A simple algorithm with no simple verification

László Csirmaz

Central European University

Abstract

The correctness of a simple sorting algorithm is resented, which algorithm is "evidently wrong" at the first sight. It is conjectured that the algorithm has no straightforward, transparent verification.

1 Introduction

Since the appearance of the fundamental work of Floyd, Naur, and Hoare [1, 3, 4], a wide variety of programs were proved to be correct. These programs were usually written by some more or less sophisticated idea in mind, and knowing the idea – i.e. how the program works –, with not too much effort, an experienced programmer can give a formal proof for the correctness (or at least for the partial correctness). In the majority of the cases the formal proof seems to be unnecessary, especially if it is made my hand, since the danger of overlooking some fine point in the proof is as big as overlooking something in the program. But there are algorithms which have no such a transparent central idea, and, so to speak, the work " incidentally." To show that they work as expected, a program verification method is unavoidable.

In this paper we intend to introduce such a program, which was found in 1974. A student wrote a simple sorting program, and made a trivial mistake. But, in spite of the bug, the program worked well. Later several proofs have been found independently by K. Balog, H. Andréka and I. Németi, and the author. None of the proofs is simple, and by no means is straightforward.

2 The program

Suppose we are given a natural number $N \geq 1$, and two arrays, A and M with indices running from 1 to N. Initially, A contains distinct real numbers, and the content of $M[k]$ is k for each $1 \leq k \leq N$. The task is to write a program which rearranges the elements of M so that the relations

$$A[M[1]] < A[M[2]] < \cdots < A[M[N]]$$

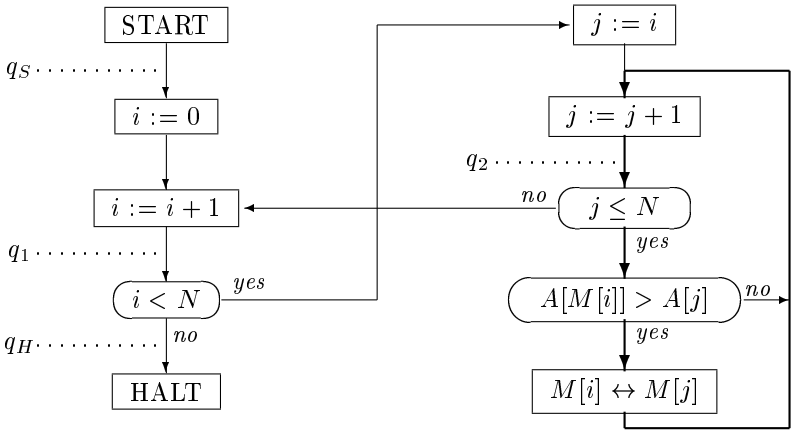

Figure 1: The sorting program

hold. An Algol-60 representation of the program is as follows.

begin
  integer i, j ;
  $i := 0$;
  for $i := i + 1$ **while** $i < N$ **do**
    begin $j := i$;
      for $j := j + 1$ **while** $j \leq N$ **do**
        if $A[M[i]] > A[j]$ **then** *exchange*$(M[i], M[j])$
    end
end.

The procedure *exchange* exchanges the values of its parameters. The "bug" is obvious: the condition in the **if** statement should be $A[M[i]] > A[M[j]]$. We claim, however, that our program is correct: it arranges the array M as required.

The flow-chart in 1 follows the program closely. From now on we use the flow-chart for reasoning about our program.

First we make some trivial observations. The program contains two nested loops. In the outer loop the variable i runs from 1 to $N - 1$, and for each value of i, j runs from $i + 1$ to N in the inner loop. In each iteration, $A[M[i]]$ and $A[j]$ are compared, and if the first is the larger then the contents of $M[i]$ and $M[j]$ are exchanged.

In table 1 we give snapshots of a sample run made at point q_2 in the inner loop. Here we have $N = 5$, and in the third and fourth column the bold face numbers are $A[M[i]]$ and $A[j]$, they are to be compared next.

We can see at once that it is a rare occasion when $A[M[i]]$ is compared to $A[M[j]]$ (i. e. when $M[j] = j$), and $A[M[i]]$ is never compared to itself, i. e. $M[j] \neq j$. Furthermore, $A[j]$ takes quite unexpected values comparing to $A[M[j]]$, but $A[j]$ and $A[M[j]]$ are always on the same side of $A[i]$. If we were able to prove this, the correctness of the program would follow at one. We shall

i	j	$A[M[.]]$					$A[.]$					$M[.]$				
		1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
1	2	**3**	5	2	1	4	3	**5**	2	1	4	1	2	3	4	5
1	3	**3**	5	2	1	4	3	5	**2**	1	4	1	2	3	4	5
1	4	**2**	5	3	1	4	3	5	2	**1**	4	3	2	1	4	5
1	5	**1**	5	3	2	4	3	5	2	1	**4**	4	2	1	3	5
1	6	1	5	3	2	4						4	2	1	3	5
2	3	1	**5**	3	2	4	3	5	**2**	1	4	4	2	1	3	5
2	4	1	**3**	5	2	4	3	5	2	**1**	4	4	2	1	3	5
2	5	1	**2**	5	3	4	3	5	2	1	**4**	4	2	1	3	5
2	6	1	2	5	3	4						4	2	1	3	5
3	4	1	2	**5**	3	4	3	5	2	**1**	4	4	2	1	3	5
4	5	1	2	**3**	5	4	3	5	2	1	**4**	4	2	1	3	5
5	6	1	2	3	5	4						4	2	1	3	5
4	5	1	2	3	**5**	4	3	5	2	1	**4**	4	2	1	3	5
4	6	1	2	3	4	5						4	3	1	5	2

Table 1: Snapshots of a sample run with $A = (3, 5, 2, 1, 4)$

see later that this is the case indeed, but it will only be a side result of out proof.

3 The assertions

We assume that the reader is familiar with the so-called Floyd-Hoare partial correctness proof method [2, 5]. Following this, we choose four points, q_S, q_H, q_1, and q_2 in the flow-chart so that they separate the program into five straight-line paths: q_S—q_1, q_1—q_2, q_2—q_2 through the heavy-line loop, q_2—q_1, finally q_1—q_H. We are going to assign assertions to these points so that each path is correct with respect to the assertions put at its start and and; moreover the assertion q_S contains only our assumptions about the arrays A and M, and q_H implies the final requirement, i. e. that A is sorted.

With q_S and q_H there is no problem. The input assertion q_S states that $N \geq 1$, the elements of A are distinct, and the $M[k] = k$ fir every integer $1 \leq k \leq N$. To be a bit more formal, we write this assertion as

$$N \geq 1, \tag{1}$$

$$(\forall k \leq N)\, M[k] = k, \tag{2}$$

$$A = AP \text{ and the elements of } A \text{ are distinct;} \tag{3}$$

When we write an assertion into several line, as we did there, it is the conjunct of the simple statements. In (2) the quantification $(\forall k \leq N)$ means "for all positive integer k such that $k \leq N$." We shall make use of the similar quantifications

$(\forall k < N)$, $(\forall i \le k \le N)$, etc. with meanings "for all positive integer k such that $k < N$," "for all positive integer k such that $i \le k \le N$," etc.

The array AP is the initial version of A; we define it in the input assertion q_S so that we can refer it in other assertions.

The output assertion q_H is as follows.

$$A = AP, \tag{4}$$
$$\text{the set } \{M[1], \ldots, M[N]\} \text{ is the same as } \{1, 2, \ldots, N\}, \tag{5}$$
$$(\forall k < N)\ A[M[k]] \text{ is the } k\text{-th (in order of magnitude)} \tag{6}$$
$$\text{in the set } \{A[1], \ldots, A[N]\}.$$

The first conjunct states that the array A remains unchanged; the second means that $M[1], \ldots, M[N]$ is a permutation of $1, 2, \ldots, N$; and in the third we claim that the index-array M contains the indices of A in the proper order, i.e. all the relations

$$A[M[1]] < A[M[2]] < \cdots < A[M[N]]$$

hold.

To make a long line short, we denote the set

$$\{M[a], \ldots, M[b-1], M[b]\}$$

by $M[a : b]$, and similarly for A. With this notation, the conjuncts (4)–(6) can be written into the more compact form

$$A = AP, \tag{7}$$
$$M[1 : N] = \{1, 2, \ldots, N\}, \tag{8}$$
$$(\forall k < N)\ A[M[k]] \text{ is the } k\text{-th in } A[1 : N]. \tag{9}$$

Next we must choose the assertion q_1 and q_2 so that the above mentioned paths between the consecutive assertions be correct. Let us go backward. We have to derive q_H from q_1 and from the additional assumption "not $i < N$." It can evidently be done if q_1 contains the conjuncts

$$A = AP \text{ and the elements of } A \text{ are distinct}, \tag{10}$$
$$M[1 : N] = \{1, 2, \ldots, N\}, \tag{11}$$
$$1 \le i \le N, \tag{12}$$
$$(\forall k < i)\ A[M[k]] \text{ is the } k\text{-th in } A[1 : N]. \tag{13}$$

We include (12) here in order to ensure that (13) is meaningful, and to give $i = n$ from the assumption "not $i < N$." Now from (1)–(3) we can derive (10)–(13) with $i = 1$, this proves the partial correctness of the pat q_S—q_1. Note that in this case (13) become

$$(\forall k < 1)\ A[M[k]] \text{ is the } k\text{-th in } A[1 : N]. \tag{14}$$

There is no positive integer less than 1, so (14) hods vacuously.

The next backward step is to choose an appropriate assertion for q_2. This must be true when the inner loop is entered (via q_1(, must remain true whenever

the control passes around the loop and returns, and must be string enough to imply the assertion (10)–(13) for the next value of i. With this last requirement in mind, we try the following conjunct for q_2:

$$A[M[i]] \text{ is just the } i\text{-th element in the set } A[1 : j-1]. \tag{15}$$

Then $M[i]$ must be an index less than j, so we also put

$$M[i] < j \tag{16}$$

Both (15) and (16) are affirmed by the snapshots. Now let us see what happens with the conjunct (15) if we follow the heavy line and arrive back to q_2. If $A[M[i]] < A[j]$ then, from (15), it follows that $A[M[i]]$ is also the i-th element in the set $A[1 : j]$. But what can we say if $A[M[i]] > A[j]$? In this case the values of $M[i]$ and $M[j]$ are exchanged, so in the next iteration (15) will be valid only if

$$A[M[j]] \text{ is the } i\text{-th element in } A[1 : j] \tag{17}$$

holds at present. (17) can be derived easily from the (inductive) conjunct

$$(\forall j \le k \le N)\ A[M[k]] \text{ is the } i\text{-th element in } A[1 : k], \tag{18}$$

which we would like to attach to q_2. But, alas, (18) is not always true. For example, using Table 1 of snapshots for the values $i = 1$, $j = 2$, $k = 5$, (18) claims

$$A[M[5]] = 4 \text{ is the least element in } \{3, 5, 2, 1, 4\},$$

which is false. So we have to refine (18). Examine a bit closely the sentence (17). We *know* from the inductive hypothesis (15) that $A[M[i]]$ is the i-th element in the set $A[1 : j-1]$, and assume that $A[M[i]] > A[j]$. At the same time we *need* that $A[M[j]]$ is the i-th element in $A[1 : j]$. Since we know something about $A[j]$, and want to derive something about $A[M[j]]$, it seems quite natural to split the investigation into two cases depending on whether $M[j] = j$ or not.

First, if $M[j] = j$ and $A[M[i]] > A[j]$ then $A[M[j]] = A[j]$ is the i-th element in $A[1 : j]$ just in case there are at least $(i-1)$ elements less than $A[j]$ in $A[1 : j]$. No try to attach it as a conjunct to q_2:

$$(\forall j \le k \le N)\ \textbf{if } M[k] = k \textbf{ then} \text{ there are at least } i-1 \text{ elements less than } A[k] \text{ in the set } A[1 : k]. \tag{19}$$

After a short glance at the snapshots, we see that it is a hopeful candidate.

Second, if $M[j] \neq j$ then $A[M[j]]$ is an element of the set $A[1 : j]$, i. e. $M[j] < j$. So we put at once

$$(\forall j \le k \le N)\ M[k] \le k. \tag{20}$$

In the lack of any better idea, we use (18) in this case, the snapshots do not contradict to this conjunct anymore:

$$(\forall j \le k \le N)\ \textbf{if } M[k] < k \textbf{ then } A[M[k]] \text{ is the } i\text{-th in the set } A[1 : k]. \tag{21}$$

No let us see were we are. From (15) we can derive (13) with $i+1$ instead of i, and from (19)–(21) we can derive (15) for the next iteration of the inner loop. Since we want (19)–(21) be valid whenever the control enters the inner loop, we put into q_1 the conjuncts

$$(\forall i \le k \le N)\ \textbf{if}\ M[k] = k\ \textbf{then} \tag{22}$$
$$\text{there are at least } i-1 \text{ elements in } A[1:k] \text{ less than } A[k];$$
$$(\forall i \le k \le N)\ M[k] \le k, \tag{23}$$
$$(\forall i \le k \le N)\ \textbf{if}\ M[k] < k\ \textbf{then}\ A[M[k]] \text{ is the } i\text{-th in } A[1:k]. \tag{24}$$

When the control first reaches q_1 then $i = 1$ and, by the assertion we made at q_S, $M[k] = k$ for every $1 \le k \le N$. Therefore (22)–(24) evidently hold. But how can we ensure that they would hold whenever the control returns from the inner loop? There is no hope: we have to put the new conjuncts into q_2, the state that as j goes higher, the initial parts in (22)–(24) become valid with the next value of i. So we put

$$(\forall i < k < j)\ \textbf{if}\ M[k] = k\ \textbf{then} \tag{25}$$
$$\text{there are at least } i \text{ elements less than } A[k] \text{ in } A[1:k],$$
$$(\forall i < k < j)\ M[k] \le k, \tag{26}$$
$$(\forall i < k < j)\ \textbf{if}\ M[k] < k\ \textbf{then} \tag{27}$$
$$A[M[k]] \text{ is the } (i+1)\text{-st element in the set } A[1:k].$$

At this point we decide it is best to start another paragraph.

4 The proof

We prove the correctness of our program in two stages. First we check that the program always terminates. This is fairly easy. The control passes through every box at most once for each possible value of the pair (i, j), therefor it reaches the HALT box before visiting $5N(N+1)$ boxes altogether.

Second, we prove the partial correctness with respect to the input and output assertions q_S and q_H defined previously. To do so, we collect and complete the disjuncts q_1 and q_2 we have found so far, and then check the partial correctness of the straight-line paths q_S—q_1, q_1—q_2, q_2—q_2, q_2—q_1, and q_1—q_H in turn.

The conjuncts of the assertion q_1 are (10)–(13) and (22)–(24). For the sake of completeness, we collect them here:

$$A = AP \text{ and the elements of } A \text{ are distinct}, \tag{28}$$
$$M[1:N] = \{1, 2, \ldots, N\}, \tag{29}$$
$$1 \le i \le N, \tag{30}$$
$$(\forall k < i)\ A[M[k]] \text{ is the } k\text{-th in } A[1:N], \tag{31}$$
$$(\forall i \le k \le N)\ \textbf{if}\ M[k] = k\ \textbf{then} \tag{32}$$
$$\text{there are at least } i-1 \text{ elements in } A[1:k] \text{ less than } A[k];$$
$$(\forall i \le k \le N)\ M[k] \le k, \tag{33}$$

$$(\forall i \le k \le N)\ \textbf{if}\ M[k] < k\ \textbf{then}\ A[M[k]] \text{ is the } i\text{-th in } A[1:k]. \tag{34}$$

With this assertion at q_1, as we have explained above, the paths q_S—q_1 and q_1—q_H are correct. Indeed, when the control reaches q_1 in the path q_S—q_1, then $i = 1$, and from this and from (1)–(3) then conjuncts follow easily. In the path q_1—q_H leading to the endpoint, (30) and the condition "not $i < N$" gives $i = N$, and then (7), (8), and (9) can be got from (28), (29), and (31), respectively.

The assertion attached to q_2 is the following twelve-line monster:

$$A = AP \text{ and the elements of } A \text{ are distinct}, \tag{35}$$
$$M[1:N] = \{1, 2, \ldots, N\}, \tag{36}$$
$$1 \le i < N, \tag{37}$$
$$(\forall k < i)\ A[M[k]] \text{ is the } k\text{-th element in } A[1:N], \tag{38}$$
$$(\forall i < k \le N)\ M[k] \le k, \tag{39}$$
$$i < j \le N + 1, \tag{40}$$
$$M[i] < j, \tag{41}$$
$$A[M[i]] \text{ is just the } i\text{-th element in } A[1:j-1], \tag{42}$$
$$(\forall j \le k \le N)\ \textbf{if}\ M[k] = k\ \textbf{then} \tag{43}$$
$$\text{there are at least } i-1 \text{ elements less than } A[k] \text{ in } A[1:k],$$
$$(\forall j \le k \le N)\ \textbf{if}\ M[k] < k\ \textbf{then} \tag{44}$$
$$A[M[k]] \text{ is the } i\text{-th element in } A[1:k],$$
$$(\forall i < k < j)\ \textbf{if}\ M[k] = k\ \textbf{then} \tag{45}$$
$$\text{there are at least } i \text{ elements less than } A[k] \text{ in } A[1:k],$$
$$(\forall i < k < j)\ \textbf{if}\ M[k] < k\ \textbf{then} \tag{46}$$
$$A[M[k]] \text{ is the } (i+1)\text{-st element in } A[1:k].$$

The conjuncts (35)–(38) are (almost faithful) copies of the conjuncts (28)–(31) of q_1; they are saved and then given back to q_1. In (39) we unify (20) and (26). (40) is necessary to ensure that when we leave the inner look then $j = N + 1$. The conjuncts (41)–(46) were discussed in detail in the previous section.

There are three further paths whose correctness need justification.

The path q_1—q_2

When the control reaches the point q_2 along from q_1 then $i < N$ and $j = i + 1$ hold as well. Therefore (35)–(39) and (43)–(44) are easy consequences of (28)–(32), and (33)-(34), respectively. (45) and (46) hold vacuously because there is no integer between i and $i + 1$. (40) is trivial, and by (32), $M[i] < i + 1 = j$, which proves (41). To only questionable conjunct is (42). (Observe that we did not bother to make (42) valid at entering the inner look.) But either $M[i] < i$, and then (34) says the $A[M[i]]$ is the i-th element in $A[1:i]$, or $M[i] = i$. (By (32) the case $M[i] > i$ cannot happen.) Since in this latter case (33) gives that there are at least $(i-1)$ elements less than $A[i]$ in the set $A[1:i]$, $A[i]$ must be the i-th element. This proves (42).

The path q_2—q_1

When the control leaves the inner look, we have "not $j < N$," i. e. by (40), $j = N + 1$. Afterwards i is increased by 1, and then the control arrives at q_1. Here the assertion (28)–(34) must hold, and we show that this is the case indeed, keeping in mind that $j = N + 1$, and that the "new" i in (28)–(32) is the same as the "old" i plus one in (35)–(46).

Now (28)–(30), (32) and (33)–(34) follow easily from (35)–(37), (39) and (45)–(46), respectively. (31) is contained in (38) except for the last value of k, when k equals to the "old" i, in which case (42) works.

The path q_2—q_2

Suppose that the assertion (35)–(46) holds at the point q_2, and the control goes around the loop and arrives back to q_2. We claim that the assertion remains valid. For conjuncts (35)–(38), (40), and (43)–(44) this is evidently true. When the control enters the look, then by (41) $M[i] < j$, and by (35), then elements of A are different. Therefore either $A[M[i]] < A[j]$, or $A[M[i]] > A[j]$, and there is no other case. We go ahead by handling these cases separately.

Case $A[M[i]] < A[j]$

Here (42) implies

$$A[M[i]] \text{ is the } i\text{-th element in } A[1 : j], \tag{47}$$

and, because the i-th element in $A[1 : j]$ is less than $A[j]$,

$$\text{there are at least } i \text{ elements less than } A[j] \text{ in } A[1 : j]. \tag{48}$$

Now we claim that

$$M[j] = j \tag{49}$$

If not, then $M[j] < j$ (by (39)), and therefore (44) gives

$$A[M[j]] \text{ is the } i\text{-th element in } A[1 : j]. \tag{50}$$

Since the elements of A are different, (47) and (50) implies $M = [i] = M[j]$, which is a contradiction.

When the control arrives back to q_2 then only the value of j has increased. So (39) and (41) trivially hold, and (47), (48), (49) show the validity of (42), (45), and (46), respectively. The completes our first case.

Case $A[M[i]] > A[j]$

The conjunct (42) says that $A[M[i]]$ is the i-th element in $A[1 : j - 1]$, therefore

$$M[j] < j \text{ and } A[M[i]] \text{ is the } (i + 1)\text{-st element in } A[1 : j]. \tag{51}$$

The claim

$$A[M[j]] \text{ is the } i\text{-th element in } A[1:j] \tag{52}$$

follows from (42), (43), and from $A[M[i]] > A[j]$ if $M[j] = j$, and form (44) if $M[j] < j$. On the other hand, (39) gives

$$M[j] < j + 1. \tag{53}$$

As the control goes ahead, the values of $M[i]$ and $M[j]$ are exchanged, and then j is increased by 1. Therefore (51), (52), and (53) show the validity of (39)–(45), (46), (42) and (41).

This completes the proof of the correctness of the sorting algorithm.

A bit more closely examination of (49), (51) and (52) gives that

$$\begin{array}{l} \textbf{if } A[M[i]] < A[j] \textbf{ then } M[j] = j, \\ \textbf{if } A[M[i]] > A[j] \textbf{ then } A[M[i]] > A[M[j]] \end{array} \tag{54}$$

always hold at point q_2. And because $A[M[j]] \neq A[j]$, this is equivalent to

$$A[M[i]] > A[j] \textbf{ if and only if } A[M[i]] > A[M[j]], \tag{55}$$

as we promised.

5 Conclusion

We have seen the correctness of a rather simple sorting algorithm, which, at the first sight, seemed to be wrong. The main invariant assertion mate at point q_2 is rather difficult. Although it makes possible to prove the correctness, gives no insight into the essence of the algorithm, and gives no explanation why the program works.

Of course, the requirement that A has only distinct elements, is unnecessary, it only makes the proof simpler.

By complexity considerations, there are simple algorithms whose correctness can be proved only by hard and difficult way (comparing to the simplicity of the algorithm, see [6]), and we think that this algorithm belongs to this class. This claim may be wrong, but a more transparent proof would not be without interest.

References

[1] K. R. Apt, Ten years of Hoare's logic, a survey, *ACM Trans. on Programming Languages and Systems* **3** (1981) pp. 431–482

[2] J. W. deBakker, *Mathematical theory of program correctness* Prentice-Hall, London, 1980

[3] R. W. Floyd, Assigning meaning to programs, in: Proc. ACM Symposioum in Applied Math **19** (1967) pp. 19–31

[4] C. A. R. Hoare, An axiomatic basis of computer programming, *Comm. ACM* **12** (1969) pp. 567–580

[5] Z. Manna, *Mathematical Theory of Computation* McGraw Hill, New York (1074)

[6] J. Spencer, Short theorem with long proofs, Am. Math.Monthly **90** (1983) pp. 365–366