While Loop

while b **do** P **od**

While Loop

$W \Leftarrow$ **while** b **do** P **od**

While Loop

$$W \Leftarrow \textbf{while } b \textbf{ do } P \textbf{ od}$$

means

$$W \Leftarrow \textbf{if } b \textbf{ then } P.\ W \textbf{ else } ok \textbf{ fi}$$

While Loop

$W \Leftarrow$ **while** b **do** P **od**

means

$W \Leftarrow$ **if** b **then** P. W **else** ok **fi**

while $n \neq \#L$ **do** $s := s + L\,n$. $n := n+1$. $t := t+1$ **od**

While Loop

$W \Leftarrow$ **while** b **do** P **od**

means

$W \Leftarrow$ **if** b **then** P. W **else** ok **fi**

$s' = s + \Sigma\, L\, [n;..\#L] \;\wedge\; t' = t + \#L - n \;\Leftarrow$

while $n \neq \#L$ **do** $s := s + L\, n$. $n := n+1$. $t := t+1$ **od**

While Loop

$W \Leftarrow$ **while** b **do** P **od**

means

$W \Leftarrow$ **if** b **then** $P.\ W$ **else** ok **fi**

to prove

$s' = s + \Sigma\, L\, [n;..\#L] \ \wedge\ t' = t + \#L - n \ \Leftarrow$

while $n{\neq}\#L$ **do** $s{:=}\ s + L\, n.\ \ n{:=}\ n{+}1.\ \ t{:=}\ t{+}1$ **od**

prove instead

$s' = s + \Sigma\, L\, [n;..\#L] \ \wedge\ t' = t + \#L - n \ \Leftarrow$

if $n{\neq}\#L$ **then** $\quad s{:=}\ s + L\, n.\ \ n{:=}\ n{+}1.\ \ t{:=}\ t{+}1.$

$s' = s + \Sigma\, L\, [n;..\#L] \ \wedge\ t' = t + \#L - n$

else ok **fi**

Exit Loop

do

A.

exit when b.

C

od

Exit Loop

$L \Leftarrow$ **do**

A.

exit when b.

C

od

Exit Loop

$L \Leftarrow$ **do**

A.

exit when b.

C

od

means

$L \Leftarrow A.$ **if** b **then** ok **else** $C.\ L$ **fi**

Exit Loop

$L \Leftarrow$ **do**

$A.$ ←

exit when $b.$

C

od

means

$L \Leftarrow A.$ **if** b **then** ok **else** $C.\ L$ **fi**

↑

Exit Loop

$L \Leftarrow$ **do**

A.

exit when b. ←

C

od

means

$L \Leftarrow A.$ **if** b **then** ok **else** $C.$ L **fi**

↑ ↑

Exit Loop

$L \Leftarrow$ **do**

A.

exit when b.

C ←

od

means

$L \Leftarrow A.$ **if** b **then** ok **else** $C.\ L$ **fi**

↑

Exit Loop

$L \Leftarrow$ **do**

A.

exit when b.

C

od

means

$L \Leftarrow A.$ **if** b **then** ok **else** $C.\ L$ **fi**

Deep Exit

do

A.

do

B.

exit 2 **when** *c*.

D

od.

E

od

Deep Exit

$P \Leftarrow$ **do**

A.

do

B.

exit 2 **when** c.

D

od.

E

od

Deep Exit

$P \Leftarrow$ **do**

A.

do

B.

exit 2 **when** c.

D

od.

E

od

means

$P \Leftarrow$

Deep Exit

$P \Leftarrow$ **do**

$A.$ ←

do

$B.$

exit 2 **when** $c.$

D

od.

E

od

means

$P \Leftarrow A.$

Deep Exit

$P \Leftarrow$ **do**

A.

do ←

B.

exit 2 **when** c.

D

od.

E

od

means

$P \Leftarrow A.\ Q$

$Q \Leftarrow$

Deep Exit

$P \Leftarrow$ **do**

A.

do

B. 

exit 2 **when** c.

D

od.

E

od

means

$P \Leftarrow A.\ Q$

$Q \Leftarrow B.$

Deep Exit

$P \Leftarrow$ **do**

A.

do

B.

exit 2 **when** c. 

D

od.

E

od

means

$P \Leftarrow A.\ Q$

$Q \Leftarrow B.$ **if** c **then** ok

Deep Exit

$P \Leftarrow$ **do**

A.

do

B.

exit 2 **when** c.

D ←

od.

E

od

means

$P \Leftarrow A.\ Q$

$Q \Leftarrow B.$ **if** c **then** ok **else** D.

Deep Exit

$P \Leftarrow$ **do**

A.

do

B.

exit 2 **when** c.

D

od. 

E

od

means

$P \Leftarrow A.\ Q$

$Q \Leftarrow B.$ **if** c **then** ok **else** $D.\ Q$ **fi**

Deep Exit

$P \Leftarrow$ **do**

A.

do

B.

exit 2 **when** c.

D

od.

$E \leftarrow ?$

od

means

$P \Leftarrow A.\ Q$

$Q \Leftarrow B.$ **if** c **then** ok **else** $D.\ Q$ **fi**

Deep Exit

$P \Leftarrow$ **do**

A.

exit 1 **when** b.

C.

do

D.

exit 2 **when** e.

F.

exit 1 **when** g.

H

od.

I

od

Deep Exit

$P \Leftarrow$ **do**

A.

exit 1 **when** b.

C.

do

D.

exit 2 **when** e.

F.

exit 1 **when** g.

H

od.

I

od

means

$P \Leftarrow$

Deep Exit

$P \Leftarrow$ **do**

$A.$ ←

exit 1 **when** b.

$C.$

do

$D.$

exit 2 **when** e.

$F.$

exit 1 **when** g.

H

od.

I

od

means

$P \Leftarrow A.$

Deep Exit

$P \Leftarrow$ **do**

A.

exit 1 **when** b. 

C.

do

D.

exit 2 **when** e.

F.

exit 1 **when** g.

H

od.

I

od

means

$P \Leftarrow A$. **if** b **then** ok

Deep Exit

$P \Leftarrow$ **do**

A.

exit 1 **when** b.

C. 

do

D.

exit 2 **when** e.

F.

exit 1 **when** g.

H

od.

I

od

means

$P \Leftarrow A$. **if** b **then** ok **else** C.

Deep Exit

$P \Leftarrow$ **do**

A.

exit 1 **when** b.

C.

do ←

D.

exit 2 **when** e.

F.

exit 1 **when** g.

H

od.

I

od

means

$P \Leftarrow A$. **if** b **then** ok **else** C. Q **fi**

$Q \Leftarrow$

Deep Exit

$P \Leftarrow$ **do**

A.

exit 1 **when** b.

C.

do

D. 

exit 2 **when** e.

F.

exit 1 **when** g.

H

od.

I

od

means

$P \Leftarrow A$. **if** b **then** ok **else** C. Q **fi**

$Q \Leftarrow D$.

Deep Exit

$P \Leftarrow$ **do**

A.

exit 1 **when** b.

C.

do

D.

exit 2 **when** e. 

F.

exit 1 **when** g.

H

od.

I

od

means

$P \Leftarrow A$. **if** b **then** ok **else** C. Q **fi**

$Q \Leftarrow D$. **if** e **then** ok

Deep Exit

$P \Leftarrow$ **do**

A.

exit 1 **when** b.

C.

do

D.

exit 2 **when** e.

F. 

exit 1 **when** g.

H

od.

I

od

means

$P \Leftarrow A$. **if** b **then** ok **else** C. Q **fi**

$Q \Leftarrow D$. **if** e **then** ok **else** F.

Deep Exit

$P \Leftarrow$ **do**

A.

exit 1 **when** b.

C.

do

D.

exit 2 **when** e.

F.

exit 1 **when** g. 

H

od.

I

od

means

$P \Leftarrow A$. **if** b **then** ok **else** C. Q **fi**

$Q \Leftarrow D$. **if** e **then** ok **else** F. **if** g **then**

Deep Exit

$P \Leftarrow$ **do**

A.

exit 1 **when** b.

C.

do

D.

exit 2 **when** e.

F.

exit 1 **when** g.

H

od.

I ←

od

means

$P \Leftarrow A$. **if** b **then** ok **else** C. Q **fi**

$Q \Leftarrow D$. **if** e **then** ok **else** F. **if** g **then** I.

Deep Exit

$P \Leftarrow$ **do**

A.

exit 1 **when** b.

C.

do

D.

exit 2 **when** e.

F.

exit 1 **when** g.

H

od.

I

od ←

means

$P \Leftarrow A$. **if** b **then** ok **else** C. Q **fi**

$Q \Leftarrow D$. **if** e **then** ok **else** F. **if** g **then** I. P

Deep Exit

$P \Leftarrow$ **do**

A.

exit 1 **when** b.

C.

do

D.

exit 2 **when** e.

F.

exit 1 **when** g.

H ←

od.

I

od

means

$P \Leftarrow A$. **if** b **then** ok **else** C. Q **fi**

$Q \Leftarrow D$. **if** e **then** ok **else** F. **if** g **then** I. P **else** H.

Deep Exit

$P \Leftarrow$ **do**

A.

exit 1 **when** b.

C.

do

D.

exit 2 **when** e.

F.

exit 1 **when** g.

H

od. ←

I

od

means

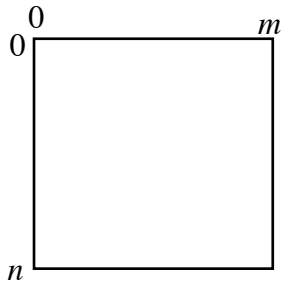
$P \Leftarrow A$. **if** b **then** ok **else** C. Q **fi**

$Q \Leftarrow D$. **if** e **then** ok **else** F. **if** g **then** I. P **else** H. Q **fi fi**

Two-Dimensional Search

Two-Dimensional Search

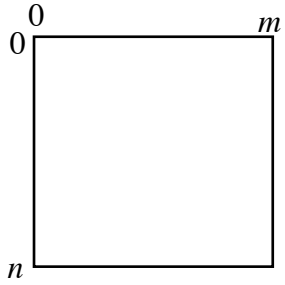
P = **if** x: $A\ (0,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**



Two-Dimensional Search

P = **if** x: $A\ (0,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

$P \Leftarrow i:=0.\ i \leq n \Rightarrow Q$

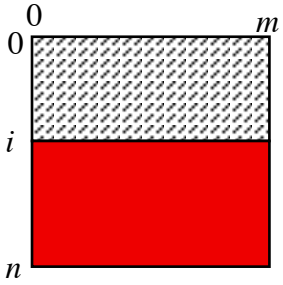


Two-Dimensional Search

P = **if** x: $A\ (0,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

Q = **if** x: $A\ (i,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

$P \Leftarrow i := 0.\ \ i \leq n \Rightarrow Q$



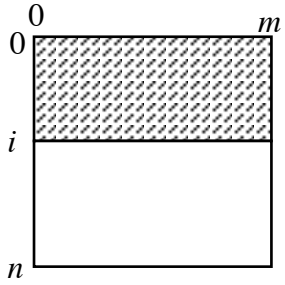
Two-Dimensional Search

P = **if** x: $A\ (0,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

Q = **if** x: $A\ (i,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

$P \Leftarrow i := 0.\ \ i \leq n \Rightarrow Q$

$i \leq n \Rightarrow Q \Leftarrow$



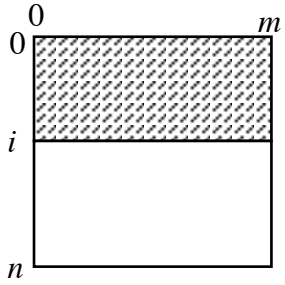
Two-Dimensional Search

P = **if** x: $A\ (0,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

Q = **if** x: $A\ (i,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

$P \Leftarrow i := 0.\ \ i \leq n \Rightarrow Q$

$i \leq n \Rightarrow Q \Leftarrow$ **if** $i=n$ **then** ok



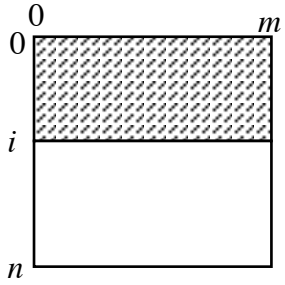
Two-Dimensional Search

P = **if** x: $A\ (0,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

Q = **if** x: $A\ (i,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

$P \Leftarrow i := 0.\ i \leq n \Rightarrow Q$

$i \leq n \Rightarrow Q \Leftarrow$ **if** $i=n$ **then** *ok* **else** $i<n \Rightarrow Q$ **fi**



Two-Dimensional Search

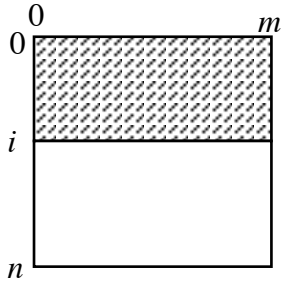
P = **if** x: $A\ (0,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

Q = **if** x: $A\ (i,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

$P \Leftarrow i := 0.\ i \le n \Rightarrow Q$

$i \le n \Rightarrow Q \Leftarrow$ **if** $i=n$ **then** *ok* **else** $i<n \Rightarrow Q$ **fi**

$i<n \Rightarrow Q \Leftarrow$



Two-Dimensional Search

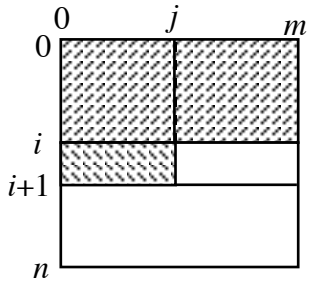
P = **if** x: $A\ (0,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

Q = **if** x: $A\ (i,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

$P \Leftarrow i:= 0.\ i \le n \Rightarrow Q$

$i \le n \Rightarrow Q \Leftarrow$ **if** $i=n$ **then** ok **else** $i<n \Rightarrow Q$ **fi**

$i<n \Rightarrow Q \Leftarrow j:= 0.\ i<n \wedge j \le m \Rightarrow R$



Two-Dimensional Search

P = **if** x: $A\ (0,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

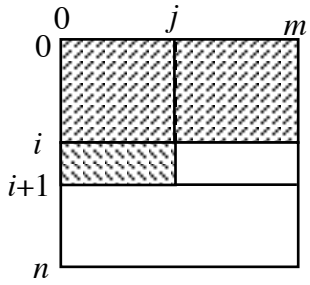
Q = **if** x: $A\ (i,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

R = **if** x: $A\ i\ (j,..m), A\ (i+1,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

$P \Leftarrow i := 0.\ i \le n \Rightarrow Q$

$i \le n \Rightarrow Q \Leftarrow$ **if** $i=n$ **then** ok **else** $i<n \Rightarrow Q$ **fi**

$i<n \Rightarrow Q \Leftarrow j := 0.\ i<n \wedge j \le m \Rightarrow R$



Two-Dimensional Search

P = **if** x: $A\ (0,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

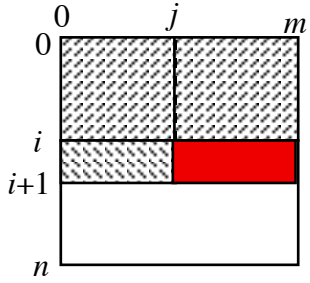
Q = **if** x: $A\ (i,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

R = **if** x: $A\ i\ (j,..m), A\ (i+1,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

$P \Leftarrow i:= 0.\ i \le n \Rightarrow Q$

$i \le n \Rightarrow Q \Leftarrow$ **if** $i=n$ **then** ok **else** $i<n \Rightarrow Q$ **fi**

$i<n \Rightarrow Q \Leftarrow j:= 0.\ i<n \wedge j \le m \Rightarrow R$



Two-Dimensional Search

P = **if** x: $A\ (0,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

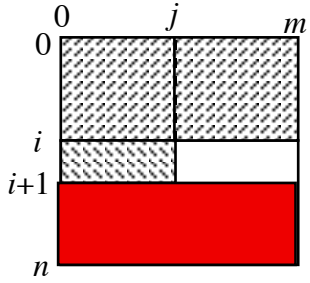
Q = **if** x: $A\ (i,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

R = **if** x: $A\ i\ (j,..m), A\ (i+1,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

$P \Leftarrow i := 0.\ i \leq n \Rightarrow Q$

$i \leq n \Rightarrow Q \Leftarrow$ **if** $i=n$ **then** ok **else** $i<n \Rightarrow Q$ **fi**

$i<n \Rightarrow Q \Leftarrow j := 0.\ i<n \wedge j \leq m \Rightarrow R$



Two-Dimensional Search

P = **if** x: $A\ (0,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

Q = **if** x: $A\ (i,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

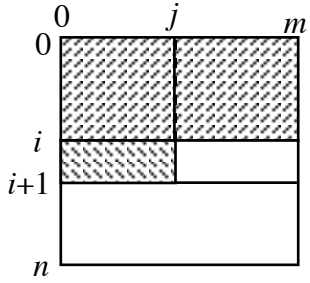
R = **if** x: $A\ i\ (j,..m), A\ (i+1,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

$P \Leftarrow i := 0.\ i \le n \Rightarrow Q$

$i \le n \Rightarrow Q \Leftarrow$ **if** $i=n$ **then** ok **else** $i<n \Rightarrow Q$ **fi**

$i<n \Rightarrow Q \Leftarrow j := 0.\ i<n \wedge j \le m \Rightarrow R$

$i<n \wedge j \le m \Rightarrow R \Leftarrow$



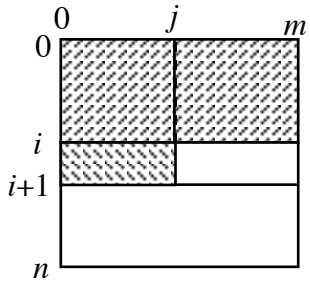
Two-Dimensional Search

P = **if** x: $A\ (0,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

Q = **if** x: $A\ (i,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

R = **if** x: $A\ i\ (j,..m), A\ (i+1,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

$P \Leftarrow i := 0.\ i \le n \Rightarrow Q$



$i \le n \Rightarrow Q \Leftarrow$ **if** $i=n$ **then** ok **else** $i<n \Rightarrow Q$ **fi**

$i<n \Rightarrow Q \Leftarrow j := 0.\ i<n \wedge j \le m \Rightarrow R$

$i<n \wedge j \le m \Rightarrow R \Leftarrow$ **if** $j=m$ **then** $i := i+1.\ i \le n \Rightarrow Q$

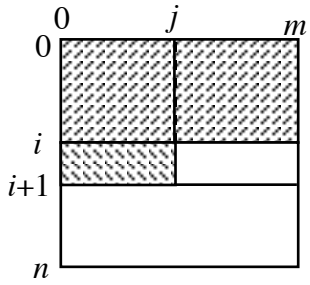
Two-Dimensional Search

P = **if** x: $A\ (0,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

Q = **if** x: $A\ (i,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

R = **if** x: $A\ i\ (j,..m), A\ (i+1,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

$P \Leftarrow i := 0.\ i \le n \Rightarrow Q$



$i \le n \Rightarrow Q \Leftarrow$ **if** $i=n$ **then** *ok* **else** $i<n \Rightarrow Q$ **fi**

$i<n \Rightarrow Q \Leftarrow j := 0.\ i<n \wedge j \le m \Rightarrow R$

$i<n \wedge j \le m \Rightarrow R \Leftarrow$ **if** $j=m$ **then** $i := i+1.\ i \le n \Rightarrow Q$ **else** $i<n \wedge j<m \Rightarrow R$ **fi**

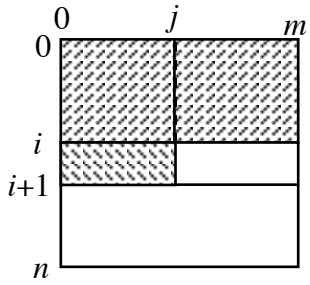
Two-Dimensional Search

P = **if** x: $A\ (0,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

Q = **if** x: $A\ (i,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

R = **if** x: $A\ i\ (j,..m), A\ (i+1,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

$P \Leftarrow i := 0.\ i \le n \Rightarrow Q$



$i \le n \Rightarrow Q \Leftarrow$ **if** $i=n$ **then** ok **else** $i<n \Rightarrow Q$ **fi**

$i<n \Rightarrow Q \Leftarrow j := 0.\ i<n \wedge j \le m \Rightarrow R$

$i<n \wedge j \le m \Rightarrow R \Leftarrow$ **if** $j=m$ **then** $i := i+1.\ i \le n \Rightarrow Q$ **else** $i<n \wedge j<m \Rightarrow R$ **fi**

$i<n \wedge j<m \Rightarrow R \Leftarrow$

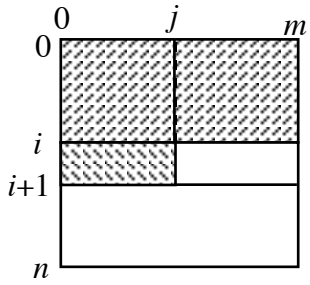
Two-Dimensional Search

P = **if** x: $A\ (0,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

Q = **if** x: $A\ (i,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

R = **if** x: $A\ i\ (j,..m), A\ (i+1,..n)\ (0,..m)$ **then** $x = A\ i'\ j'$ **else** $i'=n$ **fi**

$P \Leftarrow i := 0.\ i \le n \Rightarrow Q$



$i \le n \Rightarrow Q \Leftarrow$ **if** $i=n$ **then** ok **else** $i<n \Rightarrow Q$ **fi**

$i<n \Rightarrow Q \Leftarrow j := 0.\ i<n \wedge j \le m \Rightarrow R$

$i<n \wedge j \le m \Rightarrow R \Leftarrow$ **if** $j=m$ **then** $i := i+1.\ i \le n \Rightarrow Q$ **else** $i<n \wedge j<m \Rightarrow R$ **fi**

$i<n \wedge j<m \Rightarrow R \Leftarrow$ **if** $A\ i\ j = x$ **then** ok

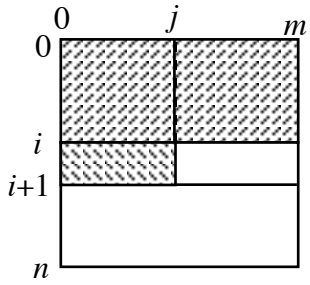
Two-Dimensional Search

$P \;=\; \textbf{if } x\text{: } A\ (0,..n)\ (0,..m) \textbf{ then } x = A\ i'\ j' \textbf{ else } i'=n \textbf{ fi}$

$Q \;=\; \textbf{if } x\text{: } A\ (i,..n)\ (0,..m) \textbf{ then } x = A\ i'\ j' \textbf{ else } i'=n \textbf{ fi}$

$R \;=\; \textbf{if } x\text{: } A\ i\ (j,..m), A\ (i{+}1,..n)\ (0,..m) \textbf{ then } x = A\ i'\ j' \textbf{ else } i'=n \textbf{ fi}$

$P \;\Leftarrow\; i{:=}\,0.\ \ i{\le}n \Rightarrow Q$



$i{\le}n \Rightarrow Q \;\Leftarrow\; \textbf{if } i{=}n \textbf{ then } ok \textbf{ else } i{<}n \Rightarrow Q \textbf{ fi}$

$i{<}n \Rightarrow Q \;\Leftarrow\; j{:=}\,0.\ \ i{<}n \wedge j{\le}m \Rightarrow R$

$i{<}n \wedge j{\le}m \Rightarrow R \;\Leftarrow\; \textbf{if } j{=}m \textbf{ then } i{:=}\,i{+}1.\ \ i{\le}n \Rightarrow Q \textbf{ else } i{<}n \wedge j{<}m \Rightarrow R \textbf{ fi}$

$i{<}n \wedge j{<}m \Rightarrow R \;\Leftarrow\; \textbf{if } A\ i\ j = x \textbf{ then } ok \textbf{ else } j{:=}\,j{+}1.\ \ i{<}n \wedge j{\le}m \Rightarrow R \textbf{ fi}$

Two-Dimensional Search

$t' \le t + n \times m \;\Leftarrow\; i := 0.\; i \le n \;\Rightarrow\; t' \le t + (n-i) \times m$

$i \le n \;\Rightarrow\; t' \le t + (n-i) \times m \;\Leftarrow\; \textbf{if } i = n \textbf{ then } ok \textbf{ else } i < n \;\Rightarrow\; t' \le t + (n-i) \times m \textbf{ fi}$

$i < n \;\Rightarrow\; t' \le t + (n-i) \times m \;\Leftarrow\; j := 0.\; i < n \wedge j \le m \;\Rightarrow\; t' \le t + (n-i) \times m - j$

$i < n \wedge j \le m \;\Rightarrow\; t' \le t + (n-i) \times m - j \;\Leftarrow$

$t := t+1.$

if $j = m$ **then** $i := i+1.\; i \le n \;\Rightarrow\; t' \le t + (n-i) \times m$

else $i < n \wedge j < m \;\Rightarrow\; t' \le t + (n-i) \times m - j$ **fi**

$i < n \wedge j < m \;\Rightarrow\; t' \le t + (n-i) \times m - j \;\Leftarrow$

if $A\ i\ j = x$ **then** ok **else** $j := j+1.\; i < n \wedge j \le m \;\Rightarrow\; t' \le t + (n-i) \times m - j$ **fi**

Two-Dimensional Search

$t' \le t + n \times m \;\Leftarrow\; i := 0.\ i \le n \;\Rightarrow\; t' \le t + (n-i) \times m$

$i \le n \;\Rightarrow\; t' \le t + (n-i) \times m \;\Leftarrow\; \textbf{if } i = n \textbf{ then } ok \textbf{ else } i < n \;\Rightarrow\; t' \le t + (n-i) \times m \textbf{ fi}$

$i < n \;\Rightarrow\; t' \le t + (n-i) \times m \;\Leftarrow\; j := 0.\ i < n \wedge j \le m \;\Rightarrow\; t' \le t + (n-i) \times m - j$

$i < n \wedge j \le m \;\Rightarrow\; t' \le t + (n-i) \times m - j \;\Leftarrow$

$t := t+1.$ 

$\textbf{if } j = m \textbf{ then } i := i+1.\ i \le n \;\Rightarrow\; t' \le t + (n-i) \times m$

$\textbf{else } i < n \wedge j < m \;\Rightarrow\; t' \le t + (n-i) \times m - j \textbf{ fi}$

$i < n \wedge j < m \;\Rightarrow\; t' \le t + (n-i) \times m - j \;\Leftarrow$

$\textbf{if } A\ i\ j = x \textbf{ then } ok \textbf{ else } j := j+1.\ i < n \wedge j \le m \;\Rightarrow\; t' \le t + (n-i) \times m - j \textbf{ fi}$

Two-Dimensional Search

$t' \le t + n \times m \;\Leftarrow\; i := 0.\; i \le n \;\Rightarrow\; t' \le t + (n-i) \times m$

$i \le n \;\Rightarrow\; t' \le t + (n-i) \times m \;\Leftarrow\;$ **if** $i = n$ **then** *ok* **else** $i < n \;\Rightarrow\; t' \le t + (n-i) \times m$ **fi**

$i < n \;\Rightarrow\; t' \le t + (n-i) \times m \;\Leftarrow\; j := 0.\; i < n \wedge j \le m \;\Rightarrow\; t' \le t + (n-i) \times m - j$

$i < n \wedge j \le m \;\Rightarrow\; t' \le t + (n-i) \times m - j \;\Leftarrow$

$t := t+1.$

if $j = m$ **then** $i := i+1.\; i \le n \;\Rightarrow\; t' \le t + (n-i) \times m$

else $i < n \wedge j < m \;\Rightarrow\; t' \le t + (n-i) \times m - j$ **fi**

$i < n \wedge j < m \;\Rightarrow\; t' \le t + (n-i) \times m - j \;\Leftarrow$

if $A\ i\ j = x$ **then** *ok* **else** $j := j+1.\; i < n \wedge j \le m \;\Rightarrow\; t' \le t + (n-i) \times m - j$ **fi**

Two-Dimensional Search

$t' \le t + n{\times}m \;\Leftarrow\; i{:=}\,0.\;\; i{\le}n \;\Rightarrow\; t' \le t + (n{-}i){\times}m$

$i{\le}n \;\Rightarrow\; t' \le t + (n{-}i){\times}m \;\Leftarrow\;$ **if** $i{=}n$ **then** *ok* **else** $i{<}n \;\Rightarrow\; t' \le t + (n{-}i){\times}m$ **fi**

$i{<}n \;\Rightarrow\; t' \le t + (n{-}i){\times}m \;\Leftarrow\; j{:=}\,0.\;\; i{<}n \wedge j{\le}m \;\Rightarrow\; t' \le t + (n{-}i){\times}m - j$

$i{<}n \wedge j{\le}m \;\Rightarrow\; t' \le t + (n{-}i){\times}m - j \;\Leftarrow$

$t{:=}\,t{+}1.$

if $j{=}m$ **then** $i{:=}\,i{+}1.\;\; i{\le}n \;\Rightarrow\; t' \le t + (n{-}i){\times}m$

else $i{<}n \wedge j{<}m \;\Rightarrow\; t' \le t + (n{-}i){\times}m - j$ **fi**

$i{<}n \wedge j{<}m \;\Rightarrow\; t' \le t + (n{-}i){\times}m - j \;\Leftarrow$

if $A\ i\ j = x$ **then** *ok* **else** $j{:=}\,j{+}1.\;\; i{<}n \wedge j{\le}m \;\Rightarrow\; t' \le t + (n{-}i){\times}m - j$ **fi**

Two-Dimensional Search

$t' \le t + n \times m \;\Leftarrow\; i := 0.\; i \le n \;\Rightarrow\; t' \le t + (n-i) \times m$

$i \le n \;\Rightarrow\; t' \le t + (n-i) \times m \;\Leftarrow\; \textbf{if } i = n \textbf{ then } ok \textbf{ else } i < n \;\Rightarrow\; t' \le t + (n-i) \times m \textbf{ fi}$

$i < n \;\Rightarrow\; t' \le t + (n-i) \times m \;\Leftarrow\; j := 0.\; i < n \wedge j \le m \;\Rightarrow\; t' \le t + (n-i) \times m - j$

$i < n \wedge j \le m \;\Rightarrow\; t' \le t + (n-i) \times m - j \;\Leftarrow$

$t := t+1.$

$\textbf{if } j = m \textbf{ then } i := i+1.\; i \le n \;\Rightarrow\; t' \le t + (n-i) \times m$

$\textbf{else } i < n \wedge j < m \;\Rightarrow\; t' \le t + (n-i) \times m - j \textbf{ fi}$

$i < n \wedge j < m \;\Rightarrow\; t' \le t + (n-i) \times m - j \;\Leftarrow$

$\textbf{if } A\ i\ j = x \textbf{ then } ok \textbf{ else } j := j+1.\; i < n \wedge j \le m \;\Rightarrow\; t' \le t + (n-i) \times m - j \textbf{ fi}$

Two-Dimensional Search

$P \Leftarrow i := 0.\ L0$

$L0 \Leftarrow$ **if** $i = n$ **then** ok **else** $j := 0.\ L1$ **fi**

$L1 \Leftarrow$ **if** $j = m$ **then** $i := i+1.\ L0$

else if $A\ i\ j = x$ **then** ok

else $j := j+1.\ L1$ **fi fi**

Two-Dimensional Search

$P \Leftarrow i := 0.\ L0$

$L0 \Leftarrow$ **if** $i = n$ **then** ok **else** $j := 0.\ L1$ **fi**

$L1 \Leftarrow$ **if** $j = m$ **then** $i := i+1.\ L0$

else if $A\ i\ j = x$ **then** ok

else $j := j+1.\ L1$ **fi fi**

in C:

```
P:   i = 0;
L0:  if (i==n);
     else {      j = 0;
          L1:  if (j==m) {i = i+1;  goto L0;}
               else if (A[i][j]==x);
                    else {j = j+1;  goto L1;}}
```

For Loop

for i:= m;..n **do** P **od**

For Loop

for i:= m;..n **do** P **od**

i is a fresh name (a local constant)

For Loop

for i:= m;..n **do** P **od**

i is a fresh name (a local constant)

m and n are integer expressions such that $m \leq n$

For Loop

for i:= m;..n **do** P **od**

i is a fresh name (a local constant)

m and n are integer expressions such that $m \leq n$

the number of iterations is $n-m$

For Loop

for i:= m;..n **do** P **od**

i is a fresh name (a local constant)

m and n are integer expressions such that $m \leq n$

the number of iterations is $n–m$

P is a specification

For Loop

for i:= m;..n **do** P **od**

For Loop

Let $F\ i$ describe the computation from from index i to the end.

for i:= m;..n **do** P **od**

For Loop

Let $F\ i$ describe the computation from from index i to the end.

for i:= m;..n **do** P **od**

$$F\ i \Leftarrow i{:}\ m,..n \wedge (P.\ F(i{+}1))$$

For Loop

Let $F\ i$ describe the computation from from index i to the end.

for i:= m;..n **do** P **od**

$$F\ i \Leftarrow i{:}\ m,..n \wedge (P.\ F(i{+}1))$$

$$F\ n \Leftarrow ok$$

For Loop

Let $F\ i$ describe the computation from from index i to the end.

$$F\ m \quad \Leftarrow \quad \textbf{for}\ i := m;..n\ \textbf{do}\ P\ \textbf{od}$$

means

$$F\ i \quad \Leftarrow \quad i{:}\ m,..n \wedge (P.\ \ F(i{+}1))$$

$$F\ n \quad \Leftarrow \quad ok$$

For Loop

example: $x'=2^n$

For Loop

example: $x' = 2^n$

define: $F\ i\ =\ x' = x \times 2^{n-i}$

For Loop

example: $x'=2^n$

define: $F\ i \ = \ x' = x \times 2^{n-i}$

refine: $x'=2^n \ \Leftarrow \ x := 1.\ F\ 0$

For Loop

example: $x'=2^n$

define: $F\ i \ = \ x' = x\times 2^{n-i}$

refine: $x'=2^n \ \Leftarrow \ x:=1.\ F\ 0$

proof: $x:=1.\ F\ 0$ expand $F\ 0$

$=$ $x:=1.\ x' = x\times 2^{n-0}$ Substitution Law and simplify

$=$ $x'=2^n$

For Loop

example: $x'=2^n$

define: $F\ i \ = \ x' = x \times 2^{n-i}$

refine: $x'=2^n \Leftarrow x := 1.\ F\ 0$

$F\ 0 \Leftarrow$ **for** $i := 0;..n$ **do** $x := 2 \times x$ **od**

For Loop

example: $x'=2^n$

define: $F\ i \ = \ x' = x \times 2^{n-i}$

refine: $x'=2^n \ \Leftarrow \ x := 1.\ F\ 0$

$F\ 0 \ \Leftarrow \ \textbf{for}\ i := 0;..n\ \textbf{do}\ x := 2 \times x\ \textbf{od}$

proof: $F\ i \ \Leftarrow \ i : m,..n \wedge (P.\ F(i+1))$

$F\ n \ \Leftarrow \ ok$

For Loop

example: $x'=2^n$

define: $F\ i \ = \ x' = x \times 2^{n-i}$

refine: $x'=2^n \ \Leftarrow \ x := 1.\ F\ 0$

$F\ 0 \ \Leftarrow \ \textbf{for}\ i := 0;..n\ \textbf{do}\ x := 2 \times x\ \textbf{od}$

proof: $F\ i \ \Leftarrow \ i : m,..n \wedge (P.\ F(i+1))$

For Loop

example: $x'=2^n$

define: $F\ i \ = \ x' = x\times 2^{n-i}$

refine: $x'=2^n \ \Leftarrow \ x:= 1.\ F\ 0$

$F\ 0 \ \Leftarrow$ **for** $i:= 0;..n$ **do** $x:= 2\times x$ **od**

proof: $F\ i \ \Leftarrow \ i: m,..n \wedge (P.\ F(i+1))$

	$i: m,..n \wedge (P.\ F(i+1))$	replace P and $F(i+1)$
$=$	$i: m,..n \wedge (x:= 2\times x.\ x' = x\times 2^{n-(i+1)})$	substitution law
$=$	$i: m,..n \wedge x' = 2\times x\times 2^{n-(i+1)}$	simplify
$=$	$i: m,..n \wedge x' = x\times 2^{n-i}$	specialize
$\Rightarrow$	$F\ i$	

For Loop

example: $x'=2^n$

define: $F\ i \ = \ x' = x \times 2^{n-i}$

refine: $x'=2^n \ \Leftarrow \ x := 1.\ F\ 0$

$F\ 0 \ \Leftarrow$ **for** $i := 0;..n$ **do** $x := 2 \times x$ **od**

proof: $F\ i \ \Leftarrow \ i : m,..n \wedge (P.\ F(i+1))$

For Loop

example: $x'=2^n$

define: $F\ i \ = \ x' = x \times 2^{n-i}$

refine: $x'=2^n \ \Leftarrow \ x := 1.\ F\ 0$

$F\ 0 \ \Leftarrow \ \textbf{for}\ i := 0;..n\ \textbf{do}\ x := 2 \times x\ \textbf{od}$

proof: $F\ i \ \Leftarrow \ i : m,..n \wedge (P.\ F(i+1))$

$F\ n \ \Leftarrow \ ok$

For Loop

example: $x'=2^n$

define: $F\ i \ = \ x' = x\times 2^{n-i}$

refine: $x'=2^n \ \Leftarrow \ x:= 1.\ F\ 0$

$F\ 0 \ \Leftarrow \ \textbf{for}\ i:= 0;..n\ \textbf{do}\ x:= 2\times x\ \textbf{od}$

proof: $F\ i \ \Leftarrow \ i: m,..n \wedge (P.\ F(i+1))$

$F\ n \ \Leftarrow \ ok$

$F\ n$

$=$ $x' = x\times 2^{n-n}$ simplify

$=$ $x'=x$

$=$ ok

For Loop

example: $t' = t + \Sigma j{:}\ m,..n{\cdot}\ G\ j$

For Loop

example: $t' = t + \Sigma j{:}\ m,..n{\cdot}\ G\ j$

define: $F\ i\ =\ t' = t + \Sigma j{:}\ i,..n{\cdot}\ G\ j$

For Loop

example: $t' = t + \Sigma j{:}\ m,..n \cdot G\ j$

define: $F\ i \ = \ t' = t + \Sigma j{:}\ i,..n \cdot G\ j$

refine: $F\ m \ \Leftarrow \ \textbf{for}\ i := m;..n\ \textbf{do}\ t' = t + G\ i\ \textbf{od}$

For Loop

example: $t' = t + \Sigma j{:}\ m,..n{\cdot}\ G\ j$

define: $F\ i \ = \ t' = t + \Sigma j{:}\ i,..n{\cdot}\ G\ j$

refine: $F\ m \ \Leftarrow \ \textbf{for}\ i{:=}\ m;..n\ \textbf{do}\ t' = t + G\ i\ \textbf{od}$

prove: $F\ i \ \Leftarrow \ i{:}\ m,..n \wedge (P.\ F(i{+}1))$

$F\ n \ \Leftarrow \ ok$

For Loop

example: $t' = t + \Sigma j{:}\ m,..n \cdot\ G\ j$

define: $F\ i \ =\ t' = t + \Sigma j{:}\ i,..n \cdot\ G\ j$

refine: $F\ m \ \Leftarrow\ \textbf{for}\ i := m;..n\ \textbf{do}\ t' = t + G\ i\ \textbf{od}$

prove: $F\ i \ \Leftarrow\ i{:}\ m,..n \wedge (P.\ F(i{+}1))$

$t' = t + \Sigma j{:}\ i,..n \cdot\ G\ j \ \Leftarrow\ i{:}\ m,..n \wedge (t' = t + G\ i.\ t' = t + \Sigma j{:}\ i{+}1,..n \cdot\ G\ j)$

$F\ n \ \Leftarrow\ ok$

For Loop

example: $t' = t + \Sigma j{:}\ m,..n \cdot\ G\ j$

define: $F\ i \ =\ t' = t + \Sigma j{:}\ i,..n \cdot\ G\ j$

refine: $F\ m \ \Leftarrow\ \textbf{for}\ i := m;..n\ \textbf{do}\ t' = t + G\ i\ \textbf{od}$

prove: $F\ i \ \Leftarrow\ i{:}\ m,..n \wedge (P.\ F(i+1))$

$t' = t + \Sigma j{:}\ i,..n \cdot\ G\ j \ \Leftarrow\ i{:}\ m,..n \wedge (t' = t + G\ i.\ t' = t + \Sigma j{:}\ i+1,..n \cdot\ G\ j)$

$F\ n \ \Leftarrow\ ok$

$t' = t + \Sigma j{:}\ n,..n \cdot\ G\ j \ \Leftarrow\ t'=t$

For Loop

example: $t' = t + \Sigma j{:}\ m,..n \cdot\ G\ j$

define: $F\ i \ =\ t' = t + \Sigma j{:}\ i,..n \cdot\ G\ j$

refine: $F\ m \ \Leftarrow\ \textbf{for}\ i := m;..n\ \textbf{do}\ t' = t + G\ i\ \textbf{od}$

prove: $F\ i \ \Leftarrow\ i{:}\ m,..n \wedge (P.\ F(i{+}1))$

$t' = t + \Sigma j{:}\ i,..n \cdot\ G\ j \ \Leftarrow\ i{:}\ m,..n \wedge (t' = t + G\ i.\ t' = t + \Sigma j{:}\ i{+}1,..n \cdot\ G\ j)$

$F\ n \ \Leftarrow\ ok$

$t' = t + \Sigma j{:}\ n,..n \cdot\ G\ j \ \Leftarrow\ t'{=}t$

If $G\ i = c$ (a constant) then

$t' = t + (n{-}m){\times}c \ \Leftarrow\ \textbf{for}\ i := m;..n\ \textbf{do}\ t' = t{+}c\ \textbf{od}$

For Loop

example: add 1 to each item in a list

For Loop

example: add 1 to each item in a list

$$\#L' = \#L \;\wedge\; \forall n: \Box L \cdot L'n = L\,n + 1$$

For Loop

example: add 1 to each item in a list

$$\#L' = \#L \;\wedge\; \forall n: \Box L \cdot L'n = L\,n + 1$$

define: $F\ i \;=\; \#L' = \#L$

$$\wedge\; (\forall n: (0,..i) \cdot L'n = L\,n)$$

$$\wedge\; (\forall n: i,..\#L \cdot L'n = L\,n + 1)$$

For Loop

example: add 1 to each item in a list

$$\#L' = \#L \;\wedge\; \forall n: \Box L \cdot L'n = L\,n + 1$$

define: $F\,i \;=\; \#L' = \#L$

$$\wedge\; (\forall n: (0,..i) \cdot L'n = L\,n)$$

$$\wedge\; (\forall n: i,..\#L \cdot L'n = L\,n + 1)$$

refine: $F\,0 \;\Leftarrow\;$ **for** $i := 0;..\#L$ **do** $L := i \rightarrow L\,i + 1 \mid L$ **od**

For Loop

example: add 1 to each item in a list

$$\#L'=\#L \;\wedge\; \forall n:\, \Box L\cdot\, L'n = L\,n + 1$$

define: $F\ i \;=\; \#L'=\#L$

$$\wedge\; (\forall n:\, (0,..i)\cdot\, L'n = L\,n)$$

$$\wedge\; (\forall n:\, i,..\#L\cdot\, L'n = L\,n + 1)$$

refine: $F\ 0 \;\Leftarrow\; \textbf{for}\ i:= 0;..\#L\ \textbf{do}\ L:= i{\rightarrow}L\ i + 1 \mid L\ \textbf{od}$

prove: $F\ i \;\Leftarrow\; i:\, 0,..\#L \wedge (L:= i{\rightarrow}L\ i + 1 \mid L.\ F(i{+}1))$

$F\ (\#L) \;\Leftarrow\; ok$

For Loop

special case: invariant

$$A\ m \Rightarrow A'n \quad \Leftarrow \quad \textbf{for}\ i := m;..n\ \textbf{do}\ i{:}\ m,..n \wedge A\ i \ \Rightarrow\ A'(i{+}1)\ \textbf{od}$$

For Loop

special case: invariant

$$A\ m \Rightarrow A'n \quad \Leftarrow \quad \textbf{for}\ i := m;..n\ \textbf{do}\ i{:}\ m,..n \wedge A\ i \ \Rightarrow\ A'(i{+}1)\ \textbf{od}$$

means

$$A\ i \Rightarrow A'n \quad \Leftarrow \quad i{:}\ m,..n \wedge (i{:}\ m,..n \ \wedge\ A\ i \Rightarrow A'(i{+}1).\ \ A(i{+}1) \Rightarrow A'n) \qquad \checkmark$$

$$A\ n \Rightarrow A'n \quad \Leftarrow \quad ok \qquad \checkmark$$

For Loop

special case: invariant

$$A\ m \Rightarrow A'n \quad \Leftarrow \quad \textbf{for}\ i := m;..n\ \textbf{do}\ i: m,..n \wedge A\ i \ \Rightarrow\ A'(i+1)\ \textbf{od}$$

example: $x' = 2^n$

For Loop

special case: invariant

$$A\ m \Rightarrow A'n \quad \Leftarrow \quad \textbf{for}\ i := m;..n\ \textbf{do}\ i: m,..n \wedge A\ i \ \Rightarrow\ A'(i+1)\ \textbf{od}$$

example: $x'=2^n$

invariant: $A\ i \ = \ x=2^i$

For Loop

special case: invariant

$$A\ m \Rightarrow A'n \quad \Leftarrow \quad \textbf{for}\ i := m;..n\ \textbf{do}\ i: m,..n \wedge A\ i \ \Rightarrow\ A'(i+1)\ \textbf{od}$$

example: $x'=2^n$

invariant: $A\ i \ = \ x=2^i$

refine: $x'=2^n \ \Leftarrow \ x := 1.\ A\ 0 \Rightarrow A'n$

For Loop

special case: invariant

$$A\ m \Rightarrow A'n \quad \Leftarrow \quad \textbf{for}\ i := m;..n\ \textbf{do}\ i: m,..n \wedge A\ i \ \Rightarrow\ A'(i+1)\ \textbf{od}$$

example: $x'=2^n$

invariant: $A\ i \ = \ x=2^i$

refine: $x'=2^n \quad \Leftarrow \quad x := 1.\ A\ 0 \Rightarrow A'n$

$$A\ 0 \Rightarrow A'n \quad \Leftarrow \quad \textbf{for}\ i := 0;..n\ \textbf{do}\ i: 0,..n \wedge A\ i \ \Rightarrow\ A'(i+1)\ \textbf{od}$$

For Loop

special case: invariant

$$A\ m \Rightarrow A'n \quad \Leftarrow \quad \textbf{for}\ i := m;..n\ \textbf{do}\ i: m,..n \wedge A\ i \ \Rightarrow\ A'(i+1)\ \textbf{od}$$

example: $x'=2^n$

invariant: $A\ i \ = \ x=2^i$

refine: $x'=2^n \quad \Leftarrow \quad x := 1.\ A\ 0 \Rightarrow A'n$

$$A\ 0 \Rightarrow A'n \quad \Leftarrow \quad \textbf{for}\ i := 0;..n\ \textbf{do}\ i: 0,..n \wedge A\ i \ \Rightarrow\ A'(i+1)\ \textbf{od}$$

$$i: 0,..n \ \wedge \ A\ i \Rightarrow A'(i+1) \quad \Leftarrow \quad x := 2{\times}x$$

Minimum Sum Segment

Given a list L of integers, possibly including negatives, write a program to find the minimum sum of any nonempty segment.

Minimum Sum Segment

Given a list L of integers, possibly including negatives, write a program to find the minimum sum of any nonempty segment.

$$[\ 4\ ;\ -2\ ;\ -8\ ;\ 7\ ;\ 3\ ;\ 0\ ;\ -1\ ]$$

Minimum Sum Segment

Given a list L of integers, possibly including negatives, write a program to find the minimum sum of any nonempty segment.

[4 ; –2 ; –8 ; 7 ; 3 ; 0 ; –1]

Minimum Sum Segment

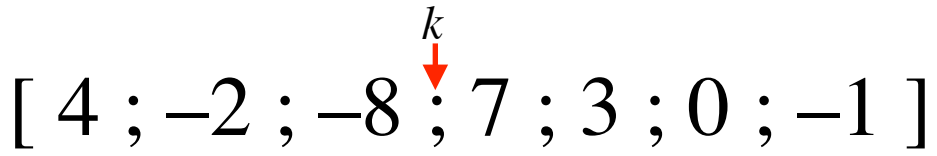
Given a list L of integers, possibly including negatives, write a program to find the minimum sum of any nonempty segment.

$$[\ 4\ ;\ -2\ ;\ -8\ ;\ 7\ ;\ 3\ ;\ 0\ ;\ -1\ ]$$

$$s' \ =\ \Downarrow i{:}\ 0,..\#L{\cdot}\ \Downarrow j{:}\ 1,..\#L{+}1{\cdot}\ \Sigma\ L\ [i;..j]$$

Minimum Sum Segment

Given a list L of integers, possibly including negatives, write a program to find the minimum sum of any nonempty segment.

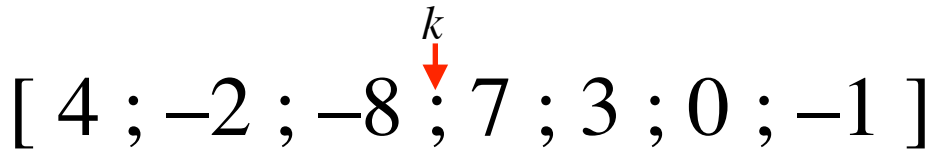


$$s' = \Downarrow i{:}\ 0,..\#L{\cdot}\ \Downarrow j{:}\ i{+}1,..\#L{+}1{\cdot}\ \Sigma\, L\,[i;..j]$$

$$A\,k = \qquad s = (\Downarrow i{:}\ 0,..k{\cdot}\ \Downarrow j{:}\ i{+}1,..k{+}1{\cdot}\ \Sigma\, L\,[i;..j])$$

Minimum Sum Segment

Given a list L of integers, possibly including negatives, write a program to find the minimum sum of any nonempty segment.

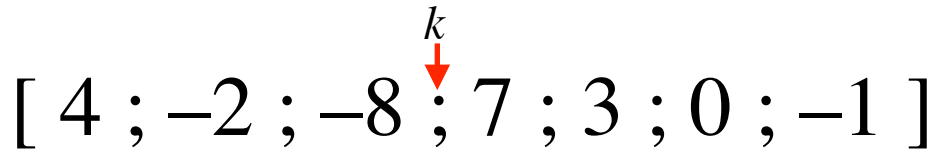


$$s' \;=\; \Downarrow i{:}\ 0,..\#L{\cdot}\ \Downarrow j{:}\ i{+}1,..\#L{+}1{\cdot}\ \Sigma\, L\,[i;..j] \quad \Longleftarrow \quad s{:=}\,\infty.\ A\,0 \Rightarrow A'(\#L)$$

$$A\,k \;=\; \qquad s \;=\; (\Downarrow i{:}\ 0,..k{\cdot}\ \Downarrow j{:}\ i{+}1,..k{+}1{\cdot}\ \Sigma\, L\,[i;..j])$$

Minimum Sum Segment

Given a list L of integers, possibly including negatives, write a program to find the minimum sum of any nonempty segment.



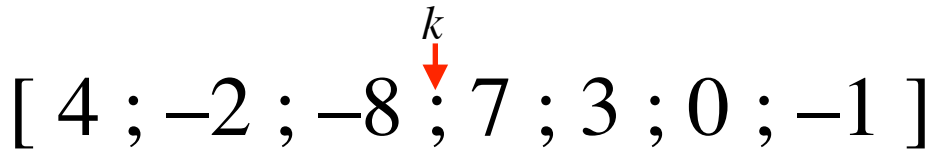
$$s' \;=\; \Downarrow i{:}\ 0,..\#L{\cdot}\ \Downarrow j{:}\ i{+}1,..\#L{+}1{\cdot}\ \Sigma\, L\,[i;..j] \quad \Leftarrow \quad s{:=}\,\infty.\ A\,0 \Rightarrow A'(\#L)$$

$$A\,0 \Rightarrow A'(\#L) \quad \Leftarrow \quad \textbf{for}\ k{:=}\,0;..\#L\ \textbf{do}\ k{:}\ 0,..\#L \wedge A\,k \ \Rightarrow\ A'(k{+}1)\ \textbf{od}$$

$$A\,k \;=\; \qquad s \;=\; (\Downarrow i{:}\ 0,..k{\cdot}\ \Downarrow j{:}\ i{+}1,..k{+}1{\cdot}\ \Sigma\, L\,[i;..j])$$

Minimum Sum Segment

Given a list L of integers, possibly including negatives, write a program to find the minimum sum of any nonempty segment.



$$s' \;=\; \Downarrow i{:}\ 0,..\#L{\cdot}\ \Downarrow j{:}\ i{+}1,..\#L{+}1{\cdot}\ \Sigma\, L\,[i;..j] \;\Leftarrow\; s{:=}\,\infty.\ A\,0 \Rightarrow A'(\#L)$$

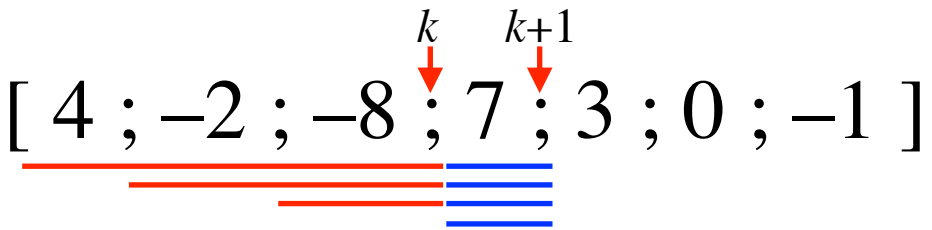
$$A\,0 \Rightarrow A'(\#L) \;\Leftarrow\; \textbf{for}\ k{:=}\,0;..\#L\ \textbf{do}\ k{:}\ 0,..\#L \wedge A\,k \;\Rightarrow\; A'(k{+}1)\ \textbf{od}$$

$$k{:}\ 0,..\#L \wedge A\,k \;\Rightarrow\; A'(k{+}1) \;\Leftarrow$$

$$A\,k \;=\; \qquad s \;=\; (\Downarrow i{:}\ 0,..k{\cdot}\ \Downarrow j{:}\ i{+}1,..k{+}1{\cdot}\ \Sigma\, L\,[i;..j])$$

Minimum Sum Segment

Given a list L of integers, possibly including negatives, write a program to find the minimum sum of any nonempty segment.



$$s' = \Downarrow i{:}\ 0,..\#L \cdot \Downarrow j{:}\ i{+}1,..\#L{+}1 \cdot \Sigma\, L\,[i;..j] \Leftarrow s := \infty.\ A\,0 \Rightarrow A'(\#L)$$

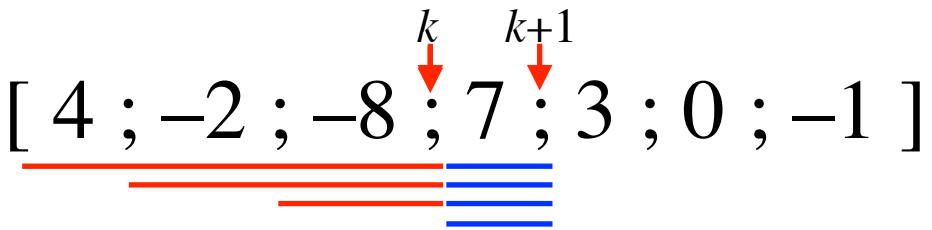
$$A\,0 \Rightarrow A'(\#L) \Leftarrow \textbf{for}\ k := 0;..\#L\ \textbf{do}\ k{:}\ 0,..\#L \wedge A\,k \Rightarrow A'(k{+}1)\ \textbf{od}$$

$$k{:}\ 0,..\#L \wedge A\,k \Rightarrow A'(k{+}1) \Leftarrow$$

$$A\,k = \quad s = (\Downarrow i{:}\ 0,..k \cdot \Downarrow j{:}\ i{+}1,..k{+}1 \cdot \Sigma\, L\,[i;..j])$$

Minimum Sum Segment

Given a list L of integers, possibly including negatives, write a program to find the minimum sum of any nonempty segment.



$$s' = \Downarrow i{:}\ 0,..\#L{\cdot}\ \Downarrow j{:}\ i{+}1,..\#L{+}1{\cdot}\ \Sigma\, L\,[i;..j] \Leftarrow s{:=}\ \infty.\ A\,0 \Rightarrow A'(\#L)$$

$$A\,0 \Rightarrow A'(\#L) \Leftarrow \textbf{for}\ k{:=}\ 0;..\#L\ \textbf{do}\ k{:}\ 0,..\#L \wedge A\,k \Rightarrow A'(k{+}1)\ \textbf{od}$$

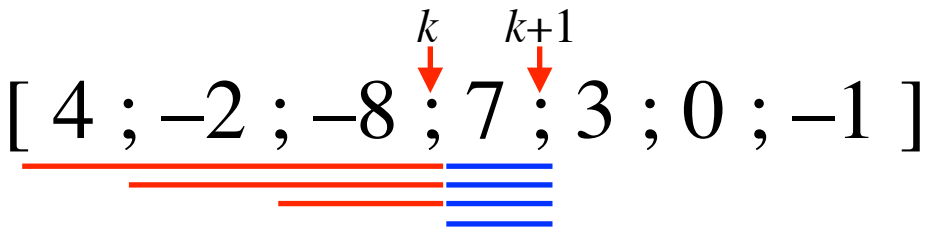
$$k{:}\ 0,..\#L \wedge A\,k \Rightarrow A'(k{+}1) \Leftarrow$$

$$A\,k = \quad s = (\Downarrow i{:}\ 0,..k{\cdot}\ \Downarrow j{:}\ i{+}1,..k{+}1{\cdot}\ \Sigma\, L\,[i;..j])$$

$$\wedge \quad c = (\Downarrow i{:}\ 0,..k{\cdot}\ \Sigma\, L\,[i;..k])$$

Minimum Sum Segment

Given a list L of integers, possibly including negatives, write a program to find the minimum sum of any nonempty segment.



$s' = \Downarrow i{:}\ 0,..\#L{\cdot}\ \Downarrow j{:}\ i{+}1,..\#L{+}1{\cdot}\ \Sigma\, L\,[i;..j] \Leftarrow s{:=}\infty.\ c{:=}\infty.\ A\,0 \Rightarrow A'(\#L)$

$A\,0 \Rightarrow A'(\#L) \Leftarrow$ **for** $k{:=}\,0;..\#L$ **do** $k{:}\ 0,..\#L \wedge A\,k \Rightarrow A'(k{+}1)$ **od**

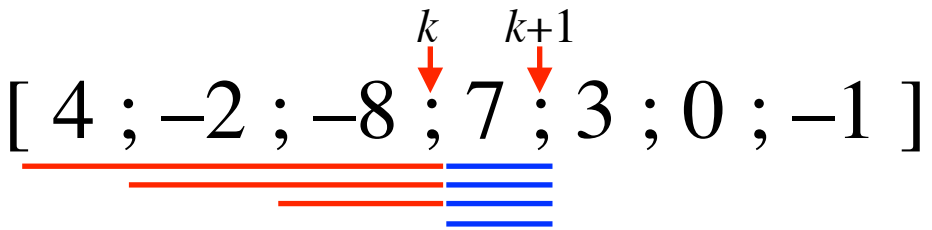
$k{:}\ 0,..\#L \wedge A\,k \Rightarrow A'(k{+}1) \Leftarrow$

$A\,k = \quad s = (\Downarrow i{:}\ 0,..k{\cdot}\ \Downarrow j{:}\ i{+}1,..k{+}1{\cdot}\ \Sigma\, L\,[i;..j])$

$\wedge \quad c = (\Downarrow i{:}\ 0,..k{\cdot}\ \Sigma\, L\,[i;..k])$

Minimum Sum Segment

Given a list L of integers, possibly including negatives, write a program to find the minimum sum of any nonempty segment.



$$s' \;=\; \Downarrow i{:}\,0,..\#L{\cdot}\ \Downarrow j{:}\,i{+}1,..\#L{+}1{\cdot}\ \Sigma\, L\,[i;..j] \;\Leftarrow\; s{:=}\,\infty.\ c{:=}\,\infty.\ A\,0 \Rightarrow A'(\#L)$$

$$A\,0 \Rightarrow A'(\#L) \;\Leftarrow\; \textbf{for}\ k{:=}\,0;..\#L\ \textbf{do}\ k{:}\,0,..\#L \wedge A\,k \;\Rightarrow\; A'(k{+}1)\ \textbf{od}$$

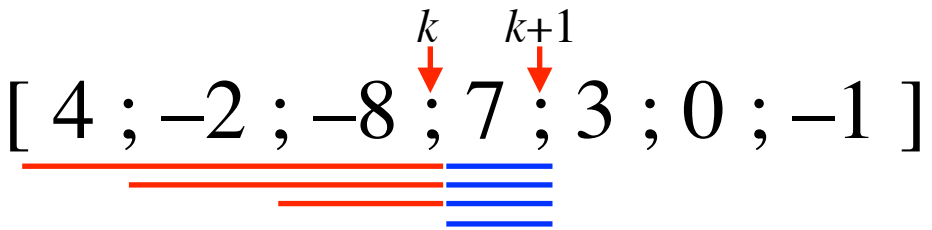
$$k{:}\,0,..\#L \wedge A\,k \;\Rightarrow\; A'(k{+}1) \;\Leftarrow\; c{:=}\,(c + L\,k)\downarrow(L\,k)$$

$$A\,k \;=\; s \;=\; (\Downarrow i{:}\,0,..k{\cdot}\ \Downarrow j{:}\,i{+}1,..k{+}1{\cdot}\ \Sigma\, L\,[i;..j])$$

$$\wedge\quad c \;=\; (\Downarrow i{:}\,0,..k{\cdot}\ \Sigma\, L\,[i;..k])$$

Minimum Sum Segment

Given a list L of integers, possibly including negatives, write a program to find the minimum sum of any nonempty segment.



$$s' \;=\; \Downarrow i{:}\ 0,..\#L{\cdot}\ \Downarrow j{:}\ i{+}1,..\#L{+}1{\cdot}\ \Sigma\, L\,[i;..j] \;\Leftarrow\; s{:=}\,\infty.\ c{:=}\,\infty.\ A\,0 \Rightarrow A'(\#L)$$

$$A\,0 \Rightarrow A'(\#L) \;\Leftarrow\; \textbf{for}\ k{:=}\,0;..\#L\ \textbf{do}\ k{:}\ 0,..\#L \wedge A\,k \;\Rightarrow\; A'(k{+}1)\ \textbf{od}$$

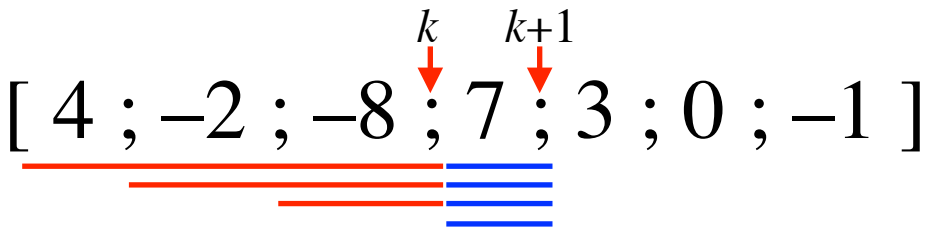
$$k{:}\ 0,..\#L \wedge A\,k \;\Rightarrow\; A'(k{+}1) \;\Leftarrow\; c{:=}\,c{\downarrow}0 + L\,k$$

$$A\,k \;=\; s \;=\; (\Downarrow i{:}\ 0,..k{\cdot}\ \Downarrow j{:}\ i{+}1,..k{+}1{\cdot}\ \Sigma\, L\,[i;..j])$$

$$\wedge\quad c \;=\; (\Downarrow i{:}\ 0,..k{\cdot}\ \Sigma\, L\,[i;..k])$$

Minimum Sum Segment

Given a list L of integers, possibly including negatives, write a program to find the minimum sum of any nonempty segment.



$$s' = \Downarrow i{:}\ 0,..\#L \cdot \Downarrow j{:}\ i{+}1,..\#L{+}1 \cdot \Sigma\, L\,[i;..j] \Leftarrow s := \infty.\ c := \infty.\ A\,0 \Rightarrow A'(\#L)$$

$$A\,0 \Rightarrow A'(\#L) \Leftarrow \textbf{for}\ k := 0;..\#L\ \textbf{do}\ k{:}\ 0,..\#L \wedge A\,k \Rightarrow A'(k{+}1)\ \textbf{od}$$

$$k{:}\ 0,..\#L \wedge A\,k \Rightarrow A'(k{+}1) \Leftarrow c := c \downarrow 0 + L\,k.\ s := s \downarrow c$$

$$A\,k = \quad s = (\Downarrow i{:}\ 0,..k \cdot \Downarrow j{:}\ i{+}1,..k{+}1 \cdot \Sigma\, L\,[i;..j])$$

$$\wedge \quad c = (\Downarrow i{:}\ 0,..k \cdot \Sigma\, L\,[i;..k])$$