2.1 Intro. to Algorithm Correctness

Thursday, January 30, 2025 1:21 PM

¿How to Argue that your algorithm is Correct?
by using "Loop invariants"

DEF: A loop invariant is a statement (true/false) about the program variables that is true before and after each iteration of a loop.

A good loop invariant has:

Initialization:- it's true before the first execution of the loop.

Maintenance:- If it is true before an iteration it is true after the iteration.

Termination:- when the loop terminates, the invariant tells us something usefull about the algorithm.

E.G

```
FUNCTION Sum( A[0..n-1])
    s ← 0
    i ← 0
    while i < n do
        s ← s + A[i]
        i ← i + 1
    return s
```

// Inv. s = sum A[0] to A[i-1]

s = sum A[0] to A[i-1] and i = n
s = sum A[0] to A[n-1]

E.G

```
FUNCTION MaxElement( A[0... n-1])        // PRE: A is not empty.
    max ← A[0]
    FOR i ← 1 to n-1 DO
        //:INV: max is the greater element in A[0..i-1]
        IF A[i] > max THEN
            max ← A[i]
    RETURN max        // max is the greatest in A[0..i-1] AND i = n
                      max is the greatest in A[0..n-1]
```

- Use of invariants:
  - Argue for correctness
  - Find missing Assumptions
  - Find Bugs
  - Design the Algorithm in the first place.

—o— EOF