university of groningen

Program Correctness

Block 5

Jorge A. Pérez
(based on slides by Arnold Meijster)

Bernoulli Institute for Mathematics, Computer Science, and AI
University of Groningen, Groningen, the Netherlands

March 9, 2023

Outline

Linear Search

We consider the following specification for computing the least natural number that satisfies some unspecified property $prop$:

$$
\begin{array}{l}
\textbf{var } k :\ \mathbb{N}; \\
\quad \{P :\ M = \text{Min}\,\{i \in \mathbb{N} \mid prop(i)\} < \infty\} \\
L \\
\quad \{Q :\ k = M\}
\end{array}
$$

- A bi-regular spec: the precondition is constant (it is independent from variable k) and the postcondition is of the form $x = X$.
- From now on, we use pre-regular preconditions anywhere in the annotation, without carrying it through every step.

Linear Search: Invariant

Notice:

$$P \equiv M = \text{Min}\,\{i \in \mathbb{N} \mid prop(i)\} < \infty$$

$$\{M < \infty \text{ (i.e. such an } M \text{ exists)}\}$$

$$\equiv M \in \mathbb{N} \ \wedge \ prop(M) \ \wedge \ \forall i \in \mathbb{N}(prop(i) \Rightarrow M \leq i)$$

0 We will iterate on k to inspect $prop(k)$, so we need a **while.**

1 Choose an invariant J and a guard B such that $J \wedge \neg B \Rightarrow Q$.

$$\boxed{\begin{array}{c} J : 0 \leq k \leq M \\ B : \neg prop(k) \end{array}}$$

Linear Search: Invariant

Notice:

$$P \equiv M = \text{Min}\,\{i \in \mathbb{N} \mid prop(i)\} < \infty$$
$$\{M < \infty \text{ (i.e. such an } M \text{ exists)}\}$$
$$\equiv M \in \mathbb{N} \wedge prop(M) \wedge \forall i \in \mathbb{N}(prop(i) \Rightarrow M \leq i)$$

0 We will iterate on k to inspect $prop(k)$, so we need a **while**.

1 Choose an invariant J and a guard B such that $J \wedge \neg B \Rightarrow Q$.

$$\boxed{\begin{array}{c} J : 0 \leq k \leq M \\ B : \neg prop(k) \end{array}}$$

$$\begin{aligned} J \wedge \neg B \equiv\ & 0 \leq k \leq M \wedge prop(k) \\ & \{prop(k) \wedge P\} \\ \Rightarrow\ & 0 \leq k \leq M \wedge M \leq k \\ \Rightarrow\ & Q :\ k = M \end{aligned}$$

Linear Search: Initialization & Variant

$$P : M = \text{Min}\,\{i \in \mathbb{N} \mid prop(i)\} < \infty\}$$
$$J : 0 \leq k \leq M$$
$$B : \neg prop(k)$$
$$Q : k = M$$

2 Initialization: Find a command T_0 such that $\{\textbf{true}\}\ T_0\ \{J\}$

Note that we use the precondition **true**, since P is pre-regular.

$\{\textbf{true}\}$
(* *use conjunct* $M \in \mathbb{N}$ *of* P *)
$\{0 \leq 0 \leq M\}$
$k := 0;$
$\{J :\ 0 \leq k \leq M\}$

3 Variant function: Choose a $vf \in \mathbb{Z}$ and prove $J \wedge B \Rightarrow vf \geq 0$

We choose $vf = M - k$. Clearly, $J \wedge B \Rightarrow M - k \geq 0$

Linear Search: Body of the Loop

$\{J \wedge B \wedge vf = V\}$

$\{J \wedge vf < V\}$

Linear Search: Body of the Loop

$\{J \wedge B \wedge vf = V\}$
  (* *definitions* J, B *and* vf *)
$\{0 \leq k \leq M \wedge \neg prop(k) \wedge M - k = V\}$

$\{J \wedge vf < V\}$

Linear Search: Body of the Loop

$\{J \wedge B \wedge vf = V\}$
(* *definitions* J, B *and* vf *)
$\{0 \leq k \leq M \wedge \neg prop(k) \wedge M - k = V\}$
(* $P \Rightarrow prop(M)$; $0 \leq k \leq M \wedge prop(M) \wedge \neg prop(k) \Rightarrow k \neq M$ *)
$\{0 \leq k < M \wedge M - k = V\}$

$\{J \wedge vf < V\}$

Linear Search: Body of the Loop

$\{J \wedge B \wedge vf = V\}$

(* *definitions* J, B *and* vf *)

$\{0 \leq k \leq M \wedge \neg prop(k) \wedge M - k = V\}$

(* $P \Rightarrow prop(M); 0 \leq k \leq M \wedge prop(M) \wedge \neg prop(k) \Rightarrow k \neq M$ *)

$\{0 \leq k < M \wedge M - k = V\}$

(* *prepare* $k := k + 1$ *)

$\{0 \leq k + 1 \leq M \wedge M - (k + 1) < V\}$

$\{J \wedge vf < V\}$

Linear Search: Body of the Loop

$\{J \wedge B \wedge vf = V\}$

(* definitions J, B and vf *)

$\{0 \leq k \leq M \wedge \neg prop(k) \wedge M - k = V\}$

(* $P \Rightarrow prop(M)$; $0 \leq k \leq M \wedge prop(M) \wedge \neg prop(k) \Rightarrow k \neq M$ *)

$\{0 \leq k < M \wedge M - k = V\}$

(* prepare $k := k + 1$ *)

$\{0 \leq k + 1 \leq M \wedge M - (k + 1) < V\}$

$k := k + 1;$

$\{J \wedge vf < V\}$

Linear Search: Body of the Loop

$\{J \wedge B \wedge vf = V\}$

(* definitions J, B and vf *)

$\{0 \leq k \leq M \wedge \neg prop(k) \wedge M - k = V\}$

(* $P \Rightarrow prop(M)$; $0 \leq k \leq M \wedge prop(M) \wedge \neg prop(k) \Rightarrow k \neq M$ *)

$\{0 \leq k < M \wedge M - k = V\}$

(* prepare $k := k + 1$ *)

$\{0 \leq k + 1 \leq M \wedge M - (k + 1) < V\}$

$k := k + 1;$

$\{0 \leq k \leq M \wedge M - k < V\}$

$\{J \wedge vf < V\}$

Linear Search: Body of the Loop

$\{J \wedge B \wedge vf = V\}$
(* definitions J, B and vf *)
$\{0 \leq k \leq M \wedge \neg prop(k) \wedge M - k = V\}$
(* $P \Rightarrow prop(M)$; $0 \leq k \leq M \wedge prop(M) \wedge \neg prop(k) \Rightarrow k \neq M$ *)
$\{0 \leq k < M \wedge M - k = V\}$
(* prepare $k := k + 1$ *)
$\{0 \leq k + 1 \leq M \wedge M - (k + 1) < V\}$
$k := k + 1;$
$\{0 \leq k \leq M \wedge M - k < V\}$
(* definitions J, and vf *)
$\{J \wedge vf < V\}$

Linear Search: Conclusion

We derived the program fragment (*linear search algorithm*):

$$
\begin{array}{l}
\textbf{var}\ \ k:\ \mathbb{Z};\\
\quad \{P:\ \ M = \mathrm{Min}\ \{i \in \mathbb{N} \mid prop(i)\} < \infty\}\\
k := 0;\\
\quad \{J : 0 \le k \le M\}\\
\qquad (^*\ vf = M - k\ ^*)\\
\textbf{while}\ \neg prop(k)\ \textbf{do}\\
\quad k := k + 1;\\
\textbf{end};\\
\quad \{Q:\ \ k = M\}
\end{array}
$$

Application: Linear Search in an Array

Given an array a of length n and a value w, compute the smallest i such that $a[i] = w$.
If such an index does not exist, the result should be n.

Application: Linear Search in an Array

Given an array a of length n and a value w, compute the smallest i such that $a[i] = w$.
If such an index does not exist, the result should be n.

const $n : \mathbb{N}$, $w : \mathbb{Z}$, $a :$ **array** $[0..n)$ **of** $\mathbb{R}$;
var $k : \mathbb{N}$;
$\{P : M = \text{Min}\,\{i \in \mathbb{N} \mid a[i] = w \lor n \leq i\}\}$
S
$\{Q : k = M\}$

Note that $M \leq n$, so $M < \infty$.

(This is Specification (8.2) in the reader.)

Application: Linear Search in an Array

We instantiate $prop(i) \equiv (n \leq i \vee a[i] = w)$ in the linear search algorithm we derived. Note that $\neg prop(i) \equiv (i < n \wedge a[i] \neq w)$:

var $k : \mathbb{Z}$;
  $\{P : M = \text{Min}\,\{i \in \mathbb{N} \mid n \leq i \vee a[i] = w\}\}$
$k := 0$;
  $\{J : 0 \leq k \leq M\}$
    (* $vf = M - k$ *)
while $k < n \wedge a[k] \neq w$ **do** (* *short circuit evaluation* *)
  $k := k + 1$;
end;
  $\{Q : k = M\}$

Outline

Up to Here...

We have seen:

- The roadmap for designing repetitions
- Linear search (Ch 8.1)

Coming next:

- Binary search (Ch 8.2)
- The Dutch National Flag problem (Ch 8.4)
- Chapters 10 and 9

Up to Here...

We have seen:

- The roadmap for designing repetitions
- Linear search (Ch 8.1)

Coming next:

- Binary search (Ch 8.2)
- The Dutch National Flag problem (Ch 8.4)
- Chapters 10 and 9

We shall use additional keywords to express conditions within annotated linear proofs:

- **suppose**, **define**, **assume**, **introduce**, ...

Useful when deriving side conditions for recurrence relations.

Monotonic functions

It is convenient to distinguish ordered arrays.

Let V be an interval of $\mathbb{Z}$, and let $f : V \to \mathbb{R}$ be a function (a sequence of numbers).

We say f is

- ascending ($\leq$ / $\leq$): if $\forall i, j \in V : (i \leq j \Rightarrow f(i) \leq f(j))$

Monotonic functions

It is convenient to distinguish ordered arrays.

Let V be an interval of $\mathbb{Z}$, and let $f : V \to \mathbb{R}$ be a function (a sequence of numbers).

We say f is

- ascending ($\leq$ / $\leq$): if $\forall i, j \in V : (i \leq j \Rightarrow f(i) \leq f(j))$
- increasing ($<$ / $<$): if $\forall i, j \in V : (i < j \Rightarrow f(i) < f(j))$

Monotonic functions

It is convenient to distinguish ordered arrays.

Let V be an interval of $\mathbb{Z}$, and let $f : V \to \mathbb{R}$ be a function (a sequence of numbers).

We say f is

- ascending ($\leq$ / $\leq$): if $\forall i, j \in V : (i \leq j \Rightarrow f(i) \leq f(j))$
- increasing ($<$ / $<$): if $\forall i, j \in V : (i < j \Rightarrow f(i) < f(j))$
- descending ($\leq$ / $\geq$): if $\forall i, j \in V : (i \leq j \Rightarrow f(i) \geq f(j))$
- decreasing ($<$ / $>$): if $\forall i, j \in V : (i < j \Rightarrow f(i) > f(j))$

Monotonic functions

It is convenient to distinguish ordered arrays.

Let V be an interval of $\mathbb{Z}$, and let $f : V \to \mathbb{R}$ be a function (a sequence of numbers).

We say f is

- ascending ($\leq$ / $\leq$): if $\forall i, j \in V : (i \leq j \Rightarrow f(i) \leq f(j))$
- increasing ($<$ / $<$): if $\forall i, j \in V : (i < j \Rightarrow f(i) < f(j))$
- descending ($\leq$ / $\geq$): if $\forall i, j \in V : (i \leq j \Rightarrow f(i) \geq f(j))$
- decreasing ($<$ / $>$): if $\forall i, j \in V : (i < j \Rightarrow f(i) > f(j))$

f is called monotonic if it has one of the above properties.

Binary Search

Consider now an ascending array a, with lenght n.

Given a value w, compute the smallest index k such that $a[k] = w$.
If such an index does not exist, the result should be $k = n$.

Q: Why is it relevant that a is ascending?

Binary Search

Consider now an ascending array a, with lenght n.

Given a value w, compute the smallest index k such that $a[k] = w$.
If such an index does not exist, the result should be $k = n$.

Q: Why is it relevant that a is ascending?
A: If a is ascending and w occurs in a then
the smallest k s.t. $a[k] = w$ is also the smallest k s.t. $w \leq a[k]$.
(Relevant if w doesn't occur in a.)

Binary Search

Consider now an ascending array a, with lenght n.

Given a value w, compute the smallest index k such that $a[k] = w$.
If such an index does not exist, the result should be $k = n$.

Q: Why is it relevant that a is ascending?
A: If a is ascending and w occurs in a then
the smallest k s.t. $a[k] = w$ is also the smallest k s.t. $w \leq a[k]$.
(Relevant if w doesn't occur in a.)

We use the following specification:

const $n : \mathbb{N}$, $w : \mathbb{Z}$, $a :$ **array** $[0..n)$ **of** $\mathbb{R}$;
var $k : \mathbb{N}$;
  $\{P :\ a$ is ascending$\}$
S
  $\{Q :\ k = \text{Min}\,\{i \in \mathbb{N} \mid n \leq i \vee w \leq a[i]\}\}$

Binary Search

Consider now an ascending array a, with lenght n.

Given a value w, compute the smallest index k such that $a[k] = w$.
If such an index does not exist, the result should be $k = n$.

Q: Why is it relevant that a is ascending?
A: If a is ascending and w occurs in a then
the smallest k s.t. $a[k] = w$ is also the smallest k s.t. $w \leq a[k]$.
(Relevant if w doesn't occur in a.)

We use the following specification:

const $n : \mathbb{N}$, $w : \mathbb{Z}$, $a :$ **array** $[0..n)$ **of** $\mathbb{R}$;
var $k : \mathbb{N}$;
$\{P :\ a \text{ is ascending}\}$
S
$\{Q :\ k = \text{Min}\,\{i \in \mathbb{N} \mid n \leq i \vee w \leq a[i]\}\}$

To enforce the informal specification, we need an active finalization:
if $k < n \wedge a[k] \neq w$ **then** $k := n$ **end**;

Massaging the Postcondition

$Q : k = \text{Min}\,\{i \in \mathbb{N} \mid n \leq i \vee w \leq a[i]\}$

Massaging the Postcondition

$Q : k = \text{Min}\,\{i \in \mathbb{N} \mid n \leq i \vee w \leq a[i]\}$

$\equiv$ {properties of Min }

$0 \leq k \;\wedge\; (n \leq k \vee w \leq a[k]) \;\wedge\; \forall i \in \mathbb{N}((n \leq i \vee w \leq a[i]) \Rightarrow k \leq i)$

Massaging the Postcondition

$Q : k = \text{Min}\, \{i \in \mathbb{N} \mid n \le i \vee w \le a[i]\}$

$\equiv$ {properties of Min }

$0 \le k \ \wedge \ (n \le k \vee w \le a[k]) \ \wedge \ \forall i \in \mathbb{N}((n \le i \vee w \le a[i]) \Rightarrow k \le i)$

$\equiv$ {contraposition: $p \Rightarrow q \equiv \neg q \Rightarrow \neg p$}

$0 \le k \ \wedge \ (n \le k \vee w \le a[k]) \ \wedge \ \forall i \in \mathbb{N}(i < k \Rightarrow (i < n \wedge a[i] < w))$

Massaging the Postcondition

$Q : k = \text{Min}\,\{i \in \mathbb{N} \mid n \leq i \vee w \leq a[i]\}$

$\equiv$ {properties of Min }

$0 \leq k \;\wedge\; (n \leq k \vee w \leq a[k]) \;\wedge\; \forall i \in \mathbb{N}((n \leq i \vee w \leq a[i]) \Rightarrow k \leq i)$

$\equiv$ {contraposition: $p \Rightarrow q \equiv \neg q \Rightarrow \neg p$}

$0 \leq k \;\wedge\; (n \leq k \vee w \leq a[k]) \;\wedge\; \forall i \in \mathbb{N}(i < k \Rightarrow (i < n \wedge a[i] < w))$

$\equiv$ {$\forall i \in \mathbb{N}(i < k \Rightarrow i < n) \equiv k \leq n$, arithmetic}

$0 \leq k \leq n \;\wedge\; (n \leq k \vee w \leq a[k]) \;\wedge\; \forall i \in \mathbb{N}(i < k \Rightarrow a[i] < w)$

Massaging the Postcondition

$Q : k = \text{Min}\,\{i \in \mathbb{N} \mid n \le i \lor w \le a[i]\}$

$\equiv$ {properties of Min }

$0 \le k \ \land\ (n \le k \lor w \le a[k]) \ \land\ \forall i \in \mathbb{N}((n \le i \lor w \le a[i]) \Rightarrow k \le i)$

$\equiv$ {contraposition: $p \Rightarrow q \equiv \neg q \Rightarrow \neg p$}

$0 \le k \ \land\ (n \le k \lor w \le a[k]) \ \land\ \forall i \in \mathbb{N}(i < k \Rightarrow (i < n \land a[i] < w))$

$\equiv$ {$\forall i \in \mathbb{N}(i < k \Rightarrow i < n) \equiv k \le n$, arithmetic}

$0 \le k \le n \ \land\ (n \le k \lor w \le a[k]) \ \land\ \forall i \in \mathbb{N}(i < k \Rightarrow a[i] < w)$

$\equiv$ {calculus; logic}

$0 \le k \le n \ \land\ (k = n \lor w \le a[k]) \ \land\ \forall i \in \mathbb{N}(i < k \Rightarrow a[i] < w)$

Massaging the Postcondition

$Q: k = \text{Min}\,\{i \in \mathbb{N} \mid n \le i \vee w \le a[i]\}$

$\equiv$ {properties of Min }

$0 \le k \;\wedge\; (n \le k \vee w \le a[k]) \;\wedge\; \forall i \in \mathbb{N}((n \le i \vee w \le a[i]) \Rightarrow k \le i)$

$\equiv$ {contraposition: $p \Rightarrow q \equiv \neg q \Rightarrow \neg p$}

$0 \le k \;\wedge\; (n \le k \vee w \le a[k]) \;\wedge\; \forall i \in \mathbb{N}(i < k \Rightarrow (i < n \wedge a[i] < w))$

$\equiv$ {$\forall i \in \mathbb{N}(i < k \Rightarrow i < n) \equiv k \le n$, arithmetic}

$0 \le k \le n \;\wedge\; (n \le k \vee w \le a[k]) \;\wedge\; \forall i \in \mathbb{N}(i < k \Rightarrow a[i] < w)$

$\equiv$ {calculus; logic}

$0 \le k \le n \;\wedge\; (k = n \vee w \le a[k]) \;\wedge\; \forall i \in \mathbb{N}(i < k \Rightarrow a[i] < w)$

$\equiv$ {logic; array a is ascending}

$0 \le k \le n \;\wedge\; (k = n \vee w \le a[k]) \;\wedge\; (k = 0 \vee a[k-1] < w)$

Massaging the Postcondition

$Q : k = \text{Min}\,\{i \in \mathbb{N} \mid n \leq i \vee w \leq a[i]\}$

$\equiv$ {properties of Min }

$0 \leq k \;\wedge\; (n \leq k \vee w \leq a[k]) \;\wedge\; \forall i \in \mathbb{N}((n \leq i \vee w \leq a[i]) \Rightarrow k \leq i)$

$\equiv$ {contraposition: $p \Rightarrow q \equiv \neg q \Rightarrow \neg p$}

$0 \leq k \;\wedge\; (n \leq k \vee w \leq a[k]) \;\wedge\; \forall i \in \mathbb{N}(i < k \Rightarrow (i < n \wedge a[i] < w))$

$\equiv$ {$\forall i \in \mathbb{N}(i < k \Rightarrow i < n) \equiv k \leq n$, arithmetic}

$0 \leq k \leq n \;\wedge\; (n \leq k \vee w \leq a[k]) \;\wedge\; \forall i \in \mathbb{N}(i < k \Rightarrow a[i] < w)$

$\equiv$ {calculus; logic}

$0 \leq k \leq n \;\wedge\; (k = n \vee w \leq a[k]) \;\wedge\; \forall i \in \mathbb{N}(i < k \Rightarrow a[i] < w)$

$\equiv$ {logic; array a is ascending}

$0 \leq k \leq n \;\wedge\; (k = n \vee w \leq a[k]) \;\wedge\; (k = 0 \vee a[k-1] < w)$

$\equiv$ {**define:** $a[-1] = -\infty$ and $a[n] = \infty$}

Massaging the Postcondition

$Q : k = \text{Min}\,\{i \in \mathbb{N} \mid n \leq i \vee w \leq a[i]\}$

$\equiv$ {properties of Min }

$0 \leq k \;\wedge\; (n \leq k \vee w \leq a[k]) \;\wedge\; \forall i \in \mathbb{N}((n \leq i \vee w \leq a[i]) \Rightarrow k \leq i)$

$\equiv$ {contraposition: $p \Rightarrow q \equiv \neg q \Rightarrow \neg p$}

$0 \leq k \;\wedge\; (n \leq k \vee w \leq a[k]) \;\wedge\; \forall i \in \mathbb{N}(i < k \Rightarrow (i < n \wedge a[i] < w))$

$\equiv$ {$\forall i \in \mathbb{N}(i < k \Rightarrow i < n) \equiv k \leq n$, arithmetic}

$0 \leq k \leq n \;\wedge\; (n \leq k \vee w \leq a[k]) \;\wedge\; \forall i \in \mathbb{N}(i < k \Rightarrow a[i] < w)$

$\equiv$ {calculus; logic}

$0 \leq k \leq n \;\wedge\; (k = n \vee w \leq a[k]) \;\wedge\; \forall i \in \mathbb{N}(i < k \Rightarrow a[i] < w)$

$\equiv$ {logic; array a is ascending}

$0 \leq k \leq n \;\wedge\; (k = n \vee w \leq a[k]) \;\wedge\; (k = 0 \vee a[k-1] < w)$

$\equiv$ {**define:** $a[-1] = -\infty$ and $a[n] = \infty$}

$0 \leq k \leq n \;\wedge\; w \leq a[k] \;\wedge\; a[k-1] < w$

Massaging the Postcondition

$Q : k = \text{Min}\,\{i \in \mathbb{N} \mid n \leq i \vee w \leq a[i]\}$

$\equiv$ {properties of Min }

$0 \leq k \;\wedge\; (n \leq k \vee w \leq a[k]) \;\wedge\; \forall i \in \mathbb{N}((n \leq i \vee w \leq a[i]) \Rightarrow k \leq i)$

$\equiv$ {contraposition: $p \Rightarrow q \equiv \neg q \Rightarrow \neg p$}

$0 \leq k \;\wedge\; (n \leq k \vee w \leq a[k]) \;\wedge\; \forall i \in \mathbb{N}(i < k \Rightarrow (i < n \wedge a[i] < w))$

$\equiv$ {$\forall i \in \mathbb{N}(i < k \Rightarrow i < n) \equiv k \leq n$, arithmetic}

$0 \leq k \leq n \;\wedge\; (n \leq k \vee w \leq a[k]) \;\wedge\; \forall i \in \mathbb{N}(i < k \Rightarrow a[i] < w)$

$\equiv$ {calculus; logic}

$0 \leq k \leq n \;\wedge\; (k = n \vee w \leq a[k]) \;\wedge\; \forall i \in \mathbb{N}(i < k \Rightarrow a[i] < w)$

$\equiv$ {logic; array a is ascending}

$0 \leq k \leq n \;\wedge\; (k = n \vee w \leq a[k]) \;\wedge\; (k = 0 \vee a[k-1] < w)$

$\equiv$ {**define:** $a[-1] = -\infty$ and $a[n] = \infty$}

$0 \leq k \leq n \;\wedge\; w \leq a[k] \;\wedge\; a[k-1] < w$

$\equiv$ {rewriting}

$Q :\; 0 \leq k \leq n \;\wedge\; a[k-1] < w \leq a[k]$

Massaging the Postcondition

$Q : k = \text{Min}\,\{i \in \mathbb{N} \mid n \le i \lor w \le a[i]\}$

$\equiv$ {properties of Min }

$0 \le k \;\land\; (n \le k \lor w \le a[k]) \;\land\; \forall i \in \mathbb{N}((n \le i \lor w \le a[i]) \Rightarrow k \le i)$

$\equiv$ {contraposition: $p \Rightarrow q \equiv \neg q \Rightarrow \neg p$}

$0 \le k \;\land\; (n \le k \lor w \le a[k]) \;\land\; \forall i \in \mathbb{N}(i < k \Rightarrow (i < n \land a[i] < w))$

$\equiv$ {$\forall i \in \mathbb{N}(i < k \Rightarrow i < n) \equiv k \le n$, arithmetic}

$0 \le k \le n \;\land\; (n \le k \lor w \le a[k]) \;\land\; \forall i \in \mathbb{N}(i < k \Rightarrow a[i] < w)$

$\equiv$ {calculus; logic}

$0 \le k \le n \;\land\; (k = n \lor w \le a[k]) \;\land\; \forall i \in \mathbb{N}(i < k \Rightarrow a[i] < w)$

$\equiv$ {logic; array a is ascending}

$0 \le k \le n \;\land\; (k = n \lor w \le a[k]) \;\land\; (k = 0 \lor a[k-1] < w)$

$\equiv$ {**define:** $a[-1] = -\infty$ and $a[n] = \infty$}

$0 \le k \le n \;\land\; w \le a[k] \;\land\; a[k-1] < w$

$\equiv$ {rewriting}

$Q : \; 0 \le k \le n \;\land\; a[k-1] < w \le a[k]$

Note: The program will not inspect $a[-1]$ and $a[n]$. Indeed, $k = 0$ and $k = n$ are only needed to reason about boundary cases.

Binary Search: Invariant and Guard

Revising the postcondition, we obtain the following specification:

const $n :\ \mathbb{N}$, $w :\ \mathbb{Z}$, $a :$ **array** $[0..n)$ **of** $\mathbb{R}$;
var $k :\ \mathbb{N}$;
  $\{P :$ a is ascending; $a[-1] = -\infty \wedge a[n] = \infty\}$
B
  $\{Q :\ 0 \leq k \leq n \wedge a[k-1] < w \leq a[k]\}$

Binary Search: Invariant and Guard

Revising the postcondition, we obtain the following specification:

const $n : \mathbb{N}$, $w : \mathbb{Z}$, $a :$ **array** $[0..n)$ **of** $\mathbb{R}$;
var $k : \mathbb{N}$;
$\{P :\ a \text{ is ascending; } a[-1] = -\infty \wedge a[n] = \infty\}$
B
$\{Q :\ 0 \leq k \leq n \wedge a[k-1] < w \leq a[k]\}$

0 We decide that we need a **while**-program:
We will inspect the array a iteratively for several indices.

1 Choose an invariant J and a guard B such that $J \wedge \neg B \Rightarrow Q$.
We use the heuristic split variable, with the new variable j:

$$J : 0 \leq j \leq k \leq n \ \wedge \ a[j-1] < w \leq a[k]$$
$$B : j \neq k$$

Clearly, $J \wedge \neg B \Rightarrow Q$.

Binary Search: Initialization & Variant

$$\boxed{\begin{array}{l} P: a \text{ is ascending; } a[-1] = -\infty \wedge a[n] = \infty \\ J: 0 \leq j \leq k \leq n \ \wedge \ a[j-1] < w \leq a[k] \\ B: j \neq k \end{array}}$$

2 Initialization: Because P is pre-regular, we can use **true** as precondition. We find a command T_0 such that $\{\textbf{true}\}\ T_0\ \{J\}$.

$$\begin{array}{l} \quad\{\textbf{true}\} \\ \qquad (^*\ n \in \mathbb{N}; \textit{calculus; use } P\ ^*) \\ \quad\{0 \leq 0 \leq n \leq n \wedge a[0-1] < w \leq a[n]\} \\ j := 0;\ k := n; \\ \quad\{J:\ 0 \leq j \leq k \leq n \wedge a[j-1] < w \leq a[k]\} \end{array}$$

3 Variant function: Choose a $vf \in \mathbb{Z}$ and prove $J \wedge B \Rightarrow vf \geq 0$. We choose $vf = k - j \in \mathbb{Z}$. Clearly, $J \wedge B \Rightarrow vf \geq 0$.

Binary Search: Body of the Loop

We will be working towards a body of the following form:

$\{J \wedge B \wedge vf = V\}$
S_0;
$\{J \wedge vf = V \wedge j \leq m < k\}$
if $a[m] < w$ **then**
$j := m + 1$;
else
$k := m$;
end
$\{J \wedge vf < V\}$

- Clearly, S_0 should involve an assignment to m, which is a point in the interval formed by j and k.
- Both '$m := j$' or '$m := k - 1$' are alternatives, but we would like to reduce by half the search area.
- Hence, we shall consider '$m := (j + k)$ **div** 2'.

Binary Search: Body of the Loop

$\{J \wedge B \wedge vf = V\}$

$\{J \wedge vf < V\}$

Binary Search: Body of the Loop

$\{J \wedge B \wedge vf = V\}$
$\{0 \leq j \leq k \leq n \ \wedge \ a[j-1] < w \leq a[k] \ \wedge \ j \neq k \ \wedge \ k - j = V\}$

if $a[m] < w$ **then**

$j := m + 1;$

else

$k := m;$

end
$\{J \wedge vf < V\}$

Binary Search: Body of the Loop

$\{J \land B \land vf = V\}$
$\{0 \leq j \leq k \leq n \ \land \ a[j-1] < w \leq a[k] \ \land \ j \neq k \ \land \ k - j = V\}$
$(^* \ (j \leq k \land j < k) \equiv (j + j \leq j + k \land j + k < k + k) \equiv 2 \cdot j \leq j + k < 2 \cdot k \ ^*)$
$\{0 \leq j \leq (j + k)$ **div** $2 < k \leq n \ \land \ a[j-1] < w \leq a[k] \ \land \ k - j = V\}$

if $a[m] < w$ **then**

$j := m + 1;$

else

$k := m;$

end
$\{J \land vf < V\}$

Binary Search: Body of the Loop

$\{J \wedge B \wedge vf = V\}$
$\{0 \leq j \leq k \leq n \;\wedge\; a[j-1] < w \leq a[k] \;\wedge\; j \neq k \;\wedge\; k - j = V\}$
$(^*\ (j \leq k \wedge j < k) \equiv (j + j \leq j + k \wedge j + k < k + k) \equiv 2 \cdot j \leq j + k < 2 \cdot k\ ^*)$
$\{0 \leq j \leq (j + k)$ **div** $2 < k \leq n \;\wedge\; a[j-1] < w \leq a[k] \;\wedge\; k - j = V\}$
$m := (j + k)$ **div** $2;$
$\{0 \leq j \leq m < k \leq n \;\wedge\; a[j-1] < w \leq a[k] \;\wedge\; k - j = V\}$
if $a[m] < w$ **then**

$j := m + 1;$

else

$k := m;$

end
$\{J \wedge vf < V\}$

Binary Search: Body of the Loop

$\{J \wedge B \wedge vf = V\}$
$\{0 \leq j \leq k \leq n \ \wedge \ a[j-1] < w \leq a[k] \ \wedge \ j \neq k \ \wedge \ k - j = V\}$
$(^* \ (j \leq k \wedge j < k) \equiv (j + j \leq j + k \wedge j + k < k + k) \equiv 2 \cdot j \leq j + k < 2 \cdot k \ ^*)$
$\{0 \leq j \leq (j + k)$ **div** $2 < k \leq n \ \wedge \ a[j-1] < w \leq a[k] \ \wedge \ k - j = V\}$

$m := (j + k)$ **div** $2;$

$\{0 \leq j \leq m < k \leq n \ \wedge \ a[j-1] < w \leq a[k] \ \wedge \ k - j = V\}$

if $a[m] < w$ **then**

$\{a[m] < w \ \wedge \ 0 \leq j \leq m < k \leq n \ \wedge \ a[j-1] < w \leq a[k] \wedge k - j = V\}$

$j := m + 1;$

else

$\{w \leq a[m] \ \wedge \ 0 \leq j \leq m < k \leq n \ \wedge \ a[j-1] < w \leq a[k] \wedge k - j = V\}$

$k := m;$

end

$\{J \wedge vf < V\}$

Binary Search: Body of the Loop

$\{J \wedge B \wedge vf = V\}$

$\{0 \leq j \leq k \leq n \;\wedge\; a[j-1] < w \leq a[k] \;\wedge\; j \neq k \;\wedge\; k - j = V\}$

(* $(j \leq k \wedge j < k) \equiv (j + j \leq j + k \wedge j + k < k + k) \equiv 2 \cdot j \leq j + k < 2 \cdot k$ *)

$\{0 \leq j \leq (j + k)$ **div** $2 < k \leq n \;\wedge\; a[j-1] < w \leq a[k] \;\wedge\; k - j = V\}$

$m := (j + k)$ **div** 2;

$\{0 \leq j \leq m < k \leq n \;\wedge\; a[j-1] < w \leq a[k] \;\wedge\; k - j = V\}$

if $a[m] < w$ **then**

$\{a[m] < w \;\wedge\; 0 \leq j \leq m < k \leq n \;\wedge\; a[j-1] < w \leq a[k] \wedge k - j = V\}$

(* *logic; calculus; prepare* $j := m + 1$ *)

$\{0 \leq m + 1 \leq k \leq n \;\wedge\; a[m + 1 - 1] < w \leq a[k] \;\wedge\; k - (m + 1) < V\}$

$j := m + 1$;

else

$\{w \leq a[m] \;\wedge\; 0 \leq j \leq m < k \leq n \;\wedge\; a[j-1] < w \leq a[k] \wedge k - j = V\}$

$k := m$;

end

$\{J \wedge vf < V\}$

Binary Search: Body of the Loop

$\{J \wedge B \wedge vf = V\}$
$\{0 \leq j \leq k \leq n \ \wedge \ a[j-1] < w \leq a[k] \ \wedge \ j \neq k \ \wedge \ k - j = V\}$
(* $(j \leq k \wedge j < k) \equiv (j + j \leq j + k \wedge j + k < k + k) \equiv 2 \cdot j \leq j + k < 2 \cdot k$ *)
$\{0 \leq j \leq (j+k)$ **div** $2 < k \leq n \ \wedge \ a[j-1] < w \leq a[k] \ \wedge \ k - j = V\}$
$m := (j+k)$ **div** 2;
$\{0 \leq j \leq m < k \leq n \ \wedge \ a[j-1] < w \leq a[k] \ \wedge \ k - j = V\}$
if $a[m] < w$ **then**
$\{a[m] < w \ \wedge \ 0 \leq j \leq m < k \leq n \ \wedge \ a[j-1] < w \leq a[k] \wedge k - j = V\}$
(* *logic; calculus; prepare* $j := m + 1$ *)
$\{0 \leq m+1 \leq k \leq n \ \wedge \ a[m+1-1] < w \leq a[k] \ \wedge \ k - (m+1) < V\}$
$j := m + 1$;
$\{0 \leq j \leq k \leq n \ \wedge \ a[j-1] < w \leq a[k] \ \wedge \ k - j < V\}$
else
$\{w \leq a[m] \ \wedge \ 0 \leq j \leq m < k \leq n \ \wedge \ a[j-1] < w \leq a[k] \wedge k - j = V\}$

$k := m$;

end
$\{J \wedge vf < V\}$

Binary Search: Body of the Loop

$\{J \wedge B \wedge vf = V\}$
$\{0 \leq j \leq k \leq n \ \wedge \ a[j-1] < w \leq a[k] \ \wedge \ j \neq k \ \wedge \ k - j = V\}$
(* $(j \leq k \wedge j < k) \equiv (j + j \leq j + k \wedge j + k < k + k) \equiv 2 \cdot j \leq j + k < 2 \cdot k$ *)
$\{0 \leq j \leq (j+k)$ **div** $2 < k \leq n \ \wedge \ a[j-1] < w \leq a[k] \ \wedge \ k - j = V\}$
$m := (j+k)$ **div** 2;
$\{0 \leq j \leq m < k \leq n \ \wedge \ a[j-1] < w \leq a[k] \ \wedge \ k - j = V\}$
if $a[m] < w$ **then**
$\{a[m] < w \ \wedge \ 0 \leq j \leq m < k \leq n \ \wedge \ a[j-1] < w \leq a[k] \wedge k - j = V\}$
(* *logic; calculus; prepare* $j := m + 1$ *)
$\{0 \leq m + 1 \leq k \leq n \ \wedge \ a[m+1-1] < w \leq a[k] \ \wedge \ k - (m+1) < V\}$
$j := m + 1$;
$\{0 \leq j \leq k \leq n \ \wedge \ a[j-1] < w \leq a[k] \ \wedge \ k - j < V\}$
else
$\{w \leq a[m] \ \wedge \ 0 \leq j \leq m < k \leq n \ \wedge \ a[j-1] < w \leq a[k] \wedge k - j = V\}$
(* *logic; calculus; prepare* $k := m$ *)
$\{0 \leq j \leq m \leq n \ \wedge \ a[j-1] < w \leq a[m] \ \wedge \ m - j < V\}$
$k := m$;

end
$\{J \wedge vf < V\}$

Binary Search: Body of the Loop

$\{J \wedge B \wedge vf = V\}$
$\{0 \leq j \leq k \leq n \ \wedge \ a[j-1] < w \leq a[k] \ \wedge \ j \neq k \ \wedge \ k - j = V\}$
(* $(j \leq k \wedge j < k) \equiv (j + j \leq j + k \wedge j + k < k + k) \equiv 2 \cdot j \leq j + k < 2 \cdot k$ *)
$\{0 \leq j \leq (j + k)$ **div** $2 < k \leq n \ \wedge \ a[j-1] < w \leq a[k] \ \wedge \ k - j = V\}$
$m := (j + k)$ **div** 2;
$\{0 \leq j \leq m < k \leq n \ \wedge \ a[j-1] < w \leq a[k] \ \wedge \ k - j = V\}$
if $a[m] < w$ **then**
$\{a[m] < w \ \wedge \ 0 \leq j \leq m < k \leq n \ \wedge \ a[j-1] < w \leq a[k] \wedge k - j = V\}$
(* *logic; calculus; prepare* $j := m + 1$ *)
$\{0 \leq m + 1 \leq k \leq n \ \wedge \ a[m + 1 - 1] < w \leq a[k] \ \wedge \ k - (m + 1) < V\}$
$j := m + 1$;
$\{0 \leq j \leq k \leq n \ \wedge \ a[j-1] < w \leq a[k] \ \wedge \ k - j < V\}$
else
$\{w \leq a[m] \ \wedge \ 0 \leq j \leq m < k \leq n \ \wedge \ a[j-1] < w \leq a[k] \wedge k - j = V\}$
(* *logic; calculus; prepare* $k := m$ *)
$\{0 \leq j \leq m \leq n \ \wedge \ a[j-1] < w \leq a[m] \ \wedge \ m - j < V\}$
$k := m$;
$\{0 \leq j \leq k \leq n \ \wedge \ a[j-1] < w \leq a[k] \ \wedge \ k - j < V\}$
end
$\{J \wedge vf < V\}$

Binary Search: Body of the Loop

$\{J \wedge B \wedge vf = V\}$
$\{0 \leq j \leq k \leq n \ \wedge \ a[j-1] < w \leq a[k] \ \wedge \ j \neq k \ \wedge \ k - j = V\}$
(* $(j \leq k \wedge j < k) \equiv (j + j \leq j + k \wedge j + k < k + k) \equiv 2 \cdot j \leq j + k < 2 \cdot k$ *)
$\{0 \leq j \leq (j + k)$ **div** $2 < k \leq n \ \wedge \ a[j-1] < w \leq a[k] \ \wedge \ k - j = V\}$
$m := (j + k)$ **div** 2;
$\{0 \leq j \leq m < k \leq n \ \wedge \ a[j-1] < w \leq a[k] \ \wedge \ k - j = V\}$
if $a[m] < w$ **then**
$\{a[m] < w \ \wedge \ 0 \leq j \leq m < k \leq n \ \wedge \ a[j-1] < w \leq a[k] \wedge k - j = V\}$
(* *logic; calculus; prepare* $j := m + 1$ *)
$\{0 \leq m + 1 \leq k \leq n \ \wedge \ a[m + 1 - 1] < w \leq a[k] \ \wedge \ k - (m + 1) < V\}$
$j := m + 1$;
$\{0 \leq j \leq k \leq n \ \wedge \ a[j-1] < w \leq a[k] \ \wedge \ k - j < V\}$
else
$\{w \leq a[m] \ \wedge \ 0 \leq j \leq m < k \leq n \ \wedge \ a[j-1] < w \leq a[k] \wedge k - j = V\}$
(* *logic; calculus; prepare* $k := m$ *)
$\{0 \leq j \leq m \leq n \ \wedge \ a[j-1] < w \leq a[m] \ \wedge \ m - j < V\}$
$k := m$;
$\{0 \leq j \leq k \leq n \ \wedge \ a[j-1] < w \leq a[k] \ \wedge \ k - j < V\}$
end
$\{J \wedge vf < V\}$

Binary Search: Body of the Loop

$\{J \wedge B \wedge vf = V\}$
$\{0 \leq j \leq k \leq n \ \wedge \ a[j-1] < w \leq a[k] \ \wedge \ j \neq k \ \wedge \ k - j = V\}$
(* $(j \leq k \wedge j < k) \equiv (j + j \leq j + k \wedge j + k < k + k) \equiv 2 \cdot j \leq j + k < 2 \cdot k$ *)
$\{0 \leq j \leq (j + k)$ **div** $2 < k \leq n \ \wedge \ a[j-1] < w \leq a[k] \ \wedge \ k - j = V\}$
$m := (j + k)$ **div** 2;
$\{0 \leq j \leq m < k \leq n \ \wedge \ a[j-1] < w \leq a[k] \ \wedge \ k - j = V\}$
if $a[m] < w$ **then**
$\{a[m] < w \ \wedge \ 0 \leq j \leq m < k \leq n \ \wedge \ a[j-1] < w \leq a[k] \wedge k - j = V\}$
(* *logic; calculus; prepare* $j := m + 1$ *)
$\{0 \leq m + 1 \leq k \leq n \ \wedge \ a[m + 1 - 1] < w \leq a[k] \ \wedge \ k - (m + 1) < V\}$
$j := m + 1$;
$\{0 \leq j \leq k \leq n \ \wedge \ a[j-1] < w \leq a[k] \ \wedge \ k - j < V\}$
else
$\{w \leq a[m] \ \wedge \ 0 \leq j \leq m < k \leq n \ \wedge \ a[j-1] < w \leq a[k] \wedge k - j = V\}$
(* *logic; calculus; prepare* $k := m$ *)
$\{0 \leq j \leq m \leq n \ \wedge \ a[j-1] < w \leq a[m] \ \wedge \ m - j < V\}$
$k := m$;
$\{0 \leq j \leq k \leq n \ \wedge \ a[j-1] < w \leq a[k] \ \wedge \ k - j < V\}$
end (* *collect branches; definitions* J *and* vf *)
$\{J \wedge vf < V\}$

Binary Search: Conclusion

const $n : \mathbb{N},\ w : \mathbb{Z},\ a :$ **array** $[0..n)$ **of** $\mathbb{R}$;
var $k,\ j,\ m : \mathbb{N}$;
  $\{P :\ a$ is ascending$\}$
$j := 0;\ k := n;$
  $\{J :\ 0 \leq j \leq k \leq n \wedge a[j-1] < w \leq a[k]\}$
    (* $vf = k - j$ *)
while $j \neq k$ **do**
  $m := (j + k)$ **div** 2;
  if $a[m] < w$ **then**
    $j := m + 1;$
  else
    $k := m;$
  end;
end;
  $\{k = \text{Min}\ \{i \in \mathbb{N} \mid i < n \Rightarrow w \leq a[i]\}\}$
if $k < n \wedge a[k] \neq w$ **then**
  $k := n;$
end;
  $\{Q :\ k = \text{Min}\ \{i \in \mathbb{N} \mid i < n \Rightarrow w = a[i]\}\}$

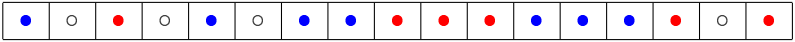
Outline

The Dutch National Flag problem (DNFP)

- A sorting problem introduced by Dijkstra.
- **Input**: An array of red, white, and blue balls.
  Output: The array re-arranged in a such way that balls of the same color are gathered together.

The Dutch National Flag problem (DNFP)

- A sorting problem introduced by Dijkstra.
- **Input**: An array of red, white, and blue balls.
  Output: The array re-arranged in a such way that balls of the same color are gathered together.
- Example: Given an array such as

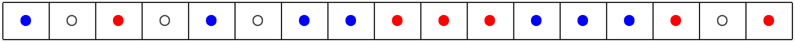
  

  the task is to transform it into

The Dutch National Flag problem (DNFP)

- A sorting problem introduced by Dijkstra.
- **Input**: An array of red, white, and blue balls.
  Output: The array re-arranged in a such way that balls of the same color are gathered together.
- Example: Given an array such as

  the task is to transform it into

- Notice: the array can only be modified by swapping two elements.
- We seek an efficient iterative procedure.
  As we will see, the choice of the invariant will be crucial.

DNFP: Statement

- An array a of length n, which stores three sorts of elements (denoted 0, 1, and 2, representing the balls of different colors).

DNFP: Statement

- An array a of length n, which stores three sorts of elements (denoted 0, 1, and 2, representing the balls of different colors).
- For each index $i \in [0..n)$, we have $a[i] = 0 \vee a[i] = 1 \vee a[i] = 2$.

DNFP: Statement

- An array a of length n, which stores three sorts of elements (denoted 0, 1, and 2, representing the balls of different colors).
- For each index $i \in [0..n)$, we have $a[i] = 0 \vee a[i] = 1 \vee a[i] = 2$.
- We can swap elements, assuming the following specification:

$$\{0 \leq i = I < n \ \wedge \ 0 \leq j = J < n \ \wedge \ a[i] = X \ \wedge \ a[j] = Y\}$$

swap(i, j)

$$\{i = I \ \wedge \ j = J \ \wedge \ a[i] = Y \ \wedge \ a[j] = X\}$$

DNFP: Statement

- An array a of length n, which stores three sorts of elements (denoted 0, 1, and 2, representing the balls of different colors).
- For each index $i \in [0..n)$, we have $a[i] = 0 \lor a[i] = 1 \lor a[i] = 2$.
- We can swap elements, assuming the following specification:

  $\{0 \le i = I < n \ \land \ 0 \le j = J < n \ \land \ a[i] = X \ \land \ a[j] = Y\}$
  swap(i, j)
  $\{i = I \ \land \ j = J \ \land \ a[i] = Y \ \land \ a[j] = X\}$

- We look for a command that, after termination, ensures that there are indices r and b such that:

$$\begin{aligned}
&0 \le r \le w \le n \\
\land\ &(\forall i : 0 \le i < r,\ a[i] = 0) \\
\land\ &(\forall i : r \le i < w,\ a[i] = 1) \\
\land\ &(\forall i : w \le i < n,\ a[i] = 2)
\end{aligned}$$

Note: This postcondition allows for zero balls of each color.

DNFP: Invariant (1/3)

What is a good invariant for the required an iterative process?

- At the beginning all balls are mixed; at the end, they are sorted. These should be two special cases of our invariant.

DNFP: Invariant (1/3)

What is a good invariant for the required an iterative process?

- At the beginning all balls are mixed; at the end, they are sorted. These should be two special cases of our invariant.
- It is natural to design an invariant that partitions the array into four segments: red, white, blue, and ‘mixed’ (unsorted).
  - At the beginning, the first three segments are empty and the mixed segment covers the entire array.
  - At the end, the mixed segment is empty.

DNFP: Invariant (2/3)

What is a good invariant for the required an iterative process?

- It is natural to design an invariant that partitions the array into four segments: red, white, blue, and ‘mixed’ (unsorted).
- There are four alternatives:

DNFP: Invariant (2/3)

What is a good invariant for the required an iterative process?

- It is natural to design an invariant that partitions the array into four segments: red, white, blue, and ‘mixed’ (unsorted).
- There are four alternatives:

0 … n

red	white	blue	unsorted

DNFP: Invariant (2/3)

What is a good invariant for the required an iterative process?

- It is natural to design an invariant that partitions the array into four segments: red, white, blue, and ‘mixed’ (unsorted).
- There are four alternatives:

0 n

red	white	blue	unsorted

0 n

red	white	unsorted	blue

DNFP: Invariant (2/3)

What is a good invariant for the required an iterative process?

- It is natural to design an invariant that partitions the array into four segments: red, white, blue, and 'mixed' (unsorted).
- There are four alternatives:

0 n

red	white	blue	unsorted

0 n

red	white	unsorted	blue

0 n

red	unsorted	white	blue

DNFP: Invariant (2/3)

What is a good invariant for the required an iterative process?

- It is natural to design an invariant that partitions the array into four segments: red, white, blue, and ‘mixed’ (unsorted).
- There are four alternatives:

0 ... n

red	white	blue	unsorted

0 ... n

red	white	unsorted	blue

0 ... n

red	unsorted	white	blue

0 ... n

unsorted	red	white	blue

- What (dis)advantages do you find in each of these options?

DNFP: Invariant (3/3)

- The preferable option is to maintain the ‘mixed’ segment in the interior of the array (why?)

DNFP: Invariant (3/3)

- The preferable option is to maintain the 'mixed' segment in the interior of the array (why?)
- The invariant modifies the postcondition by introducing a new index/variable b:

$$
\begin{aligned}
& 0 \le r \le w \le b \le n \\
\wedge\ & (\forall i : 0 \le i < r,\ a[i] = 0) \\
\wedge\ & (\forall i : r \le i < w,\ a[i] = 1) \\
\wedge\ & (\forall i : b \le i < n,\ a[i] = 2)
\end{aligned}
$$

  The indices i such that $w \le i < b$ define the 'mixed' segment. Graphically:

0	r	w	b	n
red	white	unsorted	blue	

- With this invariant, the variant function is
  The guard of the loop is

DNFP: Invariant (3/3)

- The preferable option is to maintain the 'mixed' segment in the interior of the array (why?)
- The invariant modifies the postcondition by introducing a new index/variable b:

$$
\begin{aligned}
& 0 \leq r \leq w \leq b \leq n \\
\wedge\ & (\forall i : 0 \leq i < r,\ a[i] = 0) \\
\wedge\ & (\forall i : r \leq i < w,\ a[i] = 1) \\
\wedge\ & (\forall i : b \leq i < n,\ a[i] = 2)
\end{aligned}
$$

The indices i such that $w \leq i < b$ define the 'mixed' segment. Graphically:

0	r	w	b	n
red	white	unsorted	blue	

- With this invariant, the variant function is $vf = b - w$. The guard of the loop is

DNFP: Invariant (3/3)

- The preferable option is to maintain the 'mixed' segment in the interior of the array (why?)
- The invariant modifies the postcondition by introducing a new index/variable b:

$$
\begin{aligned}
& 0 \leq r \leq w \leq b \leq n \\
\wedge\ & (\forall i : 0 \leq i < r,\ a[i] = 0) \\
\wedge\ & (\forall i : r \leq i < w,\ a[i] = 1) \\
\wedge\ & (\forall i : b \leq i < n,\ a[i] = 2)
\end{aligned}
$$

The indices i such that $w \leq i < b$ define the 'mixed' segment. Graphically:

0	r	w	b n
red	white	unsorted	blue

- With this invariant, the variant function is $vf = b - w$. The guard of the loop is $B : w < b$.

DNFP: Idea of the Body (1/3)

In the general case, we have:

						r			w				b				n
0	0	0	0	0	0	1	1	1	?	?	?	?	2	2	2	2	

We act depending on the color of the element at index w.

First case:

- If $a[w] = 1$ then

0	0	0	0	0	0	1	1	1	1	?	?	?	2	2	2	2

  and so it suffices to execute

DNFP: Idea of the Body (1/3)

In the general case, we have:

$$\begin{array}{|c|c|c|c|c|c|c|c|c||c|c|c|c||c|c|c|c|l} \multicolumn{6}{c}{} & \multicolumn{1}{c}{r} & \multicolumn{2}{c}{} & \multicolumn{1}{c}{w} & \multicolumn{3}{c}{} & \multicolumn{1}{c}{b} & \multicolumn{3}{c}{} & n \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & ? & ? & ? & ? & 2 & 2 & 2 & 2 & \\ \hline \end{array}$$

We act depending on the color of the element at index w.

First case:

- If $a[w] = 1$ then

$$\begin{array}{|c|c|c|c|c|c|c|c|c||c|c|c|c||c|c|c|c|} \hline 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & ? & ? & ? & 2 & 2 & 2 & 2 \\ \hline \end{array}$$

  and so it suffices to execute

$$w := w + 1$$

DNFP: Idea of the Body (1/3)

In the general case, we have:

						r			w				b				n
0	0	0	0	0	0	1	1	1	?	?	?	?	2	2	2	2	

We act depending on the color of the element at index w.

First case:

- If $a[w] = 1$ then

0	0	0	0	0	0	1	1	1	1	?	?	?	2	2	2	2

  and so it suffices to execute

  $$w := w + 1$$

  This yields:

0	0	0	0	0	0	1	1	1	1	?	?	?	2	2	2	2

DNFP: Idea of the Body (2/3)

In the general case, we have:

						r			w				b				n
0	0	0	0	0	0	1	1	1	?	?	?	?	2	2	2	2	

We act depending on the color of the element at index w.

Second case:

- If $a[w] = 2$ then

0	0	0	0	0	0	1	1	1	2	?	?	?	2	2	2	2

  and so we execute

DNFP: Idea of the Body (2/3)

In the general case, we have:

						r			w				b				n
0	0	0	0	0	0	1	1	1	?	?	?	?	2	2	2	2	

We act depending on the color of the element at index w.

Second case:

- If $a[w] = 2$ then

0	0	0	0	0	0	1	1	1	2	?	?	?	2	2	2	2

  and so we execute

$$\text{swap}(b-1, w);$$
$$b := b - 1$$

DNFP: Idea of the Body (2/3)

In the general case, we have:

						r			w				b				n
0	0	0	0	0	0	1	1	1	?	?	?	?	2	2	2	2	

We act depending on the color of the element at index w.

Second case:

- If $a[w] = 2$ then

0	0	0	0	0	0	1	1	1	2	?	?	?	2	2	2	2

  and so we execute

  $$\text{swap}(b - 1, w);$$
  $$b := b - 1$$

  This yields:

0	0	0	0	0	0	1	1	1	?	?	?	2	2	2	2	2

DNFP: Idea of the Body (3/3)

In the general case, we have:

						r			w				b				n
0	0	0	0	0	0	1	1	1	?	?	?	?	2	2	2	2	

We act depending on the color of the element at index w.

Final case:

- If $a[w] = 0$ then

0	0	0	0	0	0	1	1	1	0	?	?	?	2	2	2	2

  and so we execute

DNFP: Idea of the Body (3/3)

In the general case, we have:

						r			w				b				n
0	0	0	0	0	0	1	1	1	?	?	?	?	2	2	2	2	

We act depending on the color of the element at index w.

Final case:

- If $a[w] = 0$ then

0	0	0	0	0	0	1	1	1	0	?	?	?	2	2	2	2

and so we execute

$$\text{swap}(r, w);$$
$$r := r + 1; \quad w := w + 1$$

This yields:

0	0	0	0	0	0	0	1	1	1	?	?	?	2	2	2	2

DNFP: Conclusion

```
const n : ℕ, a : array [0..n) of ℤ;
var r, w, b : ℤ;
   {P : (as discussed above)}
r := 0; w := 0; b := n;
   {J : (as discussed above)}
      (* vf = b − w *)
while w < b do
   if a[w] = 1 then
      w := w + 1;
    else
         if a[w] = 2 then
            swap(b − 1, w);
            b := b − 1;
          else
         if a[w] = 0 then
            swap(r, w);
            r := r + 1; w := w + 1;
         end;
      end;
   end
      {Q : (as discussed above)}
```

The End