ICS141: Discrete Mathematics for Computer Science I

Dept. Information & Computer Sci., University of Hawaii

Jan Stelovsky

based on slides by Dr. Baek and Dr. Still

Originals by Dr. M. P. Frank and Dr. J.L. Gross

Provided by McGraw-Hill

Quiz

1. State the 1st Principle of Mathematical Induction

2. What is the difference between the 1st, 2nd and strong principles of Mathematical Induction. (Describe in plain English)

3. What is the big-O complexity of Euclid's Algorithm?

Lecture 21

Chapter 4. Induction and Recursion

4.3 Recursive Definitions and Structural Induction

Recursive Definitions

University of Hawaii

- In induction, we *prove* all members of an infinite set satisfy some predicate *P* by:
  - proving the truth of the predicate for larger members in terms of that of smaller members.
- In ***recursive definitions***, we similarly *define* a function, a predicate, a set, or a more complex structure over an infinite domain (universe of discourse) by:
  - defining the function, predicate value, set membership, or structure of larger elements in terms of those of smaller ones.

Recursion

- ***Recursion*** is the general term for the practice of defining an object in terms of *itself*
  - or of part of itself.
  - This may seem circular, but it isn't necessarily.
- An inductive proof establishes the truth of $P(k+1)$ *recursively* in terms of $P(k)$.
- There are also recursive *algorithms*, *definitions*, *functions*, *sequences*, *sets*, and other structures.

Recursively Defined Functions

University of Hawaii

- Simplest case: One way to define a function f:$\mathbf{N} \rightarrow S$ (for any set S) or series $a_n = f(n)$ is to:
  - Define $f(0)$
  - For $n > 0$, define $f(n)$ in terms of $f(0),\ldots,f(n-1)$
- **Example**: Define the series $a_n = 2^n$ where n is a nonnegative integer recursively:
  - a_n looks like $2^0, 2^1, 2^2, 2^3,\ldots$
  - Let $a_0 = 1$
  - For $n > 0$, let $a_n = 2 \cdot a_{n-1}$

Another Example

- Suppose we define $f(n)$ for all $n \in \mathbf{N}$ recursively by:
  - Let $f(0) = 3$
  - For all $n > 0$, let $f(n) = 2 \cdot f(n-1) + 3$
- What are the values of the following?
  - $f(1) =$ $2 \cdot f(0) + 3 = 2 \cdot 3 + 3 = 9$
  - $f(2) =$ $2 \cdot f(1) + 3 = 2 \cdot 9 + 3 = 21$
  - $f(3) =$ $2 \cdot f(2) + 3 = 2 \cdot 21 + 3 = 45$
  - $f(4) =$ $2 \cdot f(3) + 3 = 2 \cdot 45 + 3 = 93$

Recursive Definition of Factorial

University of Hawaii

- Give an inductive (recursive) definition of the factorial function,

  $$F(n) = n! = \prod_{1 \le i \le n} i = 1 \cdot 2 \cdots n$$

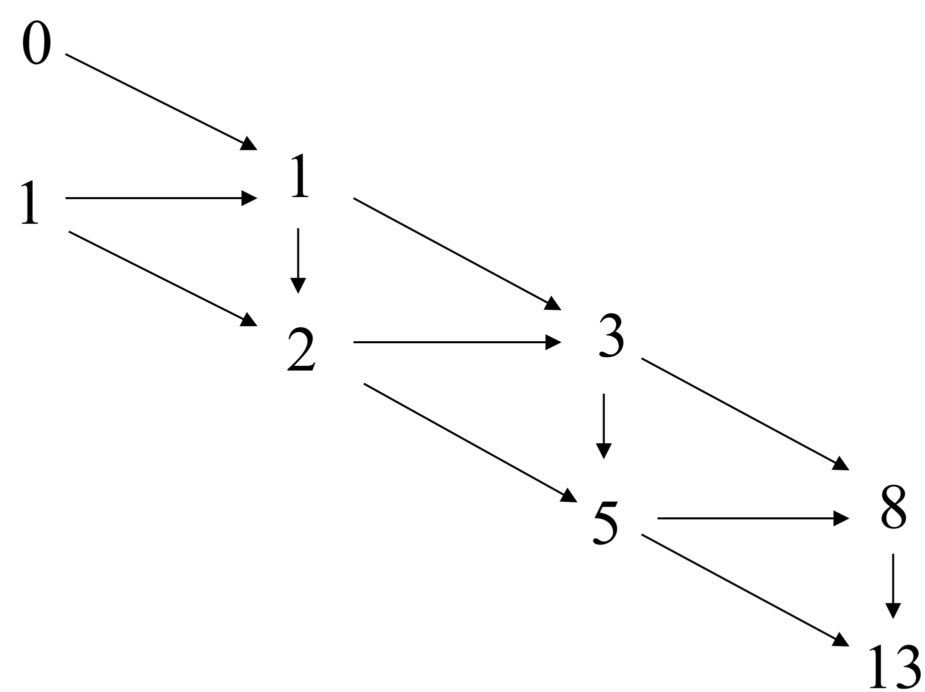
  - Basis step: $F(1) = 1$
  - Recursive step: $F(n) = n \cdot F(n-1)$ for $n > 1$
    - $F(2) = 2 \cdot F(1) = 2 \cdot 1 = 2$
    - $F(3) = 3 \cdot F(2) = 3 \cdot \{2 \cdot F(1)\} = 3 \cdot 2 \cdot 1 = 6$
    - $F(4) = 4 \cdot F(3) = 4 \cdot \{3 \cdot F(2)\} = 4 \cdot \{3 \cdot 2 \cdot F(1)\}$

      $= 4 \cdot 3 \cdot 2 \cdot 1 = 24$

The Fibonacci Numbers

- The ***Fibonacci numbers*** $f_{n \geq 0}$ is a famous series defined by:

$$f_0 = 0, \quad f_1 = 1, \quad f_{n \geq 2} = f_{n-1} + f_{n-2}$$



Inductive Proof about Fibonacci Numbers

- **Theorem:** $f_n < 2^n$. ← Implicitly for all $n \in \mathbf{N}$
- **Proof:** By induction
  - Basis step: $f_0 = 0 < 2^0 = 1$
    $f_1 = 1 < 2^1 = 2$
    Note: use of base cases of recursive definition
  - Inductive step: Use 2nd principle of induction (strong induction).
    Assume $\forall 0 \le i \le k,\ f_i < 2^i$. Then
    $f_{k+1} = f_k + f_{k-1}$ is
    $< 2^k + 2^{k-1}$
    $< 2^k + 2^k = 2^{k+1}$. ■

A Lower Bound on Fibonacci Numbers

University of Hawaii

- **Theorem:** For all integers $n \geq 3$, $f_n > \alpha^{n-2}$, where $\alpha = (1 + 5^{1/2})/2 \approx 1.61803$.
- **Proof.** (Using strong induction.)
  - Let $P(n) = (f_n > \alpha^{n-2})$.
  - **<u>Basis step</u>:**

    For $n = 3$, note that $\alpha^{n-2} = \alpha < 2 = f_3$.

    For $n = 4$, $\alpha^{n-2} = \alpha^2$

    $$= (1 + 2 \cdot 5^{1/2} + 5)/4$$
    $$= (3 + 5^{1/2})/2$$
    $$\approx 2.61803 \qquad (= \alpha + 1)$$
    $$< 3 = f_4.$$

A Lower Bound on Fibonacci Numbers: Proof Continues…

University of Hawaii

- **<u>Inductive step</u>:** For $k \geq 4$, assume $P(j)$ for $3 \leq j \leq k$, prove $P(k+1)$.
  - $f_{k+1} = f_k + f_{k-1} > \alpha^{k-2} + \alpha^{k-3}$ (by inductive hypothesis, $f_{k-1} > \alpha^{k-3}$ and $f_k > \alpha^{k-2}$).
  - Note that $\alpha^2 = \alpha + 1$.
    since $(3 + 5^{1/2})/2 = (1 + 5^{1/2})/2 + 1$
  - Thus, $\alpha^{k-1} = \alpha^2\alpha^{k-3} = (\alpha + 1)\alpha^{k-3}$
    $= \alpha\alpha^{k-3} + \alpha^{k-3} = \alpha^{k-2} + \alpha^{k-3}$.
  - So, $f_{k+1} = f_k + f_{k-1} > \alpha^{k-2} + \alpha^{k-3} = \alpha^{k-1}$.
  - Thus $P(k+1)$. ■

University of Hawaii

Recursively Defined Sets

- An infinite set *S* may be defined recursively, by giving:
  - A small finite set of *base* elements of *S*.
  - A rule for constructing new elements of *S* from previously-established elements.
  - Implicitly, *S* has no other elements but these.
- **Example:** Let $\overbrace{3\in S}^{\text{base element (basis step)}}$, and let $\overbrace{x+y\in S \text{ if } x,y\in S}^{\text{construction rule (recursive step)}}$. What is *S*?

Example *cont.*

- Let $3 \in S$, and let $x+y \in S$ if $x,y \in S$. What is S?
  - $3 \in S$ (*basis step*)
  - 6 (= 3 + 3) is in S (*first application of recursive step*)
  - 9 (= 3 + 6) and 12 (= 6 + 6) are in S (*second application of the recursive step*)
  - 15 (= 3 + 12 or 6 + 9), 18 (= 6 + 12 or 9 + 9), 21 (= 9 + 12), 24 (= 12 + 12) are in S (*third application of the recursive step*)
  - … so on
  - Therefore, $S = \{3, 6, 9, 12, 15, 18, 21, 24, \ldots\}$
    = set of *all positive multiples of* 3

The Set of All Strings

University of Hawaii

- Given an alphabet $\mathbf{\Sigma}$, the set $\mathbf{\Sigma}^*$ of all strings over $\mathbf{\Sigma}$ can be recursively defined by:
  - **Basis step**: $\lambda \in \mathbf{\Sigma}^*$ (λ : empty string)
  - **Recursive step**: $(w \in \mathbf{\Sigma}^* \wedge x \in \mathbf{\Sigma}) \rightarrow wx \in \mathbf{\Sigma}^*$
- **Example**: If $\Sigma = \{0, 1\}$ then
  - $\lambda \in \Sigma^*$ (*basis step*)
  - 0 and 1 are in Σ^* (*first application of recursive step*)
  - 00, 01, 10, and 11 are in Σ^* (*second application of the recursive step*)
  - … so on
  - Therefore, Σ^* consists of all finite strings of 0's and 1's together with the empty string

String: Example

University of Hawaii

- Show that if $\Sigma = \{a, b\}$ then aab is in Σ^*.

  Proof: We construct it with a finite number of applications of the basis and recursive steps in the definition of Σ^*:

1. $\lambda \in \Sigma^*$ by the basis step.
2. By step 1, the recursive step in the definition of Σ^* and the fact that $a \in \Sigma$, we can conclude that $\lambda a = a \in \Sigma^*$.

Proof *cont.*

3. Since $a \in \Sigma^*$ from step 2, and $a \in \Sigma$, applying the recursive step again we conclude that $aa \in \Sigma^*$.

4. Since $aa \in \Sigma^*$ from step 3 and $b \in \Sigma$, applying the recursive step again we conclude that $aab \in \Sigma^*$.

- Since we have shown $aab \in \Sigma^*$ with a finite number of applications of the basis and recursive steps in the definition we have finished the proof.

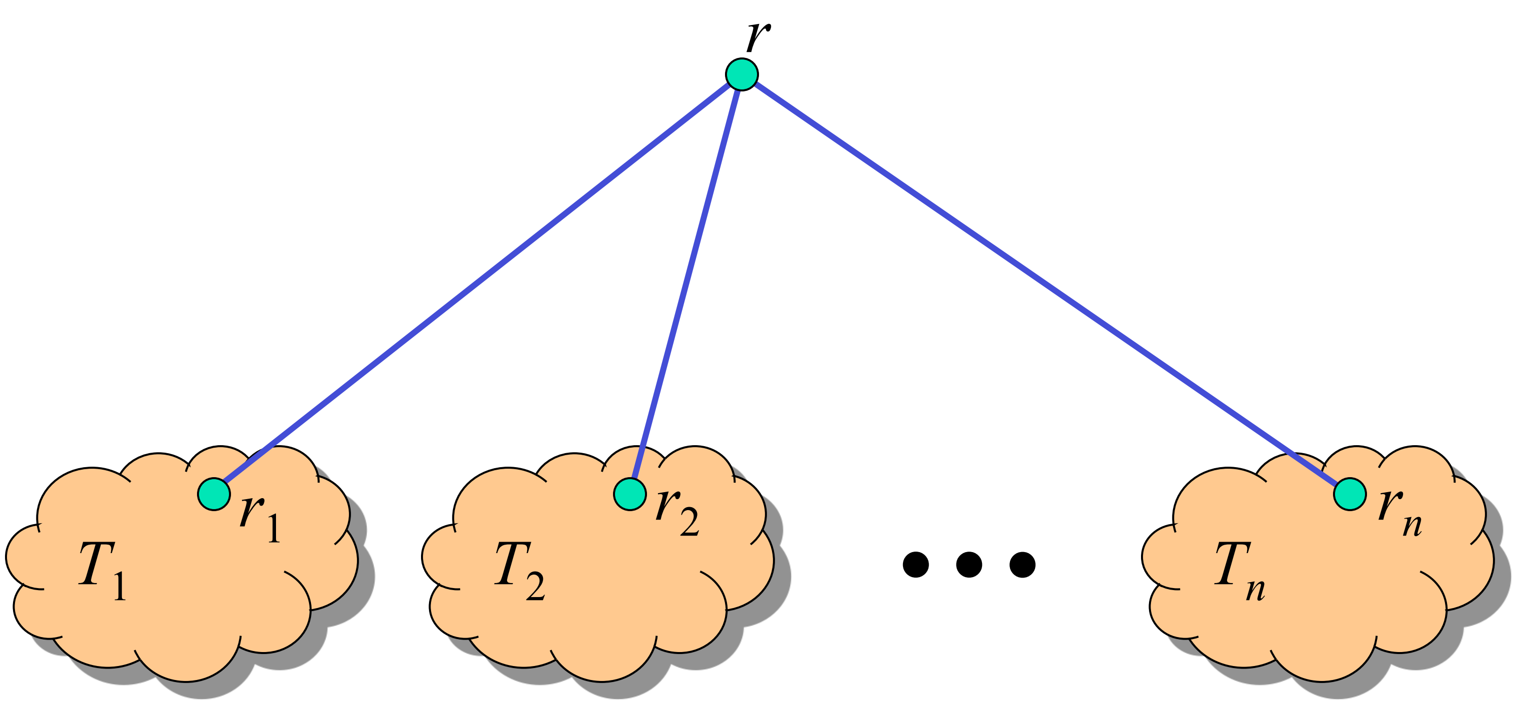
Rooted Trees

University of Hawaii

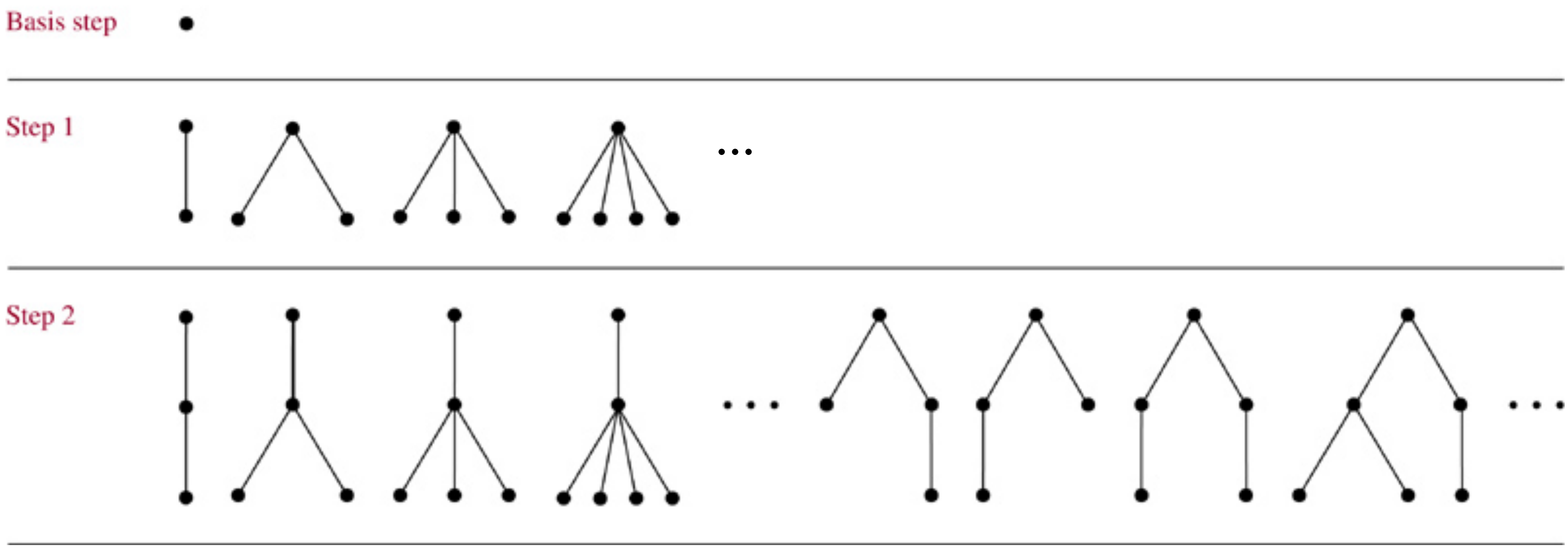
- Trees will be covered in more depth in chapter 10.
  - Briefly, a tree is a graph in which there is exactly one undirected path between each pair of nodes.
  - An undirected graph can be represented as a set of unordered pairs (called *arcs*) of objects called *nodes*.
- Definition of the set of rooted trees:
  - **Basis step**: Any single node r is a rooted tree.
  - **Recursive step**: If $T_1, \ldots, T_n$ are disjoint rooted trees with respective roots $r_1, \ldots, r_n$, and r is a node not in any of the T_i's, then another rooted tree is $\{(r, r_1), \ldots, (r, r_n)\} \cup T_1 \cup \cdots \cup T_n$.

Illustrating Rooted Tree Definition

- How rooted trees can be combined to form a new rooted tree…



Building Up Rooted Trees



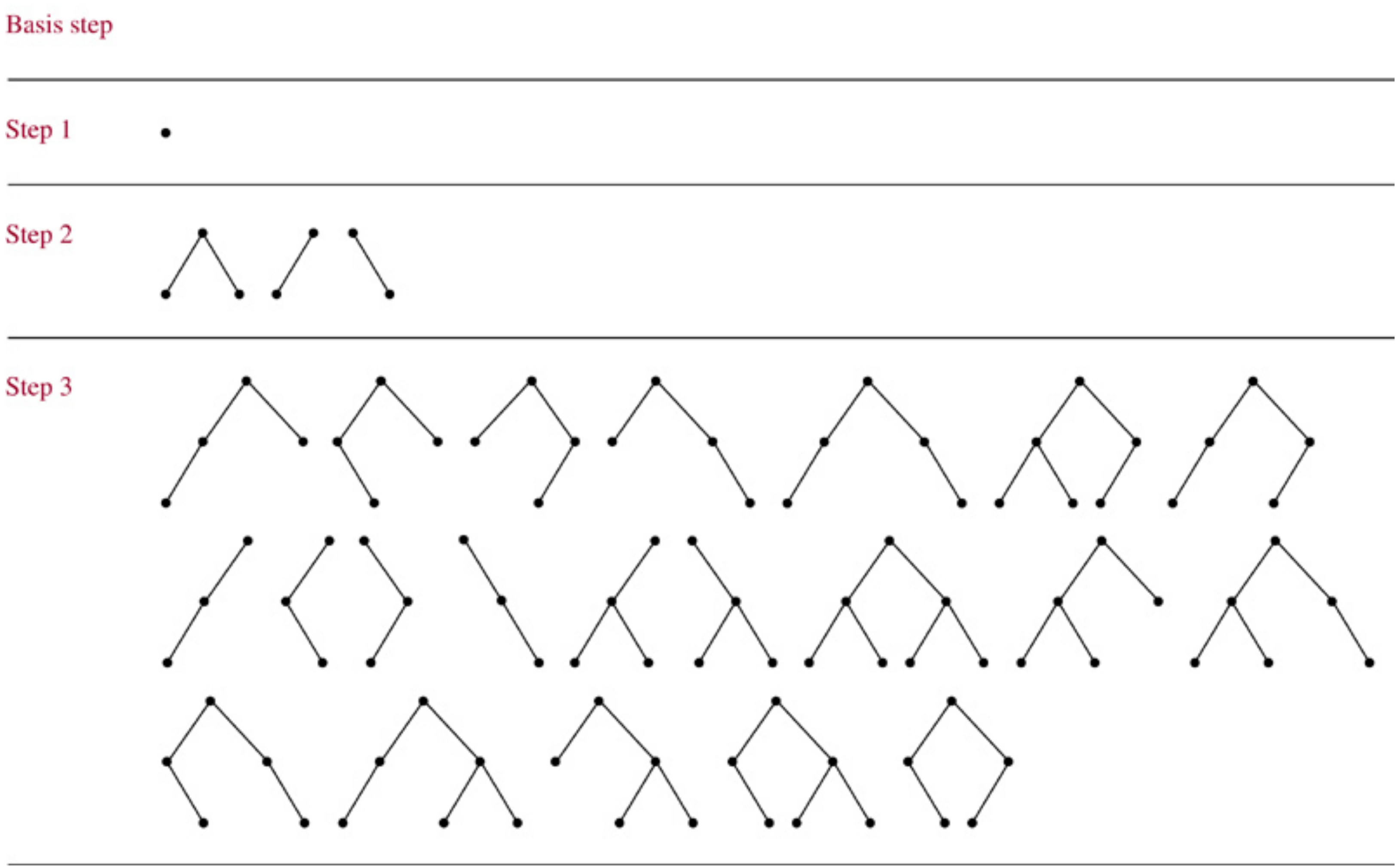
Extended Binary Trees

University of Hawaii

- A special case of rooted trees.
- Recursive definition of extended binary trees:
  - **Basis step**: The empty set $\emptyset$ is an extended binary tree.
  - **Recursive step**: If T_1, T_2 are disjoint extended binary trees, then $e_1 \cup e_2 \cup T_1 \cup T_2$ is an extended binary tree, where $e_1 = \emptyset$ if $T_1 = \emptyset$, and $e_1 = \{(r, r_1)\}$ if $T_1 \neq \emptyset$ and has root r_1, and similarly for e_2. (T_1 is the left subtree and T_2 is the right subtree.)

Building Up Extended Binary Trees

University of Hawaii



Lamé's Theorem

University of Hawaii

- **Theorem:** $\forall a,b \in \mathbf{N}$, $a \geq b > 0$, and let n be the number of steps Euclid's algorithm needs to compute gcd(a,b).
  Then $n \leq 5k$, where $k = \lfloor \log_{10} b \rfloor + 1$ is the number of decimal digits in b.
  - Thus, Euclid's algorithm is linear-time in the number of digits in b.
    (or, Euclid's algorithm is O(log a))
- **Proof:**
  - Uses the Fibonacci sequence! (See next!)

Proof of Lamé's Theorem

University of Hawaii

- Consider the sequence of division-algorithm equations used in Euclid's alg.:

$$r_0 = r_1 q_1 + r_2 \quad \text{with } 0 \le r_2 < r_1$$

$$r_1 = r_2 q_2 + r_3 \quad \text{with } 0 \le r_3 < r_2$$

$$\ldots$$

$$r_{n-2} = r_{n-1} q_{n-1} + r_n \quad \text{with } 0 \le r_n < r_{n-1}$$

$$r_{n-1} = r_n q_n + r_{n+1} \quad \text{with } r_{n+1} = 0 \text{ (terminate)}$$

Where $a = r_0$, $b = r_1$, and gcd$(a,b)=r_n$.

- The number of divisions (iterations) is *n*.

Continued on next slide…

Lamé Proof *cont.*

- Since $r_0 \geq r_1 > r_2 > \ldots > r_n$, each quotient $q_i \equiv \lfloor r_{i-1}/r_i \rfloor \geq 1$.
- Since $r_{n-1} = r_n q_n$ and $r_{n-1} > r_n$, $q_n \geq 2$.
- So we have the following relations between r and f:

$$r_n \geq 1 = f_2$$
$$r_{n-1} \geq 2r_n \geq 2f_2 = f_3$$
$$r_{n-2} \geq r_{n-1} + r_n \geq f_2 + f_3 = f_4$$
$$\ldots$$
$$r_2 \geq r_3 + r_4 \geq f_{n-1} + f_{n-2} = f_n$$
$$b = r_1 \geq r_2 + r_3 \geq f_n + f_{n-1} = f_{n+1}.$$

- Thus, if $n > 2$ divisions are used, then $b \geq f_{n+1} > \alpha^{n-1}$.
  - Thus, $\log_{10} b > \log_{10}(\alpha^{n-1}) = (n-1)\log_{10} \alpha \approx (n-1)0.208 > (n-1)/5$.
  - If b has k decimal digits, then $\log_{10} b < k$, so $n-1 < 5k$, so $n \leq 5k$.