Loop invariants to prove correctness of algorithms

Proving correctness of programs with loops:

// Precondition: Pre
while Condition c do

Statements S

// Postcondition: Post (to be proved)

To prove that the postcondition is satisfied, we write a mathematical proposition, known as a Loop Invariant (LI), and a proof structured as follows:

- Initialization: Pre implies LI
- Maintenance: LI and execution of S, implies LI
- Termination: LI \land \neg c implies Post.

Show that the number of iterations is finite separately, or include it as part of the LI.

**LI: Linear search**

```
LinearSearch(A, n, x): // Search for x in A[1..n]
Precondition: none
1: for i ← 1 to n do
2: if A[i] = x then return true
3: return false
```

**LI: Binary search**

```
BinarySearch(A, n, x): // Search for x in A[1..n]
1: p ← 1 r ← n
2: while p \leq r do
3: q ← (p + r) / 2
4: if x < A[q] then
5: r ← q − 1
6: else if x > A[q] then
7: p ← q + 1
8: else return q + 1
9: return false
```

**LI: Insertion sort**

```
InsertionSort(A, n): // Sort A[1..n]
1: for j ← 2 to n do
2: key ← A[j]
3: i ← j − 1
4: while i > 0 and A[i] > key do
6: i ← i − 1
7: A[i+1] ← key
```

**LI: Partition**

```
Precondition: none.
1: i ← p − 1
2: for j ← p to r − 1 do
3: if A[j] \leq A[r] then
4: i ← i + 1
5: Exchange A[i] and A[r]
6: Exchange A[i+1] and A[r]
7: return i + 1
```

**LI: Merge**

```
Merge(A, p, q, r): // Merge sorted subarrays A[p..q] and A[q+1..r]
Precondition: p \leq q < r, A[p+1] \leq \ldots \leq A[q], and, A[q+1] \leq A[q+2] \leq \ldots \leq A[r].
1: Copy A[p..q] into L[1..q-p+1]
2: Copy A[q+1..r] into R[1..r-q]
3: Set sentinels \( L[q-p+2] = \infty \) and \( R[r-q+1] = \infty \)
4: i ← 1, j ← 1
5: for k ← p to r do
6: if \( L[i] \leq R[j] \) then
7: \( A[k] \leftarrow L[i]; \) i ← i + 1
8: else
9: \( A[k] \leftarrow R[j]; \) j ← j + 1
```