Sorting algorithms

Sorting networks

Basics

Definition: Let J = {0, ..., n-1} be an index set and let A be a set with an order relation $\leq$. A data sequence is a mapping $a : J \rightarrow A$, i.e. a sequence of length n of data. The set of all data sequences of length n over A is denoted by A^n.

Definition: The sorting problem consists of reordering an arbitrary data sequence $a_0, ..., a_{n-1}$, $a_i \in A$ to a data sequence $a_{\varphi(0)}, ..., a_{\varphi(n-1)}$ such that

$$a_{\varphi(i)} \leq a_{\varphi(j)} \quad \text{for} \quad i < j$$

where φ is a permutation of the index set J = {0, ..., n-1}.

This definition deals only with the simplest case of sorting a data sequence in ascending order. Later, when dealing with two-dimensional processor arrays, this definition has to be generalized by introducing an index set J = {0, ..., n-1} × {0, ..., n-1} and a sorting direction $\rho : J \rightarrow \{0, ..., |J|-1\}$.

Comparator networks

Comparator networks have been introduced informally in [Knu 73] for the case J = {0, ..., n-1}. A comparator [i:j] sorts the ith and the jth element of a data sequence into nondecreasing order. Formally, a comparator is a mapping applied to the data sequence:

Definition: A comparator is a mapping $[i:j] : A^n \rightarrow A^n$, $i, j \in \{0, ..., n-1\}$ with

$$[i:j](a)_i = \min(a_i, a_j)$$

$$[i:j](a)_j = \max(a_i, a_j)$$

$$[i:j](a)_k = a_k \text{ for all } k \text{ with } k \neq i,\ k \neq j$$

for all $a \in A^n$.

Figure 1 shows the graphical representation of a comparator network with comparators [1:3] and [2:1], applied to the data sequence 6 8 5 1.

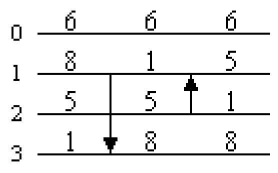

Figure 1: Graphical representation of comparators

Definition: A comparator stage S is a composition of comparators

$$S = [i_1:j_1] \cdot ... \cdot [i_k:j_k]\ ,\quad k \in \mathbb{N}$$

such that all i_r and j_s are distinct (even the i_r and j_s among each other).

Comparators within a comparator stage can be executed in parallel.

Definition: A comparator network is a composition of comparator stages.

As an example, Figure 2 shows a comparator network with two stages.

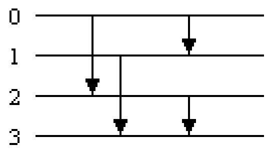

Figure 2: A comparator network with two stages

Sorting networks

Definition: A sorting network is a comparator network that sorts all input sequences.

The comparator network of the example above is not a sorting network, since it does not sort the sequence 3 1 4 2.

The problem whether an arbitrary comparator network is a sorting network or not is not easy to solve in general. It is an NP-complete problem. Besides that, sorting networks can of course be constructed systematically and proved to be correct.

Sorting networks are special cases of general sorting algorithms, since all comparisons are data-independent.

An example of a sorting network is bubblesort:

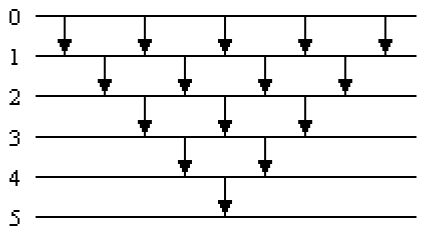

Figure 3: Sorting network bubblesort

The sorting network bubblesort consists of a first diagonal of n-1 comparators that move the greatest element to the last position. The remaining n-1 elements are sorted recursively by applying the same procedure. Bubblesort consists of $n\cdot(n-1)/2$ comparators, arranged in $2n – 3$ stages.

A similar sorting network is odd-even transposition sort. For sorting n input elements, $n\cdot(n-1)/2$ comparators but only n comparator stages are required. Each stage consists of comparators $[i: i+1]$ where i is odd or where i is even, in alternating order.

The main advantages of bubblesort and odd-even transposition sort are their simplicity and, important for hardware implementation, their locality and scalability. Locality means that comparators exist only between adjacent elements, scalability means that the structure does not change with size.

However, with an asymptotic complexitiy of $\Theta(n^2)$ comparators bubblesort and odd-even transposition sort are rather inefficient.

The lower bound for the sorting problem is $\Omega(n \log(n))$. This bound is tight even for sorting networks, as shown in [AKS 83]. However, due to its large constant, the AKS-network is impractical. But there are practical sorting networks with a complexity of $O(n\cdot\log(n)^2)$ comparators: bitonic sort, odd-even mergesort and shellsort.

The following table summarizes the complexities of some sorting networks:

	comparator stages	**comparators**
Odd-even transposition sort	$O(n)$	$O(n^2)$
Bubblesort	$O(n)$	$O(n^2)$
Bitonic sort	$O(\log(n)^2)$	$O(n\cdot\log(n)^2)$
Odd-even mergesort	$O(\log(n)^2)$	$O(n\cdot\log(n)^2)$
Shellsort	$O(\log(n)^2)$	$O(n\cdot\log(n)^2)$

The 0-1-principle

Whether an arbitrary comparator network N is a sorting network or not is independent of the input set A. It just depends on the structure of N. The 0-1-principle essentially states this fact.

Theorem: (0-1-principle)

> A comparator network with n inputs that sorts all 2^n sequences of zeroes and ones is a sorting network (i.e. it sorts all sequences of arbitrary values, too).

Proof of the 0-1-principle

The 0-1-principle is extremely useful for the proof of sorting networks.

References

[AKS 83]	M. Ajtai, J. Komlos, E. Szemeredi: An O(n log n) Sorting Network. Proceedings of the 25th ACM Symposium on Theory of Computing, 1-9 (1983)
[Knu 73]	D.E. Knuth: The Art of Computer Programming, Vol. 3 - Sorting and Searching. Addison-Wesley (1973)

Next: [0-1-principle] or ▲