

## Dutch Treat

In this handout, we study various solutions to the Dutch National Flag ball sorting program. Recall that the goal is to sort an array of colored balls to form, red, white and blue segments from left to right. Balls can be moved only by swaps.

Solutions are typically modifications of the Lomuto partitioning technique that maintain red (R), white (W), blue (B), and unexplored (U) segments. The unexplored segment can be anywhere with respect to the colored segments.

---

### RWBU Solution

```
1  Flag-Sort-RWBU(A)
2  LR ← 0 ▷ LR stands for “last red”
3  LW ← 0 ▷ LW stands for “last white”
4  scan ← 1
5  n ← length(A)
6  while scan ≤ n do ▷ ***Loop invariants tested here***
7      if isRed(A[scan]) then
8          swap(A, LW+1, scan)
9          swap(A, LR+1, LW+1)
10         LR = LR+1
11         LW = LW+1
12     else if isWhite(A[scan]) then
13         swap(A, LW+1, scan)
14         scan = scan + 1
```

*Loop Invariants:*

- (RWBU-LI1)  $A_i[1..LR_i]$  contains only red balls
- (RWBU-LI2)  $A_i[LR_i+1, LW_i]$  contains only white balls
- (RWBU-LI3)  $A_i[LW_i+1..scan_i-1]$  contains only blue balls
- (RWBU-LI4)  $elts(A_i[1..n]) = elts(A_{init}[1..n])$
- (RWBU-LI5)  $LR_i \leq LW_i < scan_i$

*Questions:*

1. Why isn't there an “unexplored” invariant?
2. Why is (RWBU-LI5) necessary?
3. Which of the following are correct replacements for lines 8–9?

```
8'      swap(A, LR+1, scan)
9'      swap(A, LW+1, scan)

8''     swap(A, LR+1, LW+1)
9''     swap(A, LW+1, scan)
```

4. How would a C programmer condense lines 8",9",10, and 11? Is this a good idea?
5. Can we replace (RWBU-LI1) – (RWBU-LI4) with the following?
  - (RWBU-LI1')  $A_i[1..LR_i]$  contains exactly the red balls in  $A_{init}[1..scan_i-1]$
  - (RWBU-LI2')  $A_i[LR_i+1, LW_i]$  contains exactly the white balls in  $A_{init}[1..scan_i-1]$
  - (RWBU-LI3')  $A_i[LW_i+1..scan_i-1]$  contains exactly the blue balls in  $A_{init}[1..scan_i-1]$
6. What is a monotonically decreasing termination metric for the **while** loop in Flag-Sort-RWBU?

## RWUB Solution

This is the solution presented in the PS4 solutions.

```

1  Flag-Sort-RWUB(A)
2  LR ← 0 ▷ LR stands for “last red”
3  scan ← 1
4  FB ← length(A)+1 ▷ FB stands for “first blue”
5  while scan < FB do ▷ ***Loop invariants tested here***
6    if isRed(A[scan]) then
7      swap(A, LR+1, scan)
8      LR = LR+1
9      scan = scan + 1
10   else if isWhite(A[scan]) then
11     scan = scan + 1
12   else ▷ isBlue(A[scan])
13     swap(A, scan, FB-1)
14     FB ← FB-1 ▷ scan does not move in this case!
```

*Loop Invariants:*

- (RWUB-LI1)  $A_i[1..LR_i]$  contains only red balls
- (RWUB-LI2)  $A_i[LR_i+1, scan_i-1]$  contains only white balls
- (RWUB-LI3)  $A_i[FB_i..n]$  contains only blue balls
- (RWUB-LI4)  $elts(A_i[1..n]) = elts(A_{init}[1..n])$
- (RWUB-LI5)  $LR_i < scan_i \leq FB_i$

*Questions:*

1. Why is (RWUB-LI5) necessary?
2. How is non-interference proved when  $scan_i = FB_i$ ?
3. What is wrong with the following invariant?

(RWBU-LI1')  $A_i[1..LR_i]$  contains exactly the red balls in  $A_{init}[1..scan_i-1]$

4. How can the above invariant be fixed?
5. What advantage(s) does Flag-Sort-RWUB have over Flag-Sort-RWBU?
6. What is a monotonically decreasing termination metric for the **while** loop in Flag-Sort-RWUB?

---

## RUWB Solution

```
1  Flag-Sort-RUWB(A)
2  scan ← 1
3  FW ← length(A)+1 ▷ FW stands for “first white”
4  FB ← length(A)+1 ▷ FB stands for “first blue”
5  while scan < FW do ▷ ***Loop invariants tested here***
6      if isRed(A[scan]) then
7          scan = scan + 1
8      else if isWhite(A[scan]) then
9          swap(A, scan, FW-1)
10         FW = FW-1 ▷ scan does not move in this case!
11     else ▷ isBlue(A[scan])
12         swap(A, scan, FB-1)
13         FB = FB-1 ▷ scan does not move in this case!
14     if FB < FW then FW ← FB
```

*Loop Invariants:*

- (RUWB-LI1)  $A_i[1..scan_i-1]$  contains only red balls
- (RUWB-LI2)  $A_i[FW_i, FB_i-1]$  contains only white balls
- (RUWB-LI3)  $A_i[FB_i..length(A_{init})]$  contains only blue balls
- (RUWB-LI4)  $elts(A_i[1..n]) = elts(A_{init}[1..n])$
- (RUWB-LI5)  $scan_i \leq FW_i \leq FB_i$

*Questions:*

1. Why is (RUWB-LI5) necessary?
2. Why is line 14 necessary?
3. Can line 14 be replaced by the following?

```
14         if FB < FW then FW ← FW-1
```

4. What is a monotonically decreasing termination metric for the **while** loop in **Flag-Sort-RUWB**?
5. The termination metric is tricky because the range  $[scan..FW-1]$  does not decrease on every loop iteration. How can the loop be rewritten so that this range decreases on every iteration?
6. How would a C programmer rewrite the loop so that it decreases on every iteration? Is this a good idea?