WIKIPEDIA
The Free Encyclopedia

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

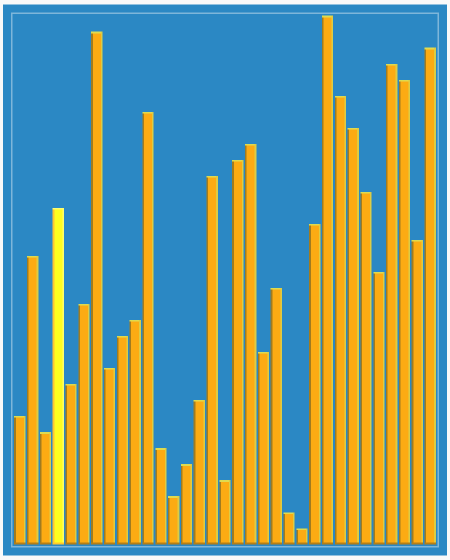
Languages
Català
Deutsch
Español
فارسی
Հայերեն
Italiano
Magyar
日本語
Polski
Português
Русский
Српски / srpski
Türkçe
Українська
Edit links

Gnome 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. *(August 2010)*

Gnome sort



Visualisation of Gnome sort.

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

Gnome sort (or **Stupid sort**) is a sorting algorithm originally proposed by Dr. Hamid Sarbazi-Azad (Professor of Computer Engineering at Sharif University of Technology) in 2000 and called "stupid sort"[1] (not to be confused with bogosort), and then later on described by Dick Grune and named "gnome sort" from the observation that it is "how a gnome sorts a line of flower pots."[2] It is a sorting algorithm which is similar to insertion sort, except that moving an element to its proper place is accomplished by a series of swaps, as in bubble sort. It is conceptually simple, requiring no nested loops. The average, or expected, running time is $O(n^2)$, but tends towards $O(n)$ if the list is initially almost sorted.[3] In practice the algorithm can run as fast as insertion sort.[*citation needed*]

The algorithm always finds the first place where two adjacent elements are in the wrong order, and swaps them. It takes advantage of the fact that performing a swap can introduce a new out-of-order adjacent pair only next to the two swapped elements. It does not assume that elements forward of the current position are sorted, so it only needs to check the position directly previous to the swapped elements.

Contents [hide]

Description [edit]

Here is pseudocode for the gnome sort using a zero-based array:

```
procedure gnomeSort(a[])
    pos := 1
    while pos < length(a)
        if (a[pos] >= a[pos-1])
            pos := pos + 1
        else
            swap a[pos] and a[pos-1]
            if (pos > 1)
                pos := pos - 1
            end if
        end if
    end while
end procedure
```

Example [edit]

Given an unsorted array, a = [5, 3, 2, 4], the gnome sort would take the following steps during the while loop.

The "current position" is highlighted in **bold**:

Current array	Action to take
[5, **3**, 2, 4]	a[pos] < a[pos-1], swap:
[3, **5**, 2, 4]	a[pos] >= a[pos-1], increment pos:
[3, 5, **2**, 4]	a[pos] < a[pos-1], swap and pos > 1, decrement pos:
[3, **2**, 5, 4]	a[pos] < a[pos-1], swap and pos <= 1, increment pos:
[2, 3, **5**, 4]	a[pos] >= a[pos-1], increment pos:
[2, 3, 5, **4**]	a[pos] < a[pos-1], swap and pos > 1, decrement pos:
[2, 3, **4**, 5]	a[pos] >= a[pos-1], increment pos:
[2, 3, 4, **5**]	a[pos] >= a[pos-1], increment pos:
[2, 3, 4, 5]	pos == length(a), finished.

Optimization [edit]

The gnome sort may be optimized by introducing a variable to store the position before traversing back toward the beginning of the list. This would allow the "gnome" to teleport back to his previous position after moving a flower pot. With this optimization, the gnome sort would become a variant of the insertion sort. The animation in the introduction to this topic takes advantage of this optimization.

Here is pseudocode for an optimized gnome sort using a zero-based array:

```
procedure optimizedGnomeSort(a[])
    pos := 1
    last := 0
    while pos < length(a)
        if (a[pos] >= a[pos-1])
            if (last != 0)
                pos := last
                last := 0
            end if
            pos := pos + 1
        else
            swap a[pos] and a[pos-1]
            if (pos > 1)
                if (last == 0)
                    last := pos
                end if
                pos := pos - 1
            else
                pos := pos + 1
            end if
        end if
    end while
end procedure
```

References [edit]

1. ^ Sarbazi-Azad, Hamid (2 October 2000). "Stupid Sort: A new sorting algorithm" (PDF). *Newsletter* (Computing Science Department, Univ. of Glasgow) (599): 4. Retrieved 25 November 2014.
2. ^ http://www.dickgrune.com/Programs/gnomesort.html
3. ^ Paul E. Black. "gnome sort". *Dictionary of Algorithms and Data Structures*. U.S. National Institute of Standards and Technology. Retrieved 2011-08-20.

External links [edit]

- Gnome sort

The Wikibook *Algorithm implementation* has a page on the topic of: ***Gnome sort***

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
	Bubble sort · Cocktail sort · Odd–even sort · Comb sort · **Gnome sort** · Quicksort · Stooge sort ·

Exchange sorts	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 | Stable sorts