Loop Invariants
Insertion and Selection Sort

Goals:

- Be able to analyze an algorithm's correctness using a given loop invariant.
- Be able to describe in words, implement using a loop invariant, and analyze the runtime of:
  - Selection sort
  - Insertion sort

Tools for Reasoning about Algorithms

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 largest value in A[0..i]
   for (int i = 1; i < A.length; i++) {
      if (A[i] > max) {
         max = A[i];
      }
   }
   return max;
}
```

Interface, not *implementation*

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

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

currentMax is the largest value in:

A[0..1]

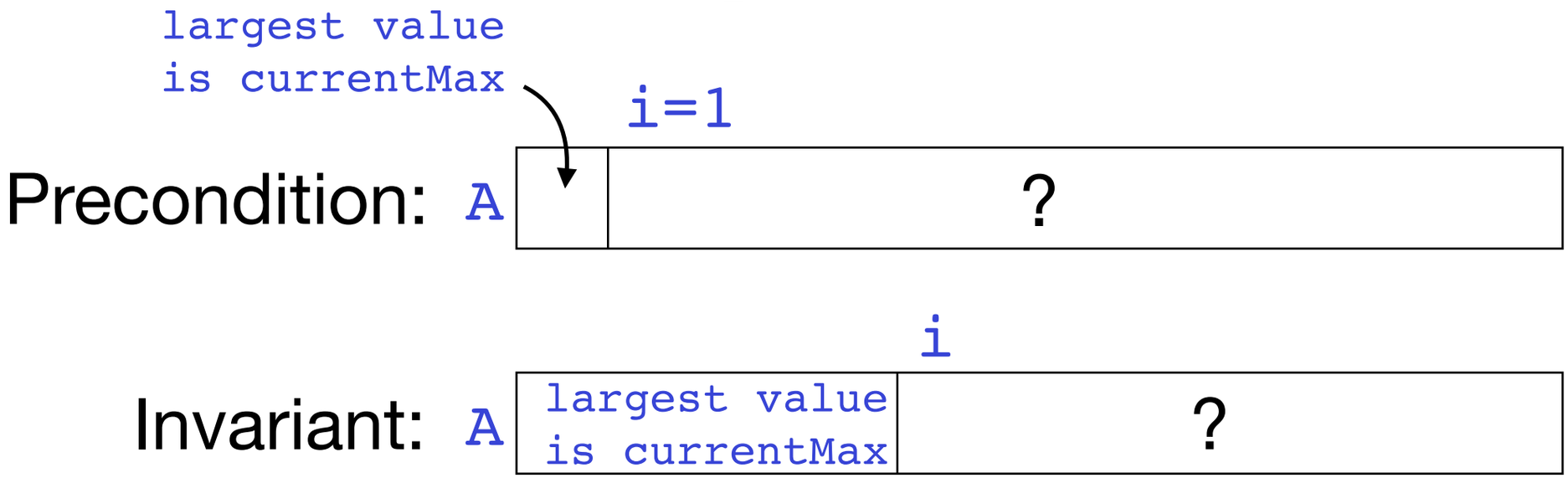
A[0..i]

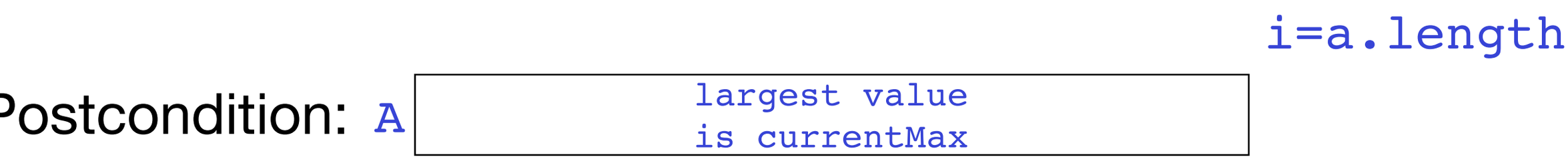
A[0..a.length]

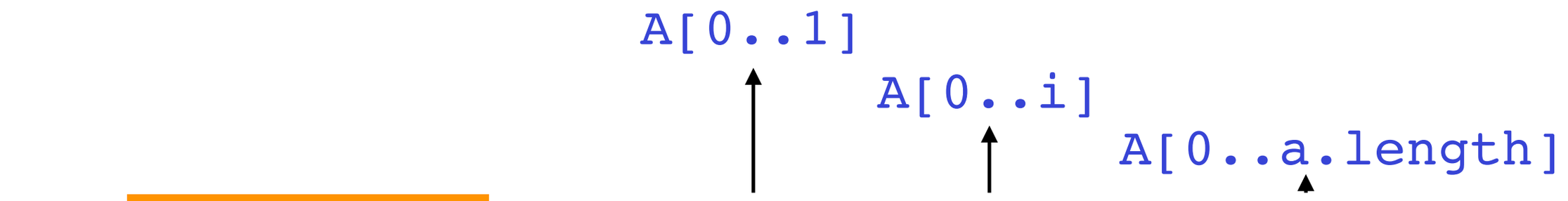
A loop invariant is true **before**, **during**, and **after** the loop.

(at the *end* of each iteration)

Loop Invariant







The loop invariant is true **before**, **during**, and **after** the loop.

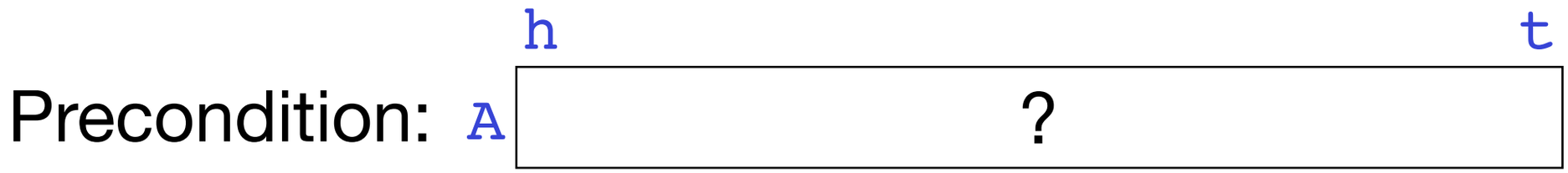
Another Example

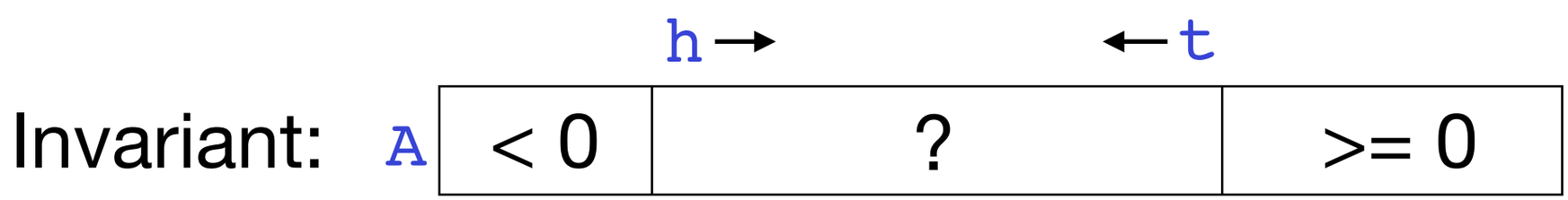
```
/** rearrange A so all negative values are to
  * the left of all non-negative values */
public void separateSign(int[] A);
```

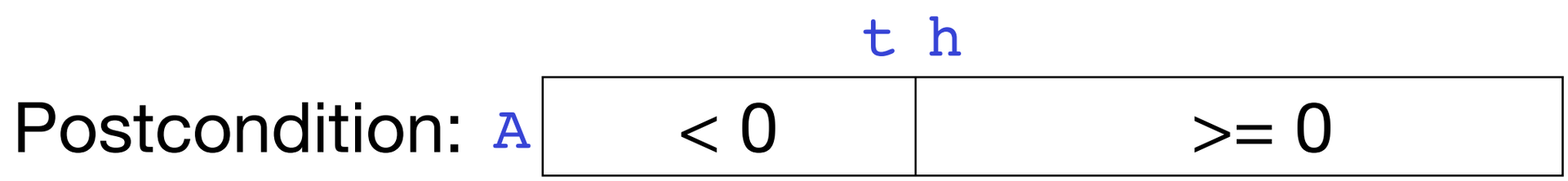
Postcondition: `A`

< 0	>= 0

```
/** rearrange A so all negative values are to
  * the left of all non-negative values */
public void separateSign(int[] A) {
```







Four concerns:

1. **Initialization**: Make the invariant true at the start.
2. **Termination**: Make the loop end when the postcondition is true.
3. **Progress**: Make progress towards the postcondition.
4. **Maintenance**: Make the invariant true after each iteration.

Insertion Sort

Insert A[i] into the sorted sublist A[0..i-1].

Selection Sort

Find the smallest element in A[i..n] and place it at A[i].

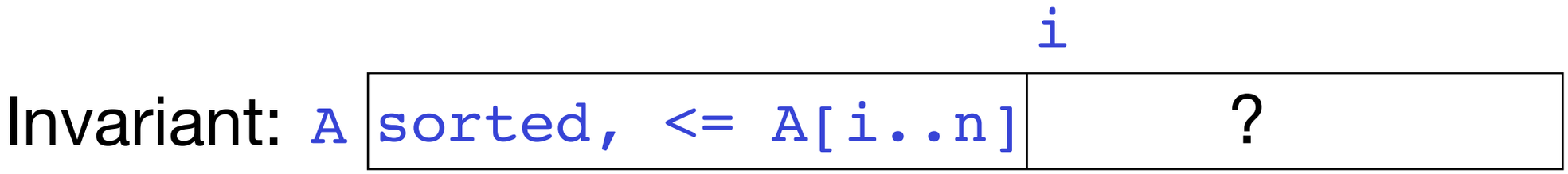
https://visualgo.net/bn/sorting

Insertion Sort

Insert A[i] into the sorted sublist A[0..i-1].

Selection Sort

Find the smallest element in A[i..n] and place it at A[i].

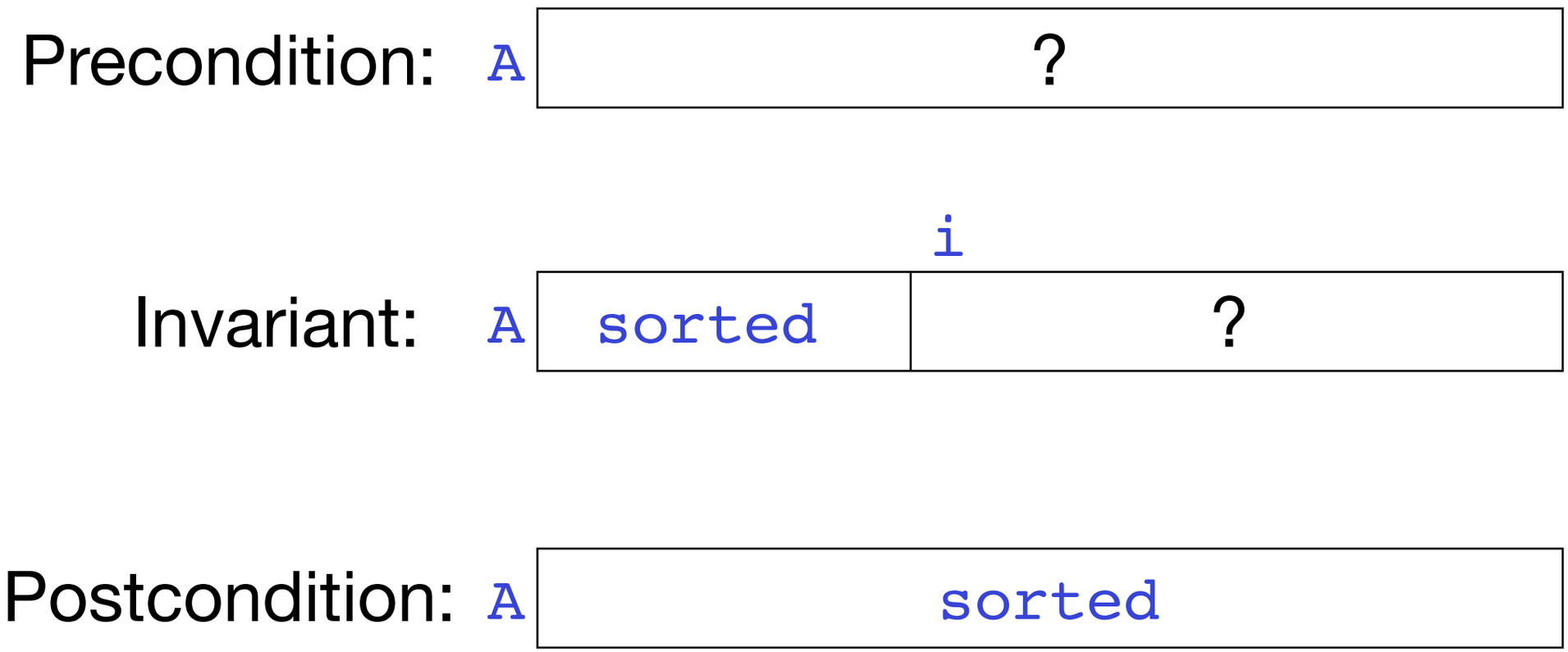


https://visualgo.net/bn/sorting

```
insertionSort(A):
  i = 0;
  while i < A.length:
    // push A[i] to its sorted position
    // increment i
```

```
selectionSort(A):
  i = 0;
  while i < A.length:
    // find min of A[i..A.length]
    // swap it with A[i]
    // increment i
```

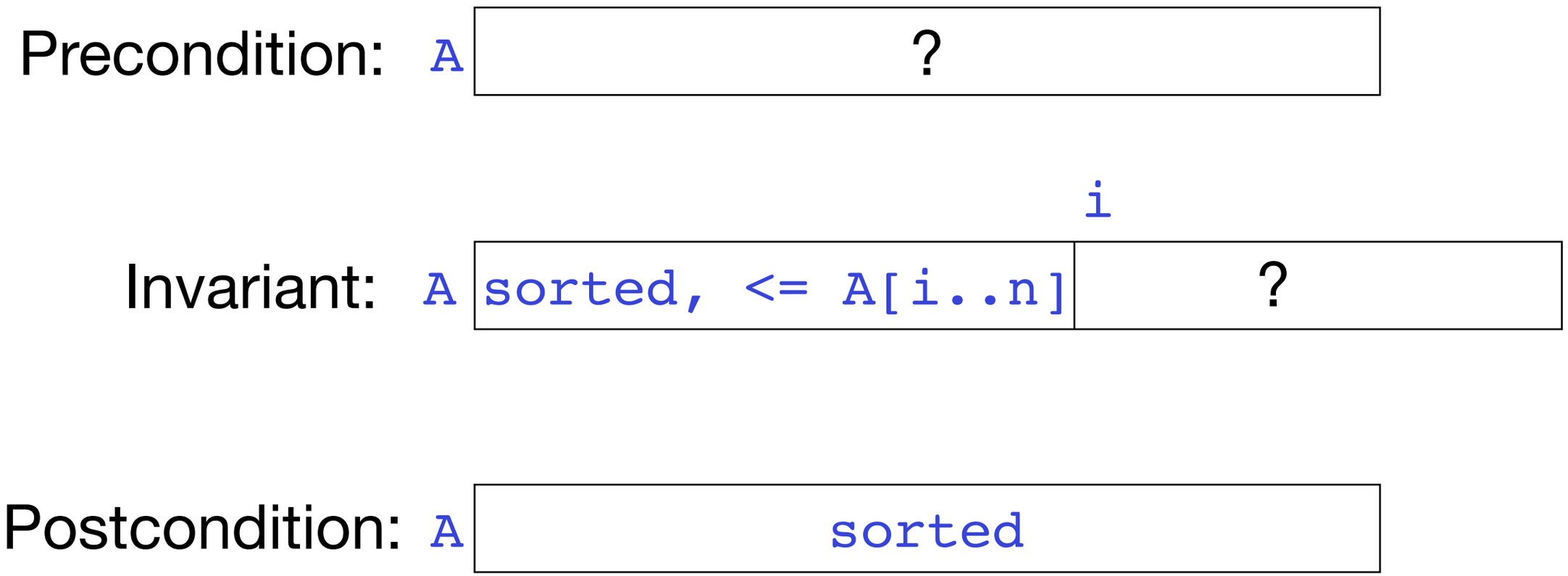
Developing InsertionSort



Four tasks:

1. **Initialization**: Make the invariant true at the start.
2. **Termination**: Make the loop end when the postcondition is true.
3. **Progress**: Make progress towards the postcondition.
4. **Maintenance**: Make the invariant true after each iteration.

Developing SelectionSort



Four concerns:

1. **Initialization**: Make the invariant true at the start.
2. **Termination**: Make the loop end when the postcondition is true.
3. **Progress**: Make progress towards the postcondition.
4. **Maintenance**: Make the invariant true after each iteration.

```
insertionSort(A):
  i = 0;
  while i < A.length:
    j = i;
    while j > 0 and A[j] > A[j-1]:
      swap(A[j], A[j-1])
      j--
    i++
```

ABCD: What's the best and worst-case asymptotic runtime complexity of insertionSort?

	Best	Worst
A	O(n)	O(n)
B	$O(n^2)$	O(n)
C	O(n)	$O(n^2)$
D	$O(n^2)$	$O(n^2)$

Why is this best-case runtime interesting?

```
insertionSort1(A):
   i = 0;
   while i < A.length:
      j = i;
      while j > 0 and A[j] < A[j-1]:
         swap(A[j], A[j-1])
         j--
      i++
```

```
insertionSort2(A):
   i = 0;
   while i < A.length:
      j = i;
      tmp = A[i];
      while j > 0 and tmp < A[j-1]:
         A[j] = A[j-1]
         j--
      i++
```

ABCD: What's the best and worst-case asymptotic runtime complexity of insertionSort2?

	Best	Worst
A	O(n)	O(n)
B	$O(n^2)$	O(n)
C	O(n)	$O(n^2)$
D	$O(n^2)$	$O(n^2)$

Practice problems

Write a precondition, postcondition, and loop invariant for the *inner* loop of InsertionSort.

Develop SelectionSort using the precondition, loop invariant, and postcondition by completing the four tasks:

- Initialization
- Termination
- Progress
- Maintenance