Week 7 Discussion

Liu Yang
lyang422@wisc.edu

This week's topic

- **Program correctness**
- **Loop invariant**

This week's topic

- **Program correctness**
    1. Partial correctness: For all valid input, if the program terminates on the input, then the program produces the **correct** output for that input.
    2. Termination: for all valid input, the program **terminates**.

This week's topic

- **Loop invariant**
  - Statement that holds each time the loop condition is about to be tested.
  - Proving loop invariant:
    - proposed invariants will be given. Generally have (at least) two loop invariants
      - one related to the looping conditions (invariant (i))
      - one related to desired output & values computed/changed in loop body (invariant (ii))
    - prove by induction (choose one you think reasonable)
      - *either* on # times loop test is done - base case is P(1)
      - *or* on # of iterations of loop (# times loop body has executed) - base case is P(0)
    - use subscripting on variables to track changes as loop body is executed

Discussion Handout

- Give the link …

Greatest Common Divisor

- Definition: The *greatest common divisor* of integers a and b, denoted $\gcd(a, b)$, is the largest integer d such that d divides both a and b. Furthermore, if c divides both a and b, then c also divides $\gcd(a, b)$.
- If a, b > 0, gcd(a, 0)=a, gcd(0, b) = b, gcd(0, 0) is undefined.
- Example:
  - gcd(15, 32) = 1
  - gcd(26, 18) = 2
  - gcd(15, 30) = 15

Greatest Common Divisor

- Properties:
  - If $a = b$, $\gcd(a, b) = a = b$
  - If $a < b$, $\gcd(a, b) = \gcd(a, b - a)$
  - If $a > b$, $\gcd(a, b) = \gcd(a - b, a)$

- Example:
  - If gcd(26, 18) = 2, then gcd(26-18, 18) = 2

8

It_GCD

- It_GCD Algorithm (GCD Algorithm in DvM):

procedure $it\_GCD(a, b)$

$(x, y) \leftarrow (a, b)$

while $x \neq y$ ***do***

    if $x < y$ ***then*** $y \leftarrow y - x$

    else $x \leftarrow x - y$

end

return x

It_GCD (Example: gcd(26, 18))

procedure $it\_GCD(a, b)$			
$(x, y) \leftarrow (a, b)$	$(x, y) = (26, 18)$	$(x, y) = (8, 18)$	$(x, y) = (8, 10)$
while $x \neq y$ ***do***	Yes	Yes	Yes
if $x < y$ ***then*** $y \leftarrow y - x$	No	$y = 18 - 8 = 10$	$y = 10 - 8 = 2$
else $x \leftarrow x - y$	$x = 26 - 18 = 8$		
end			
return x			

It_GCD (Example: gcd(26, 18))

procedure $it\_GCD(a, b)$	(con’t)		
$(x, y) \leftarrow (a, b)$	$(x, y) = (8, 2)$	$(x, y) = (6, 2)$	$(x, y) = (4, 2)$
while $x \neq y$ **do**	Yes	Yes	Yes
if $x < y$ ***then*** $y \leftarrow y - x$	No	No	No
else $x \leftarrow x - y$	$x = 8 - 2 = 6$	$x = 6 - 2 = 4$	$x = 4 - 2 = 2$
end			
return x			

It_GCD (Example: gcd(26, 18))

procedure $it\_GCD(a, b)$	(con't)		
$(x, y) \leftarrow (a, b)$	$(x, y) = (2, 2)$		
while $x \neq y$ ***do***	No		
if $x < y$ ***then*** $y \leftarrow y - x$			
else $x \leftarrow x - y$			
end			
return x	***return*** 2		

It_GCD - 0 case

- Example: $\mathrm{gcd}(15, 0)$

procedure $it\_GCD(a, b)$			
$(x, y) \leftarrow (a, b)$	$(x, y) = (15, 0)$	$(x, y) = (15, 0)$	…
while $x \neq y$ ***do***	Yes	Yes	
if $x < y$ ***then*** $y \leftarrow y - x$	No	No	
else $x \leftarrow x - y$	$x = 15 - 0 = 15$	$x = 15 - 0 = 15$	
end			
return x			

euclid_GCD (at least one of a, b is nonzero)

procedure $euclid\_GCD(a, b)$

if $a \leq b$, ***then*** $(x, y) \leftarrow (a, b)$, ***else*** $(x, y) \leftarrow (b, a)$

while $x > 0$ ***do***

  $r \leftarrow y \bmod x$

  $y \leftarrow x$

  $x \leftarrow r$

End

return y

Part a

- If x and y are both positive integers, what are the possible values that x mod y can have?

Part a

- If x and y are both positive integers, what are the possible values that x mod y can have?

$$0, 1, 2, \dots\ (y\ -\ 1)$$

euclid_GCD (Example: gcd(26, 18))

procedure $euclid\_GCD(a, b)$		
if $a \leq b$, ***then*** $(x, y) \leftarrow (a, b)$, ***else*** $(x, y) \leftarrow (b, a)$	$(x, y) = (18, 26)$	$(x, y) = (8, 18)$
while $x > 0$ ***do***	Yes	Yes
$r \leftarrow y \bmod x$	$r = 8$	$r = 2$
$y \leftarrow x$	$y = 18$	$y = 8$
$x \leftarrow r$	$x = 8$	$x = 2$
End		
return y		

euclid_GCD (Example: gcd(26, 18))

procedure $euclid\_GCD(a, b)$		
if $a \leq b$, ***then*** $(x, y) \leftarrow (a, b)$, ***else*** $(x, y) \leftarrow (b, a)$	$(x, y) = (2, 8)$	$(x, y) = (0, 2)$
while $x > 0$ ***do***	Yes	No
$\quad r \leftarrow y \bmod x$	$r = 0$	
$\quad y \leftarrow x$	$y = 2$	
$\quad x \leftarrow r$	$x = 0$	
End		
return y		***return*** 2

euclid_GCD - 0 case

procedure $euclid\_GCD(a, b)$		
if $a \leq b$, ***then*** $(x, y) \leftarrow (a, b)$, ***else*** $(x, y) \leftarrow (b, a)$	$(x, y) = (0, 15)$	
while $x > 0$ ***do***	No	
$r \leftarrow y \bmod x$		
$y \leftarrow x$		
$x \leftarrow r$		
End		
return y	***return*** 15	

Part c

- Use induction to establish the following loop invariant:
- Each time we reach the test of the *while* loop on line (3),
  (i) $0 \leq x \leq y$ and (ii) $\gcd(x, y) = \gcd(a, b)$.

euclid_GCD

procedure $euclid\_GCD(a, b)$

if $a \leq b$, ***then*** $(x, y) \leftarrow (a, b)$, ***else*** $(x, y) \leftarrow (b, a)$

while $x > 0$ ***do***

$\quad r \leftarrow y \bmod x$

$\quad y \leftarrow x$

$\quad x \leftarrow r$

End

return y

euclid_GCD

1

procedure $euclid\_GCD(a, b)$

if $a \leq b$, ***then*** $(x, y) \leftarrow (a, b)$, ***else*** $(x, y) \leftarrow (b, a)$

while $x > 0$ ***do***

    $r \leftarrow y \bmod x$

    $y \leftarrow x$

    $x \leftarrow r$

End

return y

euclid_GCD

procedure $euclid\_GCD(a, b)$

if $a \leq b$, ***then*** $(x, y) \leftarrow (a, b)$, ***else*** $(x, y) \leftarrow (b, a)$

while $x > 0$ ***do***

  $r \leftarrow y \bmod x$

  $y \leftarrow x$

  $x \leftarrow r$

End

return y

euclid_GCD

procedure $euclid\_GCD(a, b)$
if $a \leq b$, ***then*** $(x, y) \leftarrow (a, b)$, ***else*** $(x, y) \leftarrow (b, a)$
while $x > 0$ ***do***
$\quad r \leftarrow y \bmod x$
$\quad y \leftarrow x$
$\quad x \leftarrow r$
End
return y

euclid_GCD

procedure $euclid\_GCD(a, b)$

_if_ $a \leq b$, **_then_** $(x, y) \leftarrow (a, b)$, **_else_** $(x, y) \leftarrow (b, a)$

while $x > 0$ **_do_**

    $r \leftarrow y \bmod x$

    $y \leftarrow x$

    $x \leftarrow r$

End

return y

euclid_GCD

procedure $euclid\_GCD(a, b)$

if $a \leq b$, **then** $(x, y) \leftarrow (a, b)$, **else** $(x, y) \leftarrow (b, a)$

while $x > 0$ **do**

  $r \leftarrow y \bmod x$

  $y \leftarrow x$

  $x \leftarrow r$

End

return y

euclid_GCD

2

procedure $euclid\_GCD(a, b)$

if $a \leq b$, ***then*** $(x, y) \leftarrow (a, b)$, ***else*** $(x, y) \leftarrow (b, a)$

while $x > 0$ ***do***

  $r \leftarrow y \bmod x$

  $y \leftarrow x$

  $x \leftarrow r$

End

return y

Group Discussion !

Try with euclid_GCD(15, 32)

Part c

- Use induction to establish the following loop invariant:
- Each time we reach the test of the *while* loop on line (3),
  (i) $0 \leq x \leq y$ and (ii) $\gcd(x, y) = \gcd(a, b)$.

Part c

- Use induction to establish the following loop invariant:
- Each time we reach the test of the *while* loop on line (3),
  (i) $0 \le x \le y$ and (ii) $\gcd(x, y) = \gcd(a, b)$.

- Let x_n, y_n, r_n be the value of x, y, r the n-th time we reach the test of the while loop.
- P(n):$0 \le x_n \le y_n$ and $\gcd(x_n, y_n) = \gcd(a, b)$.

Part c - Base Case: k=1

procedure $euclid\_GCD(a, b)$
if $a \leq b$, ***then*** $(x, y) \leftarrow (a, b)$, ***else*** $(x, y) \leftarrow (b, a)$
while $x > 0$ ***do***
$r \leftarrow y \bmod x$
$y \leftarrow x$
$x \leftarrow r$
End
return y

- Prove by cases:
  - $a \leq b$: The condition on the first line is true.
    - $\mathbf{0} \leq \boldsymbol{x} \leq \boldsymbol{y}$: We set $x_1 = a$ and $y_1 = b$. Then $x_1 \leq y_1$. Since the specification says a and b are non-negative, $x_1 \geq 0$ holds too.
    - **gcd($\boldsymbol{x_1}, \boldsymbol{y_1}$) = gcd($\boldsymbol{a}$, $\boldsymbol{b}$)** by initialization.
  - $b < a$: The condition on the first line is false.
    - $\mathbf{0} \leq \boldsymbol{x} \leq \boldsymbol{y}$: We set $x_1 = b$ and $y_1 = a$. Then $x_1 \leq y_1$. Since the specification says a and b are non-negative, $x_1 \geq 0$ holds too.
    - **gcd($\boldsymbol{x_1}, \boldsymbol{y_1}$) = gcd($\boldsymbol{b}$, $\boldsymbol{a}$) = gcd($\boldsymbol{a}$, $\boldsymbol{b}$)** by initialization.

Part c - Inductive Step

- Induction Hypothesis: $0 \leq x_k \leq y_k$ and $\gcd(x_k, y_k) = \gcd(a, b)$.

euclid_GCD

procedure $euclid\_GCD(a, b)$

if $a \le b$, ***then*** $(x, y) \leftarrow (a, b)$, ***else*** $(x, y) \leftarrow (b, a)$

while $x > 0$ ***do***

  $r \leftarrow y \bmod x$

  $y \leftarrow x$

  $x \leftarrow r$

End

return y

euclid_GCD

procedure $euclid\_GCD(a, b)$

if $a \leq b$, ***then*** $(x, y) \leftarrow (a, b)$, ***else*** $(x, y) \leftarrow (b, a)$

k

while $x > 0$ ***do***

$0 \leq x_k \leq y_k$ and $\gcd(x_k, y_k) = \gcd(a, b)$.

    $r \leftarrow y \bmod x$

    $y \leftarrow x$

    $x \leftarrow r$

End

return y

euclid_GCD

procedure $euclid\_GCD(a, b)$

if $a \le b$, ***then*** $(x, y) \leftarrow (a, b)$, ***else*** $(x, y) \leftarrow (b, a)$

while $x > 0$ ***do***

  $r \leftarrow y \bmod x$ $r_{k+1} \leftarrow y_k \bmod x_k$

  $y \leftarrow x$

  $x \leftarrow r$

End

return y

euclid_GCD

procedure $euclid\_GCD(a, b)$
if $a \leq b$, ***then*** $(x, y) \leftarrow (a, b)$, ***else*** $(x, y) \leftarrow (b, a)$
while $x > 0$ ***do***
$r \leftarrow y \bmod x$
$y \leftarrow x$ $y_{k+1} \leftarrow x_k$
$x \leftarrow r$
End
return y

euclid_GCD

procedure $euclid\_GCD(a, b)$

if $a \leq b$, ***then*** $(x, y) \leftarrow (a, b)$, ***else*** $(x, y) \leftarrow (b, a)$

while $x > 0$ ***do***

  $r \leftarrow y \bmod x$

  $y \leftarrow x$

  $x \leftarrow r$ $x_{k+1} \leftarrow r_{k+1}$

End

return y

euclid_GCD

procedure $euclid\_GCD(a, b)$

if $a \le b$, ***then*** $(x, y) \leftarrow (a, b)$, ***else*** $(x, y) \leftarrow (b, a)$

while $x > 0$ ***do***

    $r \leftarrow y \bmod x$

    $y \leftarrow x$

    $x \leftarrow r$

End

return y

euclid_GCD

procedure $euclid\_GCD(a, b)$

if $a \leq b$, ***then*** $(x, y) \leftarrow (a, b)$, ***else*** $(x, y) \leftarrow (b, a)$

k+1

while $x > 0$ ***do***

$0 \leq x_{k+1} \leq y_{k+1}$ and $\gcd(x_{k+1}, y_{k+1}) = \gcd(a, b)$?

  $r \leftarrow y \bmod x$

  $y \leftarrow x$

  $x \leftarrow r$

End

return y

Part c - Inductive Step

- Induction Hypothesis: $\boldsymbol{0 \leq x_k \leq y_k}$ and $\gcd(x_k, y_k) = \gcd(a, b)$.
- Since x_k and y_k are both non-negative, we know
$$0 \leq y_k \bmod x_k \leq x_k - 1$$
WHY? Remainder range
- Since $r_{k+1} \leftarrow y_k \bmod x_k$ and $x_{k+1} \leftarrow r_{k+1}$, we have that
$$0 \leq x_{k+1} \leq x_k - 1 \leq x_k$$
- And since $y_{k+1} \leftarrow x_k$, therefore we have
$$0 \leq x_{k+1} \leq y_{k+1}$$

which establish the first part of the invariant.

Part c - Inductive Step

- Induction Hypothesis: $0 \leq x_k \leq y_k$ and $\mathbf{gcd}(\boldsymbol{x_k}, \boldsymbol{y_k}) = \mathbf{gcd}(\boldsymbol{a}, \boldsymbol{b})$.
- For $\gcd(x_{k+1}, y_{k+1})$, since
  - $y_{k+1} \leftarrow x_k$
  - $r_{k+1} \leftarrow y_k \bmod x_k$ and $x_{k+1} \leftarrow r_{k+1}$
- We know that

$$\gcd(x_{k+1}, y_{k+1}) = \gcd(y_k \bmod x_k, x_k)$$

Part c - Inductive Step

- Induction Hypothesis: $0 \leq x_k \leq y_k$ and $\mathbf{gcd}(\boldsymbol{x_k}, \boldsymbol{y_k}) \ = \ \mathbf{gcd}(\boldsymbol{a}, \boldsymbol{b})$.
- By theorem "Let x and y be two positive integers. Then gcd(*x*, *y*) = gcd(*y* mod *x*, *x*)", we know that
$$\gcd(x_{k+1}, y_{k+1}) = \gcd(y_k \bmod x_k, x_k) = \gcd(x_k, y_k)$$
- Thus by induction hypothesis we know that
$$\gcd(x_{k+1}, y_{k+1}) = \gcd(y_k \bmod x_k, x_k) = \gcd(x_k, y_k) = \gcd(a, b)$$
- which establish the second part of the invariant.
- Thus the inductive step holds.

Part c - Conclusion

- Therefore, by induction our loop invariants hold each time we test the while loop on line (3).

Part d – Program Correctness

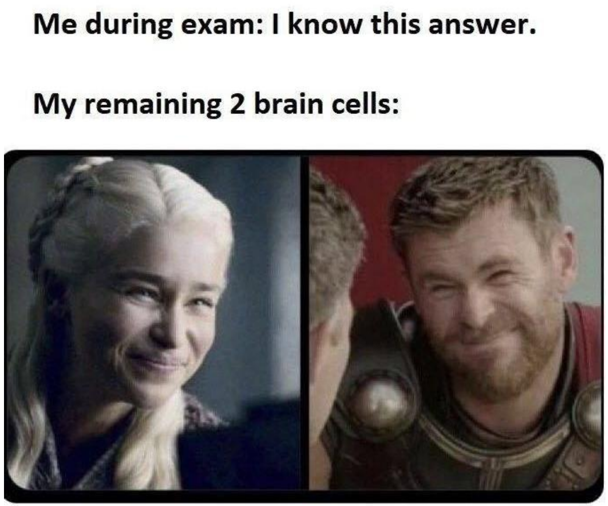
- Partial Correctness:
- Since the loop invariant (i) tells us that $x \geq 0$, and the loop terminates only if $x \leq 0$, we see that when the loop terminates, $x = 0$. In that case $\gcd(x, y) = y$ because $\gcd(0, y) = y$.
- Also by loop invariant (ii), $\gcd(x, y) = \gcd(a, b)$ at that point, which gives us partial correctness of the algorithm.

Part d – Program Correctness

- Termination:
- Now we show that the algorithm terminates. Consider the situation the k-th time we reach the test of the while loop.
- If $x_k \leq 0$, the loop terminates and the algorithm returns result.
- If $x_k > 0$, we observe that $x_{k+1} < x_k$ (proved in part c). Thus, x is a non-negative integer which decreases by at least 1 in each iteration of the loop unless it's zero already. Since x is initialized to be $\min(a, b)$, after at most $\min(a, b)$ iterations of the loop, $x = 0$, the loop terminates, and the program returns.

Discussion Participation

- Sec #
- How did the exam go/ how was your experience?



Reference

- Properties and Definitions:

https://canvas.wisc.edu/courses/212414/pages/properties-and-definitions?module_item_id=3020063

- Week 7 information page:
- https://canvas.wisc.edu/courses/212414/pages/week-7-discussion-info-mpmfe