Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
العربية
Čeština
Deutsch
Español
فارسی
Français
Հայերեն
Italiano
Magyar
日本語
Polski
Português
Русский
Српски / srpski
Türkçe
Українська
中文
Edit links

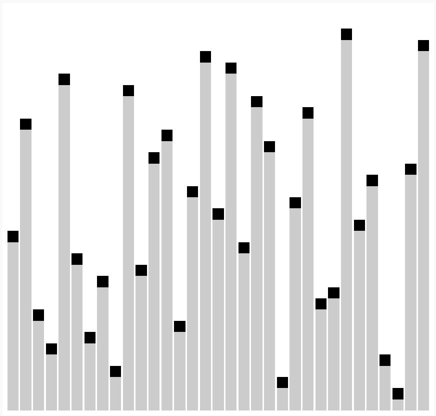
Comb sort

From Wikipedia, the free encyclopedia

This article **needs additional citations for verification**. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed. *(March 2011)*

Comb sort is a relatively simple sorting algorithm originally designed by Włodzimierz Dobosiewicz in 1980.[1] Later it was rediscovered by Stephen Lacey and Richard Box in 1991.[2] Comb sort improves on bubble sort.

Comb sort



Class	Sorting algorithm
Data structure	Array
Worst case performance	$O(n^2)$[1]
Best case performance	$O(n)$
Average case performance	$\Omega(n^2/2^p)$, where p is the number of increments[1]
Worst case space complexity	$O(1)$

Contents [hide]

Algorithm [edit]

The basic idea is to eliminate *turtles*, or small values near the end of the list, since in a bubble sort these slow the sorting down tremendously. *Rabbits*, large values around the beginning of the list, do not pose a problem in bubble sort.

In bubble sort, when any two elements are compared, they always have a *gap* (distance from each other) of 1. The basic idea of comb sort is that the gap can be much more than 1 (Shell sort is also based on this idea, but it is a modification of insertion sort rather than bubble sort).

In other words, the inner loop of bubble sort, which does the actual *swap*, is modified such that gap between swapped elements goes down (for each iteration of outer loop) in steps of shrink factor. i.e. [input size / shrink factor, input size / shrink factor^2, input size / shrink factor^3, ..., 1]. Unlike in bubble sort, where the gap is constant i.e. 1.

The gap starts out as the length of the list being sorted divided by the *shrink factor* (generally 1.3; see below), and the list is sorted with that value (rounded down to an integer if needed) as the gap. Then the gap is divided by the shrink factor again, the list is sorted with this new gap, and the process repeats until the gap is 1. At this point, comb sort continues using a gap of 1 until the list is fully sorted. The final stage of the sort is thus equivalent to a bubble sort, but by this time most turtles have been dealt with, so a bubble sort will be efficient.

The shrink factor has a great effect on the efficiency of comb sort. In the original article, the authors suggested 1.3. A value too small slows the algorithm down because more comparisons must be made, whereas a value too large means that no comparisons will be made. Lacey and Box empirically found (by testing Combsort on over 200,000 random lists) the shrink factor of 1.3 to be the best.

Pseudocode [edit]

```
function combsort(array input)
    gap := input.size //initialize gap size
    shrink := 1.3 //set the gap shrink factor

    loop until gap = 1 and swapped = false
        //update the gap value for a next comb. Below is an example
        gap := int(gap / shrink)
        if gap < 1
          //minimum gap is 1
          gap := 1
        end if

        i := 0
        swapped := false //see bubblesort for an explanation

        //a single "comb" over the input list
        loop until i + gap > input.size //see shellsort for similar idea
            if input[i] > input[i+gap]
                swap(input[i], input[i+gap])
                swapped := true // Flag a swap has occurred, so the
                                // list is not guaranteed sorted
            end if
            i := i + 1
        end loop

    end loop
end function
```

See also [edit]

- Bubble sort, a generally slower algorithm, is the basis of comb sort.
- Cocktail sort, or bidirectional bubble sort, is a variation of bubble sort that also addresses the problem of turtles, albeit less effectively.

The Wikibook *Algorithm Implementation* has a page on the topic of: ***Comb sort***

References [edit]

1. ^ *a* *b* *c* Brejová, B. (15 September 2001). "Analyzing variants of Shellsort". *Inform. Process. Lett.* **79** (5): 223–227. doi:10.1016/S0020-0190(00)00223-4.
2. ^ "A Fast Easy Sort", *Byte* Magazine, April 1991

v · t · e	**Sorting algorithms** [hide]
Theory	Computational complexity theory · Big O notation · Total order · Lists · Inplacement · Stability · Comparison sort · Adaptive sort · Sorting network · Integer sorting
Exchange sorts	Bubble sort · Cocktail sort · Odd–even sort · **Comb sort** · Gnome sort · Quicksort · Stooge sort · Bogosort
Selection sorts	Selection sort · Heapsort · Smoothsort · Cartesian tree sort · Tournament sort · Cycle sort
Insertion sorts	Insertion sort · Shellsort · Splaysort · Tree sort · Library sort · Patience sorting
Merge sorts	Merge sort · Cascade merge sort · Oscillating merge sort · Polyphase merge sort · Strand sort
Distribution sorts	American flag sort · Bead sort · Bucket sort · Burstsort · Counting sort · Pigeonhole sort · Proxmap sort · Radix sort · Flashsort
Concurrent sorts	Bitonic sorter · Batcher odd–even mergesort · Pairwise sorting network
Hybrid sorts	Block sort · Timsort · Introsort · Spreadsort · JSort
Other	Topological sorting · Pancake sorting · Spaghetti sort

Categories: Sorting algorithms | Comparison sorts