The number of values in the segment is

Example of an assertion about a sequence b. It asserts that:
1. b[0..k-1] is sorted (i.e., its values are in ascending order)
2. Everything in b[0..k-1] is <= everything in b[k..len(b)-1]

Given index h of the first element of a segment and index k of the element that follows that segment, the number of values in the segment is k - h.
b[h..k-1] has k - h elements in it.

Dutch national flag: trim

Make the red, white, blue sections initially empty:
- Range i..i+1 has 0 elements
- Main reason for this trick

Changing loop variables turns invariant into postcondition.

Finding the minimum of a sequence.

Put negative values before nonnegative ones.

Partition Algorithm

- Given a sequence b[h..k] with some value x in b[h]:
- Swap elements of b[h..k] and store in j to truthify post:

Agrees with precondition when i = h, j = k+1
Agrees with postcondition when j = i+1

Finding the minimum of a sequence.

Put negative values before nonnegative ones.

Partition Algorithm

- Given a sequence b[h..k] with some value x in b[h]:
- Swap elements of b[h..k] and store in j to truthify post:

Agrees with precondition when i = h, j = k+1
Agrees with postcondition when j = i+1

Finding the minimum of a sequence.

Put negative values before nonnegative ones.

Partition Algorithm

- Given a sequence b[h..k] with some value x in b[h]:
- Swap elements of b[h..k] and store in j to truthify post:

Agrees with precondition when i = h, j = k+1
Agrees with postcondition when j = i+1

Finding the minimum of a sequence.

Put negative values before nonnegative ones.

Partition Algorithm

- Given a sequence b[h..k] with some value x in b[h]:
- Swap elements of b[h..k] and store in j to truthify post:

Agrees with precondition when i = h, j = k+1
Agrees with postcondition when j = i+1

Finding the minimum of a sequence.

Put negative values before nonnegative ones.

Partition Algorithm

- Given a sequence b[h..k] with some value x in b[h]:
- Swap elements of b[h..k] and store in j to truthify post:

Agrees with precondition when i = h, j = k+1
Agrees with postcondition when j = i+1

Finding the minimum of a sequence.
**Partition Algorithm Implementation**

```python
def partition(h, k):
    # Partition list b[h..k] around a pivot x = b[h]
    i = h; j = k-1; x = b[h]
    # invariant: b[h..i] < x, b[i+1..k] >= x
    while i < j:
        if x == b[j+1]:
            # Move to end of block.
            _swap(b, i, j)
            j -= 1
        else:
            # b[j+1] > x
            _swap(b, i+1, j)
            i += 1
        # post: b[h..i] < x, b[i..j] >= x, and b[i+1..k] >= x
    return i
```

**Partition Algorithm Implementation**

```python
def partition(h, k):
    # Partition list b[h..k] around a pivot x = b[h]
    i = h; j = k-1; x = b[h]
    # invariant: b[h..i] < x, b[i+1..k] >= x
    while i < j:
        if x == b[j+1]:
            # Move to end of block.
            _swap(b, i, j)
            j -= 1
        else:
            # b[j+1] > x
            _swap(b, i+1, j)
            i += 1
        # post: b[h..i] < x, b[i..j] >= x, and b[i+1..k] >= x
    return i
```

**Dutch National Flag Variant**

- Sequence of integer values
  - ‘red’ = negatives, ‘white’ = 0, ‘blue’ = positive
  - Only rearrange part of the list, not all

**Dutch National Flag Algorithm**

```python
def dnf(h, k):
    # Returns partition points as a tuple (l,j)
    t = h; l, j = k
    # inv: b[h..l] < 0, b[l..j-1] > 0, b[j] = 0, b[j+1..k] > 0
    while t < j:
        if b[l] < 0:
            swap(b, l, l+1)
            t += 1
        elif b[j] == 0:
            l = j+1
        else:
            swap(b, l, j)
            t = l-1; j = j+1
        # post: b[h..l] < 0, b[l..j-1] > 0, b[j] = 0, b[j+1..k] > 0
    return (l, j)
```

**Dutch National Flag Algorithm**

```python
def dnf(h, k):
    # Returns partition points as a tuple (l,j)
    t = h; l, j = k
    # inv: b[h..l] < 0, b[l..j-1] > 0, b[j] = 0, b[j+1..k] > 0
    while t < j:
        if b[l] < 0:
            swap(b, l, l+1)
            t += 1
        elif b[j] == 0:
            l = j+1
        else:
            swap(b, l, j)
            t = l-1; j = j+1
        # post: b[h..l] < 0, b[l..j-1] > 0, b[j] = 0, b[j+1..k] > 0
    return (l, j)
```