37181 DISCRETE MATHEMATICS

©Murray Elder, UTS

Lecture 8: correctness of computer code; PMI and WOP

PLAN

- correctness of computer code
- PMI and WOP

CORRECTNESS OF COMPUTER CODE

We say a procedure/computer program/(algorithm) is correct if

- It stops after a finite number of steps.
- The output claimed to be produced by the algorithm is what is promised.

CORRECTNESS OF COMPUTER CODE

We say a procedure/computer program/(algorithm) is correct if

- It stops after a finite number of steps..
- The output claimed to be produced by the algorithm is what is promised.

Wikipedia: In computer science, a *loop invariant* is a property of a program loop that is true before (and after) each iteration.

while, for

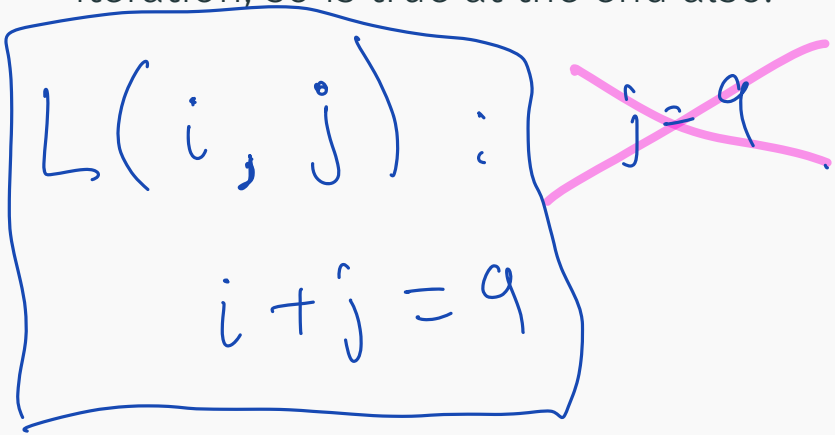
It is a logical assertion, sometimes checked within the code by an assertion call. Knowing its invariant(s) is essential in understanding the effect of a loop.

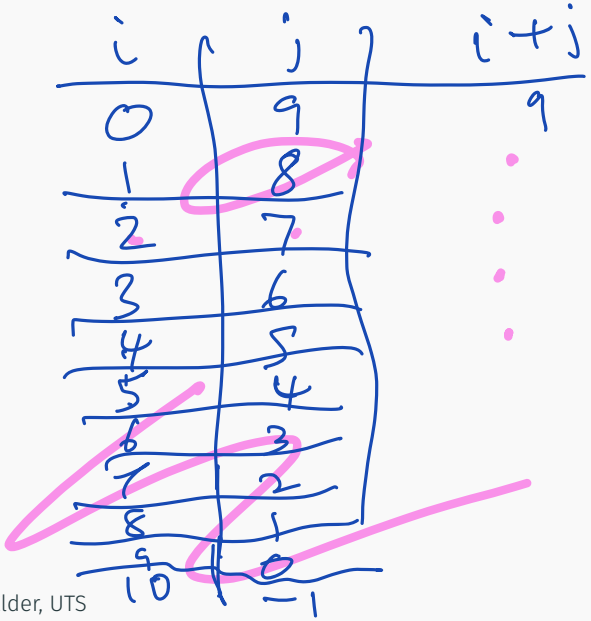
statement

Here is a fragment of slightly useless code.

```
int j = 9;
for(int i=0; i<10; i++)
    j--;
```

There is no output, but we will use this to illustrate loop invariant. Something that is true at the start, and remains true after each iteration, so is true at the end also.





Here is a fragment of slightly useless code.

```
int j = 9;
for(int i=0; i<10; i++)
   j--;
```

There is no output, but we will use this to illustrate loop invariant. Something that is true at the start, and remains true after each iteration, so is true at the end also.

Termination: for loop will stop when $i<10$ becomes false.

Loop invariant: $L(i,j)$: $i+j=9$

if $i+j=9$, then do one iteration of for loop, $i'=i+1$, $j'=j-1$

$j'+i' = j-1+i+1$
$= i+j = 9$

Here is a fragment of slightly useless code.

```
int j = 9;
for(int i=0; i<10; i++)
   j--;
```

There is no output, but we will use this to illustrate loop invariant. Something that is true at the start, and remains true after each iteration, so is true at the end also.

Termination:

Loop invariant: $i + j = 9$

```
input x,d positive integers;
q=0;
r=x;
while(r>=d)
   r=r-d;
   q++;
return (q,r)
```

q	r
0	~~x~~
1	x−d
2	x−2d
3	
4	
⋮	

if $x < d$ Stop straight away $(0, x)$

Else $x \geq d$

Division Algorithm

```
input x,d positive integers;
q=0;
r=x;
while(r>=d)
   r=r-d;
   q++;
return (q,r)
```

one iteration:
$q' = q+1$
$r' = r-d$
$q'd+r' = (q+1)d + r-d$
$= qd + d + r - d$
$= qd + r$
$= x$

Termination: because r begins as positive (x) and we are subtracting d each iteration.

Loop invariant:

$L(q,r)$: $x = qd + r$, $0 \le r < d$

Beginning
$0 \cdot d + x$
$= x$
TRUE.

EXAMPLE FROM WIKIPEDIA

```
1  int max(int n,const int a[]) {
2      int m = a[0];
3      // m equals the maximum value in a[0...0]
4      int i = 1;
5      while (i != n) {
6          // m equals the maximum value in a[0...i-1]
7          if (m < a[i])
8              m = a[i];
9          // m equals the maximum value in a[0...i]
10         ++i;
11         // m equals the maximum value in a[0...i-1]
12     }
13     // m equals the maximum value in a[0...i-1], and i==n
14     return m;
15 }
```

Termination:

Loop invariant:

$\neg(a=0 \wedge b=0)$

Euclidean algorithm: $a, b \in \mathbb{Z}_+$ (for simplicity) and $a \neq 0 \vee b \neq 0$.

The steps are:

1. Start with (a, b) such that $a \geqslant b$. (ie. put them in order).
2. While $b \neq 0$,
   compute the remainder $0 \leqslant r < b$ of a divided by b.
   set $a = b, b = r$ (and thus $a \geqslant b$ again).
3. Return a

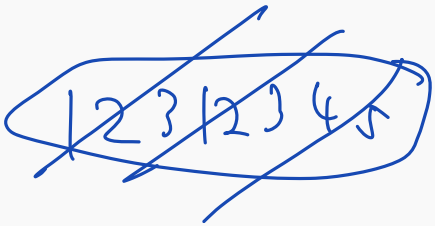
call Division algorithm

$a = qb + r$

$a' = b \qquad b' = r$

≥ 0

Termination:
b is always getting strictly smaller, but ≥ 0. Then while stops.



Input x, y:

Euclidean algorithm: ~~a, b~~ $x, y \in \mathbb{Z}_+$ (for simplicity) and ~~$a \neq 0$~~ $x \neq 0 \vee$ ~~$b \neq 0$~~ $y \neq 0$.

The steps are: Set $a = x$, $b = y$.

1. Start with (a, b) such that $a \geqslant b$. (ie. put them in order).
2. While $b \neq 0$,
   compute the remainder $0 \leqslant r < b$ of a divided by b.
   set $a = b, b = r$ (and thus $a \geqslant b$ again).
3. Return a

$a = qb + r$

if $d \mid b$ and $d \mid r$
$\rightarrow d \mid a$ as well.

Termination: ✓

Loop invariant: $L(a, b)$: $gcd(a, b) = gcd(x, y)$

True at beginning.

One iteration: $a' = b$ $b' = r$

$gcd(a', b') = gcd(b, r) \stackrel{?}{=} gcd(a, b)$ ← Need to show this

Cool thing:

$b = 0$

$\gcd(a, 0)$

$=$

a

by loop invar

$= \gcd(x, y)$

More practice on loop invariants in the homework and worksheet.

Finally, so far in this course, we have asked you to *accept* two "facts" or axioms:

WOP: Every nonempty subset of $\mathbb{N}$ has a first element.

PMI: If $P(0)$ is true and $(P(k) \to P(k+1))$ is true then $P(n)$ is true for all $n \in \mathbb{N}$.

Axiom: true without following from any other fact.

Statement

Theorem

WOP implies PMI

Proof.

Assume $P(0)$ and $(P(k) \to P(k+1))$ are both true. Define

$$S = \{i \in \mathbb{N} \mid P(i) \text{ is false}\}.$$

Suppose S is not empty.
Then S is a non empty subset of $\mathbb{N}$, so by WOP, S has a first element, $s \in S$.
$\therefore P(s)$ is false $\therefore s \neq 0$.
What about $s-1$? $s-1 \in \mathbb{N}$. $P(s-1)$ must be true because s was first element of S. □
But $P(s-1) \to P(s-1+1)$ so $P(s)$ is true.
Contradiction. $\therefore S = \emptyset$ $\therefore P(i)$ true for all $i \in \mathbb{N}$.

Theorem

PMI implies WOP

Proof.

Suppose WOP does not hold.
This means there exists a non-empty subset $S \subseteq \mathbb{N}$ which does not have a first element.

If $0 \in S$, it would be a first element for $i \geq 0$ for all $i \in \mathbb{N}$ because

$\therefore 0 \notin S$.

□

I want to use PMI, so I define a statement.

Let $P(n)$ be the statement:

$$\{0, 1, 2, \ldots, n\} \subseteq \mathbb{N} \setminus S.$$

$P(0)$? $P(0)$ is true because

$$\{0\} \subseteq \mathbb{N} \setminus S$$

because $0 \notin S$.

Suppose $P(k)$ is true some $k \in \mathbb{N}$.

Consider $P(k+1)$:

$$\{0, 1, 2, \ldots k, k+1\}.$$

I know, because $P(k)$ is true, that

$\{0, 1, 2, \ldots k\} \subseteq \mathbb{N} \setminus S$.

So I need to check that $(k+1) \notin S$.

If $k+1 \in S$, it would be a first element for S

because $0, 1, 2, \ldots k$ are not in S

so every $i \in S$ satisfies $(k+1) \leq i$

NEXT

Next lecture:

- Relations
- Functions
- one-to-one
- onto
- bijection