Postkarte

CANADA

Data Structuring Group,
Department of Computer Science
University of Waterloo
Prof. Derick Wood

Waterloo
Ontario N2L 3G1

Akademie der Wissenschaften der DDR
Karl-Weierstraß-Institut
für Mathematik
Kristel Unger
DDR 1086 Berlin
Mohrenstraße 39 · PF 1304

Dear Sir,

I would greatly appreciate a reprint of your paper

- A New Measure of Presortedness
- Roughly Sorting: A Generalization of Sorting

~~from~~ (CS 87-58 and CS 87-55)

and ~~related papers, if~~ you ~~have copies for dis~~tribution.

Thanking you in advance, Yours sincerely

Kristel Unger

sent

AUG 31 1988

377 Druckerei Aßmann Bm 915/85 5,0 214

1350 143

Printing Requisition/Graphic Services

89016

1. Please complete unshaded areas on form as applicable.
2. Distribute copies as follows: White and Yellow to Graphic Services. Retain Pink Copies for your records.
3. On completion of order the Yellow copy will be returned with the printed material.
4. Please direct enquiries, quoting requisition number and account number, to extension 3451.

TITLE OR DESCRIPTION: New Measure of Presortedness CS-87-58

DATE REQUISITIONED: October 15/87

DATE REQUIRED: ASAP

ACCOUNT NO.: 1 2 6 6 1 7 6 8 1

REQUISITIONER- PRINT: Derick Wood

PHONE: x4456

SIGNING AUTHORITY: D. Fink / D. Wood

MAILING INFO - NAME: Sue DeAngelis

DEPT.: Computer Science

BLDG. & ROOM NO.: MC 6081E

[X] DELIVER [ ] PICK-UP

Copyright: I hereby agree to assume all responsibility and liability for any infringement of copyrights and/or patent rights which may arise from the processing of, and reproduction of, any of the materials herein requested. I further agree to indemnify and hold blameless the University of Waterloo from any liability which may arise from said processing or reproducing. I also acknowledge that materials processed as a result of this requisition are for educational use only.

NUMBER OF PAGES: 13

NUMBER OF COPIES: 150

TYPE OF PAPER STOCK: [ ] BOND [ ] NCR ___ PT. [X] COVER [ ] BRISTOL [ ] SUPPLIED [ ] Alpac Ivory ~~140M~~

PAPER SIZE: [ ] 8½ x 11 [ ] 8½ x 14 [ ] 11 x 17 [ ] 10x14 Glosscoat (stock) ~~10 pt.~~

PAPER COLOUR: [ ] WHITE [ ] ___

INK: [X] BLACK [ ] Rolland Tint

PRINTING: [ ] 1 SIDE ___ PGS. [X] 2 SIDES ___ PGS.

NUMBERING: FROM ___ TO ___

BINDING/FINISHING: [ ] COLLATING [ ] STAPLING [ ] HOLE PUNCHED [ ] PLASTIC RING

FOLDING/PADDING: 7x10 Saddle Stitched

CUTTING SIZE:

Special Instructions

Special Weaver Cover with Black Ink format.

COPY CENTRE: OPER. NO. ___ BLDG. ___ MACH. NO. ___

DESIGN & PASTE-UP

OPER. NO.	TIME	LABOUR CODE
5	10	D01
		D01
		D01

TYPESETTING

	QUANTITY	OPER. NO.	TIME	LABOUR CODE
PAP000000	1	3	15	T01
PAP000000				T01
PAP000000				T01

PROOF Dylux

		OPER. NO.	TIME	LABOUR CODE
PRF 011600		5	10	P01
PRF				
PRF				

NEGATIVES

	QUANTITY	OPER. NO.	TIME	LABOUR CODE
FLM 01200		5	10	C01
FLM 081000	1	5	10	C01
FLM				C01
FLM				C01
FLM				C01

PMT

	QUANTITY	OPER. NO.	TIME	LABOUR CODE
PMT				C01
PMT				C01
PMT				C01

PLATES

	QUANTITY	OPER. NO.	TIME	LABOUR CODE
PLT 111800		5	30	P01
PLT 111700	8	11	16	P01
PLT				P01

STOCK

	QUANTITY	OPER. NO.	TIME	LABOUR CODE
KR0101400	180	16	20	001
R0L101400	720	11	60	001
				001
				001

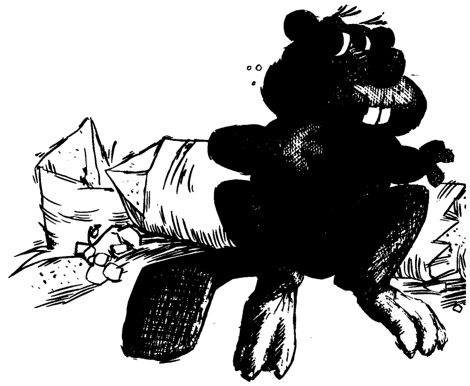
BINDERY

	QUANTITY	OPER. NO.	TIME	LABOUR CODE
RNG		1	3	B01
RNG		1	5	B01
RNG		23	5	B01
MIS00000				B01

OUTSIDE SERVICES

$ ___ COST

UNIVERSITY OF WATERLOO COMPUTER SCIENCE DEPARTMENT



A New Measure of Presortedness

Vladimir Estivill-Castro
Derick Wood

Data Structuring Group
Research Report CS-87-58

October 15, 1987

A New Measure of Presortedness *

Vladimir Estivill-Castro † Derick Wood†

October 19, 1987

Abstract

A new measure of presortedness is presented which we call $Par(X)$. It is proved that this measure is distinct from other common measures of presortedness. We design a comparison-based sorting algorithm that sorts arbitrary lists in $O(n \log n)$ comparisons and, moreover, is optimal with respect to this new measure.

1 Introduction

We present a new measure of presortedness that is derived from the study of parallel-sorting algorithms [6]. This measure which is based on the idea of presortedness presented in [3] we call $Par(X)$.

Intuitively, a list is p–sorted if, for each element x in the list, deleting the p elements immediately before and after x, the resulting list has x in its correct sorted position. We first introduce the formal definitions and the fundamental properties of $Par(X)$ in Section 2. Then, in Section 3, we compare $Par(X)$ with the measures: $Inv(X)$, the number of inversions in X; $Runs(X)$, the number of runs in X; $Exc(X) = |X|-$the number of cycles in the permutation corresponding to X, and $Rem(X)$ the minimum number of elements that need to be removed to obtain a sorted list. It is proved that $Par(X)$ is not equivalent to any of these measures. We say these functions measure presortedness since nearly sorted lists have small measure. Mehlhorn [5], who coined the term presorted, called a list presorted if it has a small $Inv(X)$ value.

If m is a measure of presortedness, an m–optimal comparison-based algorithm is an algorithm that sorts all lists, but performs particularly well for lists having small m. A definition for optimality with respect to $Inv(X)$

*This work was supported under a Natural Sciences and Engineering Research Council of Canada Grant No. A-5692

†Data Structuring Group, Department of Computer Science, University of Waterloo, WATERLOO, Ontario N2L 3G1, CANADA

was suggested by Mehlhorn in [5], the idea was studied empirically by Cook and Kim [2], but the formal concept is due to Mannila [4]. An m–optimal algorithm performs optimally with respect to a measure of presortedness if it performs as well as any sorting algorithm that uses as input not only the list X but also the value $m(X)$.

We present a Par–optimal algorithm in Section 5, while Section 4 provides a lower bound for any algorithm that sorts p–sorted lists of length n. Our algorithm is an $O(n \log n)$ comparison-based sorting algorithm that performs optimally with respect to the $Par(X)$ measure.

For a list X, $|X|$ denotes its length and for a set S, $\|S\|$ denotes its cardinality. Let $X = \langle x_1, \ldots, x_n \rangle$ and $Y = \langle y_1, \ldots, y_m \rangle$ be two lists; then their catenation is denoted by XY and is defined as $\langle x_1, \ldots, x_n, y_1, \ldots y_m \rangle$. We denote the empty list as $\langle\rangle$.

2 Definitions

Let $X = \langle x_1, x_2, \ldots, x_n \rangle$ be a list of length n of elements x_i from some linear order, that is, for all $i, j \in \{1, 2, \ldots, n\}$, $x_i \leq x_j$ or $x_j \leq x_i$. We also call X an *n–list.*

Definition 2.1 X *is p–*sorted *if and only if, for all $i, j \in \{1, 2, \ldots, n\}$, $i - j > p$ implies $x_j \leq x_i$.*

The following are immediate properties of p–sortedness:

1. If a list is p–sorted, then it is $(p + i)$–sorted, for all $i \geq 0$.
2. p–sorted does not imply $(p - 1)$–sorted.
3. A list is 0–sorted if and only if it is sorted (completely sorted).
4. The definition of being p–sorted is equivalent to:

   X is p–sorted if and only if the following two conditions are satisfied:

   (a) There is no $j < i - p$ such that $x_i < x_j$.

   (b) There is no $j > i + p$ such that $x_i > x_j$.

Definition 2.2 *Let $X = \langle x_1, \ldots, x_n \rangle$ be a list. Y is said to be an* m–subsequence *of X if $Y = \langle x_{i(1)}, x_{i(2)}, \ldots, x_{i(m)} \rangle$ and $i : \{1, 2, \ldots, m\} \rightarrow \{1, 2, \ldots n\}$ is injective and monotonically increasing. We say that Y is an* m–sublist *of a* n–list *X if $Y = \langle x_i, x_{i+1}, \ldots, x_{i+m-1} \rangle$, for some i, $1 \leq i \leq n - m + 1$.*

Igarashi and Wood [3] provided a useful equivalent local condition for p–sortedness.

Theorem 2.1 *A list X is p–sorted, if and only if, every $(2p+2)$–sublist of X is p–sorted. Moreover, for every $p \geq 0$, there is a list X satisfying the following two conditions:*

1. *X is not p–sorted.*
2. *Every $(2p+1)$–sublist of X is p–sorted.*

This theorem is the fundamental tool to prove that a given list is p–sorted.

In the examples and the following definitions we assume without loss of generality that the x_i are nonnegative integers.

We now define the new measure of presortedness.

Definition 2.3 *Let $N^{<N}$ denote the set of all finite sequences of nonnegative integers. Define Par:$N^{<N} \rightarrow N$ by:*
$Par(X) = p$ if and only if $p = min\{q | X$ is q–sorted $\}$.

The following are some basic properties of $Par(X)$:

1. For all $X = \langle x_1, x_2, \ldots, x_n \rangle$, $0 \leq Par(X) \leq n-1$.
2. $Par(X) = 0$ if and only if X is sorted.
3. For all $X = \langle x_1, x_2, \ldots, x_n \rangle$, $Par(X) = n-1$ if and only if $x_n < x_1$.

We now give the five axioms, proposed by Mannila [4], that a measure of presortedness must satisfy and we verify that Par satisfies them.

Definition 2.4 *Letting $m : N^{<N} \rightarrow N$ be some function, we say that m is* a measure of presortedness, *if and only if:*

1. *If X is in ascending order, $m(X) = 0$.*
2. *If $X = \langle x_1, x_2, \ldots, x_n \rangle$, $Y = \langle y_1, y_2, \ldots, y_n \rangle$ and $x_i \leq x_j$ if and only if $y_i \leq y_j$ for all $i, j \in \{1, 2, \ldots, n\}$, then $m(X) = m(Y)$.*
3. *If Y is a subsequence of X then $m(Y) \leq m(X)$.*
4. *If $X \leq Y$ (that is, every element of X is no greater than every element of Y), then $m(XY) \leq m(X) + m(Y)$.*
5. *For all x in N, $m(\langle x \rangle X) \leq |X| + m(X)$.*

Lemma 2.2 *$Par(X)$ is a measure of presortedness.*

Proof: We verify the five axioms for $Par(X)$.

1. This was already established as an immediate property.

2. Suppose $Par(X) = p$ and $Par(Y) = q$ and, without loss of generality, assume $p < q$. Since Y is not $(q-1)$–sorted, (q must be positive, since $0 \leq p < q$) there exist y_i and y_j such that $i - j = q$, and $y_i > y_j$. This implies $x_i > x_j$ and $i - j = q > p$. Since X is p–sorted we must have $x_i \leq x_j$, and we have obtained a contradiction.

3. Let $X = \langle x_1, \ldots, x_n\rangle$ and $Y = \langle x_{i(1)}, \ldots, x_{i(s)}\rangle$, for some s, $1 \leq s \leq n$, where $i{:}\{1, \ldots, s\} \to \{1, \ldots, n\}$ is such that $1 \leq j < k \leq s$ implies $i(j) < i(k)$. We wish to prove that $k - j > p$ implies $x_{i(j)} \leq x_{i(k)}$. Since i is and injection and monotonically increasing, $k - j \leq i(j) - i(k)$. Hence, $p < k - j$ implies $p < i(k) - i(j)$. In other words, $x_{i(j)} \leq x_{i(k)}$ as desired.

4. If $X < Y$ we claim that $Par(XY) = max\{Par(X), Par(Y)\}$. Let $Par(X) = p$, $Par(Y) = q$ and $r = max\{p, q\}$. We first prove that $Par(XY) \leq r$. We write $XY = \langle z_1, z_2, \ldots, z_{n+m}\rangle$, where $z_i = x_i$ if $1 \leq i \leq n$ and $z_i = y_{i-n}$ if $n+1 \leq i \leq n+m$. We must show that if $i - j > r$, then $z_j \leq z_i$. Suppose $i - j > r$.

Case 1: If $i \leq n$ then $j \leq i$ and $z_j = x_j \leq x_i = z_i$, because $Par(X) = p \leq r$.

Case 2: If $n+1 \leq j$, then $z_i = y_{i-n}$, $z_j = y_{j-n}$, and $(i-n)-(j-n) = i-j$. Thus $z_j \leq z_i$, because $Par(Y) = q \leq r$.

Case 3: If $j \leq n$ and $n+1 \leq i$, then $z_j \in X$ and $z_i \in Y$. Since $X \leq Y$, we conclude that $z_j \leq z_i$.

Notice that $Par(XY) \leq max\{Par(X), Par(Y)\}$ is enough to verify the axiom since $Par(X) \geq 0$ and $Par(Y) \geq 0$. The reader can now verify that $Par(XY) \geq \max\{Par(X), Par(Y)\}$.

5. By the first basic property, $Par(\langle x\rangle X) \leq |\langle x\rangle X| - 1 = |X| \leq Par(X) + |X|$, since $Par(X) \geq 0$. □

The concept of an optimal algorithm with respect to a measure of presortedness was given in a general form by Mannila [4].

Definition 2.5 *Let m be a measure of presortedness, and S a sorting algorithm which uses $T_S(X)$ comparisons on input X. We say that S is* optimal *with respect to m (or m–*optimal*) if, for some $c > 0$, we have, for all $X = \langle x_1, x_2, \ldots, x_n\rangle$:*

$$T_S(X) \leq c \cdot \max\{|X|, \log(||below(X, m)||)\}$$

where $\text{below}(X, m) = \{\pi | \pi$ *is a permutation of* $\{1, \ldots, n\}$ *and* $m(\pi(X)) \leq m(X)\}$.

3 Comparing Measures of Presortedness

We now compare our measure of presortedness with other measures.

Definition 3.1 *The number of inversions of* $X = \langle x_1, \ldots, x_n \rangle$ *is denoted by* $Inv(X)$ *and defined by:*
$Inv(X) = \|\{(i,j) | 1 \leq i < j \leq n \text{ and } x_i > x_j\}\|$.

Lemma 3.1 *For all* $X \in N^{<N}$, $Par(X) \leq Inv(X)$.

Proof: Let $Par(X) = p$. If $p = 0$, X is sorted, so $Inv(X) = 0$ and we are done.

If $p \neq 0$, since X is not $(p-1)$–sorted, there exists x_i such that $x_{i+p} < x_i$. Now consider the $p-1$ elements $x_{i+1}, x_{i+2}, \ldots, x_{i+p-1}$. If x_{i+s} with $s \in \{1, 2, \ldots, p-1\}$ is such that:

Case 1: $x_{i+s} > x_i$. Then $x_{i+s} > x_i > x_{i+p}$ and $i + p > i + s$, so we have an inversion.

Case 2: $x_{i+s} < x_i$. Then $i + s > i$ and we have an inversion.

For each $s \in \{1, 2, \ldots, p-1\}$ we have an inversion and since $x_{i+p} < x_i$, we have at least p inversions. Therefore $Inv(X) \geq p = Par(X)$. □

Lemma 3.2 *There is no* $c > 0$ *such that, for all* $X \in N^{<N}$, $Inv(X) \leq c \cdot Par(X)$.

Proof: Let $X = \langle n, n-1, \ldots, 1 \rangle$. Then $Inv(X) = n(n-1)/2$ which is $\Theta(n^2)$ while $Par(X) = n - 1$. □

Definition 3.2 *The number of maximal ascending subsequences of* X *is* $\|\{i | 1 \leq i < n \text{ and } x_{i+1} < x_i\}\| + 1$. *Since this is trivially not a measure of presortedness, we define* Runs(X) *to be this value less one.*

Lemma 3.3 *1. There is no* $c > 0$ *such that, for all* $X \in N^{<N}$, $Par(X) \leq c \cdot Runs(X)$.

2. There is no $c > 0$ *such that, for all* $X \in N^{<N}$, $Runs(X) \leq c \cdot Par(X)$.

Proof: 1. Let $X = \langle n, 2, 3, \ldots, n-1, 1 \rangle$ then $Par(X) = n-1$ but $Runs(X) = 2$.

2. Consider $X = \langle 2, 1, 4, 3, 6, 5, \ldots, n, n-1 \rangle$ then $Par(X) = 1$ while $Runs(X) = \lfloor n/2 \rfloor$. □

Definition 3.3 *The length of the largest ascending subsequence is denoted by $Las(X)$ and is defined by: $Las(X) = max\{t | \exists i(1), i(2), \ldots, i(t) \mid 1 \leq i(1) < i(2) < \cdots < i(t) \leq n$ and $x_{i(1)} < \cdots < x_{i(t)}\}$. Since $Las(X) \neq 0$ when X is sorted, we define $Rem(X) = |X| - Las(X)$. $Rem(X)$ is a measure of presortedness.*

Lemma 3.4 *For all lists X of length n, $Las(X) \geq \lceil n/(Par(X)+1) \rceil$.*

Proof: Consider $x_1, x_{1+p+1}, x_{1+2(p+1)}, \ldots, x_{1+k(p+1)}$ with $1+k(p+1) \leq n < 1+(k+1)(p+1)$. If X is p–sorted, the above sequence is in ascending order and has length $k+1$, but

$$n < 1 + (k+1)(p+1)$$

$$\Rightarrow n \leq (k+1)(p+1)$$

$$\Rightarrow n/(p+1) \leq k+1 \leq Las(X)$$

and $Las(X)$ is an integer so $Las(X) \geq \lceil n/(p+1) \rceil$.

Now consider the list $X = \langle p+1, p, p-1, \ldots, 2, 1, 2(p+1), 2(p+1)-1, \ldots \rangle$. This list is easily seen to be p–sorted and not $(p-1)$–sorted (using Theorem 2.1), so $Par(X) = p$ and $Las(X) = \lceil n/(p+1) \rceil$, proving that the bound of the above Lemma is tight. On the other hand $X = \langle p+1, 1, 2, 3, \ldots, p, 2(p+1), 2(p+1)-p, 2(p+1)-p+1, 2(p+1)-p+2, \ldots, 2(p+1)-1, 3(p+1), 3(p+1)-p, 3(p+1)-p+1, \ldots 3(p+1)-1, \ldots \rangle$ is such that $Par(X) = p$ and $Las(X) \geq p\lfloor n/(p+1) \rfloor$. This shows that, in general, we can have examples with strict inequality and with $Las(X)$ being far from $\lceil n/(Par(X)+1) \rceil$. □

Lemma 3.5 *1. $Rem(X) \leq |X|(1 - 1/(Par(X)+1))$.*

2. There is no $c > 0$ such that, for all $X \in N^{<N}$ $Par(X) \leq c \cdot Rem(X)$.

3. There is no $c > 0$ such that, for all $X \in N^{<N}$ $Rem(X) \leq c \cdot Par(X)$.

Proof: 1. $Rem(X) = |X| - Las(X) \leq |X| - |X|/(Par(X)+1)$.

2. Let $X = \langle n, 2, 3, \ldots, n-1, 1 \rangle$ then $Rem(X) = 2$ while $Par(X) = n-1$.

3. Let $X = \langle 2, 1, 4, 3, \ldots \rangle$ then $Rem(X) = \lfloor n/2 \rfloor$ but $Par(X) = 1$. □

Definition 3.4 *We now consider $Exc(X) = n-$ the number of cycles in the permutation of $\{1, 2, \ldots, n\}$ corresponding to X. Exc is also a measure of presortedness.*

Lemma 3.6 *There is no $c > 0$ such that, for all $X \in N^{<N}$, $Par(X) \leq c \cdot Exc(X)$ and there is no $d > 0$ such that, for all $X \in N^{<N}$, $Exc(X) \leq d \cdot Par(X)$.*

Proof: Let $X = \langle n, 2, 3, \ldots, n-1, 1\rangle$ then the cycles of the permutation are $(1\ n)(2)(3)\ldots(n-1)$ and then $Exc(X) = 1$ while $Par(X) = n-1$, on the other hand if $X = \langle 2, 1, 4, 3, 6, 5, \ldots, n, n-1\rangle$ we have $Par(X) = 1$ but $Exc(X) \geq \lfloor n/2 \rfloor$. □

We have compared the measure $Par(X)$ with the most common measures of presortedness and have shown:

Theorem 3.7 *$Par(X)$ is not equivalent to any of the measures of presortedness $Inv(X), Rem(X), Runs(X)$ or $Exc(X)$, but, for all $X \in N^{<N}$:*

1. *$Par(X) \leq Inv(X)$.*
2. *$Rem(X) \leq |X|(1 - 1/(Par(X)+1))$.*

We conclude that $Par(X)$ measures global presortedness and does not recognize local presortedness. In this sense $Par(X)$ is similar to $Inv(X)$.

4 A Lower Bound

We claim that any comparison-based algorithm that sorts any p–sorted list of length n requires $\Omega(\max\{n, n\log(p+1)\})$ comparisons.

To show this, consider the set

$$A_i = \{1+i(p+1), 2+i(p+1), 3+i(p+1), \ldots, p+i(p+1), p+1+i(p+1)\}$$

Let B_i be any permutation of the elements of A_i. We build a list X by catenation of the B_i:

$$X = B_0 B_1 B_2 \cdots B_{\lfloor n/(p+1)\rfloor - 1}$$

It can be directly verified that X is p–sorted. Further, this shows that there are at least $(p+1)!^{\lfloor n/(p+1)\rfloor}$ p–sorted lists. Hence, we conclude:

Theorem 4.1 *There are at least $(p+1)!^{\lfloor n/(p+1)\rfloor}$ n–lists X with $Par(X) \leq p$.*

Therefore, any comparison-based algorithm that sorts p–sorted lists requires at least $\lfloor n/(p+1)\rfloor \log((p+1)!)$ comparisons, that is, $\Omega(n\log(p+1))$ comparisons.

5 An Optimal Algorithm

We propose a comparison-based algorithm called the *Try–to–Merge Sort* that given a list as input produces the corresponding sorted list as output.

Try–to–Merge Sort is an $O(n \log n)$ worst-case sorting algorithm, but if the input list X is p–sorted, then we can certify that the algorithm requires $O(\lceil \log(p+1)+1 \rceil n)$ comparisons.

Letting $X = \langle x_1, \ldots, x_n \rangle$ define : $X_{even} = \langle x_2, x_4, \ldots, x_{2\lfloor n/2 \rfloor} \rangle$ and $X_{odd} = \langle x_1, x_3, \ldots, x_{2\lceil n/2 \rceil} \rangle$. We claim:

Theorem 5.1 *1. If X is p-sorted then X_{even} and X_{odd} are $\lfloor p/2 \rfloor$-sorted. Moreover, for every $p > 1$, there is a p-sorted list X such that X_{even} and X_{odd} are not $(\lfloor p/2 \rfloor - 1)$-sorted.*

2. For any $n \in N$, there is a list X such that $Par(X) > n$ and X_{even}, X_{odd} are both sorted.

Proof: 1. Let b_i and b_j be elements of X_{even} such that $j - i > \lfloor p/2 \rfloor$. We must prove that $b_j \geq b_i$. Now $b_j = x_{2j}$, $b_i = x_{2i}$, and $2j - 2i = 2(j-i)$. Since $j - i$ is an integer, $j - i > p/2$, and we obtain $2j - 2i > 2(p/2) = p$. Thus, since X is p–sorted, $x_{2j} \geq x_{2i}$, that is, $b_j \geq b_i$. The claim for X_{odd} is proved similarly.

Now, let $p > 1$. If p is odd, let $X = \langle x_1, x_2, \ldots, x_{p+3} \rangle$, where $x_i = 2 + i$, for $i = 1, 2, \ldots, p-1$, $x_p = 1$, $x_{p+1} = 2$, $x_{p+2} = 3+p+2$, and $x_{p+3} = 3+p+3$, and if p is even, let $X = \langle x_1, \ldots, x_{p+4} \rangle$, where $x_i = 2 + i$, for $i = 1 \ldots p$, $x_{p+1} = 1$, $x_{p+2} = 2$, $x_{p+3} = 3 + p + 3$, and $x_{p+4} = 3 + p + 4$. We claim that X is p-sorted and X_{even} and X_{odd} are not $(\lfloor p/2 \rfloor - 1)$–sorted.

Suppose p is even, $(p = 2k)$, x_1 and x_{p+1} belong to X_{odd}, more precisely, x_1 is the first element of X_{odd} and x_{p+1} is the $\lceil (p+1)/2 \rceil$ element in X_{odd}. But $\lceil (p+1)/2 \rceil - 1 = k > \lfloor p/2 \rfloor - 1$, and $x_1 = 2 + 1 = 3 > 1 = x_{p+1}$, hence X_{odd} is not $(\lfloor p/2 \rfloor - 1)$–sorted. All other cases are verified similarly.

2. Finally, let $n \in N$ and define $X = \langle x_1, \ldots x_{2n+3} \rangle$ by: $x_{2i} = i$, for $i = 1, 2, \ldots, n+1$, and $x_{2i-1} = n + 2i$, for $i = 1, 2, \ldots, n+2$. It can be verified that $Par(X) = 2n + 1$ and X_{even} and X_{odd} are sorted. □

We assume that we have a boolean function $merge(X_1, X_2, X)$ that attempts to merge the two lists X_1 and X_2 as if they are sorted. If X_1 and X_2 are sorted, it returns *true* and their merge is X, otherwise it returns *false* and X is undefined. If the input lists X_1 and X_2 have lengths n_1 and n_2, then the merging algorithm has complexity $O(n_1 + n_2)$.

5.1 Sorting Algorithm

Try–to–Merge Sort(X:list, n:integer) { $|X| = n$ }
Input: $X = \langle x_1, x_2, \ldots, x_n \rangle$.
Output: The elements in X in ascending order.

begin

if $merge(X_{even}, X_{odd}, X)$ **then** $\{successful\}$
else
begin
Try–to–Merge Sort$(X_{even}, \lfloor n/2 \rfloor)$;
Try–to–Merge Sort$(X_{odd}, \lceil n/2 \rceil)$;
$merge(X_{even}, X_{odd}, X)$
end
end

5.2 Algorithm Correctness

We assume that the procedure that performs the *merge* operation is correct.

Lemma 5.2 *For all lists X of length n,* Try–to–Merge Sort(X, n) *returns X in sorted order.*

Proof: We prove correctness by induction on the length of X.

Basis: If $|X| = 0$ then $X = \langle\rangle$, so X_{even} and X_{odd} are also empty, and as $merge(X_{even}, X_{odd}, X)$ is true, it yields $X = \langle\rangle$. Clearly the empty list is the correct result.

If $|X| = 1$ then $X_{odd} = X$ and $X_{even} = \langle\rangle$, $merge(X_{even}, X_{odd}, X)$ is true. It returns X_{odd} which is the correct result since a list of length one is always sorted.

Induction Step: Assume that the algorithm works correctly for input lists of length less than n, and we are given X of length $n \geq 2$. By the assumptions about procedure $merge(X_1, X_2, X)$ we have two cases according to whether this procedure returns *true* or *false*.

Case 1: If the merge is successful, that is, procedure *merge* returns *true*, then it returns X is sorted order.

Case 2: If the merge is unsuccessful, the algorithm performs the 'else' part. Since $n \geq 2$, $n > \lceil n/2 \rceil \geq \lfloor n/2 \rfloor$ and, by the induction hypothesis, the recursive calls to *Try–to–Merge Sort* return X_{even} and X_{odd} in sorted order. Hence, in the inner call to procedure *merge* they will be merged successfully producing as output the lists of elements in X in sorted order.

□

5.3 Algorithm Complexity

We now show that if we execute *Try–to–Merge Sort*(X,n), where n is the length of the list X, then the algorithm performs $O(n \log n)$ comparisons in the worst case.

By a worst case analysis the number of comparisons that the algorithm performs satisfies the recurrence relation:
$T(1) = 1$
$T(n) = 2T(n/2) + cn$, for some constant $c > 0$.
This has growth rate $\Theta(n \log n)$; see [1].

5.4 Par(X)–optimality

We claim that

Lemma 5.3 *If X is $(2^i - 1)$–sorted, then the maximum depth of recursion of* Try–to–Merge Sort$(X, |X|)$ *is i.*

Proof: We prove this claim by induction on i.

Basis: If $i = 0$ then X is $2^0 - 1 = 0$–sorted. In this case X is sorted, so X_{even} and X_{odd} are sorted, therefore the merging is successful. X is returned in ascending order and no recursive calls are made.

Induction Step: Assume that, for some $i \geq 1$, if $k < i$ and X is $(2^k - 1)$–sorted, then the depth of recursion of the call *Try–to–Merge Sort*(X,$|X|$) is no greater than k.

Let X be $(2^i - 1)$–sorted, and suppose we call *Try–to–Merge Sort*(X,$|X|$). If the merging on the odd and even subsequences of X is successful, then we produce the desired result with no recursive calls. But, in the worst case, the merging is unsuccessful and we call recursively:

1. *Try–to–Merge Sort*(X_{even},$\lfloor |X|/2 \rfloor$)
2. *Try–to–Merge Sort*(X_{odd},$\lceil |X|/2 \rceil$)

By Theorem 5.1 X being $(2^i - 1)$–sorted implies that X_{even} is $\lfloor \frac{2^i - 1}{2} \rfloor$–sorted, that is, X_{even} is $\lfloor 2^{i-1} - 1/2 \rfloor = (2^{i-1} - 1)$–sorted, and similarly X_{odd} is $(2^{i-1} - 1)$–sorted. By the induction hypothesis the above two recursive calls have a depth of recursion of at most $i - 1$. Therefore the maximum depth of recursion of the original call is bounded by i.

This completes our proof. $\square$

Using this lemma, and the observation that at each level of recursion we perform at most $|X|$ comparisons, we conclude:

Theorem 5.4 *If X is p–sorted and $|X| = n$, then* Try–to–Merge Sort(X, n) *requires $O(\lceil \log(p + 1) + 1 \rceil n)$ comparisons in the worst case.*

To prove that *Try–to–Merge Sort* is *Par*–optimal we need:

Lemma 5.5 *There is a $d > 0$ such that, for all $n > 1$ and for all m with $1 < m \leq n$,*

$$\log(m) + 1 \leq d\lfloor n/m \rfloor \frac{\log(m!)}{n}$$

Proof: We claim that $d = \max\{8, 4\frac{\log(3)+1}{\log(3)-1}\}$

Let $k = \lfloor n/m \rfloor$. Then $n/m < k+1$, so $1/n > 1/m(k+1)$, $n/m \geq 1$, and $k/(k+1) \geq 1/2$. Hence,

$$d\lfloor n/m \rfloor \frac{\log(m!)}{n} > d\frac{k}{k+1}\frac{\log(m!)}{m} \geq \frac{d}{2}\frac{\log(m!)}{m}$$

Case 1: $m = 2$.

Since $d \geq 8$, $\frac{d}{2}\frac{\log(m!)}{m} \geq 2 = \log(m) + 1$ as claimed.

Case 2: $m \geq 3$.

$$\frac{d}{2}\frac{\log(m!)}{m} = \frac{d}{2}\log((m!)^{1/m}) > \frac{d}{4}(\log(m) - \log 2) = (d/4)(\log(m) - 1)$$

By the definition of d and because $m \geq 3$

$$(d/4)(\log(m) - 1) \geq \frac{\log(3)+1}{\log(3)-1}(\log(m) - 1) \geq \log(m) + 1$$

□

Theorem 5.6 Try–to–Merge Sort *is Par–optimal.*

Proof: Let $T_{DMS}(X)$ be the number of comparisons that *Try–to–Merge Sort* performs on input X. Since, by Lemma 4.1, there are at least $(Par(X)+1)!^{\lfloor |X|/(Par(X)+1) \rfloor}$ lists in $below(X, Par)$, we have

$$\lfloor |X|/(Par(X)+1) \rfloor \log((Par(X)+1)!) \leq \log(||below(X, Par)||) \qquad (1)$$

and by Theorem 5.4 there is an $e > 0$ such that, for all $X \in N^{<N}$,

$$T_{DMS}(X) \leq e(\log(Par(X)+1)+1)|X| \qquad (2)$$

Let $X \in N^{<N}$.

Case 1: $Par(X) = 0$. Since X is sorted if and only if $Par(X) = 0$

$$\log(||below(X, Par)||) = \log(||\{X\}||) = \log(1) = 0$$

Moreover, in this case, the algorithm performs a successful merge. Hence, there is a $c_1 > 0$ independent of X such that

$$T_{DMS}(X) \leq c_1|X| = c_1 \max\{|X|, \log(||below(X, Par)||)\}$$

Case 2: $Par(X) > 0$. By Lemma 5.5, there is a $d > 0$ such that, for all X where $n = |X|$ and $m = Par(X) + 1$,

$$\log(Par(X) + 1) + 1 \leq d\lfloor\frac{|X|}{Par(X) + 1}\rfloor\frac{\log((Par(X) + 1)!)}{|X|}$$

Thus, there is a $d > 0$ such that

$$e(\log(Par(X)+1)+1)|X| \leq e \cdot d\lfloor\frac{|X|}{Par(X) + 1}\rfloor \log((Par(X)+1)!) \quad (3)$$

and by (1), (2), and (3), we conclude that there is a $c_2 = e \cdot d > 0$ such that

$$T_{DMS}(X) \leq c_2 \max\{|X|, \log(||below(X, Par)||)\}$$

Setting $c = \max\{c_1, c_2\}$, we conclude that there is a $c > 0$ such that, for all $X \in N^{<N}$,

$$T_{DMS}(X) \leq c \max\{|X|, \log(||below(X, Par)||)\}$$

and the theorem is proved. □

References

[1] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley Publishing Co., Reading, Mass., 1974.

[2] C.R. Cook and D.J. Kim. Best sorting algorithms for nearly sorted lists. *Communications of the ACM*, 23:620–624, 1980.

[3] Y. Igarashi and D. Wood. *Roughly Sorting — A Generalization of Sorting.* Technical Report CS-87-, Department of Computer Science, University of Waterloo, 1987.

[4] H. Mannila. Measures of presortedness and optimal sorting algorithms. *IEEE Transactions on Computers*, C-34:318–325, 1985.

[5] K. Mehlhorn. Sorting presorted files. In *4th GI Conference on Theory of Computer Science*, pages 199–212, Springer-Verlag, New York, 1979.

[6] K. Sado and Y. Igarashi. *A Parallel Pseudo-Merge Sort on a Mesh-Connected Processor Array.* Technical Report CS-85-3, Department of Computer Science, Gunma University, 1985.