G52DOA - Derivation of Algorithms

Lecture 2

Venanzio Capretta

Nottingham, Friday 5 February 2010

Dutch National Flag

Construct an algorithm that performs the following task:

- Input: An array **t** of colors (BLUE , WHITE , RED).
- Output: **t** with elements rearranged so that the BLUE ones come first, the WHITE ones are in the middle and the RED ones come last.

$$WRBRBWRBWWRRW \rightsquigarrow BBBWWWWWRRRRR$$

(The Dutch national flag consists of three horizontal bands coloured blue, white and red.)

Dijkstra's Algorithm

input $\mathbf{t}$: colour array;
$b := 0; i := 0;$
$r :=$ length of $\mathbf{t}$;
while $i < r$ do
  if $\mathbf{t}[i] = B$ then
    (swap($\mathbf{t}, b, i$);
      $b := b + 1$;
      $i := i + 1$)
  else if $\mathbf{t}[i] = W$ then
      $i := i + 1$;
  else
    ($r := r - 1$;
      swap($\mathbf{t}, r, i$))

A Run of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13
W	R	B	R	B	W	R	B	W	W	R	R	W	
$\uparrow$													$\uparrow$
$b\ i$													r

Initial State

A Run of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13
W	R	B	R	B	W	R	B	W	W	R	R	W	
$\uparrow$													$\uparrow$
$b\,i$													r

$$i < r, \mathbf{t}[i] = W \longmapsto i := i + 1$$

A Run of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13
W	R	B	R	B	W	R	B	W	W	R	R	W	
$\uparrow$	$\uparrow$												$\uparrow$
b	i												r

$$i < r, \mathbf{t}[i] = R \longmapsto r := r - 1; \mathrm{swap}(\mathbf{t}, r, i)$$

A Run of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13
W	W	B	R	B	W	R	B	W	W	R	R	R	
$\uparrow$	$\uparrow$											$\uparrow$	
b	i											r	

$$i < r, \mathbf{t}[i] = W \longmapsto i := i + 1$$

A Run of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13
W	W	B	R	B	W	R	B	W	W	R	R	R	
$\uparrow$		$\uparrow$										$\uparrow$	
b		i										r	

$$i < r, \mathbf{t}[i] = B \longmapsto \text{swap}(\mathbf{t}, b, i); b := b + 1; i := i + 1$$

A Run of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13
B	W	W	R	B	W	R	B	W	W	R	R	R	
	$\uparrow$		$\uparrow$									$\uparrow$	
	b		i									r	

$$i < r, \mathbf{t}[i] = R \longmapsto r := r - 1; \mathrm{swap}(\mathbf{t}, r, i)$$

A Run of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13
B	W	W	R	B	W	R	B	W	W	R	R	R	
	$\uparrow$		$\uparrow$								$\uparrow$		
	b		i								r		

$$i < r, \mathbf{t}[i] = R \longmapsto r := r - 1; \mathrm{swap}(\mathbf{t}, r, i)$$

A Run of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13
B	W	W	R	B	W	R	B	W	W	R	R	R	
	$\uparrow$		$\uparrow$							$\uparrow$			
	b		i							r			

$$i < r, \mathbf{t}[i] = R \longmapsto r := r - 1; \mathrm{swap}(\mathbf{t}, r, i)$$

A Run of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13
B	W	W	W	B	W	R	B	W	R	R	R	R	
	$\uparrow$		$\uparrow$						$\uparrow$				
	b		i						r				

$$i < r, \mathbf{t}[i] = W \longmapsto i := i + 1$$

A Run of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13
B	W	W	W	B	W	R	B	W	R	R	R	R	
	$\uparrow$			$\uparrow$					$\uparrow$				
	b			i					r				

$$i < r, \mathbf{t}[i] = B \longmapsto \mathsf{swap}(\mathbf{t}, b, i); b := b + 1; i := i + 1$$

A Run of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13
B	B	W	W	W	W	R	B	W	R	R	R	R	
		$\uparrow$			$\uparrow$				$\uparrow$				
		b			i				r				

$$i < r, \mathbf{t}[i] = W \longmapsto i := i + 1$$

A Run of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13
B	B	W	W	W	W	R	B	W	R	R	R	R	
		$\uparrow$				$\uparrow$			$\uparrow$				
		b				i			r				

$$i < r, \mathbf{t}[i] = R \longmapsto r := r - 1; \mathsf{swap}(\mathbf{t}, r, i)$$

A Run of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13
B	B	W	W	W	W	W	B	R	R	R	R	R	
		$\uparrow$				$\uparrow$		$\uparrow$					
		b				i		r					

$$i < r, \mathbf{t}[i] = W \longmapsto i := i + 1$$

A Run of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13
B	B	W	W	W	W	W	B	R	R	R	R	R	
		$\uparrow$					$\uparrow$	$\uparrow$					
		b					i	r					

$$i < r, \mathbf{t}[i] = B \longmapsto \mathrm{swap}(\mathbf{t}, b, i); b := b + 1; i := i + 1$$

A Run of the Algorithm

0	1	2	3	4	5	6	7	8	9	10	11	12	13
B	B	B	W	W	W	W	W	R	R	R	R	R	
			$\uparrow$					$\uparrow$					
			b					$i\ r$					

$$i = r \longmapsto \text{STOP}$$

Annotated Algorithm

$\{P\}$

```
input t : colour array;
b := 0; i := 0;
r := length of  t;
while i < r do
   if t[i] = B then
      (  swap(t, b, i);
         b := b + 1;
         i := i + 1   )
   else if t[i] = W then
         i := i + 1;
   else
      (  r := r − 1;
         swap(t, r, i)   )
```

$\{Q\}$

Annotated Algorithm

```
                      input t : colour array;
{P}                   b := 0; i := 0;
                      r := length of t;
{invariant : I}       while i < r do
                        if t[i] = B then
                          (  swap(t, b, i);
                             b := b + 1;
                             i := i + 1  )
                        else if t[i] = W then
                             i := i + 1;
                        else
                          (  r := r − 1;
                             swap(t, r, i)  )
                                                {Q}
```

Annotated Algorithm

$\{P\}$

input $\mathbf{t}$: colour array;
$b := 0; i := 0;$
$r :=$ length of $\mathbf{t}$;

$\{invariant : I\}$

```
while i < r do
   if t[i] = B then
      (   swap(t, b, i);
          b := b + 1;
          i := i + 1   )
   else if t[i] = W then
          i := i + 1;
   else
      (   r := r − 1;
          swap(t, r, i)   )
```

$\{I\}$

$\{Q\}$

Annotated Algorithm

	input $\mathbf{t}$: colour array;	
$\{P\}$	$b := 0; i := 0;$	
	$r :=$ length of $\mathbf{t}$;	$\{I\}$
$\{\text{invariant} : I\}$	while $i < r$ do	$\{I \wedge i < r\}$
	if $\mathbf{t}[i] = B$ then	
	(swap$(\mathbf{t}, b, i)$;	
	$b := b + 1$;	
	$i := i + 1$)	
	else if $\mathbf{t}[i] = W$ then	
	$i := i + 1$;	
	else	
	($r := r - 1$;	
	swap$(\mathbf{t}, r, i)$)	
		$\{Q\}$

Annotated Algorithm

	input $\mathbf{t}$: colour array;	
$\{P\}$	$b := 0; i := 0;$	
	$r :=$ length of $\mathbf{t}$;	$\{I\}$
$\{\text{invariant} : I\}$	while $i < r$ do	$\{I \land i < r\}$
	if $\mathbf{t}[i] = B$ then	$\{I \land i < r \land \mathbf{t}[i] = B\}$
	(swap$(\mathbf{t}, b, i)$;	
	$b := b + 1$;	
	$i := i + 1$)	$\{I\}$
	else if $\mathbf{t}[i] = W$ then	
	$i := i + 1$;	
	else	
	($r := r - 1$;	
	swap$(\mathbf{t}, r, i)$)	
		$\{Q\}$

Annotated Algorithm

	input $\mathbf{t}$: colour array;	
$\{P\}$	$b := 0; i := 0;$	
	$r :=$ length of $\mathbf{t}$;	$\{I\}$
$\{\text{invariant} : I\}$	while $i < r$ do	$\{I \wedge i < r\}$
	if $\mathbf{t}[i] = B$ then	$\{I \wedge i < r \wedge \mathbf{t}[i] = B\}$
	(swap$(\mathbf{t}, b, i)$;	
	$b := b + 1$;	
	$i := i + 1$)	$\{I\}$
	else if $\mathbf{t}[i] = W$ then	$\{I \wedge i < r \wedge \mathbf{t}[i] = W\}$
	$i := i + 1$;	$\{I\}$
	else	
	($r := r - 1$;	
	swap$(\mathbf{t}, r, i)$)	
		$\{Q\}$

Annotated Algorithm

	input **t** : colour array;	
$\{P\}$	$b := 0; i := 0;$	
	$r :=$ length of **t**;	$\{I\}$
$\{\text{invariant} : I\}$	while $i < r$ do	$\{I \wedge i < r\}$
	if $\mathbf{t}[i] = B$ then	$\{I \wedge i < r \wedge \mathbf{t}[i] = B\}$
	(swap$(\mathbf{t}, b, i)$;	
	$b := b + 1$;	
	$i := i + 1$)	$\{I\}$
	else if $\mathbf{t}[i] = W$ then	$\{I \wedge i < r \wedge \mathbf{t}[i] = W\}$
	$i := i + 1$;	$\{I\}$
	else	$\{I \wedge i < r \wedge \mathbf{t}[i] = R\}$
	($r := r - 1$;	
	swap$(\mathbf{t}, r, i)$)	$\{I\}$
		$\{Q\}$

Annotated Algorithm

	input $\mathbf{t}$: colour array;	
$\{P\}$	$b := 0; i := 0;$	
	$r :=$ length of $\mathbf{t}$;	$\{I\}$
$\{\text{invariant} : I\}$	while $i < r$ do	$\{I \wedge i < r\}$
	if $\mathbf{t}[i] = B$ then	$\{I \wedge i < r \wedge \mathbf{t}[i] = B\}$
	(swap($\mathbf{t}, b, i$);	
	$b := b + 1;$	
	$i := i + 1$)	$\{I\}$
	else if $\mathbf{t}[i] = W$ then	$\{I \wedge i < r \wedge \mathbf{t}[i] = W\}$
	$i := i + 1;$	$\{I\}$
	else	$\{I \wedge i < r \wedge \mathbf{t}[i] = R\}$
	($r := r - 1;$	
	swap($\mathbf{t}, r, i$))	$\{I\}$
$\{I \wedge i \geq r\}$		$\{Q\}$

Definition of the assertions

$$P \equiv \quad \mathbf{t} = \mathbf{t}_0 \wedge \text{length of } \mathbf{t} = n_0;$$

$$\begin{aligned} I \equiv \quad & 0 \leq b \leq i \leq r \leq n_0 \\ & \mathbf{t}[0..b-1] = B \wedge \mathbf{t}[b..i-1] = W \wedge \\ & \mathbf{t}[r..n_0-1] = R \wedge \mathbf{t} \sim \mathbf{t}_0; \end{aligned}$$

$$\begin{aligned} Q \equiv \quad & 0 \leq b \leq r \leq n_0 \wedge \\ & \mathbf{t}[0..b-1] = B \wedge \mathbf{t}[b..r-1] = W \wedge \\ & \mathbf{t}[r..n_0-1] = R \wedge \mathbf{t} \sim \mathbf{t}_0. \end{aligned}$$

Where
$\mathbf{t}[i..j]$ denotes the elements of **t** with indeces from i to j included; if $i > j$ there are no elements;
$\mathbf{t} \sim \mathbf{t}_0$ states that **t** and $\mathbf{t}_0$ have the same elements, although they may be in different positions.

Termination

We proved Partial Correctness:
If the program terminates, it give a correct result.

Still to prove Termination:
The program terminates on every input satisfying the precondition.

Termination of Algorithms

Termination of the algorithm:

- For every input satisfying the precondition
- The computation of the algorithm stops after a finite number of steps.

The problem of termination arises only for while loops.
Method: For every while loop

- Choose a positive integer expression in the variables, called the variant;
- Prove that the loop is executed only if the variant is positive;
- Prove that every execution of the loop decreases the variant.

Annotations for Termination

	input $\mathbf{t}$: colour array;	
	$b := 0; i := 0;$	
	$r :=$ length of $\mathbf{t}$;	
[variant : $r - i$]	while $i < r$ do	$[r - i = v_0 > 0]$
	if $\mathbf{t}[i] = B$ then	
	(swap($\mathbf{t}, b, i$);	
	$b := b + 1$;	
	$i := i + 1$)	$[r - i = v_0 - 1]$
	else if $\mathbf{t}[i] = W$ then	
	$i := i + 1$;	$[r - i = v_0 - 1]$
	else	
	($r := r - 1$;	
	swap($\mathbf{t}, r, i$))	$[r - i = v_0 - 1]$