Math/CSE 1019: Discrete Mathematics for Computer Science

Fall 2011

Suprakash Datta

datta@cse.yorku.ca

Office: CSEB 3043

Phone: 416-736-2100 ext 77875

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

Algorithms: topics

- Notation/pseudocode
- 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

What is an algorithm?

- For this course: detailed pseudocode, or a detailed set of steps
- Q: Given a problem, can we always design an algorithm to solve it?

- CSE 3101: Design and Analysis of Algorithms
  - Different design paradigms
  - Different analysis techniques
  - Intractability results

Problem 1

- Swapping two numbers in memory

```
tmp = x;
x = y;
y= tmp;
```

- Can we do it without using tmp ?

```
x = x+y;
y = x-y;
y= x-y;
```

- Why does this work?
- Does it always work?

Making change

- Want to make change for ANY amount using the fewest number of coins
- Simple "greedy" algorithm: keep using the largest denomination possible
- Works for our coins: 1,5,10, 25,100.
- Does it always work?
- Fails for the following coins: 1,5,7,10 e.g: 14 =10 + 1 +1 +1 +1, 14 = 7 + 7
- Read proof from the text

Problem 2

Q1. How do you find the max of n numbers (stored in array A?)

Formal specs:

INPUT: A[1..n] - an array of integers

OUTPUT: an element m of A such that $A[j] \le m$, $1 \le j \le \text{length}(A)$

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

How many comparisons?

Q2. Can you think of another algorithm? Take a minute….

How many comparisons does it take?

Reasoning (formally) about algorithms

1. I/O specs: Needed for correctness proofs, performance analysis. e.g. for sorting:
   INPUT: A[1..n] - an array of integers
   OUTPUT: a permutation B of A such that
   $B[1] \leq B[2] \leq \ldots \leq B[n]$

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 of algorithms

INPUT: A[1..n] - an array of integers

OUTPUT: an element m of A such that m $\leq$ A[j], 1 $\leq$ j $\leq$ length(A)

Find-max (A)

1. **max $\leftarrow$ A[1]**
2. **for j $\leftarrow$ 2 to length(A)**
3. **do if (max < A[j])**
4. **max $\leftarrow$ A[j]**
5. **return max**

Prove that for any valid Input, the output of Find-max satisfies the output condition.

Proof 1 [by contradiction]: Suppose the algorithm is incorrect. Then for some input A,

(a) max is not an element of A or (b) ($\exists$j | max < A[j]).

Max is initialized to and assigned to elements of A – (a) is impossible. For (b): after the j^{th} iteration of the for-loop (lines 2 – 4), max $\geq$ A[j]. From lines 3,4, max only increases.

Therefore, upon termination, max $\geq$ A[j], which contradicts (b).

Correctness proofs of algorithms -2

INPUT: A[1..n] - an array of integers

OUTPUT: an element m of A such that m ≤ A[j], 1 ≤ j ≤ length(A)

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 2[use loop invariants]:

(identify invariant) **<u>I(j):</u>** At the beginning of iteration j of for loop, max contains the maximum of A[1..j-1].

(Proof) True for j=2. For j > 2, assume that (j-1) holds. So at the beginning of iteration j-1, max = maximum of A[1..j-2].

Loop invariant proof - contd

Case (a) A[j] is the maximum of A[1..j]. In lines 3,4, max is set to A[j].

Case (b) 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.

You will see more non-trivial examples in CSE 2011, 3101.

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]

-- STRUCTURED PROOF TECHNIQUE
-- VERY SIMILAR TO INDUCTION

Advantages:
Rather than reason about the whole algorithm, reason about SINGLE iterations of SINGLE loops.

Problem 2

Q1. How do you find the max and min of n numbers (stored in array A?)

Q2. Can you think of a FASTER algorithm?