Recall: Horizontal Notation

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 \( \leq \) everything in \( b[k..\text{len}(b)-1] \)

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.

Partition Algorithm Implementation

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

Dutch National Flag Variant

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

```
pre: \( b \)
\( h \) \( \leq \) 0 \( \leq \) i = j+1 \( \leq \) k
post: \( b \)
\( h \) \( \leq \) i \( \leq \) j \( \leq \) k
```

Dutch National Flag Algorithm

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

Changing the Invariant

- Different invariants = different code
  - Need to change how we initialize, stop
  - Also need to change the body of the loop
Changing the Invariant

```python
def dm(b, h, k):
    """Returns: partition points""
    t = h; i = k;
    # b[h..i) < b[i..j) = b[j..k] >
    while t < i:
        if b[t] > 0:
            swap(b, t, i)
            t = t+1
        elif b[t] == 0:
            i = i+1;
        else:
            swap(b, t, j)
            j = j+1
    # b[h..i) < b[i..j) = b[j..k] >
    return (i, j)
```

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

---

Dutch National Flag Algorithm

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

---

Flag of Mauritius

- Now we have four colors!
  - Negatives: ‘red’ = odd, ‘purple’ = even
  - Positives: ‘yellow’ = odd, ‘green’ = even

- **pre:**
  - b[h..k] = ?

- **post:**
  - b[h..k] = < 0 odd, < 0 even, ≥ 0 odd, ≥ 0 even

- **inv:**
  - b[h..k] = < 0 odd, < 0 even, ≥ 0 odd, ≥ 0 even

**Need two swaps**

For two spaces

Have to check if second swap is okay

---

Flag of Mauritis

- **< 0, o**
  - h = r = s = t
  - i = ?
  - a = 0, e

- **12/1/19**