Pop-Stacks in Parallel

M.D. Atkinson
J.-R. Sack

December 12, 2001

1 Introduction

A stack may be viewed as a machine with two instructions POP and PUSH that transforms an input sequence of data items into an output sequence. The PUSH operation transfers the next item of an input stream into the stack and the POP operation transfers the most recently pushed item from the stack to an output stream. When the entire input stream has been transferred, via the stack, to the output stream, the output sequence will then be a permutation of the input sequence.

The set of possible permutations which arise in this way characterises the effect of a stack on its data and was first studied by Knuth [4]. Shortly afterwards Tarjan [7], Even and Itai [3], and Pratt [5] studied the effect of having several stacks either in series or in parallel. Since the actual values of the data items are unimportant (only that they be distinct) it is usual to denote them by $1, 2, \ldots$, although occasionally other data sets need to be considered. It is convenient to formulate the central questions that arise in terms of sorting:

1. Which permutations can be sorted into increasing order?
2. Can the sortable permutations be characterised by a finite forbidden set of permutations?
3. How many permutations of each length n can be sorted?

In the case of a single stack answers to these questions have been known for many years. Stack sortable permutations can be recognised in linear time, they are characterised by the single forbidden permutation $[2, 3, 1]$, and there are $\binom{2n}{n}/(n+1)$ sortable permutations of each length n.

The case of k stacks in series is very much harder and no satisfactory answers are known for any of the three questions above for $k \geq 2$. In response to this difficulty Avis and Newborn [1] defined a less powerful object which they called a 'pop-stack'. In a pop-stack PUSH operations are as usual but the POP operation unloads the entire stack (in the last-in, first-out manner). They introduced the respective terms *feasible* and *m-feasible* for those permutations that could be sorted by an unlimited number of pop-stacks in series and by a fixed number m of pop-stacks in series. Their main results comprised rather complex summation formulae to answer the enumeration problem for both the feasible and the m-feasible permutations. Very recently, in work which was not targeted at stacks, Bose *et al.* [2] introduced a class of permutations which they called *separable*. A permutation σ is separable if either

1. σ is the permutation [1], or
2. σ can be written as a concatenation $\alpha\beta$ where either α is a separable permutation of the symbols $1, 2, \ldots, m$ for some $1 \leq m < n$ and β is a separable permutation of $m+1, \ldots, n$ or β is a separable permutation of the symbols $1, 2, \ldots, m$ for some $1 \leq m < n$ and α is a separable permutation of $m+1, \ldots, n$.

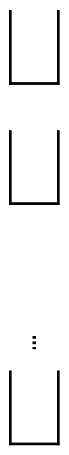
By comparing [1] and [2] it is easy to see that the separable permutations are precisely the feasible permutations and one can conclude that these permutations are characterised by the forbidden permutations $[2, 4, 1, 3]$ and $[3, 1, 4, 2]$. It then follows from [6] that the feasible permutations are enumerated by the Schröder numbers. These numbers were originally defined as the number of ways of inserting non-overlapping chords into a convex polygon but they also have various other interpretations (for more background on these numbers which occur in other generalizations of the stack structure see [4]).

For the corresponding problem of k stacks in parallel the situation is rather different. Thanks to work of Unger [8, 9] it is known that the sortable permutations can be recognised in time $O(n \log n)$ if $k \leq 3$ but that recognition is NP-complete for larger values of k. In addition, from [7, 3], it is known that, if $k \geq 2$, no finite set of forbidden permutations can characterise the sortable ones. Moreover the enumeration problem is unsolved for $k \geq 2$.

Motivated by these results we shall consider k pop-stacks in parallel (see Figure 1). We observe that, for any k, the recognition problem requires only linear time, and we prove, also for any k, that the sortable permutations are characterised by a finite forbidden set. Finally, we give an enumeration result for $k = 2$.

For clarity we use the term "MPOP" (multiple POP) for the operation of unloading an entire pop-stack.

It is also notationally convenient to consider permutations which can be sorted into decreasing order. This, of course, produces an equivalent problem in which the symbols $1, 2, \ldots, n$ are simply renamed $n, n-1, \ldots, 1$. So, from now on, we consider a system of k pop-stacks in parallel receiving data from an input



Proof We shall examine the conditions that must hold if σ cannot be sorted by a canonical series of PUSH and MPOP operations. We shall see that σ must contain a bounded set of symbols that are ordered in one of a certain number of ways; and, conversely, any set of symbols ordered in one of these ways cannot be sorted.

Suppose that the canonical series of PUSH and MPOP operations reaches a point where no further PUSH or MPOP can occur but has not terminated. All k stacks must be non-empty and we may number the stacks so that the ith stack has content $b_i \dots t_i$ where

1. b_i and t_i are, respectively, the bottom and top elements of the ith stack; and $b_i \dots t_i$ is a sequence of consecutive increasing integers.
2. $t_{i-1} < b_i$ for each $i = 2, \dots, k$

We let m denote the symbol which should next be output; note that m is not present in any of the stacks and so each b_i must precede m in the input.

When b_i was stacked it was placed in an empty stack and therefore $b_i - 1$ was not present in any of the stacks. Therefore b_i must have preceded $b_i - 1$ in the input (unless $i = 1$ and $b_1 = 1$). Note that any input sequence on the set $\{b_1, \dots, b_k, b_1 - 1, \dots, b_k - 1, m\}$ (where the symbols are not necessarily distinct) with the property that $b_1 < b_2 < \dots < b_k$, each b_i precedes m, and, for $i > 1$, each b_i precedes $b_i - 1$ would lead to a situation where, before m was stacked, all stacks would contain exactly one of the b_i.

Now let q denote the next input symbol. Since m is not one of the top of stack symbols (i.e. $m \neq t_k$) m is either the symbol q or comes after q in the input. In addition, since q cannot be stacked, either q must precede $q - 1$ or q=1. Conversely, if these conditions hold and none of the stacks is empty then no further operation is possible.

We have proved that if a permutation σ is not k-sortable then it contains a bounded set of symbols satisfying certain sequencing and ordering constraints; and conversely, if σ has a set of symbols with these constraints then it is not k-sortable. The (necessarily finite) set of permutations of these symbols that fulfil these constraints therefore has the property that one of them occurs within a permutation σ if and only if σ is not k-sortable.

This proof is constructive and, in principle, allows the forbidden sets to be computed. For example, for $k = 2$, the minimal forbidden set is

$$\{[3,2,1,4],[2,1,4,3],[2,4,1,3,5],[4,1,3,5,2],[1,4,3,5,2],[1,3,5,4,2],[1,3,5,2,4]\}$$

3 Enumeration

In this section f_n denotes the number of k-sortable permutations of length n. We shall mainly focus on the case $k = 2$ and therefore suppress notational references to k. However, we begin with a result, Lemma 1, that holds for every value of k. We define a permutation to be *reduced* if it does not contain any segment (i.e. a subsequence of consecutively placed symbols) of the form $i, i+1$ and we let g_n denote the number of reduced k-sortable permutations.

Furthermore, if $a_1 \dots a_n$ is any sequence of distinct integers we define its *normalized* form to be the permutation which arises by relabelling the items with the elements of $\{1, \dots, n\}$ so as to preserve order. For example, the normalized form of $[4, 1, 6, 3]$ is $[3, 1, 4, 2]$.

Lemma 1 $f_n = \sum_{m=1}^{n} \binom{n-1}{m-1} g_m$

Proof Every sortable permutation of length n may be written as $\sigma = [\alpha_1, \dots, \alpha_m]$ where each α_i is a maximal segment of consecutive increasing integers of length ℓ_i. The permutation σ is uniquely determined by $\ell_1, \dots, \ell_m$ together with the permutation $\bar{\sigma}$ that is the normalized form of any $[a_1, \dots, a_m]$ where $a_i \in \alpha_i$ (the normalized form is independent of the choice of $a_i \in \alpha_i$). Note that $\bar{\sigma}$ is reduced and that for a fixed reduced $\bar{\sigma}$ there are $\binom{n-1}{m-1}$ permutations $\sigma = [\alpha_1, \dots, \alpha_m]$ as the ℓ_i are allowed to vary subject to $\ell_1 + \dots + \ell_m = n$. The required equation now follows by allowing the length of $\bar{\sigma}$ to vary over all possibilities.

Lemma 2 *If $k = 2$ the sequence (g_n) is defined by the conditions $g_1 = 1, g_2 = 1, g_3 = 3$ and the recurrence $g_n = 3g_{n-1} - g_{n-2} + g_{n-3}$.*

Proof In the algorithm that sorts a reduced permutation σ, the first symbol to be output is n and just before this occurs both stacks will contain a sequence of consecutive integers increasing from bottom to top. One stack will contain $n-s+1, \dots, n$ for some s and the other $x_1, x_2, \dots$. These two sequences must be interleaved in some way to form an initial segment of σ and, since σ is reduced, the interleaving must be alternate. It follows that σ has an initial segment of one of the following forms:

A. $\sigma = [x_1, n-s+1, x_2, \dots, x_s, n, x_{s+1}, y \dots]$ with $s \geq 1$

B. $\sigma = [n-s+1, x_2, \dots, x_s, n, x_{s+1}, y, \dots]$ with $s \geq 1$

C. $\sigma = [x_1, n-s+1, x_2, \dots, x_s, n, y, \dots]$ with $s \geq 1$

D. $\sigma = [n-s+1, x_2, \dots, x_s, n, y, \dots]$ with $s \geq 2$

In all the above forms the element y denotes an integer that does not extend the increasing sequence of consecutive integers $x_1, x_2, \dots$.

We now compute, for each fixed value of s, the numbers $a_{ns}, b_{ns}, c_{ns}, d_{ns}$ of permutations of each of these forms.

A. For $s > 1$ the processes of inserting and removing the first two symbols give a one-to-one correspondence between permutations of this type and permutations $[x_2, n-s+2, x_3, \ldots, x_s, n, x_{s+1}, y, \ldots]$ of the set $\{1, \ldots, n\} \setminus \{x_1, n-s+1\}$. A permutation of the latter type, when normalized, has the form $[x_1, n-s, x_2, \ldots, x_{s-1}, n-2, x_s, y, \ldots]$ and is reduced. This shows that $a_{ns} = a_{n-2,s-1}$ and therefore, by induction, $a_{ns} = a_{n-2s+2,1}$.

   On the other hand, a_{n1} enumerates permutations of the type $[x_1, n, x_2, y, \ldots]$ and these are in one-to-one correspondence with permutations $[x_1, y, \ldots]$ on the set $\{1, \ldots, n\} \setminus \{n, x_2\}$ (here the one-to-one correspondence is obtained by inserting and removing the second and third symbols). The latter permutations are again sortable, and reduced upon nomalisation; therefore there are g_{n-2} such permutations.

   Hence $a_{ns} = a_{n-2s+2,1} = g_{n-2s}$.

B. There is a one-to-one correspondence between permutations of type A and permutations $[n-s+1, x_2, \ldots, x_s, n, x_{s+1}, y, \ldots]$ of the set $\{1, \ldots, n\} \setminus \{x_1\}$. These permutations, when normalized, are of type B, but of length $n-1$. Therefore $a_{ns} = b_{n-1,s}$ and hence $b_{ns} = g_{n-2s+1}$

C. Suppose first that $s > 2$. Then there is a one-to-one correspondence between permutations of this type and reduced sortable permutations $[x_2, n-s+2, x_3, \ldots, x_s, n, y, \ldots]$ of the set $\{1, \ldots, n\} \setminus \{x_1, n-s+1\}$ (these permutations do remain reduced when normalized since $x_2 < x_s < n-s+2$). This proves that $c_{ns} = c_{n-2,s-1}$ whenever $s > 2$ and therefore $c_{ns} = c_{n-2s+4,2}$.

   However c_{n2} enumerates permutations of the form $[x_1, n-1, x_2, n, y, \ldots]$ and these are in one-to-one correspondence with permutations $[x_1, y, \ldots]$ of $\{1, \ldots, n\} \setminus \{x_2, n-1, n\}$ which are also sortable and reduced when normalized. Thus $c_{n2} = g_{n-3}$ and, for all $s \geq 2$, $c_{ns} = g_{n-2s+1}$.

   The above argument breaks down when $s = 1$ because, for the permutations $[x_1, n, y, \ldots]$ enumerated by c_{n1}, we have the extra condition that $x_1 \neq n-1$. Thus these sequences are in one-to-one correspondences with the reduced sortable permutations $[x_1, y, \ldots]$ of length $n-1$ where $x_1 \neq n-1$. Since the permutations which do have $x_1 = n-1$ are in one-to-one correspondence with reduced sortable sequences of length $n-2$ it follows that $c_{n1} = g_{n-1} - g_{n-2}$.

D. The permutations of this type are in one-to-one correspondence with permutations $[n-s+2, x_3, \ldots, x_s, n, y, \ldots]$ on $\{1, \ldots, n\} \setminus \{n-s+1, x_2\}$ (so long as $s > 2$) and these are sortable and, on normalisation, reduced. Hence $d_{ns} = d_{n-2,s-1} = d_{n-2s+4,2}$. However the permutations enumerated by d_{n2} are of the form $[n-1, x_2, n, y, \ldots]$ which correspond to permutations $[x_2, y, \ldots]$ of length $n-2$ and there are g_{n-2} of these. Thus $d_{ns} = g_{n-2s+2}$

We can now complete the proof. We have

$$\begin{aligned} g_n &= \sum_{s\geq 1}(a_{ns} + b_{ns} + c_{ns} + d_{n,s+1}) \\ &= \sum_{s>1}(g_{n-2s} + g_{n-2s+1} + g_{n-2s+1} + g_{n-2s}) \\ &\quad + g_{n-2} + g_{n-1} + g_{n-1} - g_{n-2} + g_{n-2} \\ &= 2\sum_{r=1}^{n-1} g_{n-r} - g_{n-2} \end{aligned}$$

By differencing we now obtain $g_n - g_{n-1} = 2g_{n-1} - g_{n-2} + g_{n-3}$ as required.

Theorem 3 *If $k=2$ the sequence (f_n) is defined by the conditions $f_1 = 1, f_2 = 2, f_3 = 6$ and the recurrence $f_n = 6f_{n-1} - 10f_{n-2} + 6f_{n-3}$*

PROOF By the previous lemma $g_m = \sum_i a_i r_i^m$ where r_1, r_2, r_3 are the roots of $x^3 - 3x^2 + x - 1 = 0$ and a_1, a_2, a_3 are constants. By Lemma 1

$$\begin{aligned} f_n &= \sum_{m=1}^{n} \binom{n-1}{m-1} \sum_i a_i r_i^m \\ &= \sum_i a_i \sum_{m=1}^{n} \binom{n-1}{m-1} r_i^m \\ &= \sum_i a_i r_i (1+r_i)^{n-1} \end{aligned}$$

(by interchanging the order of summations and applying the binomial theorem). The numbers $1 + r_i$ satisfy the equation

$$0 = (x-1)^3 - 3(x-1)^2 + (x-1) - 1 = x^3 - 6x^2 + 10x - 6$$

and the theorem follows.

Obtaining a similar enumeration theorem for f_n in the case of 3 or more pop-stacks seems to be difficult since the form of the reduced permutations is more complex. Nevertheless, we conjecture that, for any fixed k, f_n satisfies a linear recurrence with constant coefficients.

References

[1] D. Avis and M. Newborn: On pop-stacks in series, Utilitas Mathematica, 19 (1981), 129–140.

[2] P. Bose, J.F. Buss, A. Lubiw: Pattern matching for permutations, Information Processing Letters 65 (1998), 277–283.

[3] S. Even and A. Itai: Queues, stacks and graphs, in Z. Kohavi and A. Paz, eds., Theory of Machines and Computations, Proc. Internat. Symp. on the Theory of Machines and Computations, Technion – Israel Inst. of Technol., Haifa, Israel, August 1971 (Academic Press, New York, 1971) 71–86.

[4] D.E. Knuth: *Fundamental Algorithms, The Art of Computer Programming* Vol. 1 (Second Edition), Addison-Wesley, Reading, Mass. (1973).

[5] V.R. Pratt: Computing permutations with double-ended queues, parallel stacks and parallel queues, Proc. ACM Symp. Theory of Computing 5 (1973), 268–277.

[6] L. Shapiro, A.B. Stephens: Bootstrap percolation, the Schröder number, and the N-kings problem, SIAM J. Discrete Math. 2 (1991), 275–280.

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

[8] W. Unger: The complexity of colouring circle graphs, Proceedings 9th Annual Symposium on Theoretical Aspects of Computer Science, 1992, Springer Lecture Notes in Computer Science 577, 389–400.

[9] W. Unger: On the k-colouring of circle graphs, Proceedings 5th Annual Symposium on Theoretical Aspects of Computer Science, 1988, Springer Lecture Notes in Computer Science 294, 61–72.