CSCI 241: Data Structures

Lecture 2

Tools for talking about algorithms

Intro to sorting

Announcements

- Lab 1 is out - get started before lab time on Thursday.

- Lots of things to get hung up on - make sure you have time to get help.

Goals

- A slide like this will appear at the start of each lecture.

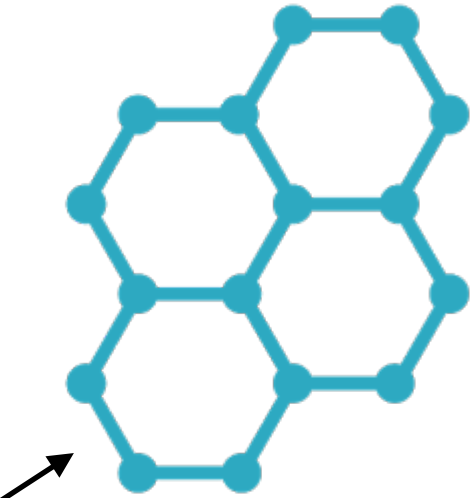
- This is to be as transparent as possible about what I expect you to get out of the lecture.

  - and consequently, what I will expect you know for assignments, quizzes, and exams.

Goals

- Understand the range index convention a..b
- Know the definition of specification, precondition, postcondition, and invariant.
- Be able to execute insertion sort and selection sort on paper.
- Be able to implement insertion sort and selection sort.

Socrative



the Socrative logo will appear on slides where I'm asking a poll question

- Go to http://socrative.com (or open the Socrative Student app)
- Click the blue "Login" button at the top right
- Click "Student Login"
- Enter "CSCI241"
- You should see A, B, C, D, E as selectable options.

Go ahead and try this out now.

Sorting Algorithms

Why?

- Arrays are the simplest and most ubiquitous data structure available to us.
- Sorting algorithms are a fundamental piece of knowledge for computer scientists
- An entry point into the practice of developing, and analyzing algorithms.

Preliminaries:
Tools for Talking about Algorithms

Range Indices

`a..b` denotes the range of consecutive integers from (and **including**) a up to (but **excluding**) b.

Examples:

- 0..5 is the range 0, 1, 2, 3, 4
- A[4..6] denotes the 4th and 5th elements of A
- 7..8 is a range containing only 7
- 6..6 is a valid range but contains no elements

Range Indices

`a..b` denotes the range of consecutive integers from (and **including**) a up to (but **excluding**) b.

- What elements are in 2..6?

A. `[3,4,5]`

B. `[2,3,4,5,6]`

C. `[3,4,5,6]`

D. `[2,3,4,5]`

Range Indices

`a..b` denotes the range of consecutive integers from (and **including**) a up to (but **excluding**) b.

- How many elements are in the range `a..b`?

A. `b-a-1`

B. `a-b-1`

C. `b-a+1`

D. `b-a`

Range Indices

`a..b` denotes the range of consecutive integers from (and **including**) a up to (but **excluding**) b.

- Recall that `A.length` gives A’s length. What range denotes all elements of A?

```
A. A[0..A.length]

B. A[0..A.length-1]

C. A[0..A.length+1]

D. A[1..A.length-1]
```

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.

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.

method implementer's responsibility

(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:

A[0..1]

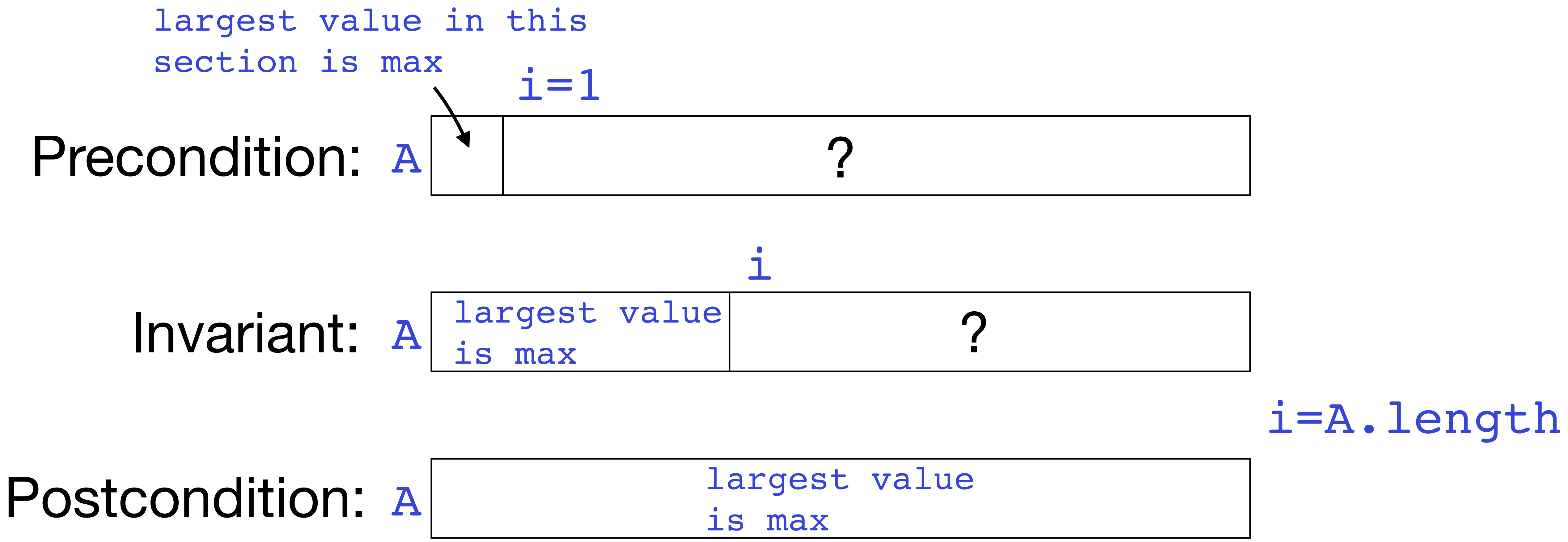
A[0..i]

A[0..A.length]

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

(at the *end* of each iteration)

Loop Invariant



largest value in this
section is max

i=1

Precondition: A

?

i

Invariant: A

largest value
is max

?

i=A.length

Postcondition: A

largest value
is max

A[0..1]

A[0..i]

A[0..a.length]

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

Mystery Algorithm

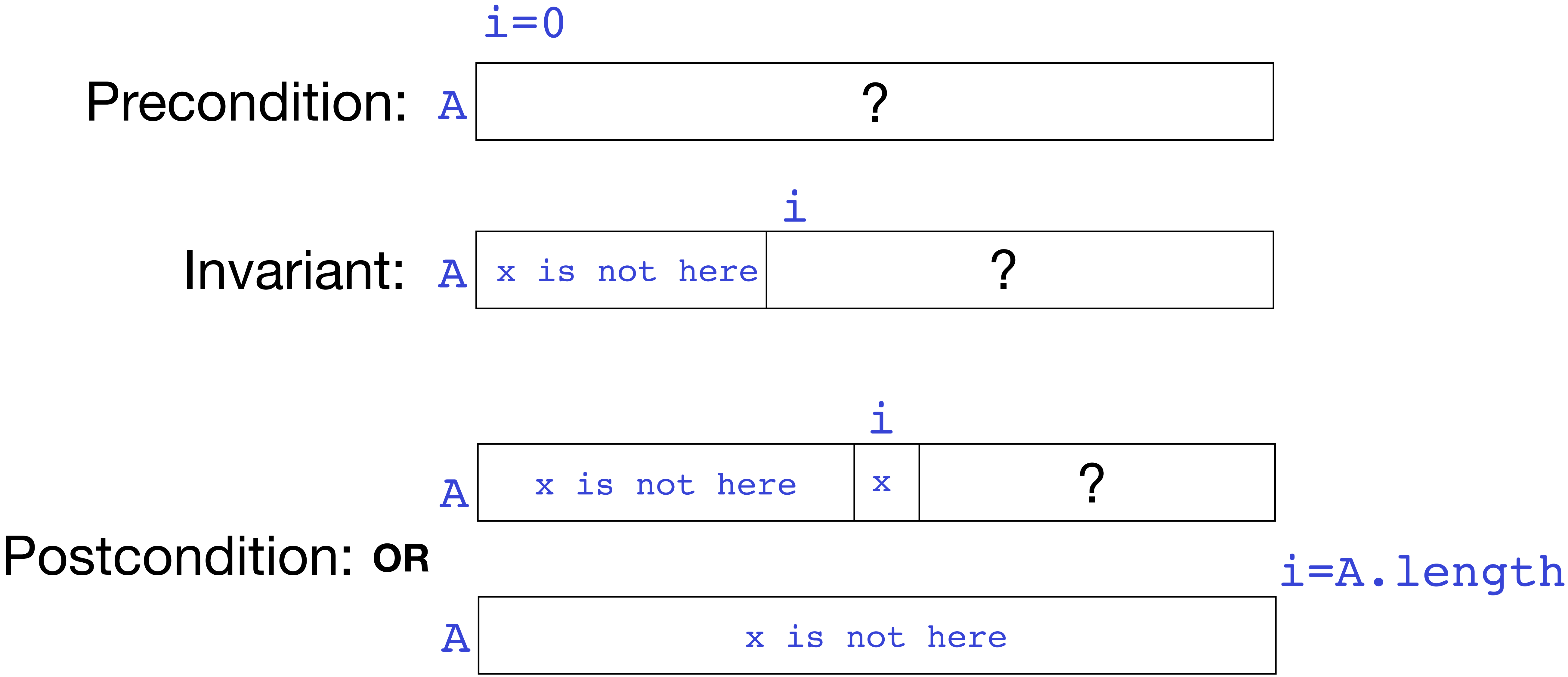
what does this do?

Inputs:

- an `int` `x`,
- an array of `int`s `A`

Output:

- final value of `i`



Precondition: A [?] i=0

Invariant: A [x is not here | ?] i

Postcondition: A [x is not here | x | ?] i

OR

A [x is not here] i=A.length

Mystery Algorithm

what does this do?

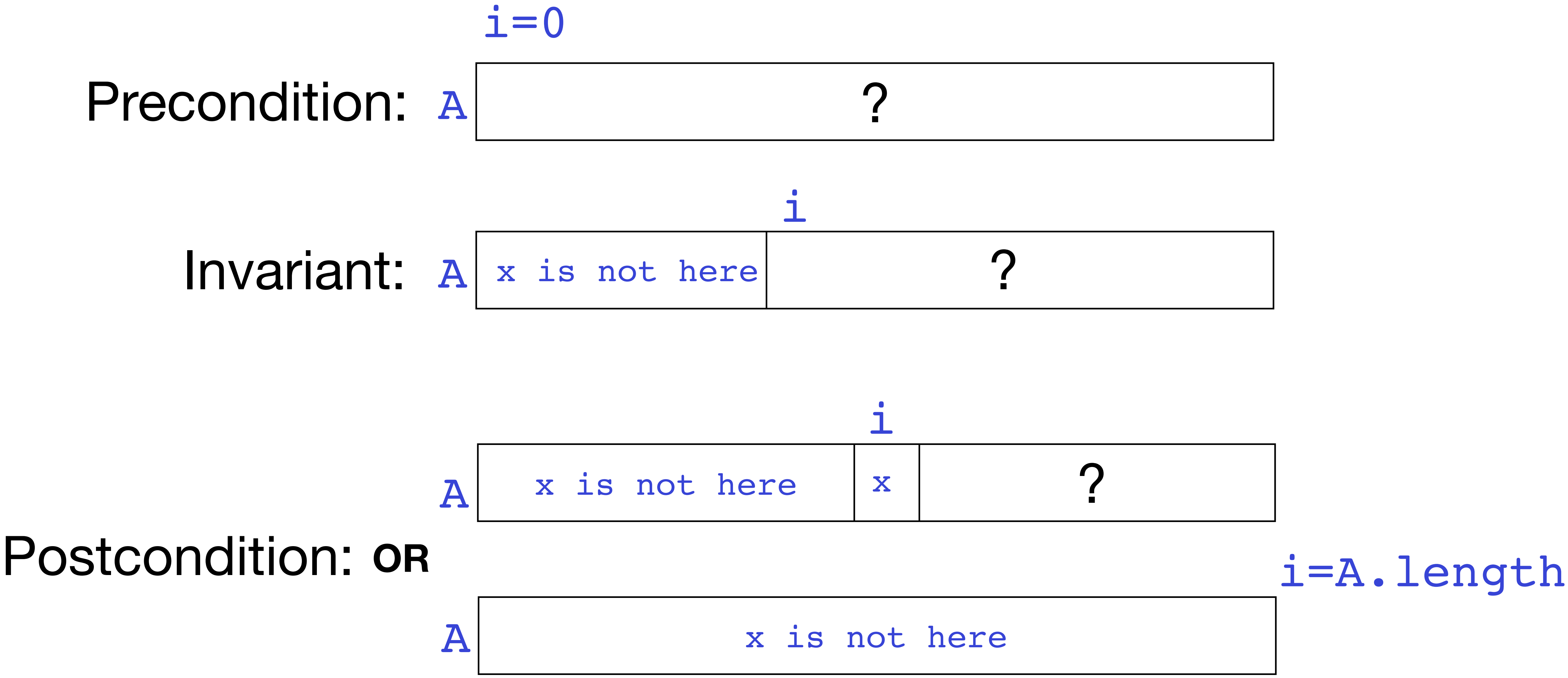
it returns the index of the first `x` if it is found in the array, or `A.length` otherwise

Inputs:

- an `int` `x`,
- an array of `int`s `A`

Output:

- final value of `i`



i=0

Precondition: A ?

i

Invariant: A x is not here ?

i

A x is not here x ?

Postcondition: OR

i=A.length

A x is not here

Interlude: Class Norms

- Let's talk about what **norms** we want to establish for our class.

  rules, conventions, expectations, etc. that we all agree to follow

- In small groups, spend 4 minutes introducing yourselves and agree on 1-3 norms for this class.
  - Can be anything, but thinking about Zoom etiquette may be useful this quarter.
  - Can relate to your expectations of me as well as of your fellow classmates.
- One member of the group: submit your norms to the open-ended poll on Socrative.