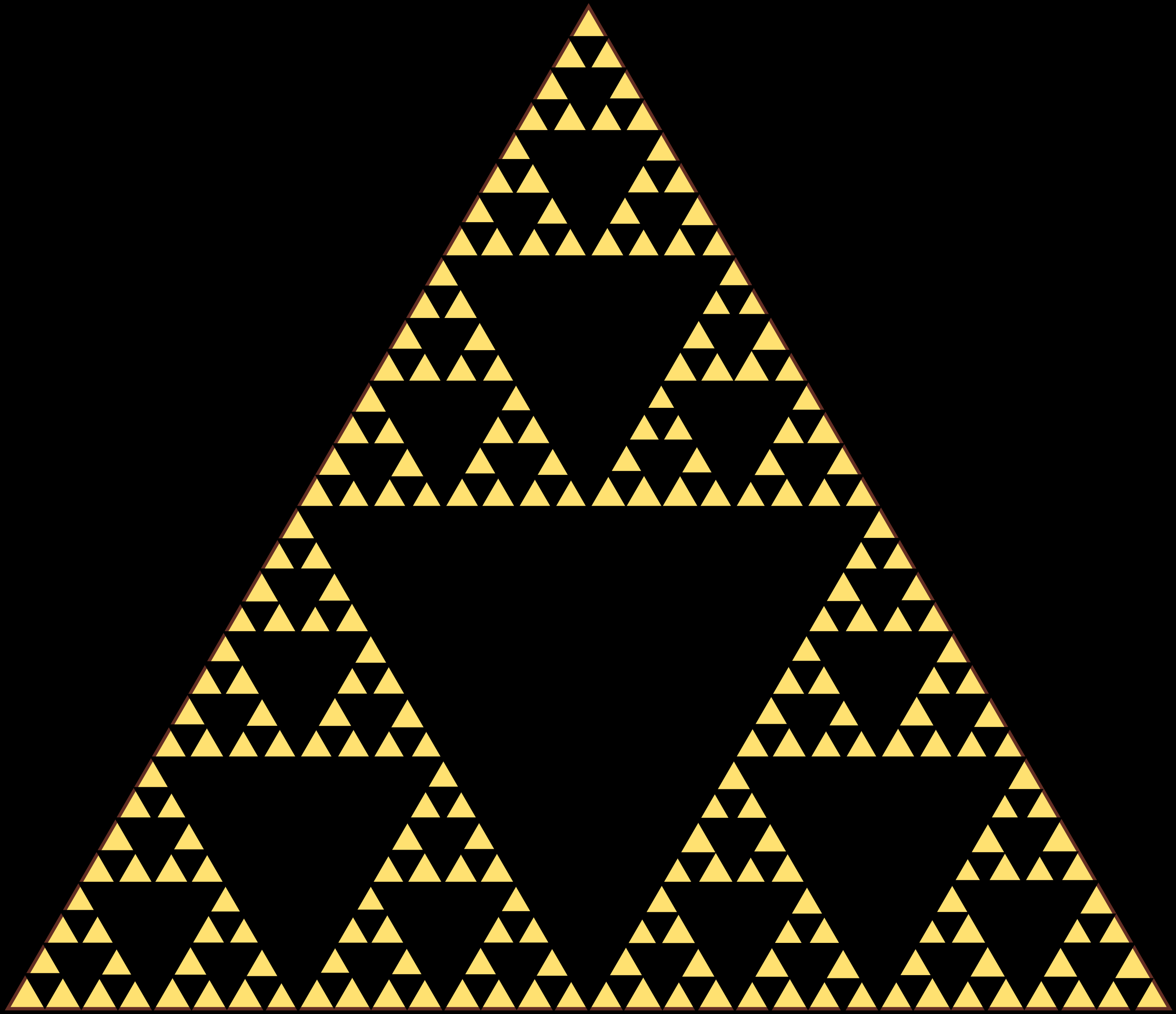
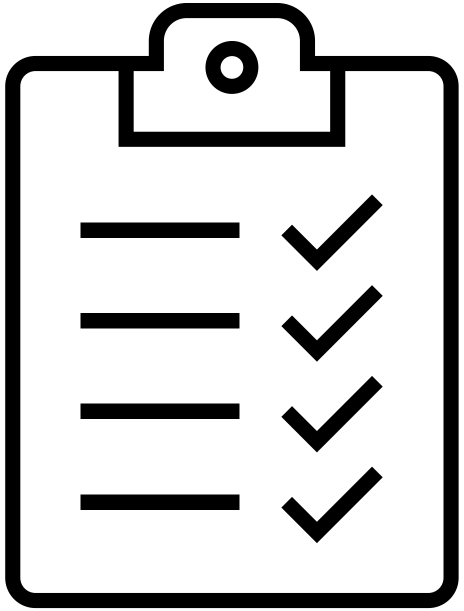
CSCI 2200
Foundations of Computer Science

Lecture 7: Recursion



Agenda



- Overview of recursion
- Recursive sets
- Recursive structures
- Recursive functions

Reminder: This course is not easy!

- Ask John von Neumann!
- “It is exceptional that one should be able to acquire the understanding of a process without having previously acquired a deep familiarity with running it, with using it, before one has assimilated it in an instinctive and empirical way.”
- “In mathematics, you do not understand things. You just get used to them.”

Google

recursion

All | Images | News | Videos | Shopping

See results about

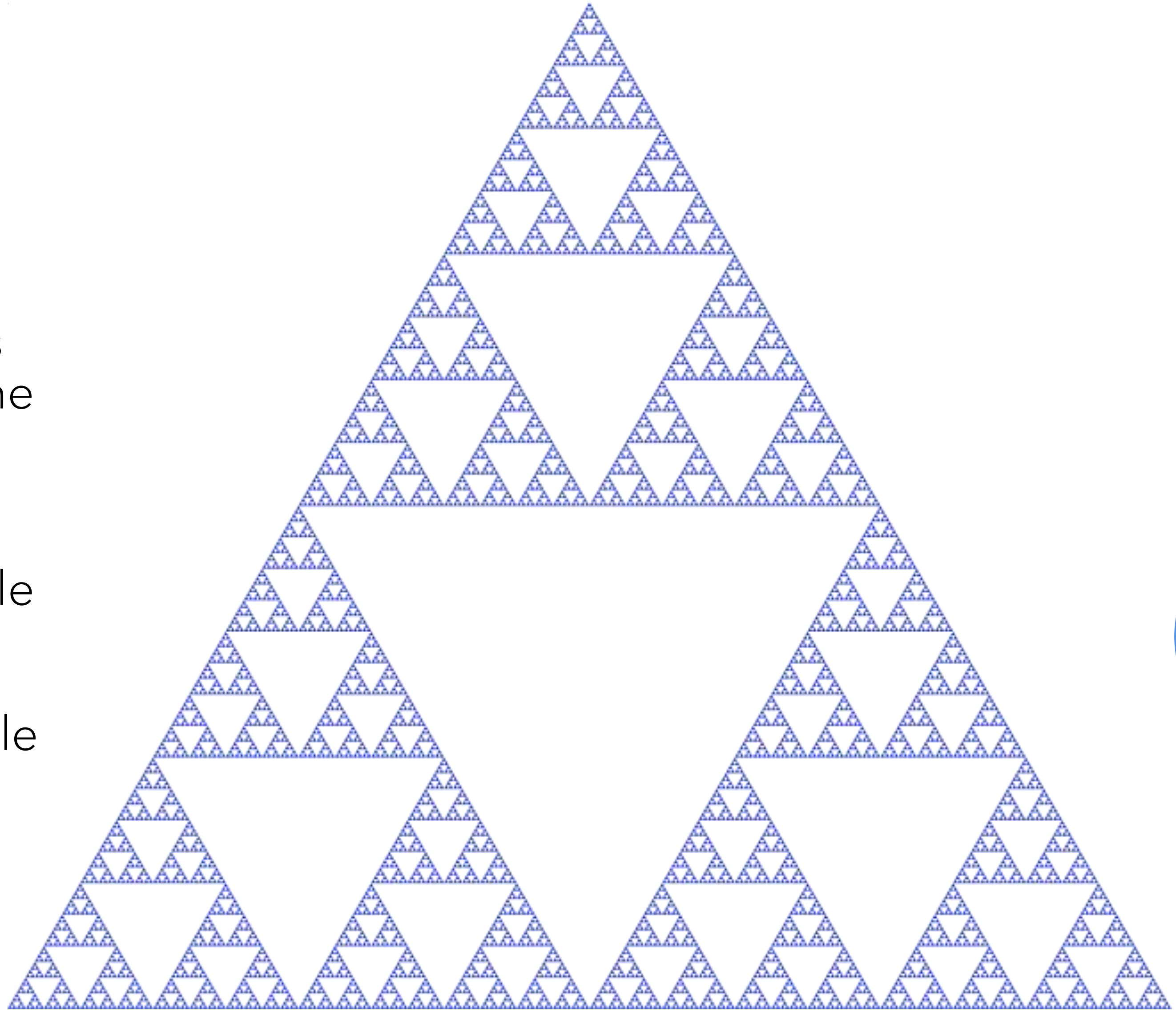
Recursion
Computer science

Did you mean: ***recursion***

Sierpinski gasket

The Sierpinski triangle (or gasket) is an example of a *fractal*: a self-similar structure in which the entire structure is contained in a smaller component of the structure.

- If you zoom in on one of the three subsections, it's identical to the whole thing.
- How do you draw a Sierpinski triangle of size S? First, draw three Sierpinski triangles of size S/2…



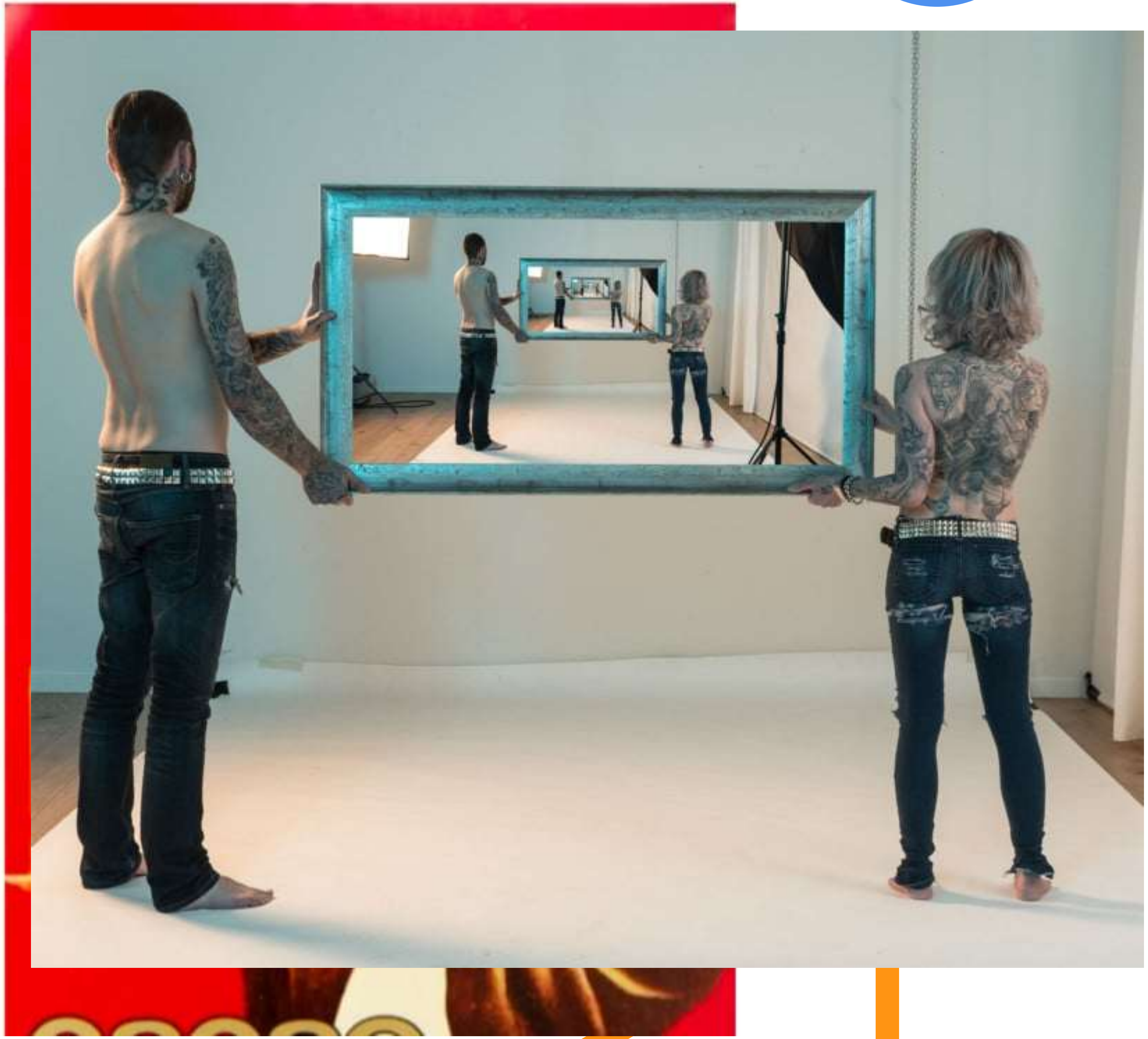
Recursion

- More generally, *recursion* refers to any situation where we use an object in its own definition. The objects in question can be functions, structures, computer programs, what have you.
- Recursion is one of the fundamental concepts in computer science, and one that you're going to have to get comfortable with.

A familiar example

- Fibonacci numbers!
- How do you calculate the 20^{th} Fibonacci number?
  - Add the 18^{th} and 19^{th}!
  - More generally, $F(n) = F(n-1) + F(n-2)$.
  - Notice: the definition of F involves F.
  - What's the catch?

The Droste effect



Where does it end? (Or begin?)

- Just like induction, recursion doesn't make sense unless you have a base case!
- What's the base case for Fibonacci?
    - $F(1) = 1$
    - and $F(2) = 1$
    - Now F is well defined for all $\mathbb{N}$

Using it in a program

```
def fib(n):
    return fib(n-1) + fib(n-2)
```

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

Recursive sets

- Let Σ^* be the set of all finite binary strings. Then we can define Σ^* as follows:
  - $\varepsilon \in \Sigma^*$
  - $x \in \Sigma^* \Rightarrow (x \cdot 0 \in \Sigma^*) \land (x \cdot 1 \in \Sigma^*)$ (dot is concatenate)
- Consider the set $\mathbb{N}$.
  - $1 \in \mathbb{N}$
  - $x \in \mathbb{N} \Rightarrow (x + 1) \in \mathbb{N}$
- … seems like almost everything can be defined recursively. 

Your turn!

Write a recursive definition for the set of all binary palindromes (that is, the set of all binary strings that read the same forwards and backwards).

Hint: Multiple base cases!

Your turn!

Write a recursive definition for the set of all binary palindromes (that is, the set of all binary strings that read the same forwards and backwards).

Hint: Multiple base cases!

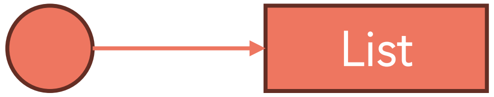
$\varepsilon \in P; 0 \in P; 1 \in P$

$x \in P \Rightarrow 0 \cdot x \cdot 0 \in P$

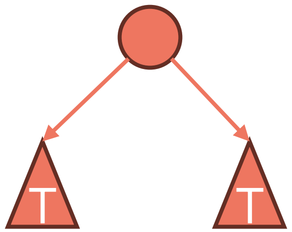
$x \in P \Rightarrow 1 \cdot x \cdot 1 \in P$

Recursive structures

- Linked lists:
  - An empty list is a LL.
  - Anything consisting of a single node followed by a LL is itself a LL.



- Binary trees:
  - An empty tree is a BT. A single node (root) is a BT.
  - Anything consisting of a single node (root) with BTs as left and right children is itself a BT.



- … seems like almost everything can be defined recursively.

Recursive functions (aka recurrence relations)

- Consider this function:
  - f(0) = 0
  - f(n) = f(n-1) + 2n – 1, for all $n \geq 1$
- What does this function calculate?
- How do you prove it?

Unrolling a recursion

- We can "stack" our recursive steps…

$$
\begin{aligned}
f(n) &= \cancel{f(n-1)} + 2n - 1 \\
\cancel{f(n-1)} &= \cancel{f(n-2)} + 2n - 3 \\
\cancel{f(n-2)} &= f(n-3) + 2n - 5 \\
&\cdots \\
\cancel{f(2)} &= \cancel{f(1)} + 3 \\
\cancel{f(1)} &= f(0) + 1
\end{aligned}
$$

Looks like $f(n) = \sum_{i=1}^{n} 2i - 1$ - can prove via induction.

Your turn!

(a) Write a recursive definition for the Fibonacci sequence.

(b) Prove via induction that $F_1 + F_2 + \ldots + F_n = F_{n+2} - 1$ for all $n \geq 1$.

Your turn!

(a) Write a recursive definition for the Fibonacci sequence.

(b) Prove via induction that $F_1 + F_2 + \ldots + F_n = F_{n+2} - 1$ for all $n \geq 1$.

$F_1 = 1; F_2 = 1$

$F_n = F_{n-1} + F_{n-2}$, for $n \geq 3$.

Your turn!

(a) Write a recursive definition for the Fibonacci sequence.

(b) Prove via induction that $F_1 + F_2 + \ldots + F_n = F_{n+2} - 1$ for all $n \geq 1$.

$F_1 = 1$; $F_2 = 1$

$F_n = F_{n-1} + F_{n-2}$, for $n \geq 3$.

Assume $F_1 + \ldots + F_n = F_{n+2} - 1$

$F_{n+3} = F_{n+2} + F_{n+1}$ (by def.)

$F_{n+3} = F_1 + \ldots + F_n + 1 + F_{n+1}$ (by I.H.)

$F_{n+3} - 1 = F_1 + \ldots + F_n + F_{n+1}$ ■

Other ways to “solve” recurrences

- Some styles of recurrences have well-known closed-form solutions.
    - Geometric series (e.g. compound interest)
- There is a “Master Theorem” that gives a closed-form solution for MANY recurrence relations.
- …
- … and we’re not covering it here. (Algo or Big Algo?)

Class survey & reminders

- HW 3 due tonight & HW 4 posted today