Discrete Mathematics
Practice 3: Induction & Recursion

Dong-Sig Han

Seoul National University
dshan@bi.snu.ac.kr

May 11, 2017
Overview

Induction

Recursion

Problems in Homework Assignment
Induction
Mathematical Induction: Introduction

- For every $n$, $P(n)$ holds ($P(k)$: a propositional function)
- **Basis step**: first domino is knocked over
- **Inductive step type 1**: if $k$-th domino is knocked over, then $(k+1)$-th domino will knocked over. 
  $P(k) \rightarrow P(k + 1)$: **Mathematical Induction**
- **Inductive step type 2**: if first, second, …, and $k$-th dominoes are knocked over, then $(k+1)$-th domino will knocked over. 
  $P(1) \land \cdots \land P(k) \rightarrow P(k + 1)$: **Strong Induction**
- Hence, all the dominoes will knocked over.
Number of Subsets of a Finite Set

**Inductive Hypothesis:** For an arbitrary nonnegative integer \( k \), every set with \( k \) elements has \( 2^k \) subsets.

- Let \( T \) be a set with \( k + 1 \) elements. Then \( T = S \cup \{a\} \), where \( a \in T \) and \( S = T - \{a\} \). Hence \( |T| = k \).
- For each subset \( X \) of \( S \), there are exactly two subsets of \( T \), i.e., \( X \) and \( X \cup \{a\} \).

By the inductive hypothesis \( S \) has \( 2^k \) subsets. Since there are two subsets of \( T \) for each subset of \( S \), the number of subsets of \( T \) is \( 2 \cdot 2^k = 2^{k+1} \).
Tiling Checkerboards

**Example:** Show that every \(2^n \times 2^n\) checkerboard with one square removed can be tiled using right triominoes.

A right triomino is an L-shaped tile which covers three squares at a time.

**Solution:** Let \(P(n)\) be the proposition that every \(2^n \times 2^n\) checkerboard with one square removed can be tiled using right triominoes. Use mathematical induction to prove that \(P(n)\) is true for all positive integers \(n\).

- **Basis Step:** \(P(1)\) is true, because each of the four \(2 \times 2\) checkerboards with one square removed can be tiled using one right triomino.

- **Inductive Step:** Assume that \(P(k)\) is true for every \(2^k \times 2^k\) checkerboard, for some positive integer \(k\).

*continued →*
Tiling Checkerboards

**Inductive Hypothesis:** Every $2^k \times 2^k$ checkerboard, for some positive integer $k$, with one square removed can be tiled using right triominoes.

- Consider a $2^{k+1} \times 2^{k+1}$ checkerboard with one square removed. Split this checkerboard into four checkerboards of size $2^k \times 2^k$, by dividing it in half in both directions.

- Remove a square from one of the four $2^k \times 2^k$ checkerboards. By the inductive hypothesis, this board can be tiled. Also by the inductive hypothesis, the other three boards can be tiled with the square from the corner of the center of the original board removed. We can then cover the three adjacent squares with a triominoe.

- Hence, the entire $2^{k+1} \times 2^{k+1}$ checkerboard with one square removed can be tiled using right triominoes.
Well-ordering Property

- Every nonempty set of nonnegative integers has a least element.
- The set of natural numbers with the order $<$ is well-ordered.
- The uncountable subset of the real numbers with the standard ordering $\leq$ cannot be well ordered.

$R$: nonempty subset of positive real numbers

$r \in R, r > 0$

$0 < \frac{r}{3} < r$

- Well-ordered subset must have a lower bound.
Well-Ordering Property

**Example:** Use the well-ordering property to prove the division algorithm, which states that if $a$ is an integer and $d$ is a positive integer, then there are unique integers $q$ and $r$ with $0 \leq r < d$, such that $a = dq + r$.

**Solution:** Let $S$ be the set of nonnegative integers of the form $a - dq$, where $q$ is an integer. The set is nonempty since $-dq$ can be made as large as needed.

- By the well-ordering property, $S$ has a least element $r = a - dq_0$. The integer $r$ is nonnegative. It also must be the case that $r < d$. If it were not, then there would be a smaller nonnegative element in $S$, namely, $a - d(q_0 + 1) = a - dq_0 - d = r - d > 0$.
- Therefore, there are integers $q$ and $r$ with $0 \leq r < d$. (uniqueness of $q$ and $r$ is Exercise 37)
Structural Induction and Binary Trees

**Theorem:** If $T$ is a full binary tree, then $n(T) \leq 2^{h(T)+1} - 1$.

**Proof:** Use structural induction.

- **BASIS STEP:** The result holds for a full binary tree consisting only of a root, $n(T) = 1$ and $h(T) = 0$. Hence, $n(T) = 1 \leq 2^{0+1} - 1 = 1$.

- **RECURSIVE STEP:** Assume $n(T_1) \leq 2^{h(T_1)+1} - 1$ and also $n(T_2) \leq 2^{h(T_2)+1} - 1$ whenever $T_1$ and $T_2$ are full binary trees.

\[
\begin{align*}
n(T) & = 1 + n(T_1) + n(T_2) & \text{(by recursive formula of } n(T)) \\
& \leq 1 + (2^{h(T_1)+1} - 1) + (2^{h(T_2)+1} - 1) & \text{(by inductive hypothesis)} \\
& \leq 2 \cdot \max(2^{h(T_1)+1}, 2^{h(T_2)+1}) - 1 \\
& = 2 \cdot 2^{\max(h(T_1), h(T_2))+1} - 1 & \text{(max}(2^x, 2^y) = 2^{\max(x, y)}) \\
& = 2^{h(t)} - 1 & \text{(by recursive definition of } h(T)) \\
& = 2^{h(t)+1} - 1
\end{align*}
\]
Recursion
Fibonacci Numbers

Example 4: Show that whenever $n \geq 3$, $f_n > \alpha^{n-2}$, where $\alpha = (1 + \sqrt{5})/2$.

Solution: Let $P(n)$ be the statement $f_n > \alpha^{n-2}$. Use strong induction to show that $P(n)$ is true whenever $n \geq 3$.

- BASIS STEP: $P(3)$ holds since $\alpha < 2 = f_3$
  
  $P(4)$ holds since $\alpha^2 = (3 + \sqrt{5})/2 < 3 = f_4$.

- INDUCTIVE STEP: Assume that $P(j)$ holds, i.e., $f_j > \alpha^{j-2}$ for all integers $j$ with $3 \leq j \leq k$, where $k \geq 4$. Show that $P(k + 1)$ holds, i.e., $f_{k+1} > \alpha^{k-1}$.
  
  Since $\alpha^2 = \alpha + 1$ (because $\alpha$ is a solution of $x^2 - x - 1 = 0$),
  
  $\alpha^{k-1} = \alpha^2 \cdot \alpha^{k-3} = (\alpha + 1) \cdot \alpha^{k-3} = \alpha \cdot \alpha^{k-3} + 1 \cdot \alpha^{k-3} = \alpha^{k-2} + \alpha^{k-3}$

- By the inductive hypothesis, because $k \geq 4$ we have
  
  $f_{k-1} > \alpha^{k-3}$, \quad $f_{k-1} > \alpha^{k-2}$.

- Therefore, it follows that
  
  $f_{k+1} = f_{k+1} + f_{k-1} > \alpha^{k-2} + \alpha^{k-3} = \alpha^{k-1}$.

- Hence, $P(k + 1)$ is true.
Lamé’s Theorem: Let $a$ and $b$ be positive integers with $a \geq b$. Then the number of divisions used by the Euclidian algorithm to find $\gcd(a,b)$ is less than or equal to five times the number of decimal digits in $b$.

**Proof:** When we use the Euclidian algorithm to find $\gcd(a,b)$ with $a \geq b$, 

- $n$ divisions are used to obtain (with $a = r_0, b = r_1$):

  \[
  r_0 = r_1 q_1 + r_2 \quad 0 \leq r_2 < r_1, \\
  r_1 = r_2 q_2 + r_3 \quad 0 \leq r_3 < r_2, \\
  \vdots \\
  r_{n-2} = r_{n-1} q_{n-1} + r_n \quad 0 \leq r_n < r_{n-1}, \\
  r_{n-1} = r_n q_n.
  \]

- Since each quotient $q_1, q_2, \ldots, q_{n-1}$ is at least 1 and $q_n \geq 2$:

  \[
  r_n \geq 1 = f_2, \\
  r_{n-1} \geq 2 \cdot r_n \geq 2 \cdot f_2 = f_3, \\
  r_{n-2} \geq r_{n-1} + r_n \geq f_3 + f_2 = f_4, \\
  \vdots \\
  r_2 \geq r_3 + r_4 \geq f_{n-1} + f_{n-2} = f_n, \\
  b = r_1 \geq r_2 + r_3 \geq f_n + f_{n-1} = f_{n+1}.
  \]

continued →
Lamé’s Theorem

- It follows that if \( n \) divisions are used by the Euclidian algorithm to find \( \gcd(a, b) \) with \( a \geq b \), then \( b \geq f_{n+1} \). By Example 4, \( f_{n+1} > \alpha^{n-1} \), for \( n > 2 \), where \( \alpha = (1 + \sqrt{5})/2 \). Therefore, \( b > \alpha^{n-1} \).

- Because \( \log_{10} \alpha \approx 0.208 > 1/5 \), \( \log_{10} b > (n-1) \log_{10} \alpha > (n-1)/5 \). Hence,

  \[
  n-1 < 5 \cdot \log_{10} b.
  \]

- Suppose that \( b \) has \( k \) decimal digits. Then \( b < 10^k \) and \( \log_{10} b < k \). It follows that \( n - 1 < 5k \) and since \( k \) is an integer, \( n \leq 5k \).

- As a consequence of Lamé’s Theorem, \( O(\log b) \) divisions are used by the Euclidian algorithm to find \( \gcd(a, b) \) whenever \( a > b \).
  - By Lamé’s Theorem, the number of divisions needed to find \( \gcd(a, b) \) with \( a > b \) is less than or equal to \( 5 (\log_{10} b + 1) \) since the number of decimal digits in \( b \) (which equals \( \lceil \log_{10} b \rceil + 1 \)) is less than or equal to \( \log_{10} b + 1 \).

Lamé’s Theorem was the first result in computational complexity
Recursive Merge Sort

• Subroutine \textit{merge}, which merges two sorted lists.

\begin{verbatim}
procedure \textit{merge}(L_1, L_2 :sorted lists)
L := empty list
while \(L_1\) and \(L_2\) are both nonempty
    remove smaller of first elements of \(L_1\) and \(L_2\) from its list;
    put at the right end of \(L\)
if this removal makes one list empty
    then remove all elements from the other list and append them to \(L\)
return \(L\) \{\(L\) is the merged list with the elements in increasing order\}
\end{verbatim}

\textbf{Complexity of Merge:} Two sorted lists with \(m\) elements and \(n\) elements can be merged into a sorted list using no more than \(m + n - 1\) comparisons.
Complexity of Merge Sort:

The number of comparisons needed to merge a list with $n$ elements is $O(n \log n)$.

- For simplicity, assume that $n$ is a power of 2, say $2^m$.
- At the end of the splitting process, we have a binary tree with $m$ levels, and $2^m$ lists with one element at level $m$.
- The merging process begins at level $m$ with the pairs of $2^m$ lists with one element combined into $2^{m-1}$ lists of two elements. Each merger takes two one comparison.
- The procedure continues, at each level ($k = m, m-1, m-1, ..., 3, 2, 1$) $2^k$ lists with $2^{m-k}$ elements are merged into $2^{k-1}$ lists, with $2^{m-k+1}$ elements at level $k-1$.
  - We know (by the complexity of the merge subroutine) that each merger takes at most $2^{m-k} + 2^{m-k} - 1 = 2^{m-k+1} - 1$ comparisons.
Complexity of Merge Sort

Summing over the number of comparisons at each level, shows that

\[ \sum_{k=1}^{m} 2^{k-1} (2^{m-k+1} - 1) = \sum_{k=1}^{m} 2^m - \sum_{k=1}^{m} 2^{k-1} = m2^m - (2^m - 1) = n \log n - n + 1, \]

because \( m = \log n \) and \( n = 2^m \).

(The expression \( \sum_{k=1}^{m} 2^{k-1} \) in the formula above is evaluated as \( 2^m - 1 \) using the formula for the sum of the terms of a geometric progression, from Section 2.4.)

In Chapter 11, we’ll see that the fastest comparison-based sorting algorithms have \( O(n \log n) \) time complexity. So, merge sort achieves the best possible big-\( O \) estimate of time complexity.
Problems in Homework Assignment
1. Prove that \( \frac{1}{2n} \leq \frac{1 \cdot 3 \cdot 5 \cdots (2n - 1)}{2 \cdot 4 \cdots 2n} \) whenever \( n \) is a positive integer.

2. Using the well-ordering principle to show that if \( x \) and \( y \) are real numbers with \( x < y \), then there is a rational number \( r \) with \( x < r < y \).