Order **10:** 31–36, 1993.

Transforming Binary Sequences Using Priority Queues

M. D. ATKINSON
Department of Mathematical and Computational Sciences, North Haugh, St. Andrews, Fife KY16 9SS, Scotland

Communicated by N. Zaguia

(Received: 10 November 1992; accepted: 20 March 1993)

Abstract. A priority queue transforms an input sequence σ into an output sequence τ which is a re-ordering of the sequence σ. The set R of all such related pairs is studied in the case that σ is a binary sequence. It is proved that R is a partial order and that $|R| = c_{n+1}$, the $(n+1)$th Catalan number. An efficient $(O(n^2))$ algorithm is given for computing the number of outputs achievable from a given input.

Mathematics Subject Classification (1991). 06A06.

Key words: Priority queue, binary sequence, enumeration.

1. Introduction

Abstract data types are a fundamental design tool in modern software systems. Although there is an infinity of possible data types there is a small number only of them which recur frequently in algorithm design (stacks, arrays, queues, dictionaries etc.) suggesting that some data types are more fundamental than others. Many of these classical data types are *Container* data types: they are holders for collections of data items and support an *Insert* operation and a *Delete* operation (often restricted in some sort of way).

In practice, a container data type is used as follows. It is initialised as being empty. Then some sequence of data items is inserted into it and a resequencing of these data items is generated by deleting items from the data type. The insertions and deletions may be interleaved (in any way, subject to the obvious restriction that a *Delete* operation is not allowed when the data type is empty). In effect, a container data type is a mechanism for transforming an input sequence into an output sequence. The functional behaviour of such a data type is essentially characterised by the relationship between the input sequences and the output sequences. An understanding of this relationship allows us to judge the capabilities of a container data type and to assess its potential applications. In general, if σ is an input sequence that gives rise to an output sequence τ then we shall say that (σ, τ) is *allowable* and speak of the *allowability* relation. The statistics of the allowability relation are

measures of the transformational capability of the data type.

In the case of a queue (where the delete operation always removes the element which has been in the queue the longest) the allowability relation is trivial since the output sequence is always the same as the input sequence. In the case of stacks (where the delete operation always removes the element which was most recently placed in the stack) the relation is much more interesting and has given rise to combinatorial connections with trees, ballot sequences, Young tableau, and triangulations of polygons (see [1], [2]). For the dictionary data type (which has an unrestricted Delete operation) the output can be any permutation of the input.

In this paper we shall consider the case of priority queues. Priority queues are characterised by two main operations *Insert* and *Delete-Minimum*. Several attractively efficient implementations of them are known; typically $O(\log n)$ operations are required for each of *Insert* and *Delete-Minimum* where n is the number of elements currently in the priority queue. Our focus here, however, is on the allowability relation R of a priority queue. The case of input and output sequences which are permutations of length n was considered in [4]. It was shown that the number of allowable pairs is $(n+1)^{n-1}$. Furthermore the following two functions were investigated

1. $t(\sigma) = |\{\tau : (\sigma, \tau) \in R\}|$,
2. $s(\tau) = |\{\sigma : (\sigma, \tau) \in R\}|$.

It was shown how to compute these functions efficiently and their combinatorial properties were studied. The allowability relation R is reflexive, and antisymmetric, but not transitive. In fact, in [3], it was proved that $R^{n-2} \neq R^{n-1} = W$, the weak order on the symmetric group.

We shall study the opposite extreme to permutation inputs (where all the elements are distinct): input sequences which are binary sequences. In this case, it turns out that the allowability relation is transitively closed and therefore is a partial order. We describe this order in two different ways:- by a partial summation condition and as the closure of a set of covering relations. Using this description we can enumerate the number of allowable pairs of binary sequences of length n:- it is c_{n+1}, the $(n+1)$th Catalan number. Then we can deduce, through natural symmetries of the partial order, various relations between the functions $t(\sigma)$ and $s(\tau)$. Finally we give efficient $(O(n^2))$ algorithms for computing the functions s, t.

2. The Partial Order

We define a partial order on the set of binary sequences of length n by specifying the covering relations to be $\alpha 01 \beta < \alpha 10 \beta$. For example, with $n = 4$, $0011 < 0101$ is a covering relation but there is no covering relation between 1001 and 0110 and, in fact, they are incomparable.

LEMMA 1. $(\sigma_1, \sigma_2, \ldots, \sigma_n) = \sigma \leqslant \tau = (\tau_1, \tau_2, \ldots, \tau_n)$ *if and only if the following*

inequalities hold:

$$\sum_{i=1}^{j} \sigma_i \leqslant \sum_{i=1}^{j} \tau_i, \quad 1 \leqslant j \leqslant n, \text{ with equality for } j = n.$$

Proof. For convenience we refer to the set of inequalities in the statement of the lemma as the *partial sum criterion.* If $\sigma < \tau$ is a covering relation the partial sum criterion obviously holds (in fact, the partial sums are all equal except for a single case where they differ by 1). Since an arbitrary relation $\sigma < \tau$ in the partial order arises from a chain of covering relations the partial sum criterion must hold for the pair (σ, τ).

Conversely, suppose the partial sum criterion holds for the pair (σ, τ). If there was a value j, $1 \leqslant j < n$, with $\sum_{i=1}^{j} \sigma_i = \sum_{i=1}^{j} \tau_i$ then there would be non-trivial decompositions $\sigma = \sigma_1\sigma_2$ and $\tau = \tau_1\tau_2$ and the partial sum criterion would hold for both the pairs (σ_1, τ_1) and (σ_2, τ_2). Then, by induction on n, we would have $\sigma_1 \leqslant \tau_1$ and $\sigma_2 \leqslant \tau_2$ from which it is clear that $\sigma \leqslant \tau$.

If no such value of j exists then strict inequality holds in the partial sum criterion except for $j = n$. It follows that

$$\sigma = 0\alpha 1 \text{ and } \tau = 1\beta 0$$

and that the partial sum criterion holds for the pair (α, β). Again, by induction, $\alpha \leqslant \beta$. Then we have

$$\begin{aligned}
\sigma &= 0\alpha 1 = 0\sigma_2\sigma_3 \ldots \sigma_{n-1}1 \leqslant \\
&\leqslant \sigma_2 0\sigma_3 \ldots \sigma_{n-1}1 \leqslant \ldots \leqslant \sigma_2\sigma_3 \ldots \sigma_{n-1}01 < \\
&< \sigma_2\sigma_3 \ldots \sigma_{n-1}10 \leqslant \sigma_2\sigma_3 \ldots 1\sigma_{n-1}0 \leqslant \ldots \leqslant 1\sigma_2\sigma_3 \ldots \sigma_{n-1}0 = \\
&= 1\alpha 0 \leqslant 1\beta 0 = \\
&= \tau \text{ as required.}
\end{aligned}$$

□

LEMMA 2. *The number of pairs* (σ, τ) *with* $\sigma \leqslant \tau$ *is* c_{n+1}, *the* $(n+1)^{\text{th}}$ *Catalan number.*

Proof. We shall use induction on n; the result is easily verified if $n = 0$ or $n = 1$. Let e_n be the number of pairs (σ, τ) with $\sigma \leqslant \tau$. These pairs either have the form

1. $(0\alpha, 0\beta)$ or $(1\alpha, 1\beta)$, or
2. $(0\alpha, 1\beta)$

In the first case it is clear that (α, β) satisfies the partial sum criterion of the previous lemma and so $\alpha \leqslant \beta$; therefore, by induction, there are $2e_{n-1}$ pairs of this form. In the second case, let j be the first index with $\sum_{i=1}^{j} \sigma_i = \sum_{i=1}^{j} \tau_i$. Then $\sigma_j = 1$ and $\tau_j = 0$ and there are decompositions $\sigma = 0\gamma 1\delta$, $\tau = 1\epsilon 0\eta$ with γ, ϵ of length $j - 2$. Clearly, the pairs (γ, ϵ) and (δ, η) satisfy the partial sum criterion and so,

again by induction, $\gamma \leqslant \epsilon$ and $\delta \leqslant \eta$. Thus there are $\sum_{j=2}^{n} e_{j-2}e_{n-j}$ pairs of the second kind. Hence

$$\begin{aligned} e_n &= 2e_{n-1} + \sum_{j=2}^{n} e_{j-2}e_{n-j} \\ &= c_n + c_n + \sum_{j=2}^{n} c_{j-1}c_{n-j+1} \\ &= \sum_{0}^{n} c_j c_{n-j} \\ &= c_{n+1} \text{ by a well-known recurrence for Catalan numbers.} \end{aligned}$$ □

3. Main Results

THEOREM 1. *$\tau \leqslant \sigma$ if and only if (σ, τ) is allowable.*

Proof. Suppose that $\tau \leqslant \sigma$. If $\tau = \sigma$ then (σ, τ) is obviously allowable so we shall assume now that $\tau < \sigma$. Then there exists some ρ with $\tau < \rho \leqslant \sigma$ where $\tau < \rho$ is a covering relation. Therefore, for some α, β,

$$\begin{aligned} \rho &= \alpha 10 \beta \\ \tau &= \alpha 01 \beta \end{aligned}$$

By induction on the length of a maximal chain of covering relations between τ and σ we may assume that (σ, ρ) is allowable. Therefore there is a sequence $\mathcal{A}$ of *Insert* and *Delete-Minimum* operations which transforms the input σ into the output ρ. Let I, D stand for *Insert, Delete-Minimum*; furthermore, subscripts 0 or 1 on I and D will, if present, refer to the element that the operation is transferring. Within $\mathcal{A}$ there must be a segment

$$\mathcal{B} = D_1 I_1^r I_0 I^s D_0, \quad \text{with } r, s \geqslant 0$$

where the initial D_1 outputs the element 1 of ρ which immediately follows α and the final D_0 outputs the next element 0. Notice that when the D_1 operation is used the priority queue cannot contain any element 0. Consider the sequence $\mathcal{A}^*$ which arises by modifying $\mathcal{A}$ by replacing the segment $\mathcal{B}$ by the segment

$$\mathcal{B}^* = I_1^r I_0 D_0 D_1 I^s.$$

It is easy to verify that, when applied to the input sequence σ, $\mathcal{A}^*$ generates the output τ which proves that (σ, τ) is allowable.

For the converse, suppose that (σ, τ) is allowable. If $\tau \not\leqslant \sigma$ there will be some initial segment $\tau_1\tau_2 \ldots \tau_i$ with $\tau_i = 1$ of τ with more 1's than the initial segment $\sigma_1\sigma_2 \ldots \sigma_i$ of σ. Immediately after τ_i has been output the priority queue cannot

contain any element 0. However, all the symbols of $\sigma_1\sigma_2 \ldots \sigma_i$ must have been inserted at this point and not all the 0 elements have been output; hence, at least one element 0 still remains in the priority queue. This contradiction completes the proof. □

COROLLARY 1. *The number of allowable pairs of binary sequences of length n is c_{n+1}.*

For any binary sequence σ let $\tilde{\sigma}$, $\bar{\sigma}$ denote the reversal and complement of σ. Since the two maps $\sigma \to \tilde{\sigma}$ and $\sigma \to \bar{\sigma}$ are each anti-isomorphisms of the partial order we have

COROLLARY 2. *(σ, τ) is allowable if and only if $(\tilde{\sigma}, \tilde{\tau})$ is allowable if and only if $(\bar{\sigma}, \bar{\tau})$ is allowable.*

COROLLARY 3. $t(\sigma) = s(\tilde{\sigma}) = s(\bar{\sigma})$.

In [3] it was shown that the k-fold compositional power of R was the relation that described the (input, output) pairs for a serial network of k priority queues (the output of each priority queue being channelled as an input to the next priority queue in the series network). But, in the binary case that we have been studying here, all these powers are equal to R itself and so we have

COROLLARY 4. *The allowable pairs are precisely the (input, output) pairs associated with a series network of priority queues of any length.*

To conclude the paper we shall present an algorithm for computing $t(\sigma)$. Because $s(\sigma) = t(\bar{\sigma})$ this algorithm is capable of calculating $s(\sigma)$ also.

Suppose that σ is expressed as $10^{u_1}10^{u_2} \ldots 10^{u_k}$. Note that there is little loss in generality in the supposition that σ begins with a 1 since $t(0\sigma) = t(\sigma)$. When σ is expressed in this way we let $t(u_1, u_2, \ldots, u_k)$ denote $t(\sigma)$.

LEMMA 3. $t(u_1, u_2, \ldots, u_k) = t(u_1, u_2, \ldots, u_{k-1})$ *if* $u_k = 0$ *and*

$$t(u_1, u_2, \ldots, u_k) = t(u_1, u_2, \ldots, u_k - 1) + t(u_1, u_2, \ldots, u_{k-2}, u_{k-1} + u_k)$$

if $u_k > 0$.

Proof. Let $\sigma = \alpha 10^{u_k}$ and consider a sequence of *Insert* and *Delete-Minimum* operations applied to σ. Suppose that, just before the first of the u_k 0's in the final block is inserted, the priority queue is empty. Then the sequence $\alpha 1$ has been inserted and output already; clearly $t(\alpha 1) = t(\alpha)$ outputs can arise in this way. If the priority queue is not empty it must contain at least one element 1 and so the

number of outputs that can arise in this case is $t(\alpha 010^{u_k-1})$. Therefore, if $u_k > 0$,

$$t(u_1, u_2, \ldots, u_k) = t(u_1, u_2, \ldots, u_{k-1}) + t(u_1, u_2, \ldots, u_{k-1} + 1, u_k - 1)$$

and, if $u_k = 0$,

$$t(u_1, u_2, \ldots, u_k) = t(u_1, u_2, \ldots, u_{k-1})$$

Iterating this recurrence yields

$$t(u_1, u_2, \ldots, u_k) = \sum_{j=0}^{u_k} t(u_1, u_2, \ldots, u_{k-2}, u_{k-1} + j)$$

from which the result follows.

This lemma is the basis of a dynamic programming algorithm. For each value $k = 0, 1, 2, \ldots$ in turn we compute the values of $t(u_1, \ldots, u_k, j)$ for $j = 0, 1, 2, \ldots, n$. When $k = 0$ these values are 1,2,3,.... Each set of values for a new k is found from the previous set in $O(n)$ steps by the recurrence of the last lemma. Since there are at most n values of k the whole calculation has at most $O(n^2)$ steps.

Acknowledgement

I thank Ursula Martin for some useful insights.

References

1. D. E. Knuth, (1973) *Fundamental Algorithms*, Addison-Wesley, Reading, Mass.
2. T. H. Cormen, C. H. Leiserson, and R. L. Rivest (1992) *Introduction to Algorithms*, McGraw-Hill, Cambridge, Mass.
3. M. D. Atkinson and Robert Beals, Priority queues and permutations, in preparation.
4. M. D. Atkinson and Murali Thiyagarajah, The permutational power of a priority queue, BIT, to appear.