Correctness

Mars Rovers – $2.5B budget

~~6~~ 7 years (so far)

software updates

2 GB flash

10 kbps https://marsmobile.jpl.nasa.gov/msl/mission/communicationwithearth/data/

~14 days

NASA is a big sponsor of formal verification

https://shemesh.larc.nasa.gov/fm/

Therac-25

radiation therapy machine

hardware interlocking mechanism replaced with buggy software

3 people killed

formal verification important for safety- and mission-critical systems

Invariants

Loop Invariant: something true at beginning of every iteration of loop

Loop Invariant Thm:

some predicate involving variables, and # iterations of loop

For predicate P, if

base case → a) P true before loop starts (after 0 iteration)

induction step → b) if P true before one iteration and guard also T then P true after next iteration

then P is truly a loop invariant

prove by induction that

$\forall n \geq 0$, if there is an nth iteration, then P is true after the nth iteration

Also want loop terminates (guard is eventually false)

P true when loop terminates (and guard is false) guarantees that the postconditions are met ((INV ∧ G) → POST)

```
Sum(A)
  total = 0
  for i = 0 to n-1
    total = total + A[i]
  return total
```

Sorting

in final locations   not sorted yet

```
SelectionSort(A,n)
   i = 0
   while i < n-1
      find min among A[i],...,A[n-1]
      swap min with A[i]
      i = i + 1
```

worst case $\Theta(n^2)$

guard

$\forall k,\ 0 \le k \le i-2 \Rightarrow A[k] \le A[k+1]$

| 4   2   1   3   6   5

1 | 2   4   3   6   5

1   2 | 4   3   6   5

1   2   3 | 4   6   5

1   2   3   4 | 6   5

1   2   3   4   5 | 6

LOOP INVARIANT: $A[0] \le A[1] \le \cdots \le A[i-1]$

and

$A[i-1] \le A[i], \ldots, A[n-1]$

and

A has original values in possibly different order

Postcondition for sorting: $A[0] \le \cdots \le A[n-1]$

and

A has same elements as at start (but in diff order)

```
InsertionSort(A,n)
   i = 1
   while i < n
      insert A[i] into correct location among A[0],...,A[i-1]
      i = i + 1
```

worst case $\Theta(n^2)$

guard

INV: $A[0] \le \cdots \le A[i-1]$ (1st i in order)

and

$A[0], \ldots, A[i-1]$ are original 1st i elements, reordered

and

$A[i], \ldots, A[n-1]$ have original values in original order

4 | 2   1   3   6   5

2   4 | 1   3   6   5

1   2   4 | 3   6   5

1   2   3   4 | 6   5

1   2   3   4   6 | 5

1   2   3   4   5   6 |

precondition: A is non-empty (so $n \geq 1$)
(or change INV part d to "$n>0 \rightarrow i \leq n-1$")

```
SelectionSort(A,n)
  i = 0
  while i < n-1
    find min among A[i],...,A[n-1]
    swap min with A[i]
    i = i + 1
```

guard

LOOP INVARIANT: a) $A[0] \leq A[1] \leq \cdots \leq A[i-1]$
and
b) $A[i-1] \leq A[i], \ldots, A[n-1]$
and
c) A has has original values in possibly a different order
and
d) $i \leq n-1$ (added for the pedants; will use to determine what i is at termination; since i is the loop counter, that isn't the interesting part of the proof)

formal versions of informal statement of invariant (no ...)
$\forall k \in \mathbb{N},\ 0 \leq k \leq i-2 \rightarrow A[k] \leq A[k+1]$
$i \geq 1 \rightarrow \forall k \in \mathbb{N},\ i \leq k \leq n-1 \rightarrow A[i-1] \leq A[k]$

Basis: a) i is initialized to 0, which makes part a empty (no k satisfies $0 \leq k \leq -2$) so vacuously true

b) i is initialized to 0, which means there is no $A[i-1]$, so b is automatically true (or, in the formal version, $i \geq 1$ is F, making the $\rightarrow$ T)

c) no assignments to A yet

d) i is initialized to 0, and by the precondition on n, $n \geq 1$ so $n-1 \geq 0 = i$
$\therefore\ i \leq n-1$

Inductive Step: suppose INV is true before the loop for the old values of the variables and the guard $i < n-1$ is true
[want to show that INV is true after the loop with the new values of the variables]

a) INV-a true before the loop means $A[0] \leq \cdots \leq A[i_{old}-1]$ [want to append $\leq A[i_{new}-1]_{new}$]

choice of min means $\min \leq A[i_{old}]_{old}, \ldots, A[n-1]_{old}$ ☆
INV-b makes $\min \geq A[i_{old}-1]$ since it is selected from things each of which is $\geq A[i_{old}-1]$
swap makes $A[i_{old}]_{new} = \min$
increment makes $i_{new} = i_{old}+1$

So: $A[0] \leq \cdots \leq A[i_{old}-1] \leq \min = A[i_{old}]_{new}$

rewrite using $i_{new} = i_{old}+1$ which is INV-a using the new values.

and $A[0] \leq \cdots \leq A[i_{new}-2] \leq A[i_{new}-1]_{new}$

b) choice of min makes $A[i_{old}] = \min \leq \underbrace{A[i_{old}+1]_{new}, \ldots, A[n-1]_{new}}_{\text{all of these are } A[k]_{old} \text{ for some } k \text{ in } i_{old}+1 \ldots n\text{; see ☆}}$

using "new" since one of these values may have changed after the swap

rewrite using $i_{new} = i_{old}+1$

and so $\underbrace{A[i_{new}-1] \leq A[i_{new}], \ldots, A[n-1]}_{\text{this is INV-b using the new variables}}$

c) only change to A is a swap, which preserves values but reorders
(technically, this is from the postcondition of swap: two elts change

c) only change to A is a swap, which preserves values but reorders
(technically, this is from the postcondition of swap: two elts change places and nothing else changes)

d) the guard is true, so $i_{old} < n-1$ and $\overset{\text{add 1 to both sides}}{\Longrightarrow}$ $i_{old}+1 < n-1+1$
the increment makes $i_{new} = i_{old}+1$
so $i_{new} < n$
which means $i_{new} \le n-1$ since i, n are integers
(pedants: add that to the invariant)

Termination: i increases each time through the loop (technically, part of INV is "i = # times through loop")
and so eventually $i < n-1$ is false

Postcondition: At termination, we have $i \ge n-1$ from the guard being false (that's why the loop terminated)
and $i \le n-1$ from INV-d

$\therefore \quad i = n-1$

plugging $i=n-1$ into INV-a and INV-b gives

$A[0] \le A[1] \le \cdots \le A[n-2] \le A[n-1]$ so A is sorted

(INV-a covers $A[0] \le \cdots \le A[n-2]$; INV-b covers $A[n-2] \le A[n-1]$)

INV-c says it contains the original elements.

Those are the required postconditions for sorting