Gnome sort

Gnome sort (dubbed **stupid sort**) is a sorting algorithm originally proposed by an Iranian computer scientist Hamid Sarbazi-Azad (professor of Computer Engineering at Sharif University of Technology)[1] in 2000. The sort was first called *stupid sort*[2] (not to be confused with bogosort), and then later described by Dick Grune and named *gnome sort*.[3]

The gnome sort is a sorting algorithm which is similar to insertion sort in that it works with one item at a time but gets the item to the proper place by a series of swaps, similar to a bubble sort. It is conceptually simple, requiring no nested loops. The average running time is $O(n^2)$ but tends towards $O(n)$ if the list is initially almost sorted.[4][note 1]

The algorithm 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 next to the previously 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.
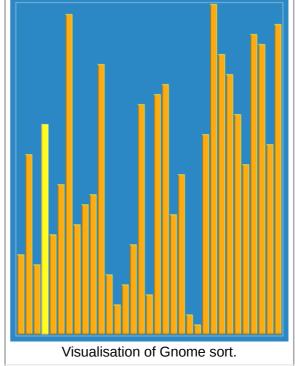
Gnome sort

Visualisation of Gnome 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	$O(1)$ auxiliary

Contents

Description

Dick Grune described the sorting method with the following story:[3]

Gnome Sort is based on the technique used by the standard Dutch Garden Gnome (Du.: tuinkabouter).

Here is how a garden gnome sorts a line of flower pots.
Basically, he looks at the flower pot next to him and the previous one; if they are in the right order he steps one pot forward, otherwise, he swaps them and steps one pot backward.

Boundary conditions: if there is no previous pot, he steps forwards; if there is no pot next to him, he is done.

— "Gnome Sort - The Simplest Sort Algorithm". *Dickgrune.com*

Code

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

```
procedure gnomeSort(a[]):
    pos := 0
    while pos < length(a):
        if (pos == 0 or a[pos] >= a[pos-1]):
            pos := pos + 1
        else:
            swap a[pos] and a[pos-1]
            pos := pos - 1
```

Example

Given an unsorted array, a = [5, 3, 2, 4], the gnome sort takes the following steps during the while loop. The current position is highlighted in bold and indicated as a value of the variable pos.

Current array	pos	Condition in effect	Action to take
[**5**, 3, 2, 4]	0	pos == 0	increment pos
[5, **3**, 2, 4]	1	a[pos] < a[pos-1]	swap, decrement pos
[**3**, 5, 2, 4]	0	pos == 0	increment pos
[3, **5**, 2, 4]	1	a[pos] ≥ a[pos-1]	increment pos
[3, 5, **2**, 4]	2	a[pos] < a[pos-1]	swap, decrement pos
[3, **2**, 5, 4]	1	a[pos] < a[pos-1]	swap, decrement pos
[**2**, 3, 5, 4]	0	pos == 0	increment pos
[2, **3**, 5, 4]	1	a[pos] ≥ a[pos-1]	increment pos
[2, 3, **5**, 4]	2	a[pos] ≥ a[pos-1]	increment pos:
[2, 3, 5, **4**]	3	a[pos] < a[pos-1]	swap, decrement pos
[2, 3, **4**, 5]	2	a[pos] ≥ a[pos-1]	increment pos
[2, 3, 4, **5**]	3	a[pos] ≥ a[pos-1]	increment pos
[2, 3, 4, 5]	4	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. With this optimization, the gnome sort would become a variant of the insertion sort.

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

```
1 procedure optimizedGnomeSort(a[]):
2     for pos in 1 to length(a):
3         gnomeSort(a, pos)
4
5 procedure gnomeSort(a[], upperBound):
6     pos := upperBound
7     while pos > 0 and a[pos-1] > a[pos]:
8         swap a[pos-1] and a[pos]
9         pos := pos - 1
```

Notes

1. *Almost sorted* means that each item in the list is not far from its proper position (not farther than some small constant distance).

References

1. Hamid, Sarbazi-Azad. "Hamid Sarbazi-Azad profile page" (http://sharif.edu/~azad/). Archived (https://web.archive.org/web/20181016164904/http://sharif.edu/~azad/#) from the original on 2018-10-16. Retrieved October 16, 2018.
2. Sarbazi-Azad, Hamid (2 October 2000). "Stupid Sort: A new sorting algorithm" (http://sina.sharif.edu/~azad/stupid-sort.PDF) (PDF). *Newsletter*. Computing Science Department, Univ. of Glasgow (599): 4. Archived (https://web.archive.org/web/20120307235904/http://sina.sharif.edu/~azad/stupid-sort.PDF#) (PDF) from the original on 2012-03-07. Retrieved 25 November 2014.
3. "Gnome Sort - The Simplest Sort Algorithm" (http://www.dickgrune.com/Programs/gnomesort.html). *Dickgrune.com*. 2000-10-02. Archived (https://web.archive.org/web/20170831222005/https://dickgrune.com/Programs/gnomesort.html#) from the original on 2017-08-31. Retrieved 2017-07-20.
4. Paul E. Black. "gnome sort" (http://xlinux.nist.gov/dads/HTML/gnomeSort.html). *Dictionary of Algorithms and Data Structures*. U.S. National Institute of Standards and Technology. Archived (https://web.archive.org/web/20110811012120/http://xlinux.nist.gov/dads//HTML/gnomeSort.html#) from the original on 2011-08-11. Retrieved 2011-08-20.

External links

- Gnome sort (http://dickgrune.com/Programs/gnomesort.html)