CSE373: Data Structures and Algorithms

Induction and Its Applications Part 1:

Algorithm Correctness, Loop Invariants, and Induction

Steve Tanimoto

Autumn 2016

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

The cover from Francesco Maurolico's *Arithmeticorum Libri Duo* (1575), which includes one of the first known proofs by mathematical induction.

Lecture Outline

- **Key Parts of an Algorithm**
  - Input, output, preconditions, postconditions, process desc.
- **Proving the Correctness of Algorithms**
  - Tracing data movements and changes
  - Example: the Swap1 procedure.
- **Using Induction to Prove Algorithm Properties**
  - Loop Invariants
  - Example: Sum_of_n_numbers.
  - Induction

What are key parts of an algorithm ?

- An algorithm is described by:
  - Input data
  - Output data
  - ***Preconditions***: specifies restrictions on input data
  - ***Postconditions***: specifies what is the result
  - Step-by-step process specification
- Example: Binary Search
  - Input data: `a:array of integer; x:integer;`
  - Output data: `found:boolean;`
  - Precondition: `a` is sorted in ascending order
  - Postcondition: `found` is true if `x` is in `a`, and `found` is false otherwise
  - Step-by-step description of the search process.

Correct algorithms

- **An algorithm is correct if:**
  - for **any correct input data**:
    - it **stops** and
    - it produces **correct output.**
  - Correct input data: satisfies precondition
  - Correct output data: satisfies postcondition

Proving correctness

- An algorithm ≈ a list of actions
- ***Proving that an algorithm is totally correct:***
  1. ***Proving that it will terminate***
  2. ***Proving that the list of actions applied to the input (which satisfies the precondition) imply that the output satisfies the postcondition***

  - This is easy to prove for simple sequential algorithms
  - This can be complicated to prove for repetitive algorithms (containing loops or recursion)
    - Use techniques based on ***loop invariants*** and ***induction***

Example – a sequential algorithm

```
Swap1(x,y):
    aux := x
    x := y
    y := aux
```

Precondition:
x = a and y = b

Postcondition:
x = b and y = a

Proof: *the list of actions applied to the input (which satisfies the precondition) imply the output satisfies the postcondition*

1. **Precondition: x = a and y = b**
2. **aux := x ⇒ aux = a**
3. **x := y ⇒ x = b**
4. **y := aux ⇒ y = a**
5. **x = b and y = a is the Postcondition**

Example – a repetitive algorithm

Algorithm Sum_of_N_numbers

Input: integer N, and
a, an array of N numbers

Output: s, the sum of the N numbers in a

```
s:=0;
k:=0;
while (k<N) do
  s:=s+a[k];
  k:=k+1;
end
```

Proof: The list of actions applied to the precondition imply the postcondition

BUT: we cannot enumerate all the actions in case of a repetitive algorithm !

We use techniques based on loop invariants and induction

Loop invariants

- A loop invariant is a logical predicate such that: if it is satisfied before entering any single iteration of the loop then it is also satisfied after that iteration.

Example: Loop invariant for Sum of n numbers

Algorithm Sum_of_N_numbers

Algorithm Sum_of_N_numbers

Input: integer N, and
a, an array of N numbers
Output: s, the sum of the N numbers in a

```
s:=0;
k:=0;
while (k<N) do
  s:=s+a[k];
  k:=k+1;
end
```

Loop invariant = induction hypothesis: At step k, s holds the sum of the first k numbers.

Using loop invariants in proofs

We must show the following 2 things about a loop invariant:

1. **Initialization:** It is true prior to the first iteration of the loop.
2. **Maintenance:** If it is true before an iteration of the loop, it remains true before the next iteration.

We also must show **Termination:** that the loop terminates.

When the loop terminates, the invariant gives us a useful property that helps show that the algorithm is correct.

Example: Proving the correctness of the Sum algorithm (1)

- Induction hypothesis: s = sum of the first k numbers

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

Before the first iteration: k=0, s=0. The first 0 numbers have sum zero (there are no numbers) ⇒ hypothesis true before entering the loop

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

- Induction hypothesis: s = sum of the first k numbers

2. *Maintenance: If hypothesis is true before step k, then it will be true before step k+1 (immediately after step k is finished).*

We assume that it is true at beginning of step k: "s is the sum of the first k numbers."

We have to prove that after executing step k, at the beginning of step k+1: "s is the sum of the first k+1 numbers."

We calculate the value of s at the end of this step

k:=k+1, s:=s+a[k+1] $\Rightarrow$ s is the sum of the first k+1 numbers.

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

- Induction hypothesis: s = sum of the first k numbers

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

The loop terminates when k=n.

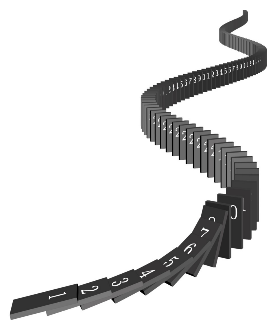
This implies s = sum of first k=n numbers.

Thus the postcondition of the algorithm is satisfied.

Q.E.D. (Quod Erat Demonstrandum; we are done.)

Loop invariants and induction

- **Proving loop invariants is a form of mathematical induction**:
  - showing that the invariant holds before the first iteration corresponds to the base case, and
  - showing that the invariant holds from iteration to iteration corresponds to the inductive step.



Mathematical induction is like a domino chain. If the relationship between each domino and the next is set up properly, all it takes is to topple the first domino, and all the rest fall down automatically.

Image by Mark Wibrow.

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