CSE 311 Foundations of Computing I

Lecture 14

Induction and Recursive Definitions

Autumn 2011

Announcements

- Reading assignments
  - Today:
    - 5.2, 5.3 7[th] Edition
    - 4.2, 4.3 6[th] Edition
    - 3.3, 3.4 5[th] Edition
  - Monday: 5.3 (7[th]), 4.3 (6[th]), 3.4 (5[th])
- Midterm next Friday, Nov 4
  - Closed book, closed notes
  - Practice midterm available Monday
  - Extra office hours Thursday, times TBA

Highlights from last lecture

- Mathematical Induction

  P(0)

  ∀ k≥0 (P(k) → P(k+1))

  ∴ ∀ n≥0 P(n)
- Induction proof layout:
  1. By induction we will show that P(n) is true for every n≥0
  2. Base Case: Prove P(0)
  3. Inductive Hypothesis: Assume that P(k) is true for some arbitrary integer k ≥ 0
  4. Inductive Step: Prove that P(k+1) is true using Inductive Hypothesis that P(k) is true
  5. Conclusion: Result follows by induction

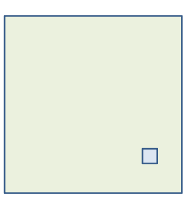
Harmonic Numbers

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots \frac{1}{n} = \sum_{k=1}^{n} \frac{1}{k}$$

Prove $H_{2^n} \geq 1 + \frac{n}{2}$ for all $n \geq 1$

Cute Application: Checkerboard Tiling with Trinominos

Prove that a $2^n \times 2^n$ checkerboard with one square removed can be tiled with: 



Strong Induction

P(0)

∀ k ((P(0) ∧ P(1) ∧ P(2) ∧ … ∧ P(k)) → P(k+1))

∴ ∀ n P(n)

Follows from ordinary induction applied to Q(n) = P(0) ∧ P(1) ∧ P(2) ∧ … ∧ P(n)

Strong Induction English Proofs

1. By induction we will show that P(n) is true for every n≥0
2. Base Case: Prove P(0)
3. Inductive Hypothesis: Assume that for some arbitrary integer k ≥ 0, P(j) is true for every j from 0 to k
4. Inductive Step: Prove that P(k+1) is true using Inductive Hypothesis that P(j) is true for all values ≤ k
5. Conclusion: Result follows by induction

Every integer ≥ 2 is the product of primes

Recursive Definitions of Functions

- F(0) = 0; F(n + 1) = F(n) + 1;
- G(0) = 1; G(n + 1) = 2 × G(n);
- 0! = 1; (n+1)! = (n+1) × n!
- H(0) = 1; H(n + 1) = $2^{H(n)}$

Fibonacci Numbers

- $f_0 = 0$; $f_1 = 1$; $f_n = f_{n-1} + f_{n-2}$

Bounding the Fibonacci Numbers

- Theorem: $2^{n/2-1} \leq f_n < 2^n$ for $n \geq 2$

Fibonacci numbers and the running time of Euclid's algorithm

- Theorem: Suppose that Euclid's algorithm takes n steps for gcd(a,b) with $a>b$, then $a \geq f_{n+1}$ so $a \geq 2^{(n-1)/2}$
  - # of steps at most one more than twice # of bits of a
- Set $r_{n+1}=a$, $r_n=b$ then Euclid's alg. computes

$r_{n+1}= q_n r_n + r_{n-1}$
$r_n= q_{n-1} r_{n-1} + r_{n-2}$
•••
$r_3= q_2 r_2 + r_1$
$r_2= q_1 r_1$

each quotient $q_i \geq 1$
$r_1 \geq 1$

Recursive Definitions of Sets

- Recursive definition
  - Basis step: $0 \in S$
  - Recursive step: if $x \in S$, then $x + 2 \in S$
  - Exclusion rule: Every element in S follows from basis steps and a finite number of recursive steps

Recursive definitions of sets

Basis: $6 \in S$; $15 \in S$;
Recursive: if $x, y \in S$, then $x + y \in S$;

Basis: $[1, 1, 0] \in S$, $[0, 1, 1] \in S$;
Recursive:
if $[x, y, z] \in S$, α in R, then $[\alpha x, \alpha y, \alpha z] \in S$
if $[x_1, y_1, z_1], [x_2, y_2, z_2] \in S$
then $[x_1 + x_2, y_1 + y_2, z_1 + z_2]$

Powers of 3

Strings

- The set Σ^* of strings over the alphabet Σ is defined
  - Basis: $\lambda \in S$ (λ is the empty string)
  - Recursive: if $w \in \Sigma^*$, $x \in \Sigma$, then $wx \in \Sigma^*$

Function definitions on recursively defined sets

$Len(\lambda) = 0$;
$Len(wx) = 1 + Len(w)$; for $w \in \Sigma^*$, $x \in \Sigma$

$Concat(w, \lambda) = w$ for $w \in \Sigma^*$
$Concat(w_1,w_2x) = Concat(w_1,w_2)x$ for w_1, w_2 in Σ^*, $x \in \Sigma$