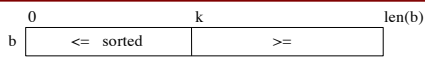
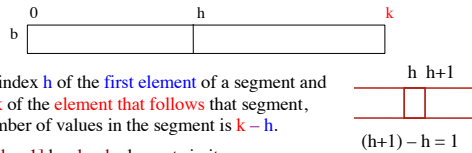


Recall: Horizontal Notation



Example of an assertion about an 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 $b[k..len(b)-1]$

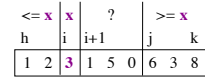


Given index h of the first element of a segment and index k of the element that follows that segment, the number of values in the segment is $k - h$.

$b[h .. k - 1]$ has $k - h$ elements in it.

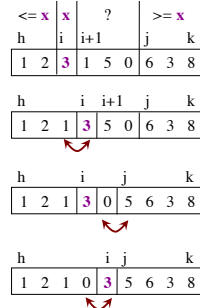
Partition Algorithm Implementation

```
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] >= x
    while i < j-1:
        if b[i+1] >= 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] >= x
    return i
```



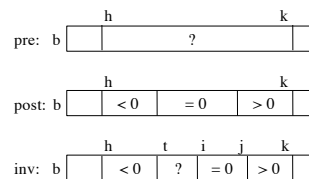
Partition Algorithm Implementation

```
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] >= x
    while i < j-1:
        if b[i+1] >= 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] >= x
    return i
```



Dutch National Flag Variant

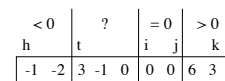
- Sequence of integer values
 - 'red' = negatives, 'white' = 0, 'blues' = positive
 - Only rearrange part of the list, not all



pre: $t = h,$
 $i = k+1,$
 $j = k$
 post: $t = i$

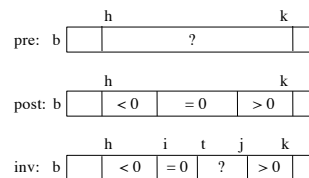
Dutch National Flag Algorithm

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



pre: $t = h,$
 $i = h,$
 $j = k$
 post: $t = j+1$

Changing the Invariant

```
def dnf(b, h, k):
    """Returns: partition points"""
    t = h; i = h; j = k;
    # b[h..t-1] < b[i..t-1] = b[t..j] ?, b[j+1..k] >
    while t < j+1:
        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
    # b[h..i-1] < b[i..j] = b[j+1..k] >
    return (i,j)
```

VS

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

Dutch National Flag Algorithm

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

< 0	?	= 0	> 0				
h	t	i	j	k			
-1	-2	3	-1	0	0	6	3

h	t	i	j	k			
-1	-2	3	-1	0	0	6	3

h	t	j	k			
-1	-2	0	0	3	6	3

Flag of Mauritius

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

pre: b [h | | | k]

post: b [<0 odd | <0 even | ≥0 odd | ≥0 even]

inv: b [<0, o | <0, e | ≥0, o | ? | ≥0, e]

Flag of Mauritius

< 0, o	< 0, e	≥ 0, o	?	≥ 0, e							
h	r	s	i	t	k						
-1	-3	-2	-4	7	5	-5	-6	1	0	2	4

h	r	s	i	t	k						
-1	-3	-5	-4	-2	5	7	-6	1	0	2	4

Need two swaps for two spaces

Flag of Mauritius

< 0, o	< 0, e	?	≥ 0, e							
h	r=s	i	t	k						
-1	-3	-7	-4	-2	-6	-5	1	0	2	4

h	r=s	i	t	k						
-1	-3	-7	-4	-2	-6	-5	1	0	2	4

BUT NOT ALWAYS!

Have to check if second swap is okay

Flag of Mauritius

< 0, o	< 0, e	≥ 0, o	?	≥ 0, e							
h	r	s	i	t	k						
-1	-3	-2	-4	7	5	-5	-6	1	0	2	4

h	r	s	i	t	k						
-1	-3	-5	-4	-2	5	7	-6	1	0	2	4

h	r	s	i	t	k						
-1	-3	-5	-4	-2	-6	7	5	1	0	2	4

h	r	s	i	t	k						
-1	-3	-5	-4	-2	-6	7	5	1	0	2	4

See algorithms.py for Python code