CSE373: Data Structures and Algorithms

Induction and Its Applications Part 2:

Strong Induction, and Example

Steve Tanimoto

Autumn 2016

This lecture material is based in part on materials provided by Ioana Sora at the Politechnic University of Timisoara.

Lecture Outline

- **Review of Induction and Strong Induction**
  - Example: A theorem about the first n natural numbers.
  - Example for Strong Induction: Making Postage
- **Example for Regular Induction: Correctness of a Decimal-to-Binary Conversion Algorithm**

Mathematical induction - Review

- Let $(\forall n \geq c)T(n)$ be a theorem that we want to prove. It includes a constant c and a natural parameter n.
- Proving that T holds for all natural values of n greater than or equal to c is done by proving following two conditions:
  1. T holds for n=c
  2. For every n>c if T holds for n-1, then T holds for n

Terminology:

T(c)	is the Base Case
T(n-1)	is the Induction Hypothesis
T(n-1) ⇒ T(n)	is the Induction Step
(∀n≥c)T(n)	is the Theorem being proved.

Mathematical induction - review

- **Strong Induction**: a variant of induction where the inductive step builds up on *all* the smaller values
- Proving that T holds for all natural values of n greater than or equal to c is done by proving following two conditions:
  1. T holds for $n=c_1, c_1+1, \ldots, c_m$
  2. If for every k from c_1 up to n-1, it is true that T(k), then T(n)

Mathematical induction – Example1

- Theorem: *The sum of the first n natural numbers is n· (n+1)/2*

  $(\forall n \geq 1)T(n) \Leftrightarrow (\forall n \geq 1) \sum_{k=1}^{n} k = n \cdot (n+1)/2$
- Proof: by *induction* on n
  1. Base case: If n=1, s(1)=1=1· (1+1)/2
  2. Inductive step: We assume that $s(n)=n\cdot(n+1)/2$, and prove that this implies $s(n+1)=(n+1)\cdot(n+2)/2$, for all n≥1

$$s(n+1)=\mathbf{s(n)}+(n+1)=\mathbf{n\cdot(n+1)/2}+(n+1)=(n+1)\cdot(n+2)/2$$

4 + 4 + 5 = 13

Making postage is the problem of selecting a group of stamps whose total value matches a given amount.

Mathematical induction – Example2

- Theorem: *Every amount of postage that is at least 12 cents can be made from 4-cent and 5-cent stamps.*
- *Proof: by induction on the amount of postage*
- *Postage (p) = m · 4 + n · 5*
- Base cases:
  - Postage(12) = 3 · 4 + 0 · 5
  - Postage(13) = 2 · 4 + 1 · 5
  - Postage(14) = 1 · 4 + 2 · 5
  - Postage(15) = 0 · 4 + 3 · 5

Mathematical induction – Example2 (cont)

- Inductive step: We assume that we can construct postage for every value from 12 up to k. We need to show how to construct k + 1 cents of postage. Since we have proved base cases up to 15 cents, we can assume that k + 1 ≥ 16.
- Since k+1 ≥ 16, (k+1)−4 ≥ 12. So by the inductive hypothesis, we can construct postage for (k + 1) − 4 cents: (k + 1) − 4 = m · 4 + n · 5
- But then k + 1 = (m + 1) · 4 + n · 5. So we can construct k + 1 cents of postage using (m+1) 4-cent stamps and n 5-cent stamps.

Correctness of algorithms

- Induction can be used for proving the correctness of repetitive algorithms:
  - Iterative algorithms:
    - Loop invariants
      - Induction hypothesis = loop invariant = relationships between the variables during loop execution
  - Recursive algorithms
    - Direct induction
      - induction hypothesis = assumption that each recursive call itself is correct (often a case for applying *strong* induction)

128	64	32	16	8	4	2	1
1	0	0	1	1	1	0	0

$$156_{10} = 10011100_2$$

Decimal-to-binary conversion means taking a number in base-10 notation and converting it into base-2 notation.

Example: Correctness proof for Decimal to Binary Conversion

```
Algorithm Decimal_to_Binary

Input: n, a positive integer
Output: b, an array of bits, the bin repr. of n,
        starting with the least significant bits

t:=n;
k:=0;
while (t>0) do
  b[k]:=t mod 2;
  t:=t div 2;
  k:=k+1;
end
```

It is a repetitive (iterative) algorithm; thus we use loop invariants and proof by induction.

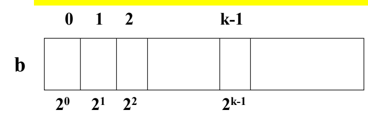
Example: Loop invariant for Decimal to Binary Conversion

```
Algorithm Decimal_to_Binary

Input: n, a positive integer
Output: b, an array of bits, the bin repr. of n

t:=n;
k:=0;
while (t>0) do
  b[k]:=t mod 2;
  t:=t div 2;
  k:=k+1;
end
```

At step k, b holds the k least significant bits of n, and the value of t, when shifted by k, corresponds to the rest of the bits.



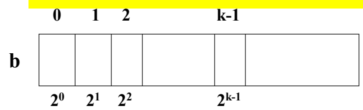
Example: Loop invariant for Decimal to Binary Conversion

```
Algorithm Decimal_to_Binary

Input: n, a positive integer
Output: b, an array of bits, the bin repr. of n

t:=n;
k:=0;
while (t>0) do
  b[k]:=t mod 2;
  t:=t div 2;
  k:=k+1;
end
```

Loop invariant: If m is the integer represented by array b[0..k-1], then n=t · 2^k+m.



Example: Proving the correctness of the conversion algorithm

- **Induction hypothesis=Loop Invariant**: If m is the integer represented by array b[0..k-1], then n=t · 2^k+m
- **To prove the correctness of the algorithm, we have to prove the 3 conditions:**
  1. ***Initialization: The hypothesis is true at the beginning of the loop.***
  2. ***Maintenance: If hypothesis is true for step k, then it will be true for step k+1.***
  3. ***Termination: When the loop terminates, the hypothesis implies the correctness of the algorithm.***

Example: Proving the correctness of the conversion algorithm *(1)*

- Induction hypothesis: If m is the integer represented by array b[0..k-1], then n=t · 2^k+m.

1. *The hypothesis is true at the beginning of the loop:*

k=0, t=n, m=0(array is empty)

n=n · 2^0+0

Example: Proving the correctness of the conversion algorithm *(2)*

- Induction hypothesis: If m is the integer represented by array b[0..k-1], then n=t · 2^k+m.

2. *If hypothesis is true for step k, then it will be true for step k+1.*

At the start of step k: assume that n=t · 2^k+m, calculate the values at the end of this step.

If t is even then: t mod 2=0, m unchanged,

t:=t / 2, k:=k+1 $\Rightarrow$ (t / 2) · $2^{(k+1)}$ + m = t · 2^k+m = n

If t is odd then: t mod 2 =1, b[k+1] is set to 1, m:=m+2^k ,

t:=(t-1)/2, k:=k+1 $\Rightarrow$ (t-1)/2 · $2^{(k+1)}$+m+2^k = t · 2^k+m = n

Example: Proving the correctness of the conversion algorithm *(3)*

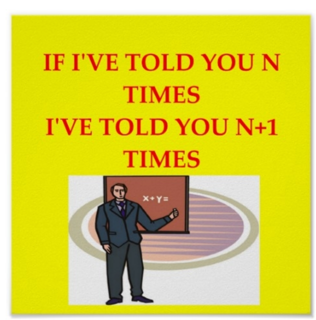
- Induction hypothesis: If m is the integer represented by array b[0..k-1], then n = t · 2^k+m

3. *When the loop terminates, the hypothesis implies the correctness of the algorithm.*

The loop terminates when t=0 implies

n = 0 · 2^k+m = m

n = m. (proved)



Poster available from
http://www.zazzle.co.nz/math_posters-228552736450241612

Bibliography

- Weiss, Ch. 1 section on induction.
- Goodrich and Tamassia: Induction and loop invariants; see, e.g., http://www.cs.mun.ca/~kol/courses/2711-w09/Induction.pdf)
- Erickson, J. Proof by Induction. Available at: http://jeffe.cs.illinois.edu/teaching/algorithms/notes/98-induction.pdf