Math/CSE 1019C: Discrete Mathematics for Computer Science
Fall 2012

Jessie Zhao
jessie@cse.yorku.ca

Course page:
http://www.cse.yorku.ca/course/1019

Introduction to Algorithms

- Why algorithms?
  - Consider the problems as special cases of general problems.
    - Searching for an element in any given list
    - Sorting any given list, so its elements are in increasing/decreasing order
- What is an algorithm?
  - For this course: detailed pseudocode, or a detailed set of steps

Algorithm (topics)

- Design of Algorithms (for simple problems)
- Analysis of algorithms
  - – Is it correct?
    Loop invariants
  - – Is it "good"?
    Efficiency
  - – Is there a better algorithm?
    Lower bounds

- Problem: Swapping two numbers in memory
  - INPUT: x=a, y=b
  - OUTPUT: x and y such that x=b and y=a.
  - tmp = x;
  - x = y;
  - y = tmp;
- Can we do it without using tmp ?
  - x = x+y;
  - y = x-y;
  - x= x-y;
- Why does this work?
- Does it always work?

- Problem: How do you find the max of n numbers (stored in array A?)
  - INPUT: A[1..n] - an array of integers
  - OUTPUT: an element m of A such that A[j] ≤ m, 1 ≤ j ≤ length(A)
- Find-max (A)
  - 1. max←A[1]
  - 2. for j ← 2 to n
  - 3. do if (max < A[j])
  - 4. max ← A[j]
  - 5. return max

Reasoning (formally) about algorithms

- 1. I/O specs: Needed for correctness proofs, performance analysis.
  - INPUT: A[1..n] - an array of integers
  - OUTPUT: an element m of A such that A[j] ≤ m, 1 ≤ j ≤ length(A)
- 2. CORRECTNESS: The algorithm satisfies the output specs for EVERY valid input
- 3. ANALYSIS: Compute the running time, the space requirements, number of cache misses, disk accesses, network accesses,….

Correctness proofs for conditional statements

- Conditional Statements
  - If (condition), do (S)

$$\frac{\begin{array}{c}(p\wedge \text{condition})\{S\}q \\ (p\wedge \neg\text{condition})\rightarrow q\end{array}}{\therefore\ p\{\text{If (condition), do (S)}\}q}$$

Note: p{S}q means whenever p is true for the input values of S and S terminates, then q is true for the output values of S.

Correctness proofs for conditional statements

- Conditional Statements
  - If (condition), do (S_1); else do (S_2)

$$\frac{\begin{array}{c}(p\wedge \text{condition})\{S_1\}q \\ (p\wedge \neg\text{condition})\{S_2\}q\end{array}}{\therefore\ p\ \{\text{If (condition), do }(S_1)\text{; else do }(S_2)\}\ q}$$

Example: partial algorithm for Find-max

3. do if (max < A[j])
4. max ← A[j]

p: T
q: max≥A[j]

Example: If x<0 then
abs:= −x
else
abs:=x
p: T
q: abs=|x|

Correctness proofs of algorithms

Find-max (A)
1. max ← A[1]
2. for j ← 2 to length(A)
3. do if (max < A[j])
4. max ← A[j]
5. return max

- Prove that for any valid Input, the output of Find-max satisfies the output condition.
- **Proof by contradiction:**
  - **Suppose the algorithm is incorrect.**
  - Then for some input A,
    - Case 1: max is not an element of A. max is initialized to and assigned to elements of A – (a) is impossible.
    - Case 2: (∃ j | max < A[j]). After the jth iteration of the for-loop (lines 2 – 4), max ≥ A[j]. From lines 3,4, max only increases.
  - Therefore, upon termination, max ≥ A[j], which contradicts (b).

Correctness proofs using loop invariants

- while (condition), do (S)
- Loop invariant
  - An assertion that remains true each time S is executed.
  - p is a loop invariant if (p∧condition){S}p is true.
  - p is true before S is executed. q and ¬condition are true after termination.

$$\frac{(p\wedge \text{condition})\{S\}p}{\therefore\ p\{\text{while condition do S}\}\ (\neg\text{condition} \wedge p)}$$

Loop invariant proofs

Find-max (A)
1. max ← A[1]
2. for j ← 2 to length(A)
3. do if (max < A[j])
4. max ← A[j]
5. return max

- Prove that for any valid Input, the output of Find-max satisfies the output condition.
- **Proof by loop invariants:**
  - Loop invariant: I(j): At the beginning of iteration j of the loop, max contains the maximum of A[1,...,j-1].
  - Proof:
    - True for j=2.
    - Assume that the loop invariant holds for the j iteration, So at the beginning of iteration k, max = maximum of A[1,...,j-1].

Loop invariant proofs

```
Find-max (A)
1. max ← A[1]
2. for j ← 2 to length(A)
3.    do if (max < A[j])
4.          max ← A[j]
5. return max
```

- For the (j+1)th iteration
  - Case 1: A[j] is the maximum of A[1,…,j]. In lines 3, 4, max is set to A[j].
  - Case 2: A[j] is not the maximum of A[1,…,j]. So the maximum of A[1,…, j] is in A[1,…,j-1]. By our assumption max already has this value and by lines 3-4 max is unchanged in this iteration.

Loop invariant proofs

- STRATEGY: We proved that the invariant holds at the beginning of iteration j for each j used by Find-max.
  - Upon termination, j = length(A)+1. (WHY?)
  - The invariant holds, and so max contains the maximum of A[1..n]

Loop invariant proofs

- Advantages:
  - Rather than reason about the whole algorithm, reason about SINGLE iterations of SINGLE loops.
- Structured proof technique
- Usually prove loop invariant via Mathematical Induction.