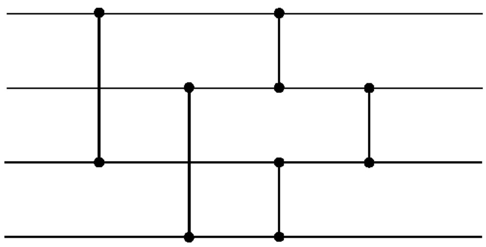
Sorting network

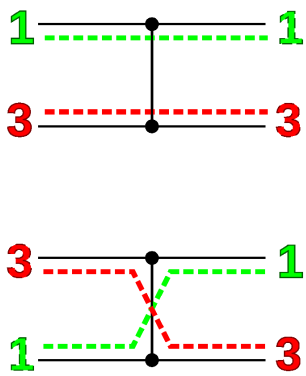
A **sorting network** is an abstract mathematical model of a network of wires and comparator modules that is used to sort a sequence of numbers. Each comparator connects two wires and sort the values by outputting the smaller value to one wire, and the larger to the other. The main difference between sorting networks and comparison sorting algorithms is that the sequence of comparisons is set in advance, regardless of the outcome of previous comparisons. This independence of comparison sequences is useful for parallel execution of the algorithms. Despite the simplicity of the model, sorting network theory is surprisingly deep and complex.



A simple sorting network consisting of four wires and five connectors

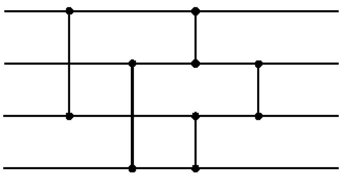
Introduction

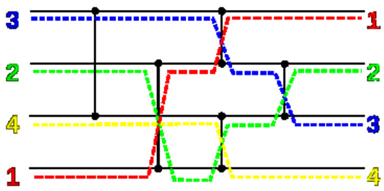
A sorting network consists of two items: comparators and wires. Each wire carries with it a value, and each comparator takes two wires as input and output. When two values enter a comparator, the comparator emits the lower value from the top wire, and the higher value from the bottom wire. A network of wires and comparators that will correctly sort all possible inputs into ascending order is called a sorting network.



Demonstration of a comparator in a sorting network.

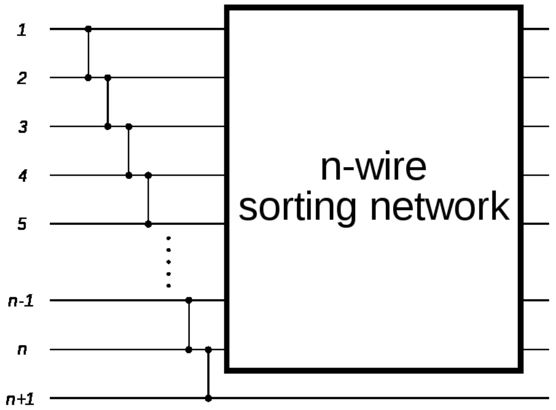
The full operation of a simple sorting network is shown below. It is easy to see why this sorting network will correctly sort the inputs; note that the first four comparators will "sink" the largest value to the bottom and "float" the smallest value to the top. The final comparator simply sorts out the middle two wires.



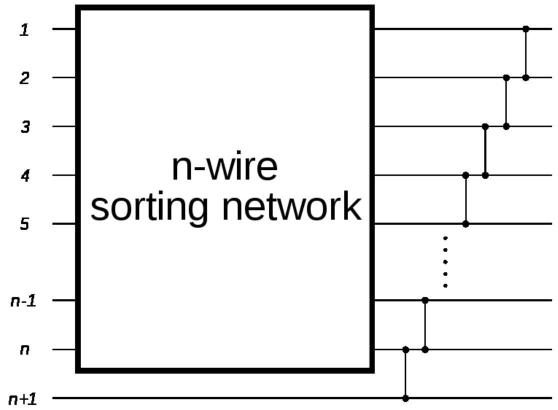


Insertion and Selection networks

We can easily construct a network of any size recursively using the principles of insertion and selection. Assuming we have a sorting network of size n, we can construct a network of size $n + 1$ by "inserting" an additional number into the already sorted subnet (using the principle behind insertion sort). We can also accomplish the same thing by first "selecting" the lowest value from the inputs and then sort the remaining values recursively (using the principle behind bubble sort).

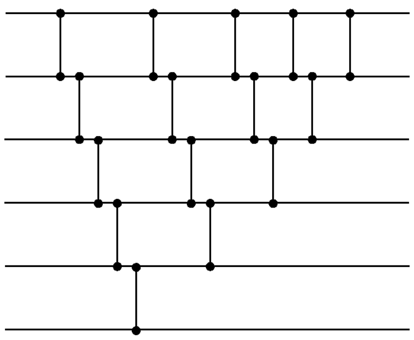


A sorting network constructed recursively that first sinks the largest value to the bottom and then sorts the remaining wires. Based on bubble sort

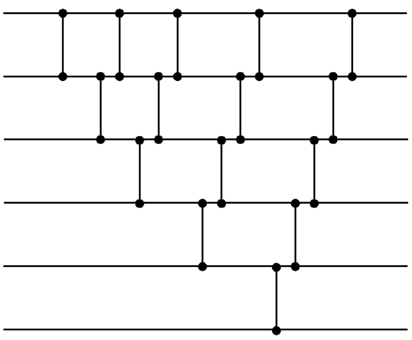


A sorting network constructed recursively that first sorts the first n wires, and then inserts the remaining value. Based on insertion sort

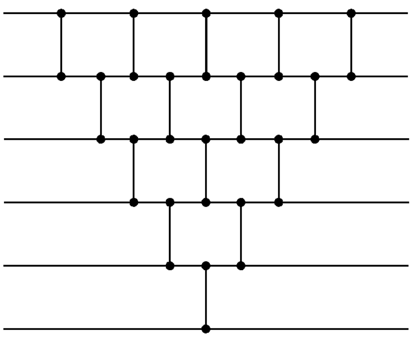
The structure of these two sorting networks are very similar. A construction of the two different variants, which collapses together comparators that can be performed simultaneously shows that, in fact, they are identical.



Bubble sorting network



Insertion sorting network



When allowing for parallel comparators, bubble sort and insertion sort are identical

Efficient networks

The insertion network has a large depth of O(n) making it impractical. There are simple networks which achieve depth O((log n)^2) (hence size O(n (log n)^2)), such as Batcher odd-even mergesort, bitonic sort, Shell sort, and the Pairwise sorting network. These networks are often used in practice.

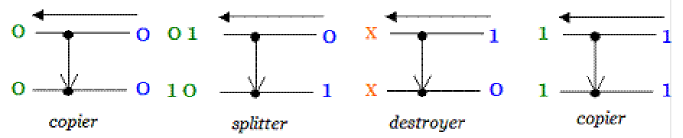
Zero-one principle

While it is easy to prove the validity of some sorting networks (like the insertion/bubble sorter), it is not always so easy. There are n! permutations of numbers in an n-wire network, and to test all of them would take a significant amount of time, especially when they are larger. However, because of the so-called zero-one principle, far fewer trials are in fact needed to test the validity of a sorting network.

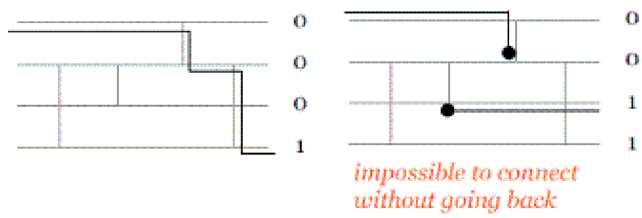
The zero-one principle states that a sorting network is valid if it can sort all 2^n sequences of 0s and 1s. This not only drastically cuts down on the number of tests needed to ascertain the validity of a network, it is of great use in creating many constructions of sorting networks as well. The principle has been proven by a special case of the Bouricius's Theorem in 1954 by W. G. Bouricius.

Topology and retrograde view

In order to reduce the number of test cases needed to assert the validity of an n-input sorting network, we can use topology of the network or retrograde software analysis (Alex Peter, 2010). Through retrograde software analysis, we analyze a network reversed in time starting from the sorted output, then going backward. If we combine the zero-one principle and retrograde analysis, we need only n test cases, in which case we have to follow gradually from 1 to 2^n testing paths. If we use topology of a sorting network, we need to check at first about n^2 test cases; basically we try to connect each input with each upper output using the network wiring without ever going back or down, and then we check that paths combined are independent regarding used comparators.



A comparator module in a sorting network observed reversed in time



Two topological test cases for an assumed invalid 4-input sorting network

Optimal sorting

The efficiency of a sorting network can be measured by its total size (the number of comparators used), or by its depth (the maximum number of comparators along any path from an input to an output). The asymptotically best known sorting network, called *AKS network* after its discoverers Ajtai, Komlós, and Szemerédi, achieves depth O(log n) and size O(n log n) for n inputs, which is asymptotically optimal. A simplified version of the AKS network was described by Paterson. While an important theoretical discovery, the AKS network has little or no practical application because of the large linear constants hidden by the Big-O notation. These are partly due to a construction of an expander graph. Finding sorting networks with size cn log n for small c remains a fundamental open problem.

Some important progress in designing optimal sorting network is done using genetic algorithm technique as well. (M. Mitchell, 1998)

For 1 to 8 inputs optimal sorting networks are known. They have respectively 0, 1, 3, 5, 9, 12, 16 and 19 comparators (Knuth, 1997). The optimal depths for up to 10 inputs are known and they are respectively 0, 1, 3, 3, 5, 5, 6, 6, 7, 7.

References

- O. Angel, A.E. Holroyd, D. Romik, B. Virag, *Random Sorting Networks* [1], Adv. in Math., 215(2):839–868, 2007.
- K.E. Batcher, *Sorting networks and their applications* [2], Proceedings of the AFIPS Spring Joint Computer Conference 32, 307–314 (1968).
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*, Second Edition. MIT Press and McGraw-Hill, 1990. ISBN 0-262-03293-7. Chapter 27: Sorting Networks, pp.704–724.
- D.E. Knuth. *The Art of Computer Programming*, Volume 3: *Sorting and Searching*, Third Edition. Addison-Wesley, 1997. ISBN 0-201-89685-0. Section 5.3.4: Networks for Sorting, pp. 219–247.
- Ajtai, M.; Komlós, J.; Szemerédi, E. (1983), "An O(n log n) sorting network", *Proceedings of the 15th Annual ACM Symposium on Theory of Computing*, pp. 1–9, doi:10.1145/800061.808726.
- M. S. Paterson, *Improved sorting networks with O*(log *N*) *depth*, Algorithmica 5 (1990), no. 1, pp. 75–92, doi:10.1007/BF01840378.
- M. Mitchell, *An Introduction to Genetic Algorithms*, The MIT Press, 1998. ISBN 0-262-63185-7. Chapter 1: Overview, pp. 21–27
- Alex Peter, *Retrograde Software Analysis*, UniBook, 2010, pp. 8-13

External links

- Sorting Networks [3]
- Sorting Networks [4]
- List of Sorting Networks [5]
- Sorting networks and the END algorithm [6]

References

[1] http://arxiv.org/abs/math/0609538
[2] http://www.cs.kent.edu/~batcher/sort.ps
[3] http://www.iti.fh-flensburg.de/lang/algorithmen/sortieren/networks/sortieren.htm
[4] http://www.cs.uky.edu/~lewis/essays/algorithms/sortnets/sort-net.html
[5] http://pages.ripco.net/~jgamble/nw.html
[6] http://www.cs.brandeis.edu/~hugues/sorting_networks.html

Article Sources and Contributors

Sorting network *Source*: http://en.wikipedia.org/w/index.php?oldid=402886042 *Contributors*: Aperisic, ArthurDenture, Booyabazooka, Danakil, David Eppstein, Dcoetzee, EmilJ, Ezhiki, Gaius Cornelius, Garrettw87, Goodrone, GregorB, Kope, Kostmo, Kri, Linas, Lowellian, Mhym, Novamo, Octotron, Oskar Sigvardsson, Ott2, Qutezuce, Rajah, Sae1962, Satchmo, Sn0rt, The Anome, Wmahan, 33 anonymous edits

Image Sources, Licenses and Contributors

Image:SimpleSortingNetwork2.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:SimpleSortingNetwork2.svg *License*: Creative Commons Attribution 3.0 *Contributors*: User:Oskar Sigvardsson

Image:Sorting-network-comparator-demonstration.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Sorting-network-comparator-demonstration.svg *License*: Creative Commons Attribution 3.0 *Contributors*: User:Oskar Sigvardsson

Image:SimpleSortingNetworkFullOperation.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:SimpleSortingNetworkFullOperation.svg *License*: Creative Commons Attribution 3.0 *Contributors*: User:Oskar Sigvardsson

Image:Recursive-bubble-sorting-network.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Recursive-bubble-sorting-network.svg *License*: Creative Commons Attribution 3.0 *Contributors*: User:Oskar Sigvardsson

Image:Recursive-insertion-sorting-network.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Recursive-insertion-sorting-network.svg *License*: Creative Commons Attribution 3.0 *Contributors*: User:Oskar Sigvardsson

Image:Six-wire-bubble-sorting-network.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Six-wire-bubble-sorting-network.svg *License*: Creative Commons Attribution 3.0 *Contributors*: User:Oskar Sigvardsson

Image:Six-wire-insertion-sorting-network.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Six-wire-insertion-sorting-network.svg *License*: Creative Commons Attribution 3.0 *Contributors*: User:Oskar Sigvardsson

Image:Six-wire-pyramid-sorting-network.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Six-wire-pyramid-sorting-network.svg *License*: Creative Commons Attribution 3.0 *Contributors*: User:Oskar Sigvardsson

Image:Reversed.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Reversed.gif *License*: unknown *Contributors*: Aperisic

Image:Connect_sorting_network.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Connect_sorting_network.gif *License*: unknown *Contributors*: Aperisic

License