Gnome sort

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

Gnome sort, originally proposed by Hamid Sarbazi-Azad in 2000 and called Stupid sort [1], and then later on described by Dick Grune and named "Gnome sort",[2] 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 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. The average runtime is $O(n^2)$ [4]

.

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 right before or after 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 before the swapped elements.

Description

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 else pos := pos + 1 end if end while end procedure

Example

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:
[2, **3**, 5, 4]	a[pos] >= a[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

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.

References

[1] http://sina.sharif.edu/~azad/stupid-sort.PDF

[2] http://www.cs.vu.nl/~dick/gnomesort.html

[3] Paul E. Black. "gnome sort" (http://www.itl.nist.gov/div897/sqg/dads/HTML/gnomeSort.html). *Dictionary of Algorithms and Data Structures*. U.S. National Institute of Standards and Technology. . Retrieved 2010-01-20.

[4] "What is the Average Big-O Complexity of Gnome sort?" (http://stackoverflow.com/questions/2066541/what-is-the-average-big--complexity-of-gnome-sort). *StackOverflow*. .

External links

- Gnome sort (http://www.cs.vu.nl/~dick/gnomesort.html)

Article Sources and Contributors

Gnome sort *Source*: http://en.wikipedia.org/w/index.php?oldid=404060947 *Contributors*: Abdull, Abu adam, Ahy1, Arvindn, Beej71, Bluemoose, Booyabazooka, Bpcrochet, CAPS LOCK, CJLL Wright, CiaPan, Cyhawk, DanielKO, Dcoetzee, DevastatorIIC, Drmies, FMartinMaroto, FatalError, Fredrik, Fuzzie, Gpetty, Grendelkhan, Happypal, Honza Záruba, Irate, Ittakezou0, J.Rackham, JaGa, Jamelan, Jesin, JimD, Jll, Jwinius, MBlume, Maghnus, Mankarse, Matthew V Ball, Merendoglu, Nimur, Nneonneo, Orderud, Oskar Sigvardsson, Otus, Oğuz Ergin, Pacers, Pfalstad, Porges, Progmaker, Pulveriser, Qz, Rhanekom, Robin Stocker, Signalhead, Sligocki, Themania, Thumperward, Toebes, VPeric, Veganfanatic, Vijay.imt007, Wesselbindt, Ztothefifth, Олександр Кравчук, 107 anonymous edits

License