1972

Parallel neighbor-sort (or the glory of the induction principle)

A. Nico Habermann
Carnegie Mellon University

PARALLEL NEIGHBOR-SORT

(OR THE GLORY OF THE INDUCTION PRINCIPLE)

Nico Habermann
Carnegie-Mellon University
Pittsburgh, Pa.

August 1972

This work was supported by the Advanced Research Projects Agency of the Office of the Secretary of Defense (F44620-70-C-0107) and is monitored by the Air Force Office of Scientific Research.

ABSTRACT

Array A[1:N] ($N \geq 2$) is to be sorted in ascending order using procedure NS(i) which arranges the values of an adjacent pair A[i], A[i+1] in the right order. It is shown that a parallel process P which can perform at least $N \div 2$ executions of procedure NS in parallel will sort array A in p steps where $p \leq N$. The proof is based on the observation that the distance of the position of a number in array A after p steps and its final position is bounded by $N - p$.

1. INTRODUCTION

Elements of a given array A[1:N] (where $N \geq 2$) are to be sorted in ascending order assuming that nothing is known about the initial order.

Suppose array A can only be rearranged by sorting adjacent pairs (A[i], A[i+1]) with procedure NS declared as:

```
procedure NS(i); integer i;
if i < N do
if A[i] > A[i+1] do
begin integer w; w ← A[i]; A[i] ← A[i+1]; A[i+1] ← w end;
```

This requirement -- that only allows operations on neighboring pairs of elements of array A -- restricts of course the range of sorting algorithms considerably. But this restriction is plausible if parallel processes are considered which are capable of carrying out many sort operations on elements of A simultaneously.

The parallel processes considered here are supposed to be able to execute in parallel at least $N \div 2$ instances of procedure NS. Such a process could for instance order in one step all elements with odd index with regard to their neighbor element with even index. This paper addresses itself to the question of how many parallel steps would be needed to sort array A[1:N] with procedure NS. The answer turns out to be that the number of parallel steps required is less or equal to N. This seems plausible when thinking of the smallest or largest number, but not so obvious when observing how an "in between" number may wander about before arriving at its destination.

Example

						parallel step p
11	7	8	4	2	1	0
7	11	4	8	1	2	1
7	4	11	1	8	2	2
4	7	1	11	2	8	3
4	1	7	2	11	8	4
1	4	2	7	8	11	5
1	2	4	7	8	11	6

The numbers 2 and 7 are starting off in the wrong direction, the numbers 4 and 8 begin heading in the right direction but pass their destination. But it seems as if a number v cannot wander too far from its destination. The example shows that any number v is at most two positions off from its destination for p = N-2 and only one for p = N-1. The generalization of this observation is: a number v is after p steps at most N-p positions away from its destination.

The larger part of the sequel deals with the correctness proof of this observation. The existence of the upperbound N for the number of steps p needed to sort array A can easily be derived from it.

2. REPHRASING THE PROBLEM

The sorting of array A[1:N] ($N \geq 2$) with arbitrary values can easily be translated into a mapping of $I \rightarrow I$ where $I = \{1,2,\ldots,N\}$. The idea is to replace each value v by the index of its destination which will be reached after sorting. Thus, to sort array A means to exchange elements of A until for all $j \in \{1,2,\ldots,N\}$ $A[j] = j$.

A sequential process that sorts array A using procedure NS is described by

```
integer k,z;
for z ← N-1 step -1 until 1 do
   for k ← 1 step 1 until z do NS(i)
```

It is well known that this process would certainly not be the most efficient one if the restriction of "rearranging neighbors only" was removed. It requires 1/2 N(N-1) sequential executions of procedure NS, whereas Quick Sort procedures require in the order of 1/2 N $\log_2$ N comparisons. On the other hand, pairwise comparison is attractive if many of those can be performed in parallel at no additional cost. But the sequential process described above cannot be transformed easily into a parallel process because of its iterative nature.

Instead, parallel processes that will be considered use two parallel statements, named S_{odd} and S_{even}:

```
S_odd:  parbegin NS(1); NS(3); ...; NS(1 + (N-1) ÷ 2*2) parend
S_even: parbegin NS(2); NS(4); ...; NS(N ÷ 2*2) parend
```

The result of executing a parallel statement should be unambiguously interpretable without requiring knowledge of the order in which variables are accessed. It would not make sense for instance to use two consecutive indices as parameters to calls of procedure NS in a parallel statement, e.g., the result of

```
parbegin ...; NS(i); NS(i+1); ... parend
```

cannot be interpreted uniquely. For instance,

```
(A[i], A[i+1], A[i+2]) = (11, 7, 3)
```

could be mapped by this parallel statement into

(7, 3, 11) or (3, 7, 11) or even (7, 11, 7)!!

Parallel statements S_{odd} and S_{even} do not have such ambiguities, because each element A[i] is involved in not more than one instance of procedure NS.

Let process P_{odd} be defined by

```
integer p;
for p = 1 step 1 until N do
   if p = p ÷ 2*2 then S_even else S_odd
```

and process P_{even} by

```
integer p;
for p = 1 step 1 until N do
   if p = p ÷ 2*2 then S_odd else S_even
```

Process P_{odd} starts with S_{odd}, process P_{even} starts with S_{even}. The main theorem to prove is

Theorem 1. Process P_{odd} or process P_{even} sort array A in ascending order (or: array A is sorted by p alternating execution of S_{odd} and S_{even} -- no matter which is first, and $p \leq N$).

The proof is based on the following result:

Theorem 2. $ABS(i_j(p) - j) \leq N-p$ (1)

for all $j \in N$,

where $i_j(p)$ is the index of j in array A after p steps (i.e., p alternating execution of S_{odd} and S_{even}, no matter which one is taken first).

The proof of theorem 1 follows immediately from (1):

$$ABS(i_j(n) - j) = 0 \qquad \text{for } p = N \text{ and}$$

for all $j \in \{1,\ldots,N\}$, i.e., the index of j after N steps is j, or A[j] = j.

3. PROOF OF THEOREM 2

Theorem 2 puts an upper limit on how far a number j is out of balance so to speak. This depends on

$\alpha_j(p)$ ::= how many numbers to the left of j are greater than j after p steps

$\beta_j(p)$ ::= how many numbers to the right of j are less than j after p steps.

(The subscript j will be omitted from now on where confusion is unlikely.)

The numbers $\leq j$ are together $[i(p) - \alpha(p)] + \beta(p)$, and so

$$i(p) - j = \alpha(p) - \beta(p) \tag{2}$$

and initially

$$i(0) - j = \alpha(0) - \beta(0) \tag{3}$$

The goal is to prove that $0 \leq \alpha(p) \leq N-p$ (4)

and $0 \leq \beta(p) \leq N-p$ (5)

The proof of theorem 2 follows then from (2), (4) and (5).

$$-(N-p) \leq -\beta(p) \leq \alpha(p)-\beta(p) = i(p)-j \leq \alpha(p) \leq N-p$$

Let B[1:N] ($N \geq 2$) be an array of which initially

B[k] = k for k=1,...,m and $1 \leq m \leq N-1$

B[k] = 0 for k=m+1,...,N

Array B consists of the sequence 1,...,m followed by at least one 0.

S_{odd} or S_{even} applied to array B transform no other pairs than (k,0) into (0,k) for $k \in \{1,...,m\}$. Thus, the non-zero numbers will not overtake each other and the zeros preserve also their relative order. The case [P_{odd}, m is odd] or [P_{even}, m is even] is denoted by C_{even}; the other case by C_{odd}.

The pair (B[m], B[m+1]) is compared in the first step iff C_{even} applies.

Lemma 1. The number m, initially in B[m], is in B[m+p] after p steps of C_{even} or p+1 steps of C_{odd} for $0 \le p \le N-m$.

Proof. The first step of C_{odd} has no effect on array B since no pair (1,0) is considered. After the first step C_{odd} has reached the same state as the initial state of C_{even}. Thus, whatever is achieved in p steps of C_{even}, is obtained in p+1 steps of C_{odd}. Consider C_{even}. The statement is true for p=0. Suppose it is true for $p=p_0$ and p_0 is a number such that $0 \le p_0 \le N-m-1$. So, $(B[m+p_0], B[m+p_0+1]) = (m,0)$, since m is the rightmost non-zero number. The pairs considered in the p^{th} step of C_{even} are

$$(B[m+p-1+2*v], B[m+p+2*v])$$

for all v such that $2 \le m+p+2*v \le N$. Hence, one of the pairs considered in step p_0+1 is

$$(B[m + (p_0+1) - 1], B[m + (p_0+1)])$$

for v = 0 [for which value $2 \le m+p_0+1+2*v \le N$ is true]. But this pair equals (m,0) according to the supposition and it is subsequently transformed into (0,m). Hence, the assumption that $B[m+p_0] = m$ for $p = p_0$ implies that

$B[m+p_0+1] = m$ for $p = p_0+1$. Thus, the statement is true for all p such that $0 \leq p \leq N-m$ according to the induction principle.

There is obviously a duality in array B of zeros and non-zero numbers and so, lemma 1 about the rightmost non-zero number carries over into a statement about the leftmost 0:

Lemma 2. The leftmost 0, initially in $B[m+1]$, is in $B[m-p+1]$ after p steps of C_{even} or p+1 steps of C_{odd} for $0 \leq p \leq m$

Lemma 3. Number $k \in \{1,\ldots,m\}$, initially in $B[k]$, is in $B[2*k+p-m]$ after p steps of C_{even} or p+1 steps of C_{odd} for $m \leq k + p \leq N$.

Proof. As observed in the proof of lemma 1, C_{odd} required one step more to achieve the same as C_{even}. Consider C_{even}. The statement is true for k=m (lemma 1). Suppose it is true for $k = k_0 \in \{1,\ldots,m\}$, i.e.,

$$B[2*k_0 + p-m] = k_0 \quad \text{for all p such that} \quad m \leq k_0-1+p \leq N.$$

It is true for $p = m+1-k_0$, because $B[m - (m+1-k_0) + 1] = B[k_0]$ contains the leftmost zero according to lemma 2 and so $B[k_0-1] = k_0-1$. Suppose it is true for $p = p_0$ and $m \leq k_0 + p_0 \leq N$, i.e., $B[2*(k_0-1) + p_0-m] = k_0-1$. But it is known that for this value of p

$$B[2*k_0 + p_0-m] = k_0 \quad \text{and so}$$
$$B[2*k_0 - p_0-m] = 0.$$

In step p_0 $B[2*k_0 + p_0-m]$ got the value k_0, hence in step p_0+1 the pair $(B[2*k_0 + p_0-2-m], B[2*k_0 + p_0-1-m])$ is considered. It equals $(k_0-1,0)$ and is subsequently transformed into $(0,k_0-1)$. Thus,

$$B[2*(k_0-1) + (p_0+1) - m] = k_0-1 \quad \text{for } p = p_0+1$$

Applying the induction principle twice shows that the statement is true.

Let $\ell^*(p)$ denote the number of non-zeros left of any zero in array B after p steps of P_{odd} or P_{even}.

Lemma 4. $\ell^*(p) \leq N-p$

Proof. Number $k_0 \in \{1,\ldots,m\}$ is moved from $B[k_0]$ to $B[N-m+k_0]$ in $p = N-k_0$ steps of C_{even} or $p = N-k_0+1$ steps of C_{odd} according to lemma 3. This number k_0 has passed all zeros of array B. But so have all numbers k for which $k_0 \leq k \leq m$ or $N-p+1 \leq k \leq m$. Thus, for given p the numbers that possibly did not pass all zeros of array B is less than N-p+1 or $\ell^*(p) \leq N-p$.

Let C[1:N] be an array with the same number of non-zero and zero elements as array B[1:N]. The non-zero elements are the numbers $1,\ldots,m$ $(1 \leq m \leq N-1)$ from left to right. These numbers are not necessarily stored in consecutive elements of array C. Thus, if $i_j^*(p)$ and $i_j(p)$ denote the indices of number $j \in \{1,\ldots,m\}$ after p steps of P_{odd} or P_{even} in arrays B and C respectively,

$$i_j^*(0) \leq i_j(0)$$

for all $j \in \{1,\ldots,m\}$.

Lemma 5. $i_j^*(p) \leq i_j(p)$ for all $j \in \{1,\ldots,m\}$ and $p \in \{0,\ldots,N\}$

Proof. Let j = m. According to lemma 1

$$i_m^*(p) = i_m^*(0) + p \leq i_m(0) + p = i_m(p) \quad \text{for } p \leq N-i_m(0)$$
$$(\text{or } p \leq N-i_m(0) + 1 \text{ for } C_{odd}) \quad \text{and}$$
$$i_m^*(p) \leq i_m(p) = N \text{ for } N-i_m(0) + 1 \leq p \leq N.$$

Suppose the relation is true for $j = k+1$. The objective is to prove that this implies that the relation is also true for $j=k$. For $p=0$ $i_k^*(0) \le i_k(0)$ is true. Suppose it is true for $p = p_0 \in \{0,\ldots N-1\}$. It could not happen that, if $i_k^*(p_0) = i_k(p)$,

$$B[i_k^*(p_0) + 1] = 0 \quad \text{and} \quad C[i_k(p) + 1] \neq 0$$

because it would mean that

$$i_{k+1}^*(p_0) > i_k^*(p_0) + 1 = i_k(p_0) + 1 = i_{k+1}(p_0),$$

and this is in contradiction with the supposition that the relation is true for all $p \in \{0,\ldots,N\}$ for $j = k+1$. It can easily be derived that the relation is also true for $p = p_0+1$ in all other cases (viz. either $B[i_k^*(p_0) + 1] \neq 0$ or both $B[i_k^*(p_0) + 1] = C[i_k(p) + 1] = 0$ or $i_k^*(p_0) < i_k(p)$). Applying the induction principle twice shows that

$$i_j^*(p) \le i_j(p) \text{ for all } j \in \{1,\ldots,m\} \text{ and } p \in \{0,\ldots,N\}$$

Let $\ell(p)$ denote the number of non-zeros in array C to the left of at least one zero after p steps .

Lemma 6. $\ell(p) \le \ell^*(p)$ for all $p \in \{0,\ldots,N\}$

Proof. If there are not any non-zeros to the right of all zeros then $\ell(p) = \ell^*(p) = m$. Let the numbers $k+1,\ldots,m$ be to the right of all zeros in array B and the numbers $j+1,\ldots,m$ to the right of all zeros in array C. Then $B[n-m+k] = 0$, which is the rightmost zero in array B, $C[N-m+j] = 0$, which is the rightmost zero in array C and $\ell(p) = j$ and $\ell^*(p) = k$. Suppose $k < j$.

This implies that j has all zeros to its left in array B and so $i_j^*(p) = N-m+j$. But $i_j(p) < N-m+j$ in array C and so $i_j^*(p) > i_j(p)$. But this relation is in contradiction with lemma 5.

Let us now return to array A[1:N]. The objective was to show that $\alpha_j(p) \leq N-p$ for all $j \in \{1,\ldots,N\}$. There are no numbers in array A greater than N and so the relation is trivial for j = N: $\alpha_N(p) = 0 \leq N-p$ is true. Consider a number $j \in \{1,\ldots,N-1\}$ and the mapping $T_j(A) \rightarrow C$ defined by

```
T_j: integer i,m; m ← 0;
     for i ← 1 step 1 until N do
     if A[i] > j then begin m ← m+1; C[i] ← m end
                 else C[i] ← 0
```

The result of the mapping is that array C contains at least one number $\neq 0$ and at least one 0. The non-zero numbers are ordered in ascending order from left to right, but not necessarily in contiguous locations.

Lemma 7. $T_j.S(A) = S.T_j(A)$ for all $j \in \{1,\ldots,N-1\}$, i.e., the index relation between numbers $\leq j$ in array A and zeros in array C is invariant for parallel steps S_{odd} and S_{even}.

Proof. Take a number $j \in \{1,\ldots,N-1\}$ and let (k_1,k_2) be a number pair in array A that is considered in S_{even} or S_{odd}.

If $k_1 > j$ and $k_2 > j$ then

$$\left.\begin{aligned} T_j.S(k_1,k_2) &= T_j(k_1,k_2) \\ &\text{or } = T_j(k_2,k_1) \end{aligned}\right\} (i,i+1)$$

and $S.T_j(k_1,k_2) = S(i,i+1) = (i,i+1)$.

If $k_1 \leq j$ and $k_2 \leq j$, the result is both ways (0,0).

If $k_1 > j$ and $k_2 \leq j$ then

$T_j.S(k_1,k_2) = T_j(k_2,k_1) = (0,i)$

and $S.T_j(k_1,k_2) = S(i,0) = (0,i)$

If $k_1 \leq j$ and $k_2 > j$, the result is both ways $(0,i)$.

The statement can be generalized to include an arbitrary sequence of steps S_{odd} and S_{even}.

Lemma 8. $\alpha_j(p) \leq \ell_j(p)$ for all $j \in \{1,\ldots,N-1\}$ and $p \in \{0,\ldots,N\}$

Proof. Let $V_j(p) = \{k \mid k > j \text{ and } i_k(p) < i_j(p)\}$.

$\alpha_j(p)$ = number of elements of $V_j(p)$.

$T_j(k \in V_j(p)) = k' > 0$ and $i_{k'}(p) = i_k(p) < i_j(p)$ and

$T_j(j) = 0$ in $C[i_j(p)]$ according to lemma 7.

Hence, $\alpha(p)$ = number of non-zeros left of $C[i_j(p)] = 0$, whereas $\ell(p)$ = number of non-zeros left of the rightmost 0. Thus, $\alpha(p) \leq \ell(p)$.

The crucial relation

$$\alpha_j(p) \leq N-p \tag{4}$$

for all $j \in \{1,\ldots,N\}$ and $p \in \{0,\ldots,N\}$ can now be derived from

$$\alpha_N(p) = 0 \quad \text{for } j = N \text{ and all } p \in \{0,\ldots,N\}$$

and for all $j \in \{1,\ldots,N-1\}$ and all $p \in \{0,\ldots,N\}$

$$\ell_j^*(p) \leq N-p \qquad \text{(lemma 4)}$$

$$\ell_j(p) \leq \ell_j^*(p) \qquad \text{(lemma 6)}$$

$$\alpha_j(p) \leq \ell_j(p) \qquad \text{(lemma 8)}$$

For the proof of the main theorem, that array A[1:N] can be sorted in N parallel steps, we also needed the relation

$$\beta_j(p) \leq N-p$$

It is obvious that the proof that this relation is true can be constructed by systematic changes of left, right and less than, greater than in the lemmas. The check is left to the reader. It is surprising that a theorem which seemed quite obvious required such an elaborate proof.

ACKNOWLEDGEMENTS

I am grateful to Anita Jones, Larry Snyder, and Tim Teitelbaum for discussing the solution with me. It was a pleasure to observe how E. W. Dijkstra constructed variations and alternatives of the algorithms P_{odd} and P_{even}. I am also grateful to R. Floyd who pointed out to me that an algorithm to sort the reverse sequence, N, N-1,..., 1, also sorts any other permutation of these numbers. This theorem is essentially used where arrays B and C are discussed (p. 8) and proved for algorithms P_{odd} and P_{even} in the form of lemmas 5 and 6.

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
Computer Science Department Carnegie-Mellon University Pittsburgh, Pa. 15213	UNCLASSIFIED 2b. GROUP

3. REPORT TITLE

PARALLEL NEIGHBOR-SORT (OR THE GLORY OF THE INDUCTION PRINCIPLE)

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Scientific Final

5. AUTHOR(S) (First name, middle initial, last name)

Nico Habermann

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
August 1972	15	

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
F44620-70-C-0107 b. PROJECT NO. 9769 c. 61102F d. 681304	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
TECH OTHER	Air Force Office of Scientific Rsch (NM) 1400 Wilson Blvd. Arlington, Va. 22209

13. ABSTRACT

Array A[1:N] (N ≥ 2) is to be sorted in ascending order using procedure NS(i) which arranges the values of an adjacent pair A[i], A[i+1] in the right order. It is shown that a parallel process P which can perform at least N ÷ 2 executions of procedure NS in parallel will sort array A in p steps where p ≤ N. The proof is based on the observation that the distance of the position of a number in array A after p steps and its final position is bounded by N - p.

DD FORM 1 NOV 65 1473

Security Classification