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
- \( T(n-1) \implies T(n) \) is the Induction Step
- \( (\forall n \geq c) T(n) \) is the Theorem being proved.

Mathematical induction – Example1

- Theorem: The sum of the first \( n \) natural numbers is \( n \cdot (n+1)/2 \)
  \( (\forall n \geq 1) T(n) \iff (\forall n \geq 1) \sum_{k=1}^{n} k = n \cdot (n+1)/2 \)
- Proof: by induction on \( n \)
  1. Base case: If \( n = 1 \), \( s(1) = 1 = 1 \cdot (1+1)/2 \)
  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 \( n \geq 1 \)

\[
s(n+1) = s(n) + (n+1) = n \cdot (n+1)/2 + (n+1) = (n+1) \cdot (n+2)/2
\]

Making postage is the problem of selecting a group of stamps whose total value matches a given amount.
Mathematical induction – Example 2

• 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 \( p = m \cdot 4 + n \cdot 5 \)
• Base cases:
  – Postage(12) = 3 \cdot 4 + 0 \cdot 5
  – Postage(13) = 2 \cdot 4 + 1 \cdot 5
  – Postage(14) = 1 \cdot 4 + 2 \cdot 5
  – Postage(15) = 0 \cdot 4 + 3 \cdot 5

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 \( k + 1 \geq 16 \).

Since \( k+1 \geq 16 \), \((k+1)−4 \geq 12\). So by the inductive hypothesis, we can construct postage for \((k + 1) − 4\) cents: \((k + 1) − 4 = m \cdot 4 + n \cdot 5\).

But then \( k + 1 = (m + 1) \cdot 4 + n \cdot 5\). 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)

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 \\
  \quad b[k] := t \mod 2; \\
  \quad t := t \div 2; \\
  \quad 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 \( n \)

\[
t := n; \quad k := 0; \quad \text{while } (t > 0) \text{ do } \\
  \quad b[k] := t \mod 2; \quad \text{t := t \div 2;} \\
  \quad k := k + 1; \quad \text{end}
\]

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

\[
t := n;
k := 0;
\text{while } (t>0) \text{ do}
\begin{align*}
    b[k] &:= t \mod 2; \\
    t &:= t \div 2; \\
    k &:= k+1;
\end{align*}
\text{end}
\]

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

Example: Proving the correctness of the conversion algorithm

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

1. **The hypothesis is true at the beginning of the loop:**
   
   \(k=0\), \(t=n\), \(m=0\) (array is empty)

   \(n = n \cdot 2^0 + 0\)

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

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 \(n = t \cdot 2^k + m\), calculate the values at the end of this step.

   - If \(t\) is even then: \(t \mod 2 = 0\), \(m\) unchanged,
     \[
     t = t / 2, \quad k = k+1 \implies \left( t / 2 \right) \cdot 2^{k+1} + m = t \cdot 2^k + m = n
     \]
   - If \(t\) is odd then: \(t \mod 2 = 1\), \(b[k+1]\) is set to 1, \(m' = m + 2^k\),
     \[
     t = (t-1)/2, \quad k = k+1 \implies (t-1)/2 \cdot 2^{k+1} + m + 2^k = t \cdot 2^k + m = n
     \]

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

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

   The loop terminates when \(t=0\) implies
   
   \(n = 0 \cdot 2^k + m = m\)
   
   \(n = m\). \text{(proved)}
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