CSCI 241

Scott Wehrwein

Preliminaries: Algorithmic Reasoning Tools

Goals

Know the purpose and contents of a method specification

Know the definition and implications of preconditions and postconditions

Know the definition of loop invariant and be able to illustrate one with an array diagram.

Specification

```
/** return the max value in A
  * precondition: A is nonempty
  * postcondition: max value of A is returned */
public int findMax(int[] A) {
  int max = A[0];
  // invariant: max is the max of A[0..i]
  for (int i = 1; i < A.length; i++) {
    if (A[i] > max) {
      max = A[i];
    }
  }
  return max;
}
```

A method specification is a comment above the method that details the precise **behavior** of the method.

↳ ***what*** it does; not ***how*** it does it

Precondition, Postcondition

```
/** return the max value in A
  * precondition: A is nonempty
  * postcondition: max value of A is returned */
public int findMax(int[] A) {
   int max = A[0];
   // invariant: max is the max of A[0..i]
   for (int i = 1; i < A.length; i++) {
     if (A[i] > max) {
       max = A[i];
     }
   }
   return max;
}
```

caller's responsibility

The precondition is true **before** method execution.

The postcondition is true **after** method execution.

implementer's responsibility

Precondition, Postcondition

We can also state pre and postconditions
for smaller bits of code:

```
// swap a and b if necessary so a <= b
// precondition: ints a, b have values
// postcondition: a <= b
if (a > b) {
    int tmp = a;
    a = b;
    b = tmp;
}
```

This is most often useful for loops - example in a few slides.

Invariant

An invariant is a thing that always remains true.

- a fact you can count on
- a fact you must make sure can be counted on

Invariants are useful for **reasoning about** algorithms and data structures.

Example: an invariant pertaining to Java arrays is that `A.length` always contains the length of the array.

Loop Invariant

```
/** return the max value in A
   * precondition: A is nonempty
   * postcondition: max value of A is returned */
public int findMax(int[] A) {
   int max = A[0];
   // invariant: max is the max of A[0..i]
   for (int i = 1; i < A.length; i++) {
      if (A[i] > max) {
         max = A[i];
      }
   }
   return max;
}
```

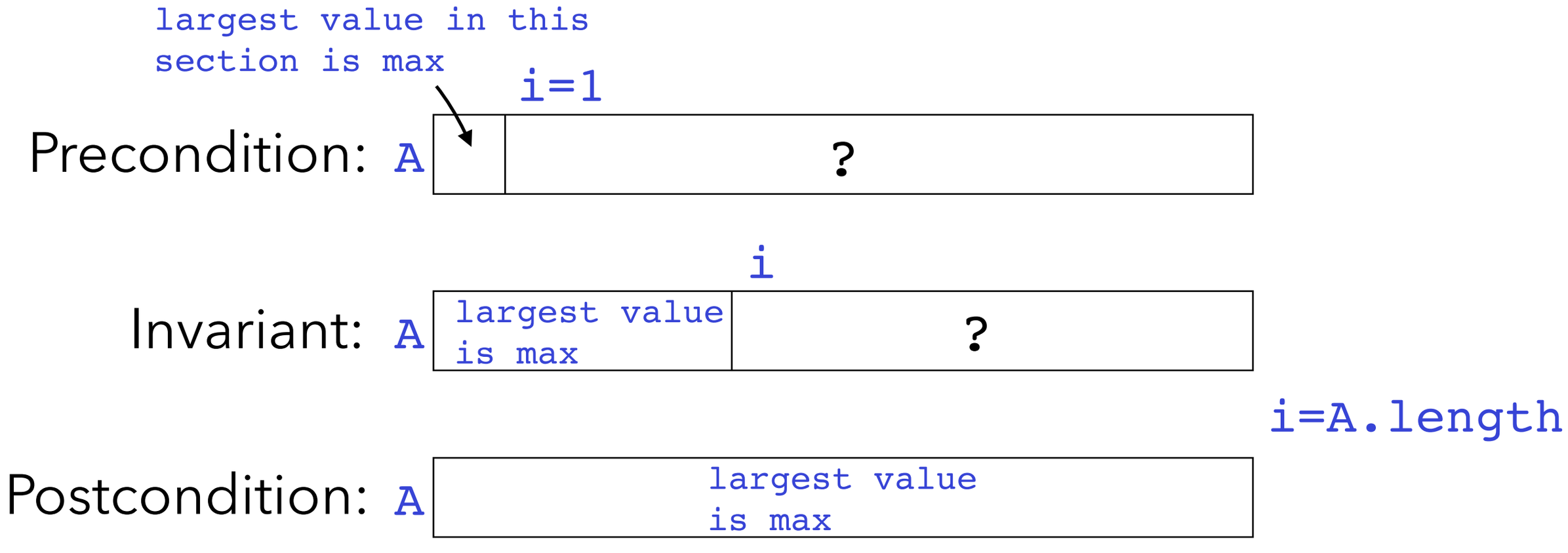
`max is the largest value in:`

Before the loop: `A[0..1]`

After iteration i: `A[0..i]`

After iteration n: `A[0..A.length]`

A loop invariant is true **before the loop begins** and **after each iteration**.




```
int max = A[0];
// invariant: max is the max of A[0..i]
for (int i = 1; i < A.length; i++) {
  if (A[i] > max) {
    max = A[i];
  }
}
```

max is the largest value in:

Before the loop: `A[0..1]`

After iteration i: `A[0..i]`

After iteration n: `A[0..A.length]`

A loop invariant is true **before the loop begins** and **after each iteration**.