CORRECTNESS PROOF OF AN IN-PLACE PERMUTATION

A. J. W. DUIJVESTIJN

Abstract.

The correctness of an in-place permutation algorithm is proved. The algorithm exchanges elements belonging to a permutation cycle. A suitable assertion is constructed from which the correctness can be deduced after completion of the algorithm.

An in-place rectangular matrix transposition algorithm is given as an example.

Key words and phrases: Proof of programs, algorithm, program correctness, theory of programming.

Introduction.

The in-place permutation problem deals with the rearrangement of the elements of a given vector $VEC[i]$, $i = 1(1)G$, $G \geq 1$, using an arbitrary permutation $f(i)$ of the integers $1, \ldots, G$.

The problem that has to be solved is: write an algorithm that permutes the elements of $VEC$ without using extra storage. That means if $VEC[i] = \alpha_i$ before the permutation then $VEC[i] = \alpha_{f(i)}$ after the permutation.

The solution of the permutation problem is given by the following algorithm:

```plaintext
procedure permute (VEC, f, G); value G; integer G; array VEC;
    integer procedure f;
    comment $f(x)$ is the index of $VEC$ where the element can be found that has to be moved to $VEC[x]$;
begin integer k, ko, kn, wr;
    for k := 1 step 1 until G do
        begin
            kn := f(k);
            for ko := kn while kn < k do kn := f(ko);
            if kn \neq k then begin comment exchange ($VEC[kn]$, $VEC[k]$);
```

A special case of the permutation problem arises in the transposition of a rectangular matrix without using extra storage [2, 3]. In case the matrix $A[i,j]$, $i=1(1)m$ and $j=1(1)n$ is columnwise mapped onto a vector $VEC[k]$, $k=1(1)m*n$, $G=m*n$, the function $f$ is defined as follows in ALGOL-60:

```algon
text
integer procedure f(x); value x; integer x;
comment $f(x)$ is the index of $VEC$ where the element can be found that has to be moved to $VEC[x]$;
begin integer w;
    w := $x-1 + n$;
    $f$ := $(x-w*n-1)*m + w + 1$
end
```

The algorithm for which a correctness proof is given in this note is essentially that of R. F. Windley [1].

**Correctness of the algorithm.**

It has to be proved that the algorithm performs the following:

$1$) $\forall i (1 \leq i \leq G \rightarrow VEC[i] = \alpha_{f(i)}$.

First we introduce a function $\psi_k(i)$ that is defined for $k \leq i \leq G$ with $1 \leq k \leq G$:

$\psi_k(i) =$ the first $f^{(s)}(i)$ with $f^{(s)}(i) \geq k$, $s \geq 1$.

The expression $f^s$ means: $f$ if $s=1$, otherwise $ff^{(s-1)}$. Consequently $\psi_k(i) = f^{(s)}(i) \geq k$, and $f^{(s)}(i) < k$ with $1 \leq t < s$, $s \geq 1$.

We prove certain properties of the function $\psi$.

**Property 1** is a property of the permutation $f$:

$\forall i (1 \leq i \leq G \rightarrow \exists e1 (1 \leq e1 \leq G \wedge i = f(e1))$.

and

$\forall i (1 \leq i \leq G \rightarrow \exists e2 (1 \leq e2 \leq G \wedge e2 = f(i))$.
Property 2.

(2) $\forall i (k \leq i \leq G \rightarrow \exists e1 (k \leq e1 \leq G \land i = \psi_k(e1)))$

and

(3) $\forall i (k \leq i \leq G \rightarrow \exists e2 (k \leq e2 \leq G \land e2 = \psi_k(i)))$.

Proof. Let $V_{k,G}$ be the set of integers: $V_{k,G} = \{i: k \leq i \leq G\}$, then property 2 says that $\psi_k(i)$ is a permutation on $V_{k,G}$.

Apparently property 2 is true for $k = 1$ since $\psi_1(i) = f(i)$ (property 1).

Assuming property 2 is true for $k$ (induction assumption), we prove that property 2 is also true for $k + 1$.

According to the induction assumption there exists exactly one element $e_1 \in V_{k,G}$ such that $k = \psi_k(e_1)$ and exactly one element $e_2 \in V_{k,G}$ such that $e_2 = \psi_k(k)$. (A direct consequence of (2) and (3)).

We consider two cases:

Case 1. $e_1 > k$. Then clearly $e_2 > k$. Consider the sets $V^*_{k+1,G} = V_{k+1,G} \setminus e_1$ and $V^{**}_{k+1,G} = V_{k+1,G} \setminus e_2$.

According to the induction assumption we have:

(4) $\forall a (a \in V^*_{k+1,G} \rightarrow \exists b (b \in V^{**}_{k+1,G} \land b = \psi_k(a)))$

and

(5) $\forall b (b \in V^{**}_{k+1,G} \rightarrow \exists a (a \in V^*_{k+1,G} \land b = \psi_k(a)))$

Since $b = \psi_k(a) > k$ it follows from the definition of $\psi$:

$b = f^s(a), s \geq 1$ and $f^t(a) < k$ for $1 \leq t < s$

that

$b = f^s(a) \geq k + 1, s \geq 1$ and $f^t(a) < k < k + 1$ for $1 \leq t < s$;

(6) we conclude $b = \psi_{k+1}(a)$.

Hence it follows that:

(7) $\forall a (a \in V^*_{k+1,G} \rightarrow \psi_k(a) = \psi_{k+1}(a))$.

Furthermore we prove $e_2 = \psi_{k+1}(e_1)$.

From the definition of $\psi$ and the induction assumption it follows:

$\exists s (s \geq 1 \land k = f^s(e_1) \land \forall t (1 \leq t < s \rightarrow f^t(e_1) < k))$

and

$\exists r (r \geq 1 \land e_2 = f^r(k) \land \forall u (1 \leq u < r \rightarrow f^u(k) < k))$. 
Clearly \( e_2 = f^{s+r}(e_1) \geq k + 1 \), \( s+r \geq 2 \) and \( f^p(e_1) < k + 1 \) with \( 1 \leq p \leq s+r \). Hence

\[ (8) \quad e_2 = \psi_{k+1}(e_1). \]

Using (4), (5), (6), (7) and (8) we conclude:

\[ (9) \quad \forall a (a \in V_{k+1,G} \rightarrow \exists b (b \in V_{k+1,G} \land b = \psi_{k+1}(a)) ) \]

and

\[ (10) \quad \forall b (b \in V_{k+1,G} \rightarrow \exists a (a \in V_{k+1,G} \land b = \psi_{k+1}(a)) ). \]

**Case 2.** \( e_1 = k \). In this case \( e_1 = e_2 = k \). Furthermore \( V_{k+1,G}^* = V_{k+1,G}^* = V_{k+1,G} \) and according to (4), (5), (6) and (7) we have:

\[ (11) \quad \forall a (a \in V_{k+1,G} \rightarrow \exists b (b \in V_{k+1,G} \land b = \psi_{k+1}(a)) ) \]

and

\[ (12) \quad \forall b (b \in V_{k+1,G} \rightarrow \exists a (a \in V_{k+1,G} \land b = \psi_{k+1}(a)) ). \]

Using (9), (10), (11) and (12) then by induction property 2 is true for all \( k \leq G \).

We can now formulate property 3 and 4.

**Property 3.** If \( \psi_k(e_1) = k \) and \( \psi_k(k) = e_2 \), while \( e_1 > k \) and \( e_2 > k \) then according to (8) \( e_2 = \psi_{k+1}(e_1) \).

**Remark.** In case \( e_1 = e_2 = k \), \( \psi_{k+1}(e_1) \) is not defined.

**Property 4.** \( \psi_k(i) = \psi_{k+1}(i) \) for all \( i > k \) except that \( i \) for which \( \psi_k(i) = k \) (see (6) and (7)).

We prove the truth of the assertion \( E_1 \land E_2 \) on a certain label in the program. The definition of \( E_1 \) and \( E_2 \) is as follows:

\[ (E_1) \quad \forall i (1 \leq i < k \rightarrow VEC[i] = \alpha_{f(i)}) \]

and

\[ (E_2) \quad \forall i (k \leq i \leq G \rightarrow VEC[\psi_k(i)] = \alpha_{f(i)}). \]

The structure of the program is:

for \( k := 1 \) step 1 until \( G \) do
begin ... end;

This program is equivalent with the program:

\( k := 1; \)
\( L: \text{if } k > G \text{ then goto } Exh; \)
begin ... end;
\( k := k+1; \text{ goto } L; \ Exh: \)
We prove $\vdash E_1 \land E_2$ on label $L$ for all $k$, $1 \leq k \leq G + 1$.

**Proof.** If $k = 1$ then $\vdash E_1 \land E_2$ since $E_1$ is true ($1 \leq i < 1$ is false so the implication is true) and since $\psi_1(i) = f(i)$ the assertion $E_2$ reads:

$$\forall i (1 \leq i \leq G \rightarrow VEC[\psi_1(i)] = VEC[f(i)] = \alpha_{f(k)}(i))$$

which is clearly true.

Assuming that $\vdash E_1 \land E_2$ on $L$ for a certain $k = k_1$ ($1 \leq k_1 \leq G$) the following statements are executed before returning to label $L$.

$$L: \; kn = f(k);$$
$$\text{for } k_0 := kn \text{ while } kn < k \text{ do } kn := f(k_0);$$
$$L_1: \; \text{if } kn = k \text{ then exchange } (VEC[kn], VEC[k]);$$
$$L_2: \; k := k + 1; \; \text{goto } L;$$

The labels $L_1$ and $L_2$ are merely introduced as a reference. At label $L_1$ we have $kn = \psi_k(k)$. Consequently $kn \geq k$. In case $kn = k$, $VEC[kn]$ and $VEC[k]$ are exchanged. Since $\vdash E_1 \land E_2$ on $L$ it follows $\vdash E_1 \land E_2$ on $L_1$.

We consider two cases:

**Case 1.** $kn > k$. From $\vdash E_1 \land E_2$ on $L_1$ we have $VEC[\psi_k(k)] = VEC[kn] = \alpha_{f(k)}$. After exchanging $VEC[kn]$ and $VEC[k]$, $VEC[k] = \alpha_{f(k)}$ at label $L_2$.

Therefore the following assertion holds at $L_2$:

$$\forall i (1 \leq i \leq k \rightarrow VEC[i] = \alpha_{f(k)}).$$

Hence

$$\forall i (1 \leq i < k + 1 \rightarrow VEC[i] = \alpha_{f(k)}).$$

Finally $\vdash E_1$ at $L$ for $k = k_1 + 1$.

Since $kn = \psi_k(k) > k$ then according to property 2 there exist elements $e_1$ and $e_2$, $e_1 > k$, $e_2 > k$ such that:

$$e_2 = \psi_k(k) \; \text{ and } \; k = \psi_k(e_1)$$

and according to property 3:

$$e_2 = \psi_{k+1}(e_1).$$

Apparently $e_2 = kn$.

At label $L_1$ we have

$$VEC[k] = VEC[\psi_k(e_1)] = \alpha_{f(e_1)}, \; \text{since } e_1 > k.$$

At label $L_2$

$$VEC[kn] = VEC[\psi_k(k)] = VEC[e_2] = \alpha_{f(e_1)}.$$
Using property 3 at L2,

\[ VEC[e2] = VEC[\gamma_{k+1}(e1)] = \alpha_{f(e1)}. \]

From \( \vdash E2 \) we deduce at label L1:

\[ \forall i(k < i \leq G \wedge i + e1 \rightarrow VEC[\gamma_k(i)] = \alpha_{f,0}). \]

Using property 4 we get at L2

\[ \forall i(k < i \leq G \wedge i + e1 \rightarrow VEC[\gamma_k(i)] = VEC[\gamma_{k+1}(i)] = \alpha_{f,0}). \]

Combining (13), (14) and (15) we have at L2:

\[ \forall i(k + 1 \leq i \leq G \rightarrow VEC[\gamma_{k+1}(i)] = \alpha_{f,0}). \]

Passing from label L2 to label L \( k := k + 1 \). Hence \( \vdash E2 \) at L for \( k = k1 + 1 \).

**CASE 2.** \( kn = k \). In this case \( \gamma_k(k) = k \) and no exchange takes place.

From \( \vdash E2 \) at L and at L1 and L2 we deduce:

\[ VEC[\gamma_k(k)] = VEC[k] = \alpha_{f,0}). \]

Combining (16) with \( \vdash E1 \) we get at L2

\[ \forall i(1 \leq i \leq k \rightarrow VEC[i] = \alpha_{f,0}). \]

Hence

\[ \forall i(1 < i < k + 1 \rightarrow VEC[i] = \alpha_{f,0}) \text{ at L2}. \]

From \( \vdash E2 \) and since there does not exist an element \( e1 > k \) with \( \gamma_k(k) = e1 \), and from property 4 it follows that:

\[ \forall i(k + 1 \leq i \leq G \rightarrow VEC[\gamma_{k+1}(i)] = \alpha_{f,0}) \text{ at L2}. \]

Combining (18) and (19) at L2 and using the assignation \( k := k + 1 \) in passing from label L2 to label L we get: \( \vdash E1 \wedge E2 \) at L for \( k = k1 + 1 \). By induction it follows that: \( \vdash E1 \wedge E2 \) at L for all \( k = 1(1)G + 1 \). Moreover \( \vdash E1 \wedge E2 \wedge k = G + 1 \) at label Exh. In that case \( E1 \) confirms the truth of (1).

**Remark 1.** The algorithm can be changed slightly in case of a matrix transposition. It suffices that the for loop runs from \( k = 2(1)G - 2 \), because \( A[1,1] \) and \( A[m,n] \) do not move. In case all elements have been moved up to \( G - 2 \) then the \( G - 1 \)th element is in place. Even in the general case the range of the for loop can be taken \( k = 1(1)G - 1 \).
Remark 2. Looking at the invariant $\rightarrow E_1 \wedge E_2$ we observe that $E_2$ describes the initial state of the program for $k = 1$. $E_1$ is then "empty". $E_1$ describes the final state for $k = G + 1$. $E_2$ is then "empty".

Acknowledgements.

The author wishes to thank F. Göbel and J. Engelfriet for their discussions and remarks.

REFERENCES