THE SWITCHYARD PROBLEM: SORTING USING NETWORKS OF-QUEUES'AND STACKS

BY

ROBERTTARJAN

STAN-CS-71-213

APRIL, 1971

COMPUTER SCIENCE DEPARTMENT

School of Humanities and' Sciences

STANFORD UNIVERSITY

THE SWITCHYARD PROBLEM:

SORTING USING NETWORKS OF QUEUES AND STACKS

Robert Tarjan
Computer Science Department
Stanford University

'Abstract

The problem of sorting a sequence of numbers using a network of queues and stacks is presented. A characterization of sequences sortable using parallel queues is given, and partial characterizations of sequences sortable using parallel stacks and networks of queues are given.

Keywords and Phrases: Sorting, network, queue, stack.

This research was supported by the Hertz Foundation and the National Science Foundation (GJ-992).

The Switchyard Problem:

Sorting Using Networks of Queues and Stacks

Inspired by Knuth [2], p. 234, we wish to consider the following problem: suppose we are presented with the layout of a railroad switchyard [Figure 1]. If a train is driven into one end of the yard, what rearrangements of the cars may be made before the train comes out the other end?

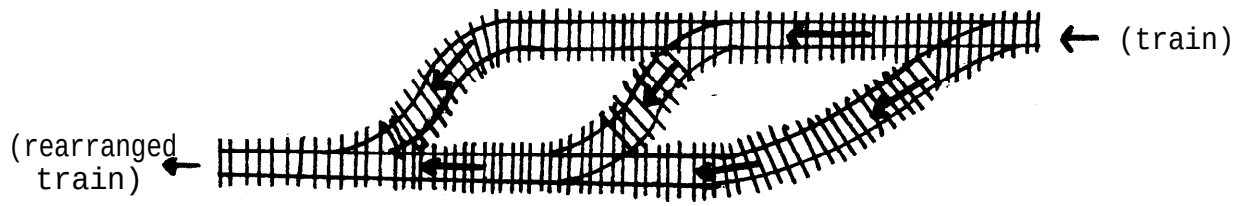

Figure 1: A railroad switchyard. What rearrangements are possible'?

In order to get a handle on the problem, we must introduce some formalization. A switchyard is an acyclic directed graph, with a unique source and a unique sink. Each vertex represents a siding. The vertex/siding is assumed to have indefinite storage space and may be a stack, a queue, or a deque of some sort (see Knuth [2] p. 234). A stack is a siding which has the property that the last element inserted is the first to be removed. A queue has the property that the first element inserted is the first to be removed. In the switchyard, the sidings associated with the source and sink are assumed to be queues.

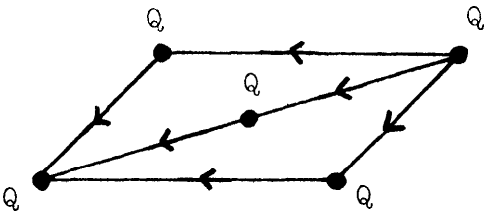

Figure 2: Abstract representation of switchyard in Figure 1. The vertices are queues.

Suppose a finite sequence of numbers $s = (s_1, s_2, \ldots, s_n)$ is placed in the source queue of a switchyard. We may rearrange s by moving the elements of s through the switchyard. At each step, an element is moved from some siding to another siding along an arc of the switchyard. After a suitable number of such moves, all elements will be in the sink queue. If they are in order, smallest to largest, we have sorted the sequence s using the switchyard. We wish to analyze the sequences s which may be sorted in a switchyard Y .

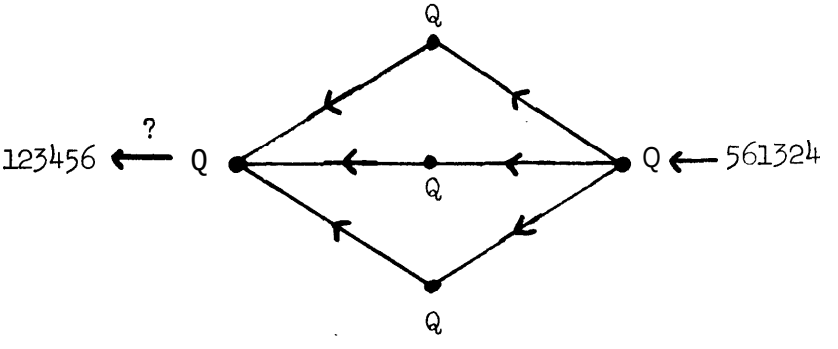

Figure 3: Can (561324) be sorted through the switchyard above?

We lose nothing in our formalism by allowing storage only on the vertices, and not on the arcs of the switchyard. We ignore questions

concerning the finite size of the sidings; assuming small sidings complicates the problem considerably. A circuit in the switchyard will allow us to sort any sequence; thus we do not allow circuits. Having established our model, we proceed to discover its properties.

Notice that no fixed switchyard is sufficient to sort all sequences. This may be proved very easily.

Lemma 1: Let Y be a switchyard. Then there are an infinite number of sequences which Y will not sort.

Proof. Consider moving a sequence s of length ℓ through Y . If Y has v vertices, then at any step there are at most v-1 possible moves. After at most $v\ell$ moves, all elements of s must have reached the sink queue. Thus there are at most $(v-1)^{v\ell}$ possible move sequences. However, there are $\ell!$ possible permutations of the numbers $1,2,\ldots,\ell$, and for large ℓ , $\ell! > (v-1)^{v\ell}$. Thus for large ℓ , same permutations of length ℓ are unsortable.

Lemma 1 gives a very crude upper bound on the size of the smallest sequence unsortable in a given switchyard Y . We will be able to compute the length of the smallest unsortable sequence exactly for certain switchyards. Let us characterize the sequences sortable in some simple switchyards.

A parallel network of m queues is a switchyard with a source queue, a sink queue, and m queues. An arc connects the source queue to each of the m queues, and an arc connects each of the m queues to the sink queue. A parallel network of m stacks is identical to a

parallel network of m queues except each of the m queues is replaced by a stack.

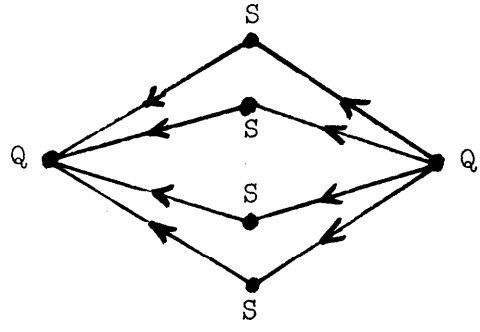

Figure 4: A parallel network of four stacks.

Consider any sequence of numbers s . Let the length of the longest strictly increasing subsequence of s be i(s) . Let the length of the longest strictly decreasing subsequence of s be d(s) .

Lemma 2: Let Y be a parallel network of m queues. Let s be a sequence of numbers. Then s is sortable in Y if and only if $d(s) \leq m$.

Proof. If $d(s) > m$, then in any movement of s through Y , there must be two elements s_i and s_j of s such that $i < j$, $s_i > s_j$, and s_i and s_j both pass through the same one of the m queues. But in this case, s_i and s_j will be out of order in the sink queue. Thus if $d(s) > m$, s is unsortable in Y .

To prove the converse, we use an algorithm to find the longest decreasing subsequence of a sequence [4]. This algorithm will give us a sorting procedure using m queues, where $m = d(s)$. Imagine the sequence s to be sitting in the

source queue. First we move all elements to the parallel queues. Number the parallel queues from 1 to m . Put the first element of the sequence in queue 1 . At the i-th step, put the i-th element s_i of the sequence s in the first compatible queue; that is, in the first queue such that the last element q in this queue satisfies $s_i \geq q$. After all elements are inserted into queues, exactly m queues will contain numbers and the numbers in each queue will be in order. Move the numbers to the sink queue, smallest first, largest last. This completes the sort. We leave it to the reader to verify this procedure.

Lemma3: Let Y be a parallel network of m stacks. Let s be a sequence of numbers. Suppose we wish to sort s by first inserting all the elements of s into the parallel stacks and then moving all the elements into the sink queue. s is sortable in this way if and only if i(s) <m .

Proof. We may prove Lemma 3 in the same way as Lemma 2; a similar algorithm gives a sort if one exists.

Corollary: Let Y be a parallel network of m stacks. Let s be a sequence of numbers, such that the last element s_n is the smallest element of s . Then s is sortable in Y if and only if $i(s) \leq m$.

Corollary: Let Y be a parallel network of m stacks. Then the shortest sequence unsortable in Y is of length m+2 .

A pattern is a finite permutation $p = (p_1, \ldots, p_k)$ of the integers $1, 2, \ldots, k$. Let s be a sequence of numbers. We say that s contains the pattern p if there is a 1-1 mapping ϕ of p into s such that if $\phi(p_i) = s_{i'}$ and $\phi(p_j) = s_{j'}$, $i < j$ implies $i' < j'$ and $p_i < p_j$ if and only if $s_{i'} < s_{j'}$. As an example, the sequence (561324) contains the pattern (312) in ten different ways. Using the notion of a pattern, we may cast Lemma 2 and Lemma 3 into a new form.

Lemma 2*: Let Y be a parallel network of m queues. Let s be a sequence of numbers. Then s is sortable in Y if and only if s does not contain the pattern $(m, m-1, \ldots, 1)$.

Lemma 3*: Let Y be a parallel network of m stacks. Let s be a sequence of numbers. Then s is sortable in Y using complete insertion into the parallel stacks followed by complete deletion if and only if s does not contain the pattern $(1, 2, \ldots, m+1)$.

Even and Itai [1] give characterizations similar to those of Lemma 2 and Lemma 3 based upon coloring a graph corresponding to a sequence to be sorted. If we relax the conditions we place on sorting using parallel stacks, the problem of characterizing the sortable sequences becomes much harder. For instance, we have the following necessary condition:

Lemma 4: Let Y be a parallel network of m stacks and let s be a sequence. Then if s is sortable in Y, s does not contain the pattern $(2, 3, 4, \ldots, m+1, 1)$.

The condition given in Lemma 4 is sufficient for one stack [2], but is not sufficient for two stacks or more. For instance, the sequence (27416385) is unsortable using two parallel stacks, though it does not contain the pattern (2341) . In general, given a sequence s , we may construct a corresponding graph which Even and Itai [1] call the union graph. The vertices of the graph are the elements of the sequence. If s_i , s_j , s_k match the pattern (231) then s_i and s_j are connected by an arc. The sequence is sortable-using m parallel stacks if and only if the corresponding union graph is colorable using m colors. This gives a nice algorithm for deciding whether a sequence is sortable using two parallel stacks, but beyond that we have no good decision procedures.

We may conjecture that some finite set of patterns characterize the sequences sortable in a switchyard. However, using the concept of the union graph, we may disprove this for the case of two parallelstacks.

Letnma5: There are an infinite set of permutations, none of which contains another as a pattern, and such that each permutation is unsortable using two parallel stacks.

Proof: Let us construct a diagram corresponding to a permutation $p = (p_1,\ldots,p_n)$. We plot i on the x axis, p_i on the y axis, and we connect points which are joined by an arc in the corresponding union graph.

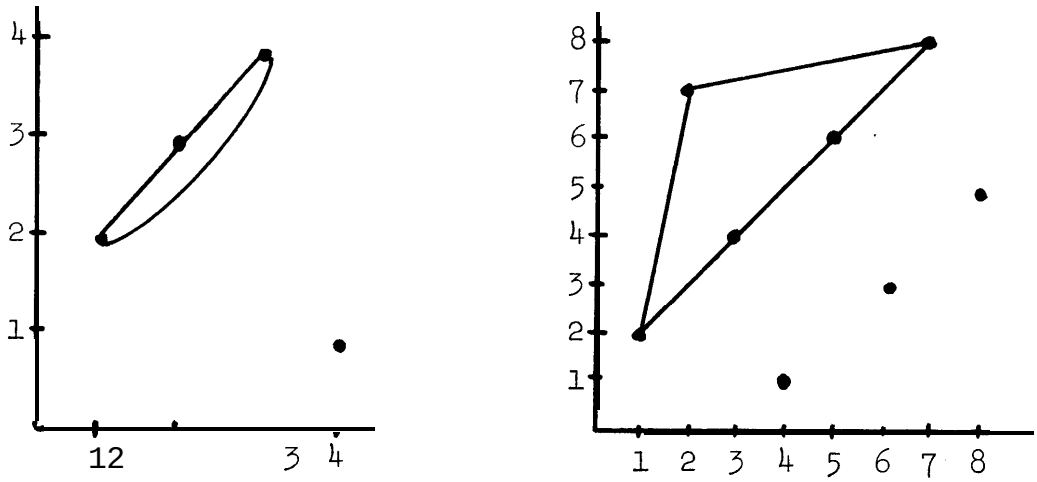

Figure 5: Diagrams for (2341) and (27416385).

Given the second example in Figure 5, we may extend the idea to construct a permutation whose union graph is a cycle of length $2n+1$, for arbitrary $n \geq 2$. $(2, 4n-1, 4, 1, 6, 3, 8, 5, \ldots, 4n, 4n-3)$ is the general permutation. Since the union graph of this permutation is a cycle of odd length, the permutation is unsortable using two stacks. Further, no permutation of this type contains another of this type as a pattern.

Let us return to the case of arbitrary switchyards. We will assume that all sidings are queues. Given a switchyard Y, we associate with it a capacity c(Y) computed as follows:

(1) Number the sidings of the switchyard from 1 to m so that no arc runs from a higher numbered siding to a lower numbered one. (This is always possible in an acyclic directed graph.)

(2) Attach a capacity to each siding from 1 to m : Label siding m (the sink) with 1. Attach 1 to all arcs entering siding m. At step i, add up all capacities of arcs out of siding m-i+1. Attach this capacity to siding m-i+1 and to all arcs entering it.

(3) When the labelling is completed, the capacity of the source is c(Y) .

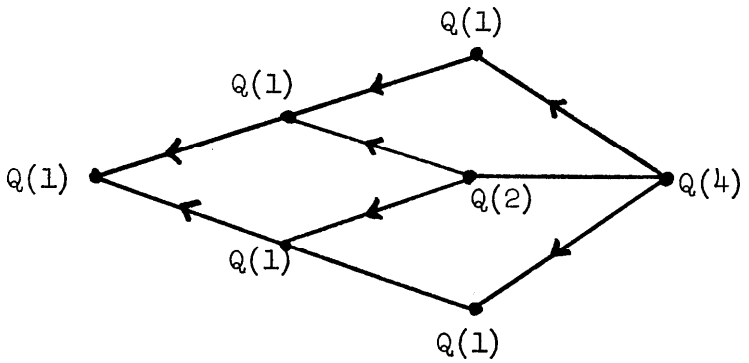

Figure 6: A switchyard and its siding capacities.

Lemma 6: Let Y be a switchyard of queues. Let Y' be the switchyard formed by reversing the direction of all arcs of Y . Then c(Y) = c(Y') .

Proof: The value of c(Y) is actually the number of different paths from the source to the sink. Thus c(Y) is independent of the direction of its calculation. The number of paths from source to sink in Y is the same as the number of paths from source to sink in Y' .

It is more useful in what follows to regard the calculation of c(Y) as proceding from the sink back towards the source. We may state a relationship between c(Y) and sequences sortable in Y .

Lemma 7: Let Y be a switchyard of queues and let s be a sequence. If s contains the pattern (c(Y)+1, c(Y), ... 2, 1) then s is unsortable in Y .

Proof: By induction. Clearly, Lemma 7 holds for a two-siding switchyard. Suppose the result is true for all switchyards with m-l or fewer sidings. Let Y be a switchyard with m sidings, and let s be a sequence which contains the pattern $(c(Y)+1, c(Y), \ldots, 2, 1)$. Let the queues adjacent to the source have capacities $c_1, \ldots, c_k$ $\sum c_i$. Then any sequence of moves of s through the switchyard must overload one of the queues adjacent to the source. That is, for some i , the subsequence of s which passes through queue i adjacent to the source must contain the pattern $(c_i+1, c_{i-1}, \ldots, 2, 1)$. By the induction hypothesis, this subsequence is unsortable in the remainder of the switchyard, and thus the entire sequence is unsortable.

Lemma 8: Let Y be a switchyard of queues and let s be a sequence. If s is no longer than c(Y) , then s is sortable in Y .

Proof. We proceed by induction. The result is trivial for a two-siding switchyard. Suppose the result is true for all switchyards with m-l or fewer sidings. Consider a switchyard with m sidings. Do the calculation of c(Y) from the sink queue to the source queue. Each arc a_i out of the source queue has a capacity $c(a_i)$; $\sum c(a_i) = c(Y)$. Further, the arcs have an imposed ordering given by the number of the queues on their front end. Move the lowest elements of the sequence to the highest numbered adjacent queue, the next lowest elements to the next highest numbered adjacent queue, and so on. At most $c(a_i)$ elements are allowed to pass through arc a_i . Once this has been done, the elements at the highest numbered queue may be sorted through the rest of the network by the induction hypothesis. The number of elements in this queue does not

exceed its capacity, and the other elements of the original sequence do not interfere. Then the elements at the next highest queue may be sorted, and so on. Thus the entire sequence may be sorted.

Lemmas 7 and 8 give us the length of the shortest sequence unsortable in a switchyard of queues. Lemma 7 gives a necessary condition for sortability. We have already seen that the condition is sufficient in the case of parallel queues; however, it is doubtful that the condition is in general sufficient. The situation is somewhat analogous to that of parallel stacks, though we presently know of no counterexample to show that the converse of Lemma 7 is false.

If we allow stacks or deques in an arbitrary switchyard things become even more complicated. Let us examine one such case. A series network of m stacks is a directed simple path of length m+2 . The two end sidings are queues, and the m intermediate sidings are stacks.

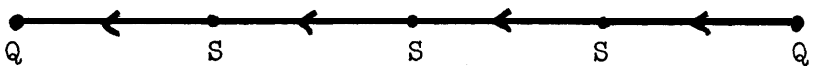

Figure 8: A series network of three stacks. What is the shortest unsortable sequence?

Lemma 9: Let Y be a series network of 2 stacks. Then the shortest unsortable sequence in Y is of length 7 .

Proof: (2435761) is unsortable using two stacks, as the reader may easily verify. Conversely, every sequence of length 6 or less may be sorted using two stacks. Exhaustive case analysis will verify this fact.

Lemma 10 [3]: Suppose sequences of length k or less may be sorted using m stacks in series. Then sequences of length 2k or less may be sorted using m+1 stacks in series.

Proof: Suppose that the hypothesis of the lemma is true and that a sequence of length 2k is given. Sort the first k elements into the last of the m+1 stacks, so that the k elements are in order, smallest on top, largest on the bottom. Sort the remaining k elements through the network so that they enter the last stack in order, smallest to largest. Instead of placing these elements in the last stack, merge them with those already in the last stack by always moving the smallest element to the sink queue when it is available to be moved. 2k elements may be sorted in this way.

Corollary: If $m > 2$, a sequence of length $3 \cdot 2^{m-1}$ or less may be sorted using m stacks in series.

In the case of one or two stacks in series, we know the length of the smallest unsortable sequence. Lemma 10 gives a lower bound in the general case. Knuth [3], using an argument along the lines of Lemma 1, gives an asymptotic upper bound of $k4^m$ for the shortest sequence unsortable using m stacks in series. The upper and lower bounds are not close in general. The author has constructed a sequence of length 41 which is unsortable using three stacks in series; beyond this, getting the upper and lower bounds closer together seems hard.

Conclusions

We have defined the switchyard problem and given a solution in certain special cases. In general, the problem seems very difficult and many questions are unanswered.

References

[1] Even, S. and Itai, A. Queues, stacks, and graphs. Unpublished.

[2] Knuth, D. E. The Art of Computer Programming, Volume 1. Addison-Wesley Publishing Company, Reading, Mass., 1968.

[3] Knuth, D. E. The Art of Computer Programming, Volume 3. To appear.

[4] Schensted, C. Longest increasing and decreasing subsequences. Canadian Journal of Mathematics, Vol. XIII, No. 2 (1961).