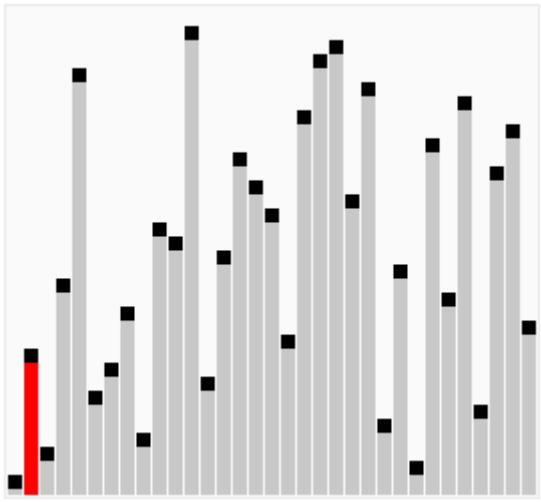
Cocktail shaker sort

Cocktail shaker sort,[1] also known as **bidirectional bubble sort**,[2] **cocktail sort**, **shaker sort** (which can also refer to a variant of selection sort), **ripple sort**, **shuffle sort**,[3] or **shuttle sort**, is a variation of bubble sort that is both a stable sorting algorithm and a comparison sort. The algorithm differs from a bubble sort in that it sorts in both directions on each pass through the list. This sorting algorithm is only marginally more difficult to implement than a bubble sort, and solves the problem of turtles in bubble sorts. It provides only marginal performance improvements, and does not improve asymptotic performance; like the bubble sort, it is not of practical interest (insertion sort is preferred for simple sorts), though it finds some use in education.

Cocktail shaker sort



Class	Sorting algorithm
Data structure	Array
Worst-case performance	$O(n^2)$
Best-case performance	$O(n)$
Average performance	$O(n^2)$
Worst-case space complexity	$O(1)$

Contents

Pseudocode

The simplest form goes through the whole list each time:

```
procedure cocktailShakerSort(A : list of sortable items) is
    do
        swapped := false
        for each i in 0 to length( A ) - 2 do:
            if A[ i ] > A[ i + 1 ] then // test whether the two elements are in the wrong order
                swap( A[ i ], A[ i + 1 ] ) // let the two elements change places
                swapped := true
            end if
        end for
        if not swapped then
            // we can exit the outer loop here if no swaps occurred.
            break do-while loop
        end if
        swapped := false
        for each i in length( A ) - 2 to 0 do:
            if A[ i ] > A[ i + 1 ] then
                swap( A[ i ], A[ i + 1 ] )
                swapped := true
            end if
        end for
```

```
    while swapped // if no elements have been swapped, then the list is sorted
end procedure
```

The first rightward pass will shift the largest element to its correct place at the end, and the following leftward pass will shift the smallest element to its correct place at the beginning. The second complete pass will shift the second largest and second smallest elements to their correct places, and so on. After *i* passes, the first *i* and the last *i* elements in the list are in their correct positions, and do not need to be checked. By shortening the part of the list that is sorted each time, the number of operations can be halved (see bubble sort).

This is an example of the algorithm in MATLAB/OCTAVE with the optimization of remembering the last swap index and updating the bounds.

```
function A = cocktailShakerSort(A)
% `beginIdx` and `endIdx` marks the first and last index to check
beginIdx = 1;
endIdx = length(A)-1;
while beginIdx <= endIdx
    newBeginIdx = endIdx;
    newEndIdx = beginIdx;
    for ii= beginIdx:endIdx
        if A(ii) > A(ii + 1)
            [A(ii+1), A(ii)] = deal(A(ii), A(ii+1));
            newEndIdx=ii;
        end
    end

    % decreases `endIdx` because the elements after `newEndIdx` are in correct order
    endIdx = newEndIdx - 1;

    for ii= endIdx:-1:beginIdx
        if A(ii) > A(ii + 1)
            [A(ii+1), A(ii)] = deal(A(ii), A(ii+1));
            newBeginIdx = ii;
        end
    end
    % increases `beginIdx` because the elements before `newBeginIdx` are in correct order
    beginIdx = newBeginIdx + 1;
end
end
```

Differences from bubble sort

Cocktail shaker sort is a slight variation of bubble sort.[1] It differs in that instead of repeatedly passing through the list from bottom to top, it passes alternately from bottom to top and then from top to bottom. It can achieve slightly better performance than a standard bubble sort. The reason for this is that bubble sort only passes through the list in one direction and therefore can only move items backward one step each iteration.

An example of a list that proves this point is the list (2,3,4,5,1), which would only need to go through one pass of cocktail sort to become sorted, but if using an ascending bubble sort would take four passes. However one cocktail sort pass should be counted as two bubble sort passes. Typically cocktail sort is less than two times faster than bubble sort.

Another optimization can be that the algorithm remembers where the last actual swap has been done. In the next iteration, there will be no swaps beyond this limit and the algorithm has shorter passes. As the cocktail shaker sort goes bidirectionally, the range of possible swaps, which is the range to be tested, will reduce per pass, thus reducing the overall running time slightly.

Complexity

The complexity of the cocktail shaker sort in big O notation is $O(n^2)$ for both the worst case and the average case, but it becomes closer to $O(n)$ if the list is mostly ordered before applying the sorting algorithm. For example, if every element is at a position that differs by at most k ($k \geq 1$) from the position it is going to end up in, the complexity of cocktail shaker sort becomes $O(kn)$.

The cocktail shaker sort is also briefly discussed in the book *The Art of Computer Programming*, along with similar refinements of bubble sort. In conclusion, Knuth states about bubble sort and its improvements:

> But none of these refinements leads to an algorithm better than straight insertion [that is, insertion sort]; and we already know that straight insertion isn't suitable for large *N*. [...] In short, the bubble sort seems to have nothing to recommend it, except a catchy name and the fact that it leads to some interesting theoretical problems.
>
> — D. E. Knuth[1]

References

1. Knuth, Donald E. (1973). "Sorting by Exchanging". *Art of Computer Programming*. 3. Sorting and Searching (1st ed.). Addison-Wesley. pp. 110–111. ISBN 0-201-03803-X.
2. Black, Paul E.; Bockholt, Bob (24 August 2009). "bidirectional bubble sort". In Black, Paul E. (ed.). *Dictionary of Algorithms and Data Structures* (http://xlinux.nist.gov/dads/HTML/bidirectionalBubbleSort.html). National Institute of Standards and Technology. Retrieved 5 February 2010.
3. Duhl, Martin (1986). "Die schrittweise Entwicklung und Beschreibung einer Shuffle-Sort-Array Schaltung". *HYPERKARL aus der Algorithmischen Darstellung des BUBBLE-SORT-ALGORITHMUS*. *Projektarbeit* (in German). Technical University of Kaiserslautern.

Sources

- Hartenstein, R. (July 2010). "A new World Model of Computing" (https://web.archive.org/web/20130807043613/http://www.inf.pucminas.br/sbc2010/anais/pdf/semish/st03_02.pdf) (PDF). *THE GRAND CHALLENGE TO REINVENT COMPUTING*. Belo Horizonte, Brazil: CSBC. Archived from the original (http://www.inf.pucminas.br/sbc2010/anais/pdf/semish/st03_02.pdf) (PDF) on 2013-08-07. Retrieved 2011-01-14.

External links

- Interactive demo of cocktail sort (http://www.hermann-gruber.com/lehre/sorting/Shaker/Shaker-en.html)
- Java source code and an animated demo of cocktail sort (called bi-directional bubble sort) and several other algorithms (https://web.archive.org/web/20061008105719/http://www.cs.ubc.ca/~harrison/Java/sorting-demo.html)
- ".NET Implementation of cocktail sort and several other algorithms" (https://web.archive.org/web/20120212173240/http://www.sharpdeveloper.net/content/archive/2007/08/14/dot-net-data-structures-and-algorithms.aspx). Archived from the original (http://www.sharpdeveloper.

net/content/archive/2007/08/14/dot-net-data-structures-and-algorithms.aspx) on 2012-02-12.