SORTING PERMUTATIONS WITH NETWORKS OF STACKS

M.D. Atkinson

TR-210, AUGUST 1992

School of Computer Science, Carleton University
Ottawa, Canada, KIS 5B6

Sorting Permutations with Networks of Stacks

M.D. Atkinson [*]
School of Computer Science
Carleton University, Ottawa
CANADA K1S 5B6

In his paper [3] R.E. Tarjan introduced the problem of sorting permutations with networks of stacks and queues. He made significant progress in cases where the nodes of the network were all queues and began the study of networks of stacks (see also [2]). In the latter case his results were not so complete and he observed that "many questions are unanswered". Since that time there appears to have been no progress on the problem although it is mentioned by Knuth in [1, pp169-170]. In this paper we introduce an infinite permutation group associated with a network of stacks and study its structure. We then specialise to the case of a network of k stacks in series and give some extensions of Tarjan's work.

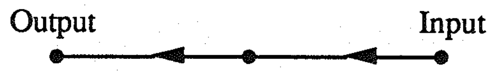
Formally, a network of stacks is a directed graph whose nodes are stacks. An edge from node S to node T signifies the possibility of popping the top element from stack S and pushing it onto stack T. Such an operation is called a *transfer*. In the network there are two distinguished nodes: the input node and the output node. It is assumed that there is at least one directed path from the input to the output. A permutation is said to be sortable by the network if, when placed in the input stack (first symbol at the top, last symbol at the bottom), there is a sequence of transfers which results in the output stack containing all the symbols in increasing order from bottom to top.

In Tarjan's paper the input and output nodes were defined to be queues. However, in interesting networks, the in-degree of the input and out-degree of the output are each zero and whether they are stacks or queues is immaterial. By taking them to be stacks we achieve some economy of notation.

The simplest non-trivial network is

---

[*] This research was supported by the National Science and Engineering Research Council of Canada under Grant No. A4219



The permutations $\sigma=[s_1, s_2,..., s_n]$ sortable by this network are usually called *stack sortable* permutations and are characterised by the condition that there is no triple $i<j<k$ such that $s_j>s_i>s_k$. Another way of phrasing this condition is to say that no subsequence of σ is ordered like [2,3,1], or that σ does not contain the pattern [2,3,1]. The number of stack sortable permutations on 1,2,...,n is well-known to be the n th Catalan number

$$c_n = \frac{1}{n+1}\binom{2n}{n}$$

We adopt the following notation. Stacks will be denoted by upper-case letters and the corresponding subscripted lower case letter will be used to denote the positions of the stack. Thus $s_1,s_2,....$ denote the positions of the stack S (s_1 being the top position).

Consider a directed edge (S,T) in a general network of stacks. When an element is transferred from S to T all other elements of S move one position towards the top of S and all elements of T move one position further down so a transfer from S to T corresponds to the infinite permutation $\pi(S,T)$ defined by

$$s_i \rightarrow s_{i-1},\ i>1$$

$$s_1 \rightarrow t_1$$

$$t_i \rightarrow t_{i+1},\ i>1$$

(all positions in other stacks being fixed).

The *group of the network* is defined to be the group generated by all permutations $\pi(S,T)$ for all edges (S,T) in the network. It is convenient to extend the $\pi(S,T)$ notation to every pair of nodes S,T in the network to mean the operation of transferring an element from the top of S onto T whether or not S and T are connected by an edge. This operation has the same permutational description as the one above. It is easy to verify that $\pi(S,T) = \pi(T,S)^{-1}$ and that $\pi(S,T)\pi(T,U) = \pi(S,U)$. We shall say that a network is *connected* if it is connected as an undirected graph. For connected networks $\pi(S,T)$ belong to the group for every pair S,T of nodes.

THEOREM 1 If G is the group of a connected network with n>2 nodes then

1. The derived group G' is the restricted symmetric group on the set of all stack positions, and

2. G/G' is a free abelian group of rank n-1.

Proof. If S,T,U,V are any 4 distinct nodes then, by direct computation of commutators,

$[\pi(S,T),\pi(U,V)] = 1$, and

$[\pi(S,T),\pi(T,U)] = (t_1,u_1)$, a transposition.

Since G' is generated by the conjugates of all commutators of generators of G and each of theses commutators and their conjugates are finitary permutations G' is a group of finitary permutations.

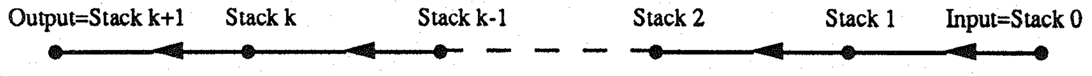
Furthermore, G' contains every transposition. To see this, let S,T,U be any 3 nodes. As above, $(t_1,u_1)\in G'$. Conjugating this element by powers of $\pi(S,U)$ shows that every transposition $(t_1,u_i)\in G'$. Then conjugating these elements by powers of $\pi(S,T)$ and $\pi(T,U)$ shows that G' contains every (t_i,t_j) and (t_i,u_j). Thus G' is the restricted symmetric group.

Let E be a set of edges which forms a spanning tree of the network. The relations $\pi(S,T)\pi(T,U) = \pi(S,U)$ prove that G is generated by the n-1 elements $\pi(S,T)$ with $(S,T)\in E$. To prove that their images in G/G' generate G/G' freely suppose that there was some relation

$$\prod_{(S,T)\in F}\pi(S,T)^{f(S,T)} \in G'$$

where F is a subset of E with all $f(S,T)\neq 0$. The graph formed by F is a forest so there is some node T incident with only one edge $(S,T)\in F$. Then $\pi(S,T)$ maps each t_i to t_{i+1} and every other $\pi(U,V)$ with $(U,V)\in F$ fixes each of $t_1,t_2,\ldots$. Hence $\prod_{(S,T)\in F}\pi(S,T)^{f(S,T)}$ is not a finitary permutation and so no relation of the stipulated form can exist.

We now turn our attention to series networks where the network topology is as shown:



DEFINITION A permutation σ is said to be *k-stack sortable* if it can be sorted by a series network of k+2 stacks (k internal stacks, that is, k stacks not including the input or output stack).

The definition is less anomalous than it seems since the input and output stacks play little part in sorting the permutation. Notice that 1-stack sortability is precisely the same as stack sortability. In our discussion of k-stack sortable permutations greek letters will denote permutations (not always of $\{1,2,\ldots,n\}$).

LEMMA 1 A permutation is k-stack sortable if and only if it is the product of k 1-stack sortable permutations.

Proof. If a permutation $\pi=[p_1,p_2,\ldots,p_n]$ is input to a stack and is output in the order $\rho=[r_1,r_2,\ldots,r_n]$ then $\rho=\pi\sigma$ (permutation product) where the permutation σ depends only on the particular sequencing of push and pop operations and not on π. The permutations σ that a stack can induce are precisely the inverses of 1-stack sortable permutations since π is sortable if and only if there is some σ for which ρ is the identity permutation.

For a network of k+2 stacks in series the output to each stack becomes the input to the next stack and so an input permutation π is related to an output permutation by an equation $\rho=\pi\sigma_1\sigma_2\ldots\sigma_k$ where each σ_i is the inverse of a 1-stack sortable permutation. The k-stack sortable permutations are precisely those for which the equation $\pi\sigma_1\sigma_2\ldots\sigma_k = 1$ can be solved.

COROLLARY The number of k-stack sortable permutations is at most $c_n{}^k \approx 4^{nk}$, where c_n is the nth Catalan number.

Any algorithm to sort a permutation may be described by the sequence of stack movements that it requires. Let X_i denote the operation of transferring an element from the top of stack i-1 to the top of stack i. A word in $X_1,\ldots,X_{k+1}$ will represent an algorithm if and only if it contains equal numbers of each symbol X_i and if, in every initial segment, the number of symbols X_i is never less than the number of symbols X_{i+1} ($i=1,2,\ldots,k$). (The latter condition is the one that ensures that no operation attempts to remove a symbol from an empty stack). We shall call these words *well-formed.* If w is a well-formed word of length (k+1)n we may define an associated Young tableau with k+1 rows of length n; in the ith row we place the positions of occurrence of the symbol X_i within w. Conversely any Young tableau with k+1 rows of length n defines a well-formed word in $X_1,\ldots,X_{k+1}$. The hook formula from the theory of Young tableaux allows us to determine the exact number of well-formed words.

LEMMA 2 There are $\frac{(kn+n)!2!3!...k!}{n!(n+1)!...(n+k)!}$ well-formed words. For fixed k and $n \to \infty$ this is $O\left((k+1)^{(k+1)n}\right)$.

We can see from this lemma that, whereas for k=1 there is a one-to-one correspondence between well-formed words and k-stack sortable permutations, for k>1 there are very many more algorithms than permutations. It is often possible to demonstrate that two well-formed words sort the same permutation by using relations that hold in the group of the network. Some simple relations are:

$X_i X_{i+j} = X_{i+j} X_i$, if $j>1$

$X_{i-1} X_i X_{i+1} X_i = X_i X_{i-1} X_i X_{i+1}$

$X_{i-1} X_i^2 X_{i+1} = X_i X_{i+1} X_{i-1} X_i$

There are many others, too many it seems to make this the basis of a method for testing when two words sort the same permutation.

To each well-formed word $w=w(X_1,...,X_{k+1})$ there is a well-formed word $w^*=w(X_{k+1},...,X_2,X_1)^R$ (where u^R denotes the reverse of u) and $w^{**}=w$.

LEMMA 3 If σ is the k-stack sortable permutation corresponding to the well-formed word w then $\theta\sigma^{-1}\theta$ is the permutation that corresponds to w^* where θ is the involution $i \to n+1-i$.

Proof. In an input $[p_1,p_2,...,p_n]$ p_1 is at the top of the input stack while for an output $[r_1,r_2,...,r_n]$ r_1 is at the bottom of the output stack. Suppose that the permutation σ carries input permutations $\pi=[p_1,p_2,...,p_n]$ to output permutations $\rho=[r_1,r_2,...,r_n]$ by the equation $\rho=\pi\sigma$. Then the permutation σ^*, which corresponds to w^*, carries $[r_n,r_{n-1},...,r_1]=\rho\theta$ to $[p_n,p_{n-1},...,p_1]=\pi\theta$, so $\rho\theta\sigma^* = \pi\theta = \rho\sigma^{-1}\theta$. Hence $\sigma^*=\theta\sigma^{-1}\theta$.

The central question about k-stack sortable permutations is: for which values of n is every permutation on n symbols k-stack sortable? To investigate this question we introduce the concept of criticality. We shall say that an integer n is k-*critical* (or simply *critical* if the value of k need not be emphasised) if there exists a permutation of n symbols which is not k-stack sortable but every permutation of fewer than n symbols is k-stack sortable. In addition, if n is k-critical then any permutation of n symbols which is not k-stack sortable will also be called k-critical.

It is well known that both permutations of {1,2} can be 1-stack sorted and that [2,3,1] is the only permutation of {1,2,3} which cannot be 1-stack sorted. Thus 3 is 1-critical and [2,3,1] is the unique 1-critical permutation.

Tarjan [3] reported that 7 was 2-critical (though not with this terminology) and, by computer search, we have proved

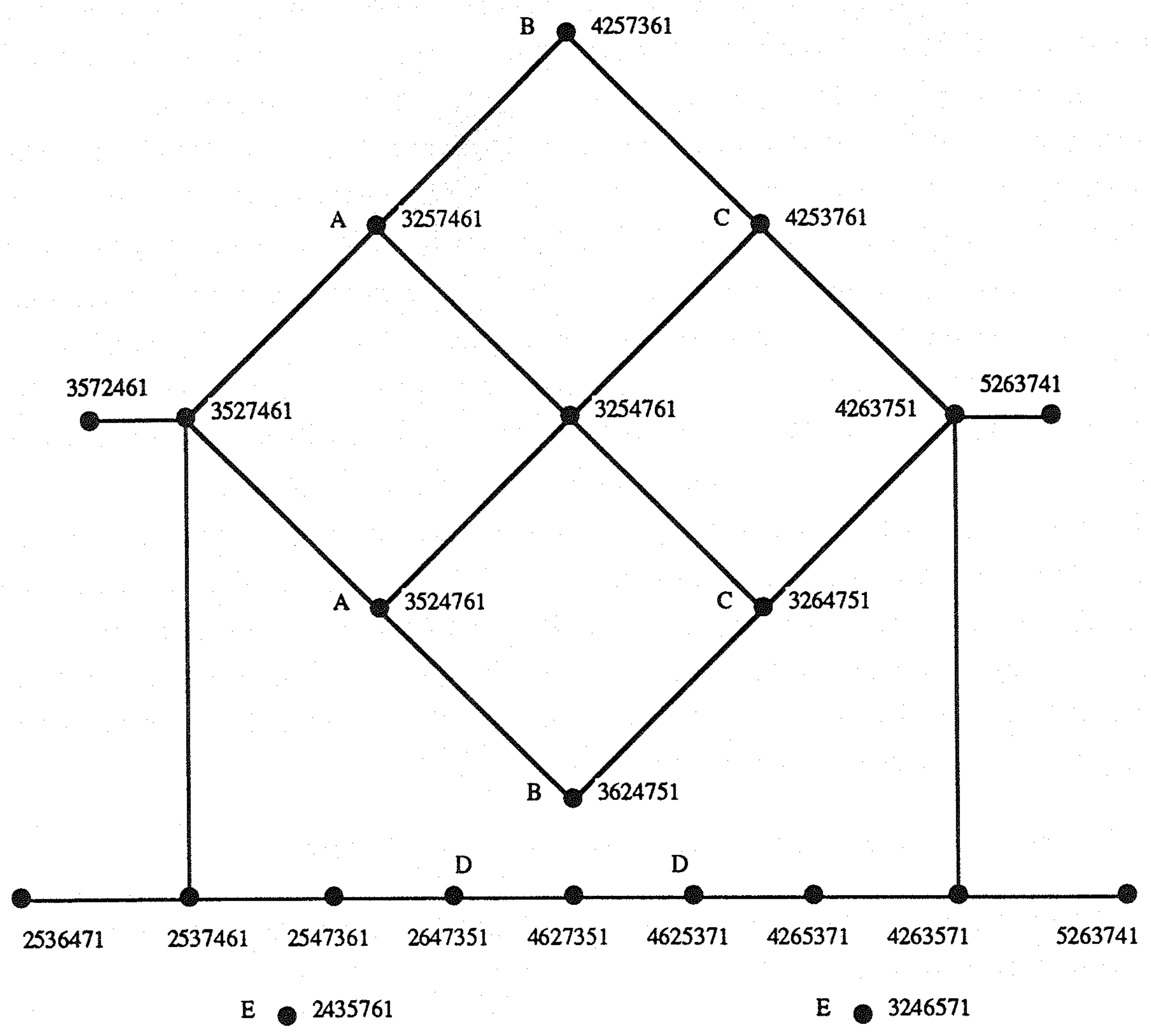
THEOREM 2 There are precisely 22 2-critical permutations and they are

2435761	2536471	2537461	2463571	2547361	2647351
3246571	3254761	3264751	3257461	4253761	4263571
4263751	5263741	4257361	4265371	3524761	3624751
3527461	4625371	4627351	3572461		

(for brevity we have written 2435761 instead of [2,4,3,5,7,6,1] etc.)

Permutations which cannot be 1-stack sorted are precisely those which contain some pattern [2,3,1], the unique 1-critical permutation. This phenomenon does not extend to larger numbers of stacks. The permutation [5,2,7,4,6,1,8,3] is not 2-stack sortable yet it contains none of the 22 2-critical patterns.

The 22 2-critical permutations are drawn in graph form below. Two permutations are joined by an edge if one can be obtained from another by interchanging two symbols. The symmetry of order 2 is the one corresponding to the operation w$\rightarrow$ w*.



This graph S is, of course, a subgraph of the 5040 node graph T of all permutations of degree 7. In T every node has degree 21 and so a random 22 node subgraph of T would have an expected number of edges $\frac{22\times21\times21}{2\times5039} < 1$. The fact that S is so much denser than a random subgraph is some indication of the similarities among the 22 2-critical permutations.

The k-critical values for $k\geq3$ are unknown. Despite this it is possible to find properties of k-critical permutations for general values of k as we show below.

THEOREM 3 Critical permutations of $\{1,2,...,n\}$ end in 1.

Proof. We begin by showing that a critical permutation cannot have 1 as its penultimate symbol. Let σ be k-critical and of the form $\mu1t$. The permutation $\mu1$ can be sorted using k stacks. The algorithm that does this will eventually output t-1 and, since 1 is the first symbol that is output, all of $\mu 1$ will have been read at this point. Thus if we insert, at this stage of the algorithm, instructions $X_1 X_2 \ldots X_{k+1}$ which move the next input symbol directly to the output, we shall obtain an algorithm that sorts σ. This contradicts the k-criticality of σ. Next we show that if $\sigma = \alpha u 1 \beta$ is sortable with k stacks then so is $\sigma' = \alpha 1 u \beta$. The sorting algorithm for $\alpha u 1 \beta$ begins with an initial sequence of operations $r(X_1,X_2,\ldots,X_k)X_1$ where the last X_1 places u in the first stack (the word r contains no X_{k+1} because the symbol 1 has not yet been ouput). Without loss of generality the algorithm may be taken to next transfer symbol 1 through each stack in turn so that the entire sorting algorithm for $\alpha u 1 \beta$ has the form $r(X_1,X_2,\ldots,X_k)X_1\ X_1\ X_2 \ldots X_{k+1}\ s(X_1,\ldots, X_{k+1})$. But now $r(X_1,X_2,\ldots,X_k)\ X_1\ X_2 \ldots X_{k+1}\ X_1\ s(X_1,\ldots, X_{k+1})$ sorts σ'. The proof of the theorem is now completed by a reverse induction showing that critical permutations cannot have 1 in position j, j=n-1,...,2,1. The base of this induction was established above and the inductive step is justified as follows. Assume that for j=n-1, n-2,..., h+1 any permutation with 1 in place j is not critical (that is, in this case, any such permutation is sortable) and consider a permutation $\sigma = \alpha 1 u \beta$ with its 1 in place h. Since $\alpha u 1 \beta$ has 1 in place h+1 it is sortable and so therefore is σ.

THEOREM 4 Let σ be a k-critical permutation of $\{1,2,\ldots,n\}$. Then

1.If $2 \leq b < n$ σ cannot end in a permutation of $\{1,2,3,\ldots,b\}$. In particular, the penultimate symbol of σ cannot be 2,

2.If $0 \leq i < k$, n-i cannot occur among the first k-i positions of σ,

3.If k is even σ cannot contain an adjacent pair i,i+1 and if k is odd σ cannot contain an adjacent pair i+1,i.

Proof.

1. Suppose to the contrary that there is a k-critical permutation of the form $\sigma = \alpha\beta$ where β is a permutation of $\{1,\ldots,b\}$ (and α is a permutation of $\{b+1,\ldots,n\}$). Now consider $\sigma' = \alpha b$, a permutation of fewer symbols, which, since σ is critical, is k-stack sortable. The sorting algorithm for σ' cannot transfer any symbol of α to the output stack before first transferring b from the input stack (because b is smaller than any symbol of α). At this point there is no loss in generality in assuming that b is transferred directly through each stack in turn to the output stack, and so the sorting algorithm for σ' has the form $r(X_1,X_2,\ldots,X_k)X_1X_2\ldots X_{k+1}s(X_2,\ldots,X_k)$. However β is also k-stack sortable (since

$b \neq n$) and has a sorting algorithm $t(X_1,\ldots,X_{k+1})$. Then, clearly, $r(X_1,X_2,\ldots,X_k)t(X_1,\ldots,X_{k+1})s(X_2,\ldots,X_k)$ is a sorting algorithm for σ which is impossible. Finally, the penultimate symbol of σ cannot be 2 otherwise, by the previous theorem, σ would end in a permutation of $\{1,2\}$.

2. Suppose $0 \leq i < k$ and n-i occurs among the first k-i positions, that is, σ has the form α,n-i,β with at most k-i-1 symbols in α. The permutation $\alpha\beta$ is k-stack sortable and we consider its sorting algorithm A. After all the symbols of α have been transferred from the input there will be a set U of at least i+1 empty internal stacks. Later in the algorithm just after n-i-1 has been placed on the output stack only i symbols remain to be output and so one of the stacks S_u in the set U will be empty. A sorting algorithm for α,n-i,β can be devised as follows. We follow the algorithm A until all the symbols of α have been transferred from the input stack, transfer n-i from the input through each stack in turn until it reaches the empty stack S_u, resume algorithm A until n-i-1 has been output, transfer n-i (now the unique symbol on stack S_u) through each stack in turn until it reaches the output, and finally resume and complete algorithm A

3. If k is even and $\sigma=\alpha i,i+1,\beta$ is critical then $\alpha i\beta$ is k-stack sortable. We apply its sorting algorithm to σ except that whenever it is called upon to transfer i from one stack to the next we transfer instead the pair $\{i,i+1\}$. After an even number of such moves the pair $\{i,i+1\}$ is in the right order. The case k odd is similar; here the pair $\{i+1,i\}$ would be interchanged an odd number of times and would again end in the right order.

Knuth [1, p169] observed that if every permutation on m symbols was (k-1)-stack sortable then every perrmutation on 2m symbols would be k-stack sortable. Since we shall use some variations on his basic method we outline it here along with some observations that will be useful below.

If every permutation on m symbols is (k-1)-stack sortable then, given an arbitrary input permutation σ and arbitrary output permutation τ, there will be an algorithm (the one that sorts $\sigma\tau^{-1}$) which produces τ from σ in a series network of k+1 stacks. Now consider an arbitrary permutation σ of 2m symbols and a series network of k+2 stacks $S_0,S_1,\ldots,S_{k+1}$. the first m symbols of σ can be sorted in ascending order from top to bottom on stack S_k (regarding S_k as the output stack of a series network of k+1 stacks). The remaining symbols of σ are (k-1)-stack sortable but rather than sort them on stack S_k we transfer each one from S_k to S_{k+1} immediately it is placed on S_k. As we do this we interleave transfers of the first m symbols of σ from S_k to S_{k+1} in such a way as to merge all the symbols onto S_{k+1} in sorted order.

In the next theorem we use results about critical permutations to improve on Knuth's method.

THEOREM 5 If every permutation on m symbols can be sorted with k-1 stacks then every permutation on n=2m+1 symbols can be sorted with k stacks.

Proof. Suppose for a contradiction that the theorem is false. Since every permutation on 2m symbols can be sorted with k stacks n must be k-critical. Let σ be a critical permutation of 1..n (which necessarily ends in 1). We shall derive our contradiction by showing that σ is k-sortable. Note that m+1 is (k-1)-critical. If the first m+1 symbols of σ could be reverse (k-1)-stack sorted we would be able to sort σ so we may assume that this is not so and therefore σ_{m+1} is larger than its predecessors. We now sort the first m symbols into the last stack smallest at the top. Next we stack σ_{m+1} in the first stack. If we ignored σ_{m+1} it would be possible to sort the remaining symbols by the Knuth sort-merger process achieving any desired order for the symbols not among the first m symbols of σ; the ordering we actually require is one where the symbols less than σ_{m+1} are in increasing order and the symbols greater than it are in decreasing order. We do this process up to the point that it has output $\sigma_{m+1} - 1$. So now the input is exhausted and the final stack is empty, and it is possible to sort the remaining symbols (save for σ_{m+1}) in decreasing order without using the last stack except for to transfer directly to the output. Rather than directing the symbols to the output we direct them to the last stack. Eventually we reach a point where the symbols are in reverse order on the last stack except for σ_{m+1} which is still on the first stack. It is now an easy matter to finish off the process of sorting σ.

COROLLARY With k stacks $2^{k-2}.7 - 1$ symbols can be sorted.

Proof. Let $u_2 = 6$ and, for k>2, $u_k = 2u_{k+1}+1$. Then the previous result shows that with k stacks u_k symbols may be sorted. However, it is easily shown by induction that $u_k = 2^{k-2}.7 - 1$.

Tarjan [3] reported that he had found a permutation on 41 symbols which cannot be sorted with 3 internal stacks. We now describe a permutation on 38 symbols which cannot be sorted with 3 internal stacks. We require the following auxiliary permutation:

LEMMA 4 The permutation [2,3,1,5,6,4] cannot be sorted by a series network with 3 internal stacks by any algorithm which has an interim stage where all the symbols are in stack 2.

Proof. Suppose it were possible to sort this permutation in the manner described. Then after symbol 1 has been transferred from the input stack into stack 1 not all of 2,3, and 1 can be in stack 1 (otherwise their order in stack 1 would be 1,3,2 from top to bottom, and when they were transferred all to stack 2 their order in stack 2 would be 2,3,1 from top to bottom; but the permutation [2,3,1] cannot be sorted by the remaining stack). So there exists a symbol, x say, among {1,2,3} which must have reached stack 2 before the symbol 5 is transferred from the input. According to the hypothesis the symbols 5,6,4 must all reach stack 2 before symbol x has been transferred from it. However 5,6,4 is not 1-stack sortable and so the order of the symbols {x,4,5,6} in stack 2 cannot be 6,5,4,x from top to bottom. Therefore there are two symbols $y<z$ among {4,5,6} which have y closer to the top of stack 2 than z. So stack 2 will contain a pattern y,z,x from top to bottom which is not sortable using the remaining internal stack.

Note: Computer searches have shown that [2,3,1,5,6,4] is the only permutation on 6 symbols with this property and there are 38 such when n=7.

LEMMA 5 There exists a permutation on 38 points which cannot be 3-stack sorted.

Proof. Consider the pattern π=[1,8,3,5,2,7,9,6,4]. This contains both the pattern [2,3,1,5,6,4] (as [3,5,2,7,9,6]) and the pattern [1,6,4,2,7,5,3] (as [1,8,5,2,9,6,4]). The latter pattern is the reversal of one of the 22 2-unsortable permutations. Suppose that this pattern occurs within some 3-stack sortable permutation. The algorithm that sorts the permutation cannot, at any interim stage, have all the symbols of π on stack 2 (by Lemma 4). Nor can there be an interim stage where all of the symbols of π are on stack 1 (otherwise, top to bottom, stack 1 would contain a pattern [3,5,7,2,4,6,1] and such a permutation cannot, by Theorem 2, be sorted by the remaining two internal stacks).

We shall show that the permutation

τ=[2,7,5,3,8,6,4; 16,21,19,17,22,20,18; 9,14,12,10,15,13,11;
30,37,32,34,31,36,38,35,33; 23,28,26,24,29,27,25; 1]

cannot be sorted by a series network of 3 internal stacks. The punctuation in this permutation gives some guidance to its structure. It is of the form $[\alpha_2,\alpha_4,\alpha_3,\alpha_6,\alpha_5,\alpha_1]$ where $\alpha_2,\alpha_4,\alpha_3,\alpha_5$ are each of pattern [1,6,4,2,7,5,3] which is a reversal of one of the 22 2-unsortable permutations in Lemma ???, α_6 is of pattern π, and $\alpha_1=1$. Moreover the subscript order 2,4,3,6,5,1 is in accord with the relative values of the symbols in $\alpha_2,\alpha_4,\alpha_3,\alpha_6,\alpha_5,\alpha_1$.

Suppose, for a contradiction, that τ is 3-stack sortable. We first prove that just before the first symbol of α_6 is transferred into the first stack at least one symbol of $\alpha_2\alpha_4\alpha_3$ must be in stack 3. To prove this suppose the contrary, that all the symbols of $\alpha_2\alpha_4\alpha_3$ are in stacks 1 and 2. Just before the first symbol of α_4 was transferred to stack 1 not all the symbols of α_2 can be on stack 1 otherwise stack 1 contains a permutation that when input to the two remaining stacks can be sorted:- however, from top to bottom, this permutation has pattern [3,5,7,2,4,6,1] which is impossible. Therefore stack 2 contains a symbol of α_2. By precisely the same argument it subsequently receives a symbol of α_4 and then a symbol of α_3. But now stack 2 contains a permutation, top to bottom, of shape [2,3,1] and this is impossible to sort with the remaining internal stack.

The presence in stack 3 of one of the symbols of $\alpha_2\alpha_4\alpha_3$ is a strong restriction on the subsequent operation of a sorting algorithm. It implies that stack 3 cannot receive any symbol of $\alpha_6\alpha_5$ until the whole of the permutation has been removed from the input stack (for it is only after this happens that symbol 1 can be output so, before the whole permutation has been read, stack 3 must retain a symbol of $\alpha_2\alpha_4\alpha_3$ on top of which no larger symbol can be placed).

According to the remarks at the beginning of the proof at least one symbol of α_6 must be transferred from stack 1 to stack 2 before the first symbol of α_5 is read. We now prove that, when the first symbol of α_6 is placed in stack 2, stack 2 cannot, in fact, contain any symbol of $\alpha_2\alpha_4\alpha_3$. Suppose that it did contain some such symbol, z say, and that a symbol y of α_6 was placed on top of it. We would obtain a contradiction as follows. The symbols z,y would have to remain in stack 2 at least until all the symbols of α_5 had been removed from the input stack. But before all the symbols of α_5 have been removed from the input stack one of them, x say, must be placed into stack 2 (since not all the symbols of α_5 can be simultaneously in stack 1). At this point stack 2 would contain the symbols x,y,z (top to bottom) which have pattern [2,3,1] and these symbols cannot be sorted using the remaining internal stack 3.

We can now deduce that, when the first symbol of α_6 is placed in stack 2, stack 3 must contain not just one symbol from $\alpha_2\alpha_4\alpha_3$ but at least one symbol from each of α_2, α_4, α_3. The reason is that symbols of α_2, α_4, α_3 have all previously passed through stack 2 and so must be in stack 3.

Next we prove that when the first element of α_6 is placed on stack 2 not all of the symbols of α_2 are in stack 3. Suppose that they were. Before the symbols of α_6 were removed from the input stack, stack 2 had already received at least one symbol from each of α_4 and α_3; these symbols (x and y say) have left stack 2 when the first symbol of α_6 is placed there so must be in stack 3. This implies that the subsequence 7,5,3,8,6,4,x (and the subsequence 7,5,3,8,6,4,y) of τ have been placed in decreasing order on stack 3. But it is impossible to reverse-sort the pattern [5,3,1,6,4,2,7] in this way since [3,5,7,2,4,6,1] is not 2-stack sortable.

At the point that the first symbol of α_6 is placed in stack 2 there must be at least one symbol $x \in \alpha_2$ which is on stack 1. Suppose that this symbol is transferred to stack 2 before all the symbols of α_6 have been removed from the input stack. By the argument of the preceding paragraph x can proceed no further until some of the symbols of stack 3 have been output and this cannot happen until all the input stack has been cleared. So, by the time that the symbols of α_5 have all been removed from the input stack one of them, y say, will have been transferred to stack 2 (they cannot all be in stack 1) and will be above x. But it is now impossible to sort τ since y can proceed no further until stack 3 is emptied, and this cannot happen because it would require x to be output and x lies below y in stack 2. It follows therefore that x cannot be placed in stack 2 before all the symbols of α_6 have been transferred from the input stack. Hence stack 2 will now contain some of the symbols of α_6 and the others will be above x in stack 1. It is impossible to complete the sorting of τ from this configuration. No symbol of α_6 can be placed in stack 3 until that stack has output all its current symbols from $\alpha_2\alpha_4\alpha_3$ and this cannot happen until x also has been output. But to remove x from stack 1 would require that all the symbols of α_6 were on stack 2 and this is impossible.

Acknowledgement. Part 2 of Theorem 4 was observed and proved by Edward Norminton with whom I had several productive conversations in the course of this work

References

1. S. Even, A. Itai, Queues, stacks and graphs, in Theory of Machines and Computations, (Z. Kohavi, A. Paz, editors) (Proceedings of International Symposium on the Theory of Machines and Computations, Technion - Israel Inst. of Technol., Haifa, Israel, August 1971), Academic Press, New York, 1971, pp. 71-86.

2. D.E. Knuth, Sorting and Searching, The Art of Computer Programming vol. 3, Addison-Wesley, Reading, Massachusetts, 1972.

3. R.E. Tarjan, Sorting using networks of queues and stacks, Journal of the ACM 19 (1972), 341-346.

School of Computer Science, Carleton University
Recent Technical Reports

TR-169 **Trade-Offs in Non-Reversing Diameter**
Hans L. Bodlaender, Gerard Tel and Nicola Santoro, February 1990

TR-170 **A Massively Parallel Knowledge-Base Server using a Hypercube Multiprocessor**
Frank Dehne, Afonso Ferreira and Andrew Rau-Chaplin, April 1990

TR-171 **Parallel Processing of Quad Trees on the Hypercube (and PRAM)**
Frank Dehne, Afonso Ferreira and Andrew Rau-Chaplin, April 1990

TR-172 **A Note on the Load Balancing Problem for Coarse Grained Hypercube Dictionary Machines**
Frank Dehne and Michel Gastaldo, May 1990

TR-173 **Self-Organizing Doubly-Linked Lists**
R.S. Valiveti and B.J. Oommen, May 1990

TR-174 **A Presortedness Metric for Ensembles of Data Sequences**
R.S. Valiveti and B.J. Oommen, May 1990

TR-175 **Separation of Graphs of Bounded Genus**
Ljudmil G. Aleksandrov and Hristo N. Djidjev, May 1990

TR-176 **Edge Separators of Planar and Outerplanar Graphs with Applications**
Krzystof Diks, Hristo N. Djidjev, Ondrej Sykora and Imrich Vrto, May 1990

TR-177 **Representing Partial Orders by Polygons and Circles in the Plane**
Jeffrey B. Sidney and Stuart J. Sidney, July 1990

TR-178 **Determining Stochastic Dependence for Normally Distributed Vectors Using the Chi-squared Metric**
R.S. Valiveti and B.J. Oommen, July 1990

TR-179 **Parallel Algorithms for Determining K-width-Connectivity in Binary Images**
Frank Dehne and Susanne E. Hambrusch, September 1990

TR-180 **A Workbench for Computational Geometry (WOCG)**
P. Epstein, A. Knight, J. May, T. Nguyen, and J.-R. Sack, September 1990

TR-181 **Adaptive Linear List Reorganization under a Generalized Query System**
R.S. Valiveti, B.J. Oommen and J.R. Zgierski, October 1990

TR-182 **Breaking Substitution Cyphers using Stochastic Automata**
B.J. Oommen and J.R. Zgierski, October 1990

TR-183 **A New Algorithm for Testing the Regularity of a Permutation Group**
V. Acciaro and M.D. Atkinson, November 1990

TR-184 **Generating Binary Trees at Random**
M.D. Atkinson and J.-R. Sack, December 1990

TR-185 **Uniform Generation of Combinatorial Objects in Parallel**
M.D. Atkinson and J.-R. Sack, January 1991

TR-186 **Reduced Constants for Simple Cycle Graph Separation**
Hristo N. Djidjev and Shankar M. Venkatesan, February 1991

TR-187 **Multisearch Techniques for Implementing Data Structures on a Mesh-Connected Computer**
Mikhail J. Atallah, Frank Dehne, Russ Miller, Andrew Rau-Chaplin, and Jyh-Jong Tsay, February 1991

TR-188 **Generating and Sorting Jordan Sequences**
Alan Knight and Jörg-Rüdiger Sack, March 1991

TR-189 **Probabilistic Estimation of Damage from Fire Spread**
Charles C. Colbourn, Louis D. Nel, T.B. Boffey and D.F. Yates, April 1991

TR-190	**Coordinators: A Mechanism for Monitoring and Controlling Interactions Between Groups of Objects** Wilf R. LaLonde, Paul White, and Kevin McGuire, April 1991
TR-191	**Towards Decomposable, Reusable Smalltalk Windows** Kevin McGuire, Paul White, and Wilf R. LaLonde, April 1991
TR-192	**PARASOL: A Simulator for Distributed and/or Parallel Systems** John E. Neilson, May 1991
TR-193	**Realizing a Spatial Topological Data Model in a Relational Database Management System** Ekow J. Otoo and M.M. Allam, August 1991
TR-194	**String Editing with Substitution, Insertion, Deletion, Squashing and Expansion Operations** B John Oommen, September 1991
TR-195	**The Expressiveness of Silence: Optimal Algorithms for Synchronous Communication of Information** Una-May O'Reilly and Nicola Santoro, October 1991
TR-196	**Lights, Walls and Bricks** J. Czyzowicz, E. Rivera-Campo, N. Santoro, J. Urrutia and J. Zaks, October 1991
TR-197	**A Brief Survey of Art Gallery Problems in Integer Lattice Systems** Evangelos Kranakis and Michel Pocchiola, November 1991
TR-198	**On Reconfigurability of Systolic Arrays** Amiya Nayak, Nicola Santoro, and Richard Tan, November 1991
TR-199	**Constrained Tree Editing** B. John Oommen and William Lee, December 1991
TR-200	**Industry and Academic Links in Local Economic Development: A Tale of Two Cities** Helen Lawton Smith and Michael Atkinson, January 1992
TR-201	**Computational Geometry on Analog Neural Circuits** Frank Dehne, Boris Flach, Jörg-Rüdiger Sack, Natana Valiveti, January 1992
TR-202	**Efficient Construction of Catastrophic Patterns for VLSI Reconfigurable Arrays** Amiya Nayak, Linda Pagli, Nicola Santoro, February 1992
TR-203	**Numeric Similarity and Dissimilarity Measures Between Two Trees** B. John Oommen and William Lee, February 1992
TR-204	**Recognition of Catastrophic Faults in Reconfigurable Arrays with Arbitrary Link Redundancy** Amiya Nayak, Linda Pagli, Nicola Santoro, March 1992
TR-205	**The Permutational Power of a Priority Queue** M.D. Atkinson and Murali Thiyagarajah, April 1992
TR-206	**Enumeration Problems Relating to Dirichlet's Theorem** Evangelos Kranakis and Michel Pocchiola, April 1992
TR-207	**Distributed Computing on Anonymous Hypercubes with Faulty Components** Evangelos Kranakis and Nicola Santoro, April 1992
TR-208	**Fast Learning Automaton-Based Image Examination and Retrieval** B. John Oommen and Chris Fothergill, June 1992
TR-209	**On Generating Random Intervals and Hyperrectangles** Luc Devroye, Peter Epstein and Jörg-Rüdiger Sack, July 1992
TR-210	**Sorting Permutations with Networks of Stacks** M.D. Atkinson, August 1992