BIT 12 (1972), 318–324

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 $VEC[i]$, $i=1(1)G$, $G \geqq 1$, 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:

```
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 ≠ k then begin comment exchange (VEC[kn], VEC[k]);
```

Received Feb. 10, 1972. Revised April 11, 1972.

```
                    wr := VEC[kn]; VEC[kn] := VEC[k];
                    VEC[k] := wr
                  end
  end
end
```

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:

```
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:

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

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

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

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

We prove certain properties of the function ψ.

Property 1 is a property of the permutation f:

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

and

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

PROPERTY 2.

$$\forall i\big(k \leqq i \leqq G \rightarrow \exists e1(k \leqq e1 \leqq G \wedge i = \psi_k(e1))\big) \tag{2}$$

and

$$\forall i\big(k \leqq i \leqq G \rightarrow \exists e2(k \leqq e2 \leqq G \wedge e2 = \psi_k(i))\big)\,. \tag{3}$$

PROOF. Let $V_{k,G}$ be the set of integers: $V_{k,G} = \{i: k \leqq i \leqq G\}$, 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 $e1 \in V_{k,G}$ such that $k = \psi_k(e1)$ and exactly one element $e2 \in V_{k,G}$ such that $e2 = \psi_k(k)$. (A direct consequence of (2) and (3)).

We consider two cases:

CASE 1. $e1 > k$. Then clearly $e2 > k$. Consider the sets $V^*_{k+1,G} = V_{k+1,G} \setminus e1$ and $V^{**}_{k+1,G} = V_{k+1,G} \setminus e2$.

According to the induction assumption we have:

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

and

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

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

$$b = f^{(s)}(a),\ s \geqq 1 \text{ and } f^t(a) < k \text{ for } 1 \leqq t < s$$

that

$$b = f^s(a) \geqq k+1,\ s \geqq 1 \text{ and } f^{(t)}(a) < k < k+1 \text{ for } 1 \leqq t < s;$$

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

Hence it follows that:

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

Furthermore we prove $e2 = \psi_{k+1}(e1)$.

From the definition of ψ and the induction assumption it follows:

$$\exists s\big(s \geqq 1 \wedge k = f^s(e1) \wedge \forall t(1 \leqq t < s \rightarrow f^t(e1) < k)\big)$$

and

$$\exists r\big(r \geqq 1 \wedge e2 = f^r(k) \wedge \forall u(1 \leqq u < r \rightarrow f^u(k) < k)\big)\,.$$

Clearly $e2 = f^{s+r}(e1) \geqq k+1$, $s+r \geqq 2$ and $f^p(e1) < k+1$ with $1 \leqq p < s+r$. Hence

(8) $$e2 = \psi_{k+1}(e1)\,.$$

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

(9) $$\forall a\big(a \in V_{k+1,G} \to \exists b(b \in V_{k+1,G} \wedge b = \psi_{k+1}(a))\big)$$

and

(10) $$\forall b\big(b \in V_{k+1,G} \to \exists a(a \in V_{k+1,G} \wedge b = \psi_{k+1}(a))\big)\,.$$

Case 2. $e1 = k$. In this case $e1 = e2 = k$. Furthermore $V^{*}_{k+1,G} = V^{**}_{k+1,G} = V_{k+1,G}$ and according to (4), (5), (6) and (7) we have:

(11) $$\forall a\big(a \in V_{k+1,G} \to \exists b(b \in V_{k+1,G} \wedge b = \psi_{k+1}(a))\big)$$

and

(12) $$\forall b\big(b \in V_{k+1,G} \to \exists a(a \in V_{k+1,G} \wedge b = \psi_{k+1}(a))\big)\,.$$

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

We can now formulate property 3 and 4.

Property 3. If $\psi_k(e1) = k$ and $\psi_k(k) = e2$, while $e1 > k$ and $e2 > k$ then according to (8) $e2 = \psi_{k+1}(e1)$.

Remark. In case $e1 = e2 = k$, $\psi_{k+1}(e1)$ 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 $E1 \wedge E2$ on a certain label in the program. The definition of $E1$ and $E2$ is as follows:

($E1$) $$\forall i(1 \leqq i < k \to VEC[i] = \alpha_{f(i)})$$

and

($E2$) $$\forall i(k \leqq i \leqq G \to VEC[\psi_k(i)] = \alpha_{f(i)})\,.$$

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:  if k > G then goto Exh;
    begin ... end;
    k := k+1; goto L; Exh:
```

We prove $\vdash E1 \wedge E2$ on label L for all k, $1 \leqq k \leqq G+1$.

PROOF. If $k=1$ then $\vdash E1 \wedge E2$ since $E1$ is true ($1 \leqq i < 1$ is false so the implication is true) and since $\psi_1(i) = f(i)$ the assertion $E2$ reads:

$$\forall i(1 \leqq i \leqq G \rightarrow VEC[\psi_1(i)] = VEC[f(i)] = \alpha_{f(i)}) \text{ which is clearly true .}$$

Assuming that $\vdash E1 \wedge E2$ on L for a certain $k = k_1$ ($1 \leqq k_1 \leqq G$) the following statements are executed before returning to label L.

```
L: kn = f(k);
   for ko := kn while kn < k do kn := f(ko);
L1: if kn ≠ k then exchange (VEC[kn], VEC[k]);
L2: k := k+1; goto L;
```

The labels $L1$ and $L2$ are merely introduced as a reference. At label $L1$ we have $kn = \psi_k(k)$. Consequently $kn \geqq k$. In case $kn \neq k$, $VEC[kn]$ and $VEC[k]$ are exchanged. Since $\vdash E1 \wedge E2$ on L it follows $\vdash E1 \wedge E2$ on $L1$. We consider two cases:

CASE 1. $kn > k$. From $\vdash E1 \wedge E2$ on $L1$ 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 $L2$.

Therefore the following assertion holds at $L2$:

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

Hence

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

Finally $\vdash E1$ at L for $k = k1 + 1$.

Since $kn = \psi_k(k) > k$ then according to property 2 there exist elements $e1$ and $e2$, $e1 > k$, $e2 > k$ such that:

$$e2 = \psi_k(k) \quad \text{and} \quad k = \psi_k(e1)$$

and according to property 3:

$$e2 = \psi_{k+1}(e1) .$$

Apparently $e2 = kn$.

At label $L1$ we have

$$VEC[k] = VEC[\psi_k(e1)] = \alpha_{f(e1)}, \text{ since } e1 > k .$$

At label $L2$

$$VEC[kn] = VEC[\psi_k(k)] = VEC[e2] = \alpha_{f(e1)} .$$

Using property 3 at $L2$,

$$(13) \qquad VEC[e2] = VEC[\psi_{k+1}(e1)] = \alpha_{f(e1)} .$$

From $\vdash E2$ we deduce at label $L1$:

$$(14) \qquad \forall i(k < i \leqq G \wedge i \neq e1 \rightarrow VEC[\psi_k(i)] = \alpha_{f(i)}) .$$

Using property 4 we get at $L2$

$$(15) \qquad \forall i(k < i \leqq G \wedge i \neq e1 \rightarrow VEC[\psi_k(i)] = VEC[\psi_{k+1}(i)] = \alpha_{f(i)}) .$$

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

$$\forall i(k+1 \leqq i \leqq G \rightarrow VEC[\psi_{k+1}(i)] = \alpha_{f(i)}) .$$

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 $\psi_k(k) = k$ and no exchange takes place. From $\vdash E2$ at L and at $L1$ and $L2$ we deduce:

$$(16) \qquad VEC[\psi_k(k)] = VEC[k] = \alpha_{f(k)} .$$

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

$$(17) \qquad \forall i(1 \leqq i \leqq k \rightarrow VEC[i] = \alpha_{f(i)}) .$$

Hence

$$(18) \qquad \forall i(1 \leqq i < k+1 \rightarrow VEC[i] = \alpha_{f(i)}) \text{ at } L2 .$$

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

$$(19) \qquad \forall i(k+1 \leqq i \leqq G \rightarrow VEC[\psi_{k+1}(i)] = \alpha_{f(i)}) \text{ at } L2 .$$

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 $\vdash E1 \wedge E2$ we observe that $E2$ describes the initial state of the program for $k=1$. $E1$ is then "empty". $E1$ describes the final state for $k=G+1$. $E2$ is then "empty".

Acknowledgements.

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

REFERENCES

1. P. F. Windley, *Transposing matrices in a digital computer*, Computer Journal 2 (April 1959, 47–48.
2. J. Bootroyd, *Algorithm 302, Transpose vector stored array*, Comm. ACM 10 (May 1967), 292–293.
3. S. Laflin and M. A. Brebner, *Algorithm 380, In situ-transposition of a rectangular matrix*, Comm. ACM 13 (May 1970), 324–326.

TECHNOLOGICAL UNIVERSITY TWENTE
ENSCHEDE
THE NETHERLANDS