Announcements

• Final Exam conflicts due tonight at 11:59pm
• Final Exam review sessions on the 14th
• Labs on 5/9 and 5/10 will be office hours
• Assignment 5
  * Due 11:59pm on ***Wednesday*** May 10th
• Lab 13 is out

Dutch National Flag Variant

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

```
pre: h k
post: h < 0 i = 0 > 0
inv: h < 0 ? = 0 > 0
```

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[i-1] < 0:
            swap(b,i-1,t)
            t = t+1
        elif b[i-1] == 0:
            i = i - 1
        else:
            swap(b,i-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)
```

Flag of Mauritius

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

```
pre: h k
post: h < 0 o < 0 e
inv: h < 0, o ? = 0, e > 0
```

Linear Search

```
def linear_search(b,c,h):
    """Returns: first occurrence of c in b[h..]"
    # Store in i the index of the first c in b[h..]
    i= h
    # invariant: c is not in b[0..i-1]
    while i < len(b) and b[i] != c:
        i = i+ 1
    # post: c is not in b[h..i-1]
    #          i >= len(b) or b[i] == c
    return i if i < len(b) else -1
```

Analyzing the Loop
1. Does the initialization make inv true?
2. Is post true when inv is true and condition is false?
3. Does the repetend make progress?
4. Does the repetend keep the invariant inv true?
Binary Search

- **Vague:** Look for \( v \) in sorted sequence segment \( b[h..k] \).
- **Better:**
  - **Precondition:** \( b[h..k-1] \) is sorted (in ascending order).
  - **Postcondition:** \( b[h..i] <= v \) and \( v < b[i+1..k-1] \)
- Below, the array is in non-descending order:

```
pre: b <= v
h                        i k
post: b >= v
h i j                      k
```

New statement of the invariant guarantees that we get leftmost position of \( v \) if found

```
if v is 3, set i to 0
if v is 4, set i to 5
if v is 5, set i to 7
if v is 9, set i to 10
```

**Called binary search because each iteration of the loop cuts the array segment still to be processed in half.**

Example \( b = [1, 2, 3, 3, 3, 4, 4, 6, 7, 7] \)

Sorting: Arranging in Ascending Order

```
0 1 2 3 4 5 6 7 8 9
```

**Insertion Sort:**

```
i = 0
while i < n:
    push_down(b, i)
    i = i + 1
```

```
def push_down(b, i):
    j = i
    while j > 0:
        if b[j-1] > b[j]:
            swap(b, j-1, j)
        j = j - 1
```

```
0 1 2 3 3 4 4 6 6 7
```

**QuickSort**

```
def quick_sort(b, h, k):
    """Sort the array fragment b[h..k]"""
    if b[h..k] has fewer than 2 elements:
        return
    j = partition(b, h, k)
    if b[j-1] <= b[j] <= b[j+1..k]:
        quick_sort(b, h, j-1)
        quick_sort(b, j+1..k)
    else:
        quick_sort(b, h, j-1)
        quick_sort(b, j+1..k)
```

```
pre: b <= x
h i j+1 k
post: <= x x >= x
```