School of Technology

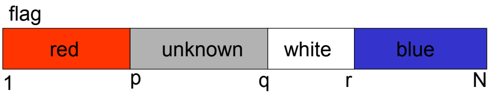
The Dutch National Flag by Edsger Dijkstra

David Lightfoot

The task

- Given a row of beads, each of which is either red, white or blue, re-arrange the beads so that they appear in the order of the Dutch National Flag: red, white, blue.
- Do this *in situ* by exchanging pairs of beads.
- Do this in as few operations as possible.

Dijkstra's approach: invariant

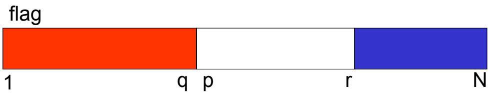


type Bead = (red, white, blue);
var flag: **array** [1 .. N] **of** Bead;

introduce three variables: *p*, *q* and *r*

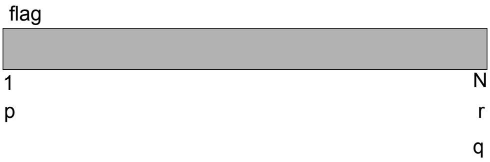
1 <= p <= q <= r <= N &
flag[1 .. p – 1] = red &
flag[q + 1 .. r] = white &
flag[r + 1 .. N] = blue

Final state



- This is the final state: q + 1 = p

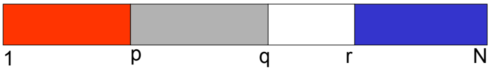
Initial state



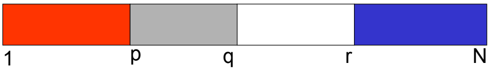
Initialisation:

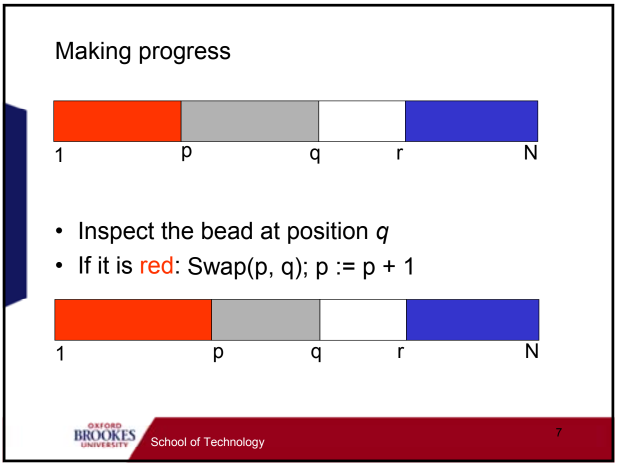
p := 1; q := N; r := N

Making progress



- Inspect the bead at position *q*
- If it is white: q := q - 1



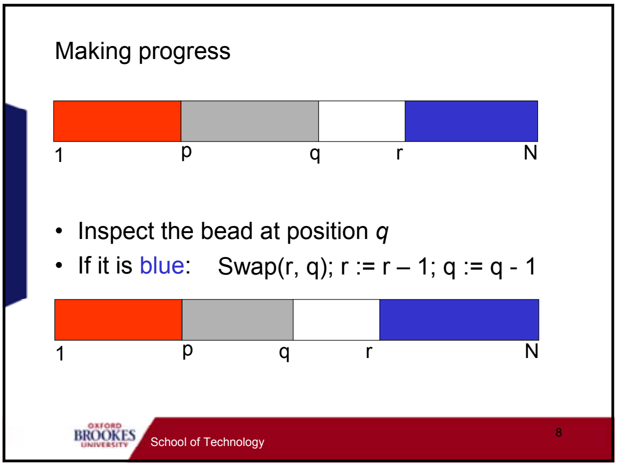


Making progress

1
p
q
r
N

- Inspect the bead at position *q*
- If it is red: Swap(p, q); p := p + 1

1
p
q
r
N

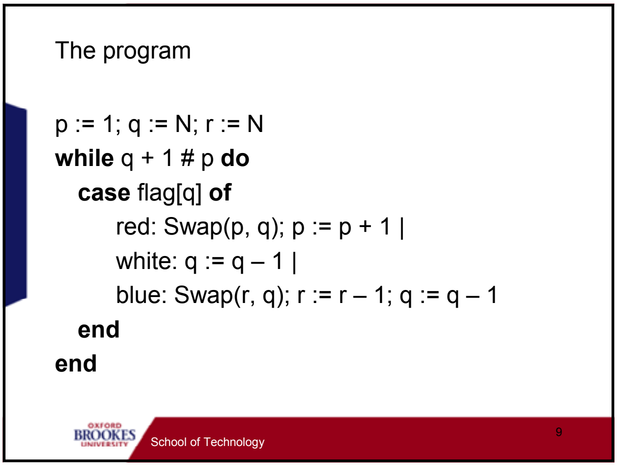


Making progress

1
p
q
r
N

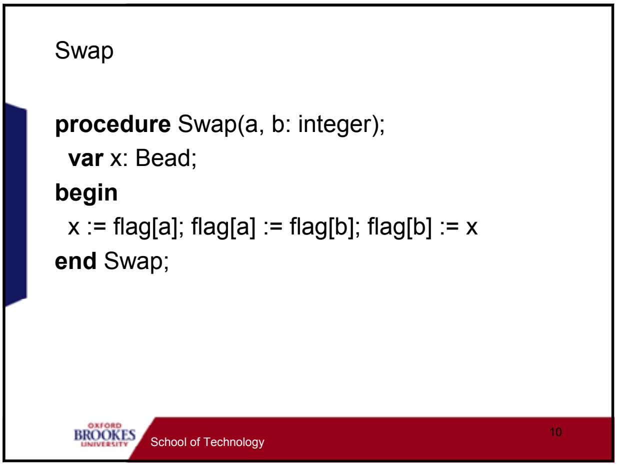
- Inspect the bead at position *q*
- If it is blue: Swap(r, q); r := r – 1; q := q - 1

1
p
q
r
N



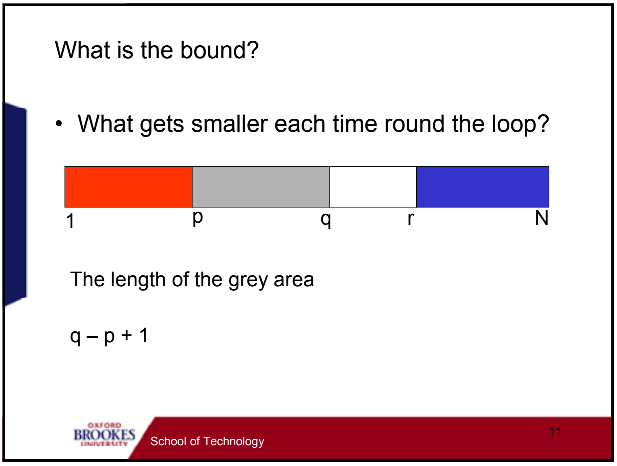
The program

```
p := 1; q := N; r := N
while q + 1 # p do
   case flag[q] of
         red: Swap(p, q); p := p + 1 |
         white: q := q – 1 |
         blue: Swap(r, q); r := r – 1; q := q – 1
   end
end
```



Swap

```
procedure Swap(a, b: integer);
  var x: Bead;
begin
  x := flag[a]; flag[a] := flag[b]; flag[b] := x
end Swap;
```

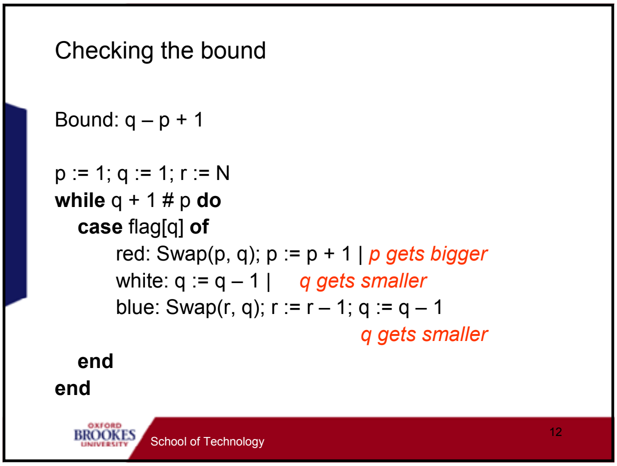


What is the bound?

- What gets smaller each time round the loop?

The length of the grey area

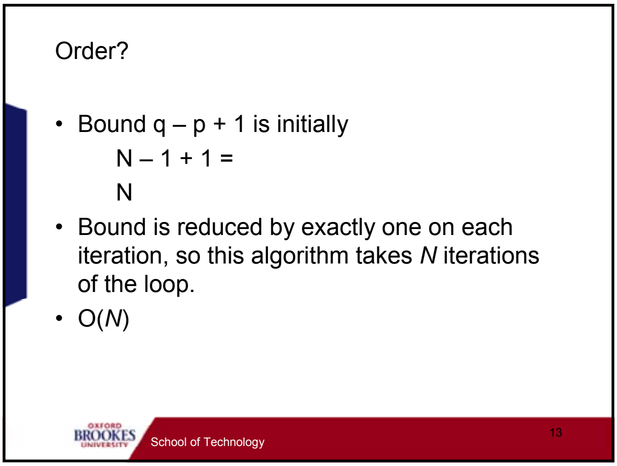
q – p + 1



Checking the bound

Bound: q – p + 1

```
p := 1; q := 1; r := N
while q + 1 # p do
   case flag[q] of
         red: Swap(p, q); p := p + 1 | p gets bigger
         white: q := q – 1 |      q gets smaller
         blue: Swap(r, q); r := r – 1; q := q – 1
                                       q gets smaller
   end
end
```

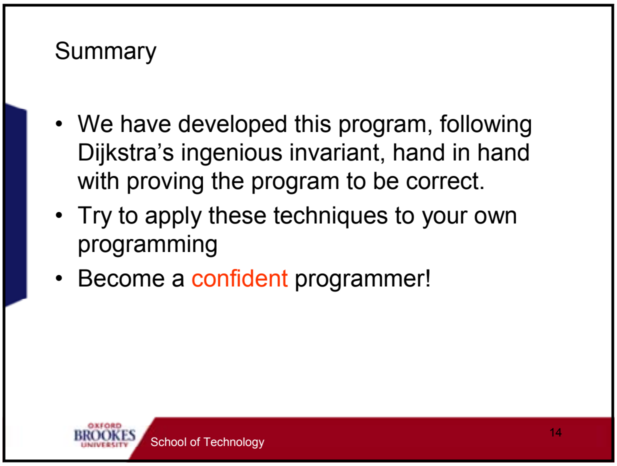


Order?

- Bound q – p + 1 is initially

  N – 1 + 1 =

  N
- Bound is reduced by exactly one on each iteration, so this algorithm takes *N* iterations of the loop.
- O(*N*)



Summary

- We have developed this program, following Dijkstra’s ingenious invariant, hand in hand with proving the program to be correct.
- Try to apply these techniques to your own programming
- Become a confident programmer!