Recall: Horizontal Notation

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>k</th>
<th>len(b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>&lt;= sorted</td>
<td>&gt;=</td>
<td></td>
</tr>
</tbody>
</table>

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.

Recall: Horizontal Notation

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>h</th>
<th>k</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>&lt;= sorted</td>
<td>&gt;=</td>
<td></td>
</tr>
</tbody>
</table>

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

Partition Algorithm Implementation

```python
def partition(b, h, k):
    # Partition list b[h..k] around a pivot x = b[h]
    i = h; j = k; x = b[h]
    while i < j:
        if b[i+1] > x:
            i = i+1
        else:
            j = j-1
        if i < j:
            swap(b,i,i+1)
            if b[i+1] > x:
                i = i+1
            else:
                j = j-1
    return i
```

Dutch National Flag Algorithm

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

Dutch National Flag Variant

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

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 dnf(h, k):
    """Returns partition points""
    r = h
    s = k
    while t < h:
        if b[t] < 0:
            swap(b, t, i)
            i = i + 1; t = t + 1
        elif b[t] == 0:
            t = t + 1
        else:
            swap(b, t, j)
            j = j + 1
    return (i, j)
```

Dutch National Flag Algorithm

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

Flag of Mauritius

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

Flag of Mauritius

<table>
<thead>
<tr>
<th>&lt;0, o</th>
<th>&lt;0, e</th>
<th>≥0, o</th>
<th>≥0, e</th>
</tr>
</thead>
<tbody>
<tr>
<td>h</td>
<td>t</td>
<td>i</td>
<td>k</td>
</tr>
<tr>
<td>-1</td>
<td>-3</td>
<td>-7</td>
<td>-4</td>
</tr>
<tr>
<td>-1</td>
<td>-3</td>
<td>-7</td>
<td>-4</td>
</tr>
</tbody>
</table>

Flag of Mauritius

<table>
<thead>
<tr>
<th>&lt;0, o</th>
<th>&lt;0, e</th>
<th>≥0, o</th>
<th>≥0, e</th>
</tr>
</thead>
<tbody>
<tr>
<td>h</td>
<td>t</td>
<td>i</td>
<td>k</td>
</tr>
<tr>
<td>-1</td>
<td>-3</td>
<td>-7</td>
<td>-4</td>
</tr>
<tr>
<td>-1</td>
<td>-3</td>
<td>-7</td>
<td>-4</td>
</tr>
</tbody>
</table>

Have to check if second swap is okay

BUT NOT ALWAYS!

Need two swaps for two spaces

See algorithms.py for Python code

VS