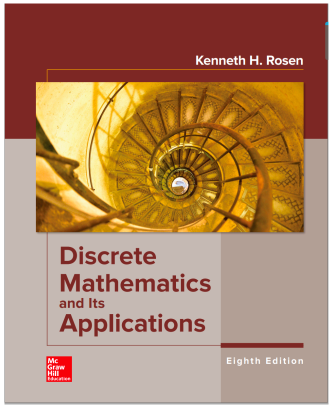
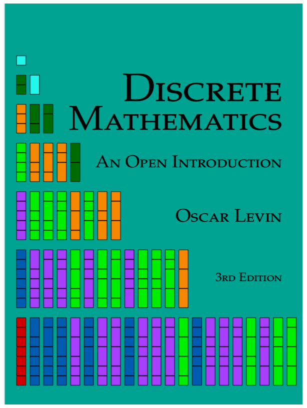
Vietnam National University of HCMC

International University

School of Computer Science and Engineering





Induction and Recursion

Assoc. Prof. Dr. Nguyen Van Sinh

nvsinh@hcmiu.edu.vn

Outline (Sep 27-30th 2022)

- Induction
- Recursion

What is mathematical induction?

- **A method of proof. It does not generate answers: it only can prove them.**
- **It is a method of proving that something holds.**
- **A propositional function P(n) is true for all positive integers n.**

Example:

$$1 + 2 + 3 + \ldots + n = \frac{n(n+1)}{2}, \quad n = 1, 2, 3, \ldots$$

or

$$n \leq 2^n, \quad n = 1, 2, 3, \ldots$$

What is mathematical induction?

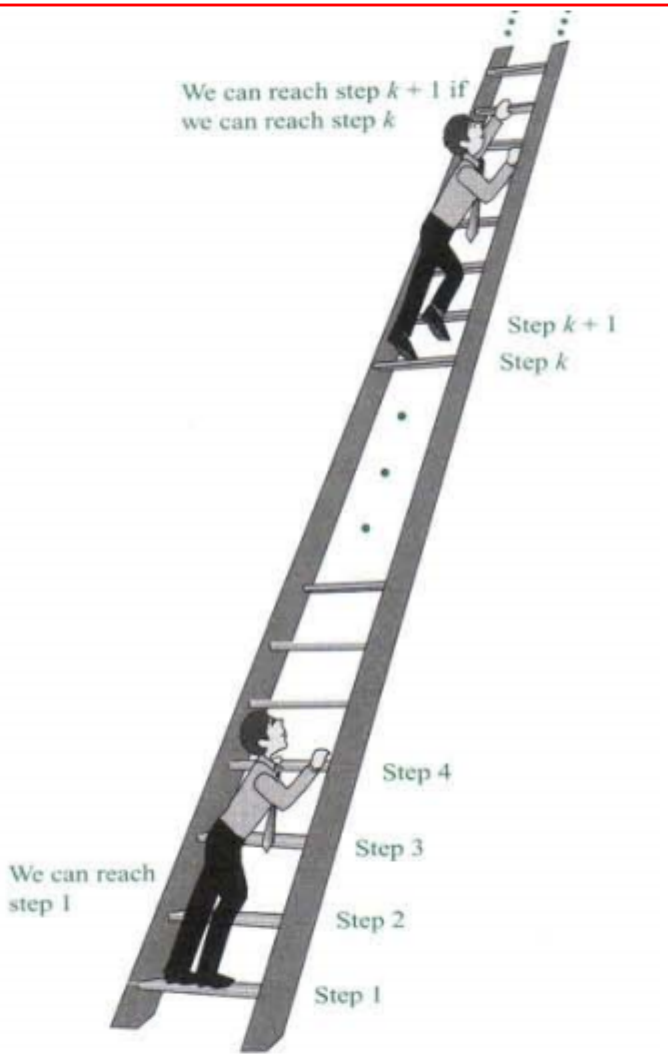
Suppose we have an infinite ladder, and we want to know if we *can reach every step* on this ladder.

We know the following two things:

1. We can reach the base of the ladder
2. If we can reach a particular step, then we can reach the next step

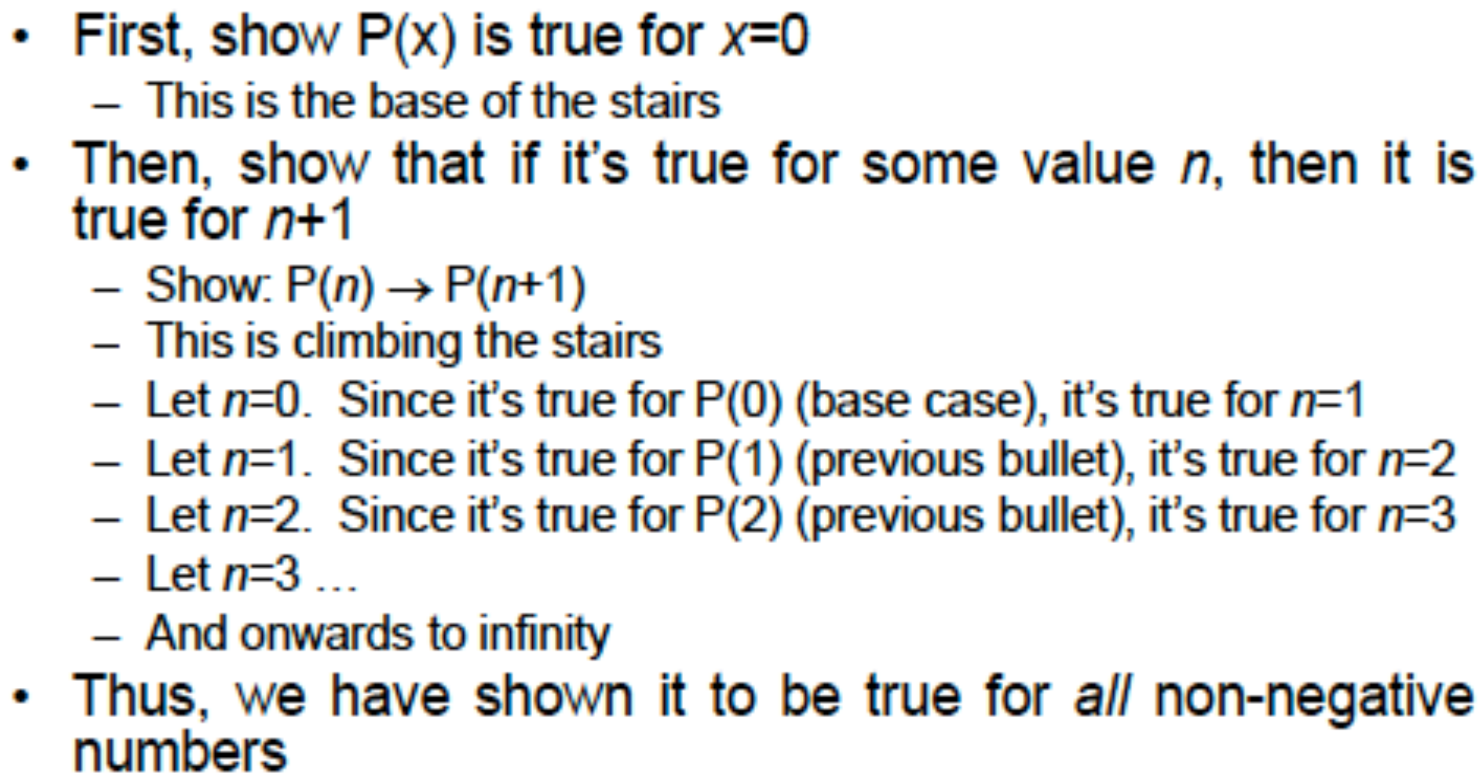
Can we conclude that we can reach every step of the ladder?

What is mathematical induction?



Understanding induction

Suppose we want to prove that P(x) holds for all x



- First, show P(x) is true for *x*=0
  - This is the base of the stairs
- Then, show that if it's true for some value *n*, then it is true for *n*+1
  - Show: P(*n*) → P(*n*+1)
  - This is climbing the stairs
  - Let *n*=0. Since it's true for P(0) (base case), it's true for *n*=1
  - Let *n*=1. Since it's true for P(1) (previous bullet), it's true for *n*=2
  - Let *n*=2. Since it's true for P(2) (previous bullet), it's true for *n*=3
  - Let *n*=3 …
  - And onwards to infinity
- Thus, we have shown it to be true for *all* non-negative numbers

Proof structure

Three parts:

- Base case(s): show it is true for one element
  - (get to the stair's base platform)
- Inductive hypothesis: assume it is true for any given element
  - (assume you are on a stair)
  - **Must be clearly labeled!!!**
- Show that if it true for the next higher element
  - (show you can move to the next stair)

Induction example

- Show that the sum of the first *n* odd integers is n^2
  - Example: If *n* = 5, 1+3+5+7+9 = 25 = 5^2
  - Formally, Show

$$\forall n \; \mathrm{P}(n) \text{ where } \mathrm{P}(n) = \sum_{i=1}^{n} 2i - 1 == n^2$$

- Base case: Show that P(1) is true

$$\mathrm{P}(1) = \sum_{i=1}^{1} 2(i) - 1 == 1^2$$

$$= \; 1 == 1$$

Example continued

- Inductive hypothesis: assume true for *k*
  - Thus, we assume that P(*k*) is true, or that

$$\sum_{i=1}^{\boxed{k}} 2i-1 = \boxed{k}^2$$

  - Note: we don't yet know if this is true or not!

- Inductive step: show true for *k*+1
  - We want to show that:

$$\sum_{i=1}^{\boxed{k+1}} 2i-1 = \boxed{(k+1)}^2$$

Example continued

- Recall the inductive hypothesis: $\sum_{i=1}^{k} 2i-1 = k^2$

- Proof of inductive step:

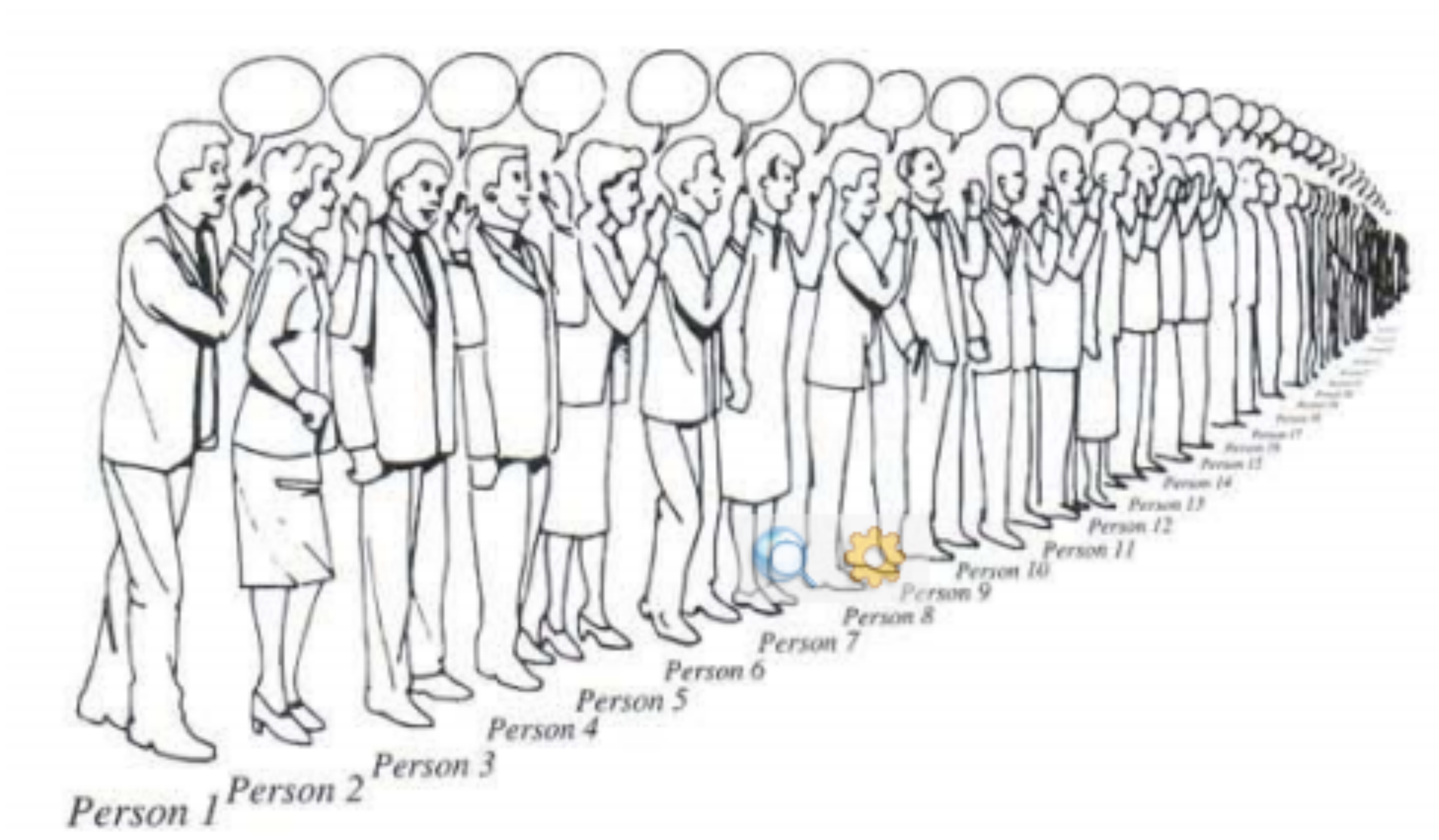
$$\sum_{i=1}^{k+1} 2i-1 = (k+1)^2$$

$$2(k+1)-1+\boxed{\sum_{i=1}^{k} 2i-1} = k^2+2k+1$$

$$2(k+1)-1+\boxed{k^2} = k^2+2k+1$$

$$k^2+2k+1 = k^2+2k+1$$

Example: people telling secrets



Example: Falling Dominos

What did we show?

- Base case: P(1)
- If P(*k*) was true, then P(*k*+1) is true
  - i.e., P(*k*) → P(*k*+1)

- We know it's true for P(1)
- Because of P(*k*) → P(*k*+1), if it's true for P(1), then it's true for P(2)
- Because of P(*k*) → P(*k*+1), if it's true for P(2), then it's true for P(3)
- Because of P(*k*) → P(*k*+1), if it's true for P(3), then it's true for P(4)
- Because of P(*k*) → P(*k*+1), if it's true for P(4), then it's true for P(5)
- And onwards to infinity

- Thus, it is true for all possible values of *n*

- In other words, we showed that:

$$\left[\mathrm{P}(1) \wedge \forall k\left(\mathrm{P}(k) \rightarrow \mathrm{P}(k+1)\right)\right] \rightarrow \forall n\, \mathrm{P}(n)$$

Example 2

- Show the sum of the first *n* positive even integers is $n^2 + n$
  - Rephrased:

$$\forall n\, P(n) \quad \text{where } P(n) = \sum_{i=1}^{n} 2i == n^2 + n$$

- The three parts:
  - Base case
  - Inductive hypothesis
  - Inductive step

Example continued

- Base case: Show P(1):

$$P(1) = \sum_{i=1}^{1} 2(i) == 1^2 + 1$$
$$= 2 == 2$$

- Inductive hypothesis: Assume

$$P(k) = \sum_{i=1}^{k} 2i == k^2 + k$$

- Inductive step: Proof that:

$$P(k+1) = \sum_{i=1}^{k+1} 2i == (k+1)^2 + (k+1)$$

Example continued

- Recall our inductive hypothesis:

$$P(k) = \sum_{i=1}^{k} 2i == k^2 + k \quad (1)$$

Proof that:

$$\sum_{i=1}^{k+1} 2i == (k+1)^2 + k + 1$$

$$2(k+1) + \boxed{\sum_{i=1}^{k} 2i} \quad \text{substitute "red rectangle" by (1)}$$

$$2(k+1) + \boxed{k^2 + k}$$

$$k^2 + 3k + 2 == k^2 + 3k + 2$$

Example 3

- S that $n! < n^n$ for all $n > 1$

- Base case: $n = 2$

  $2! < 2^2$

  $2 < 4$

- Inductive hypothesis: assume $k! < k^k$
- Inductive step: show that $(k+1)! < (k+1)^{k+1}$

$$(k+1)! = (k+1)\boxed{k!} < (k+1)\boxed{k^k} < (k+1)(k+1)^k = (k+1)^{k+1}$$

Strong induction

- Weak mathematical induction proves P(0) is true and assumes P(*k*) is true, and uses that (and only that!) to show P(*k*+1) is true

- Strong mathematical induction proves P(0), P(1), …, P(b) are true, and assumes P(*k*-*b*), P(*k*-*b*+1), …, P(*k*) are all true, and uses that to show that P(*k*+1) is true.

$$\left[P(k-b) \wedge P(k-b+1) \wedge P(k-b+2) \wedge \ldots \wedge P(k)\right] \rightarrow P(k+1)$$

Example

Show that every postage amount 12 cents or more can be formed using only the coins 4 and 5 cent stamps?

Proof using Mathematical Induction

- Show base case: P(12):
  - 12 = 4 + 4 + 4
- Inductive hypothesis: Assume P(k) is true
- Inductive step: Show that P(k+1) is true
  - If P(k) uses a 4 cent stamp, replace that stamp with a 5 cent stamp to obtain $P(k+1)$
  - If P(k) does not use a 4 cent stamp, it must use only 5 cent stamps
    - Since $k > 10$, there must be at least three 5 cent stamps
    - Replace these with four 4 cent stamps to obtain k+1
- Note that only $P(k)$ was assumed to be true

Same Proof using Strong Induction

- Show base cases: P(12), P(13), P(14), and P(15)
  - 12 = 4 + 4 + 4
  - 13 = 4 + 4 + 5
  - 14 = 4 + 5 + 5
  - 15 = 5 + 5 + 5
- Inductive hypothesis: Assume P(*k*-3), P(*k*-2), P(*k*-1), P(*k*) are all true
  - For $k \geq 15$
- Inductive step: Show that P(*k*+1) is true
  - We will obtain P(*k*+1) by adding a 4 cent stamp to P(*k*+1-4)
  - Since we know P(*k*+1-4) = P(*k*-3) is true, our proof is complete
- Note that P(*k*-3), P(*k*-2), P(*k*-1), P(*k*) were all assumed to be true

Errors in Induction

For all positive integers n, $3^n - 2$ is even.

P(n): $3^n - 2$ is even.

Induction hypothesis: Assume P(n) is true

Inductive case: We want to show P(n+1) is true.

Because $3^{n+1} - 2 = 3*3^n - 2 = 2*3^n + (3^n - 2)$

- $2*3^n$ is even
- $(3^n - 2)$ is even by Induction hypothesis

So $3^{n+1} - 2$ is even.

Errors in Induction

- A camel can always carry all the straw in a barn.
- P(n): camel can carry n straws
- Base case: P(1) is true
- Induction hypothesis: Assume P(n-1) is true
- Inductive case: Since the camel can carry n-1 straws, it has no problem to carry one more straw.

Recursion

Recursion means defining something, such as a function, in terms of itself

- For example, let $f(x) = x!$
- We can define f(x) *as* $f(x) = x * f(x-1)$

Recursive definition

Two parts of a recursive definition:

Base case and a Recursive step

.

Example: the set of positive integers

- Basis step: $1 \in S$
- Recursive step: if $x \in S$, then $x+1 \in S$

The set of odd positive integers

- $1 \in S$
- If $x \in S$, then $x+2 \in S$

Recursion example

– Find $f(1)$, $f(2)$, $f(3)$, and $f(4)$, where $f(0) = 1$

a) Let $f(n+1) = f(n) + 2$
- $f(1) = f(0) + 2 = 1 + 2 = 3$
- $f(2) = f(1) + 2 = 3 + 2 = 5$
- $f(3) = f(2) + 2 = 5 + 2 = 7$
- $f(4) = f(3) + 2 = 7 + 2 = 9$

b) Let $f(n+1) = 3f(n)$
- $f(1) = 3 * f(0) = 3*1 = 3$
- $f(2) = 3 * f(1) = 3*3 = 9$
- $f(3) = 3 * f(2) = 3*9 = 27$
- $f(4) = 3 * f(3) = 3*27 = 81$

Fibonacci sequence

Definition of the Fibonacci sequence

- Non-recursive: $$F(n) = \frac{\left(1+\sqrt{5}\right)^n - \left(1-\sqrt{5}\right)^n}{\sqrt{5}\cdot 2^n}$$
- Recursive: $F(n) = F(n\text{-}1) + F(n\text{-}2)$

  or: $F(n{+}1) = F(n) + F(n\text{-}1)$

Must always specify base case(s)!

- $F(1) = 1$, $F(2) = 1$
- Note that some will use $F(0) = 1$, $F(1) = 1$

Bad recursive definitions

- Consider:
  - $f(0) = 1$
  - $f(n) = 1 + f(n-2)$
  - What is $f(1)$?

- Consider:
  - $f(0) = 1$
  - $f(\mathrm{n}) = 1+f(-\mathrm{n})$
  - What is $\mathrm{f}(1)$?

Why are these definitions bad?

More examples of recursion: defining strings

- Terminology
  - λ is the empty string: “”
  - Σ is the set of all letters: { a, b, c, …, z }
    - The set of letters can change depending on the problem
- We can define a set of strings Σ^* as follows
  - Base step: $\lambda \in \Sigma^*$
  - If $w \in \Sigma^*$ and $x \in \Sigma$, then $wx \in \Sigma^*$
  - Thus, Σ^* s the set of all the possible strings that can be generated with the alphabet

Structural induction

A technique for proving a property of a recursively defined object.
It is very much like an inductive proof, except that in the inductive step we try to show that if the statement holds for each of the element used to construct the new element, then the result holds for the new element too.

Example. Prove that if T is a full binary tree, and h(T) is the height of the tree then the number of elements in the tree $n(T) \leq 2^{h(T)+1} - 1$.

Solution: see pages 377 in the textbook (next slide).

Structural induction…

Proof: We prove this inequality using structural induction.

BASIS STEP: For the full binary tree consisting of just the root r the result is true because $n(T) = 1$ and $h(T) = 0$, so that $n(T) = 1 \leq 2^{0+1} - 1 = 1$.

INDUCTIVE STEP: For the inductive hypothesis we assume that $n(T_1) \leq 2^{h(T_1)+1} - 1$ and $n(T_2) \leq 2^{h(T_2)+1} - 1$ whenever T_1 and T_2 are full binary trees. By the recursive formulae for $n(T)$ and $h(T)$ we have $n(T) = 1 + n(T_1) + n(T_2)$ and $h(T) = 1 + \max(h(T_1), h(T_2))$.

We find that

$$\begin{aligned}
n(T) &= 1 + n(T_1) + n(T_2) && \text{by the recursive formula for } n(T)\\
&\leq 1 + (2^{h(T_1)+1} - 1) + (2^{h(T_2)+1} - 1) && \text{by the inductive hypothesis}\\
&\leq 2 \cdot \max(2^{h(T_1)+1}, 2^{h(T_2)+1}) - 1 && \text{because the sum of two terms is at most 2 times the larger}\\
&= 2 \cdot 2^{\max(h(T_1),h(T_2))+1} - 1 && \text{because } \max(2^x, 2^y) = 2^{\max(x,y)}\\
&= 2 \cdot 2^{h(T)} - 1 && \text{by the recursive definition of } h(T)\\
&= 2^{h(T)+1} - 1.
\end{aligned}$$

This completes the inductive step. ◁

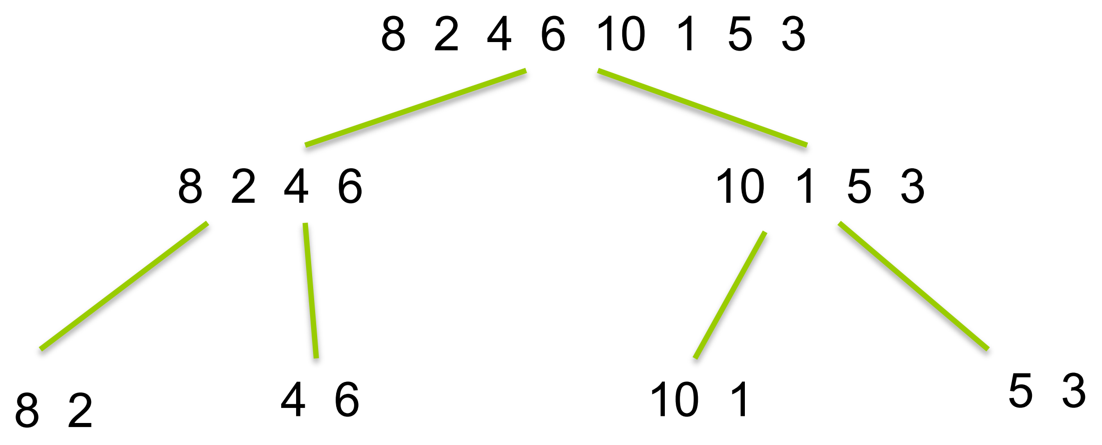
Recursive Algorithm

Example 1. *Given a and n, compute a^n*

```
procedure power (a : real number, n: non-negative integer)
if n = 0 then power (a, n) := 1
          else power (a, n) := a. power (a, n-1)
```

Recursive algorithms: Sorting

Here is the recursive algorithm Merge sort. It **merges** two sorted lists to produce a new sorted list



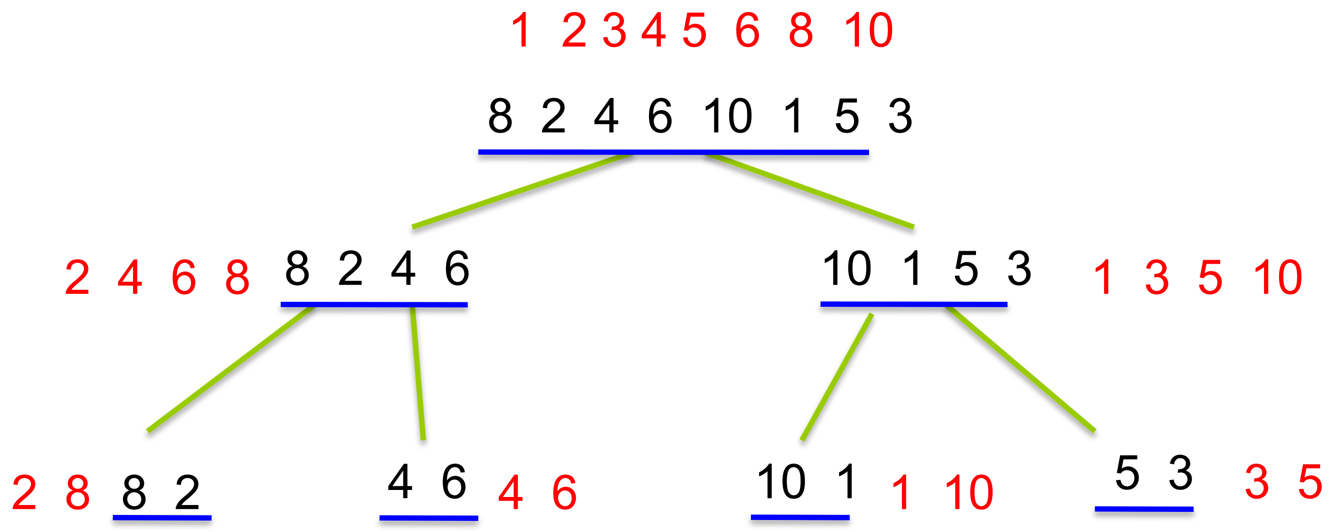
Mergesort

The merge algorithm "merges" two sorted lists

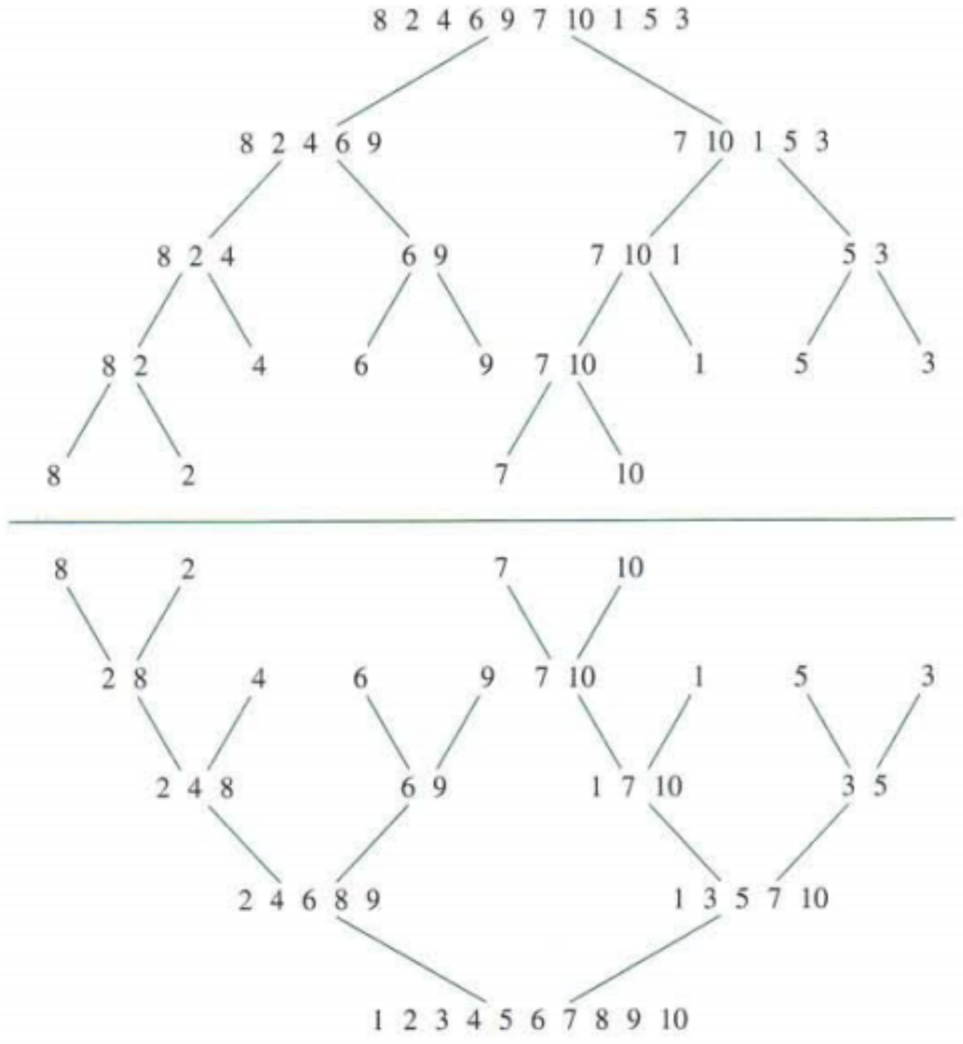
2 4 6 8 merged with 1 3 5 10 will produce 1 2 3 4 5 6 8 10

```
procedure mergesort (L = $a_1, a_2, a_3, \dots a_n$)
if n > 0 then
        m:= n/2
        L1 := $a_1, a_2, a_3, \dots a_m$
        L2 := $a_{m+1}, a_{m+2}, a_{m+3}, \dots a_n$
        L := merge (mergesort(L1), mergesort(L2))
```

Example of Mergesort

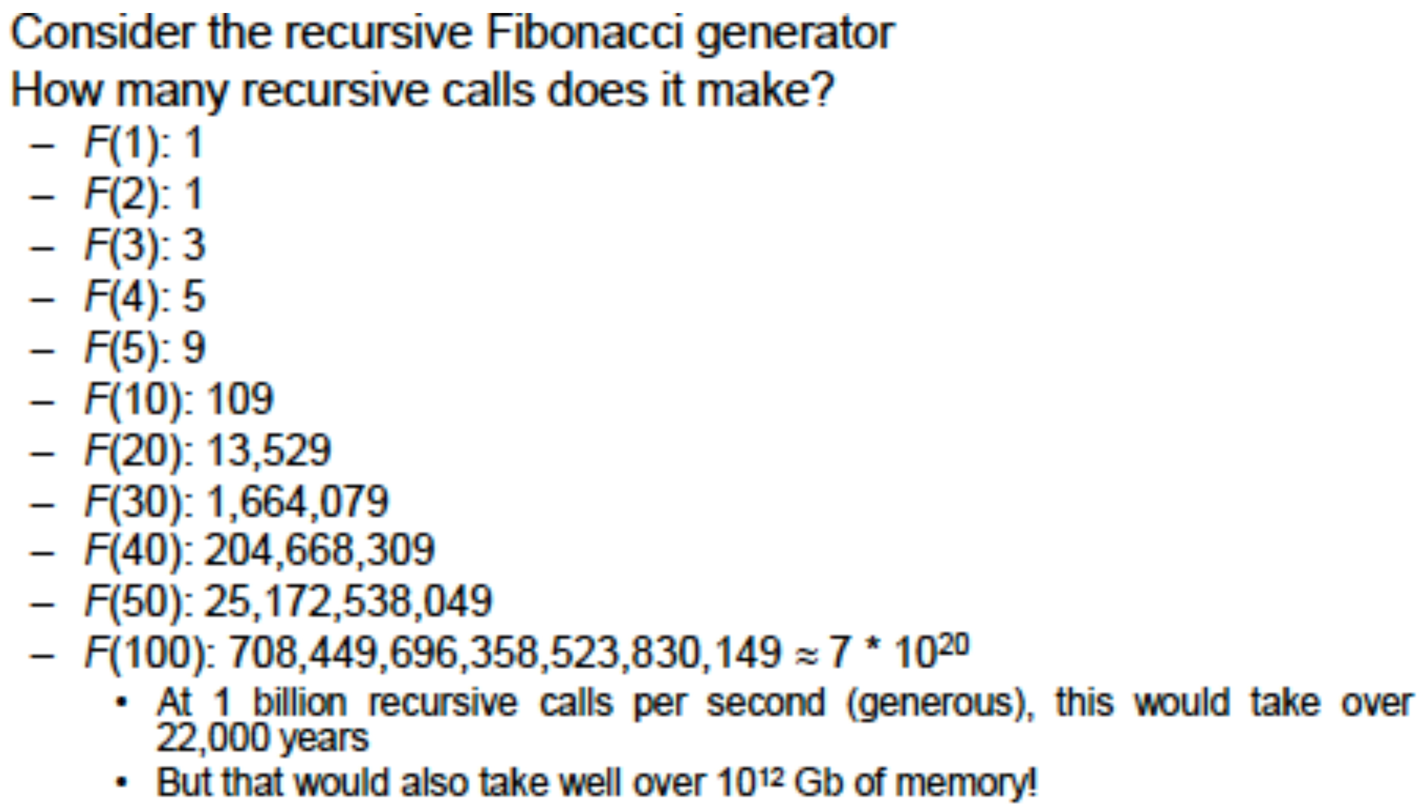


Example of Mergesort



Pros and Cons of Recursion

While recursive definitions are easy to understand



Consider the recursive Fibonacci generator

How many recursive calls does it make?

- *F*(1): 1
- *F*(2): 1
- *F*(3): 3
- *F*(4): 5
- *F*(5): 9
- *F*(10): 109
- *F*(20): 13,529
- *F*(30): 1,664,079
- *F*(40): 204,668,309
- *F*(50): 25,172,538,049
- *F*(100): 708,449,696,358,523,830,149 $\approx 7 * 10^{20}$
  - At 1 billion recursive calls per second (generous), this would take over 22,000 years
  - But that would also take well over 10^{12} Gb of memory!

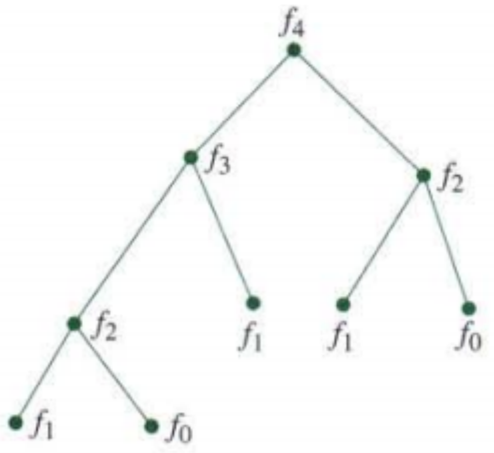
Iterative solutions for Fibonacci sequence are much faster (see check in chapter 5)

Recursive program on computer

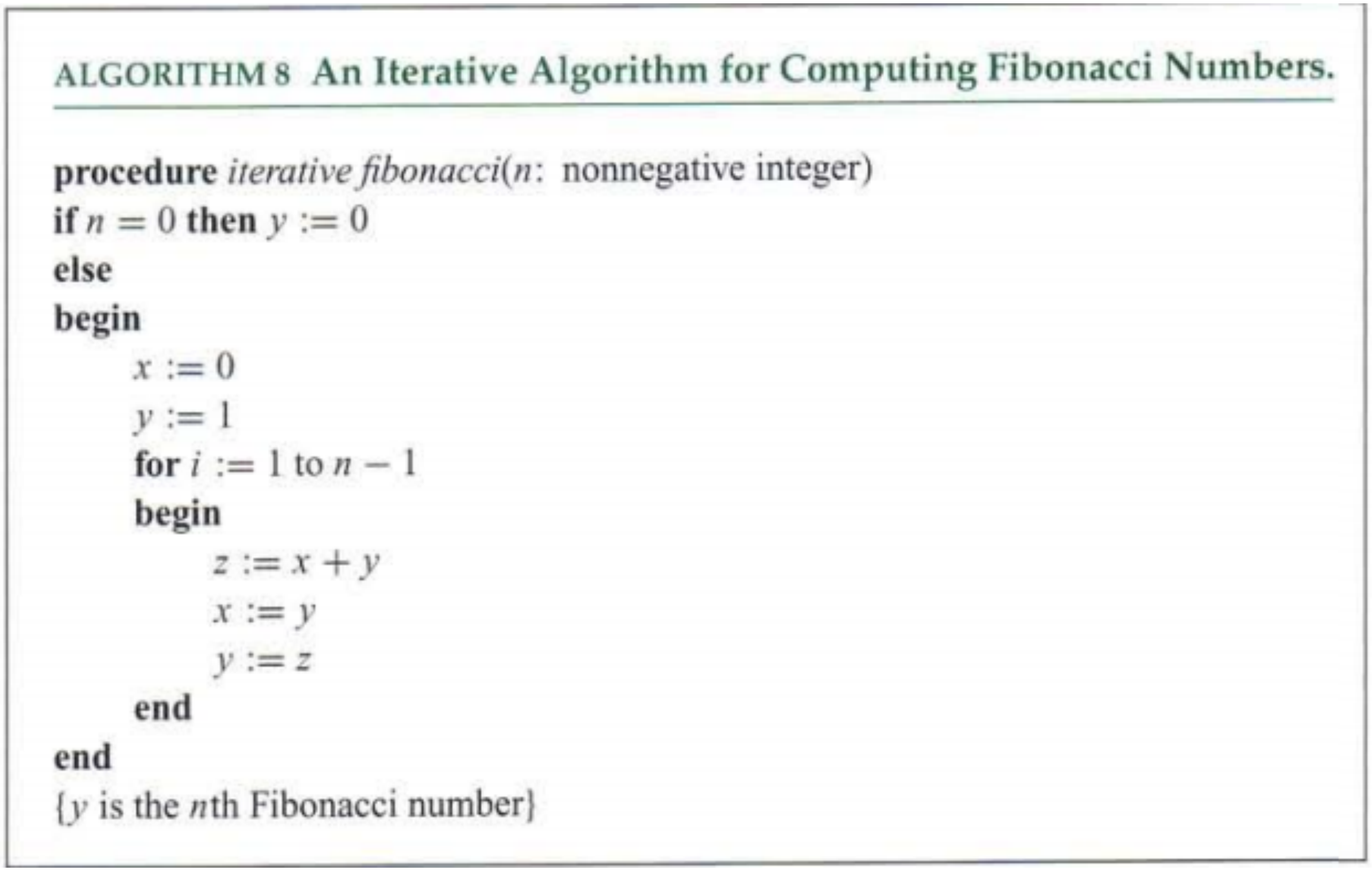
ALGORITHM 7 A Recursive Algorithm for Fibonacci Numbers.

```
procedure fibonacci(n: nonnegative integer)
if $n = 0$ then fibonacci(0) := 0
else if $n = 1$ then fibonacci(1) := 1
else fibonacci(n) := fibonacci(n − 1) + fibonacci(n − 2)
```

Ex: evaluating f_4 recursively



Recursive program on computer



ALGORITHM 8 An Iterative Algorithm for Computing Fibonacci Numbers.

```
procedure iterative fibonacci($n$: nonnegative integer)
if $n = 0$ then $y := 0$
else
begin
    $x := 0$
    $y := 1$
    for $i := 1$ to $n - 1$
    begin
        $z := x + y$
        $x := y$
        $y := z$
    end
end
{$y$ is the $n$th Fibonacci number}
```