CIS 575. Introduction to Algorithm Analysis
Material for February 14, 2024

Proving Correct an Iterative Algorithm on Arrays

1 Verifying a Loop Reading From an Array

For program that involve arrays, loop invariants typically involve (implicit) quantifiers. To illustrate this, consider a program for finding the *last* occurrence in A of an element x that is known to exist in A. With r the result, and with $x \in A[lo..hi]$ a shorthand for $\exists i \in lo..hi : x = A[i]$, a formal specification is

Precondition $x \in A[1..n]$ (thus $n \geq 1$)

Postcondition $1 \leq r \leq n$, $A[r] = x$, $x \notin A[r+1..n]$

We claim that this specification can be implemented by the algorithm

$r \leftarrow n$
while $A[r] \neq x$
  $r \leftarrow r - 1$
return r

To prove this, we shall use the loop invariant given by

$$1 \leq r \leq n,\ x \in A[1..r],\ x \notin A[r+1..n]$$

and shall now verify all items in the checklist.

Establish We must prove that $1 \leq n \leq n$ and that $x \in A[1..n]$ which follows from the precondition, and that $x \notin A[n+1..n]$ which trivially holds (as $A[n+1..n]$ is empty).

Maintain When the loop guard $A[r] \neq x$ is true, we infer from the loop invariant that $x \notin A[r..n]$ and that $x \in A[1..r-1]$ (and thus $1 \leq r - 1$) which since $r' = r - 1$ shows that the invariant is re-established:

$$1 \leq r' \leq n,\ x \in A[1..r'],\ x \notin A[r'+1..n]$$

Terminate Since the invariant ensures that $r \geq 1$, and since 1 is subtracted from r in each iteration, the loop cannot go on forever.

Correctness At loop exit we have $A[r] = x$ which together with the loop invariant implies the postcondition.