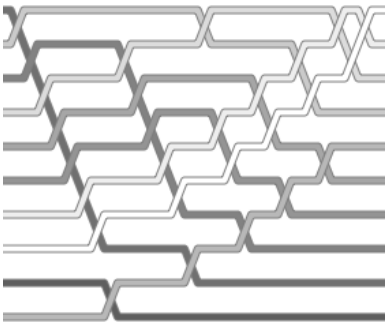
Bubble sort



A visual representation of how bubble sort works.

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

Bubble sort, also known as **sinking sort**, is a simple sorting algorithm that works by repeatedly stepping through the list to be sorted, comparing each pair of adjacent items and swapping them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted. The algorithm gets its name from the way smaller elements "bubble" to the top of the list. Because it only uses comparisons to operate on elements, it is a comparison sort. The equally simple insertion sort has better performance than bubble sort, so some have suggested no longer teaching the bubble sort.[1] [2]

Analysis

Performance

Bubble sort has worst-case and average complexity both $O(n^2)$, where n is the number of items being sorted. There exist many sorting algorithms with substantially better worst-case or average complexity of $O(n \log n)$. Even other $O(n^2)$ sorting algorithms, such as insertion sort, tend to have better performance than bubble sort. Therefore, bubble sort is not a practical sorting algorithm when n is large.

The only significant advantage that bubble sort has over most other implementations, even quicksort, but not insertion sort, is that the ability to detect that the list is sorted is efficiently built into the algorithm. Performance of bubble sort over an already-sorted list (best-case) is $O(n)$. By contrast, most other algorithms, even those with better average-case complexity, perform their entire sorting process on the set and thus are more complex. However, not only does insertion sort have this mechanism too, but it also performs better on a list that is substantially sorted (having a small number of inversions).

Rabbits and turtles

The positions of the elements in bubble sort will play a large part in determining its performance. Large elements at the beginning of the list do not pose a problem, as they are quickly swapped. Small elements towards the end, however, move to the beginning extremely slowly. This has led to these types of elements being named rabbits and turtles, respectively.

Various efforts have been made to eliminate turtles to improve upon the speed of bubble sort. Cocktail sort achieves this goal fairly well, but it retains $O(n^2)$ worst-case complexity. Comb sort compares elements separated by large gaps, and can move turtles extremely quickly before proceeding to smaller and smaller gaps to smooth out the list. Its average speed is comparable to faster algorithms like quicksort.

Step-by-step example

Let us take the array of numbers "5 1 4 2 8", and sort the array from lowest number to greatest number using bubble sort algorithm. In each step, elements written in **bold** are being compared.

First Pass:

(**5** **1** 4 2 8) → (**1** **5** 4 2 8), Here, algorithm compares the first two elements, and swaps them.

(1 **5** **4** 2 8) → (1 **4** **5** 2 8), Swap since 5 > 4

(1 4 **5** **2** 8) → (1 4 **2** **5** 8), Swap since 5 > 2

(1 4 2 **5** **8**) → (1 4 2 **5** **8**), Now, since these elements are already in order (8 > 5), algorithm does not swap them.

Second Pass:

(**1** **4** 2 5 8) → (**1** **4** 2 5 8)

(1 **4** **2** 5 8) → (1 **2** **4** 5 8), Swap since 4 > 2

(1 2 **4** **5** 8) → (1 2 **4** **5** 8)

(1 2 4 **5** **8**) → (1 2 4 **5** **8**)

Now, the array is already sorted, but our algorithm does not know if it is completed. The algorithm needs one **whole** pass without **any** swap to know it is sorted.

Third Pass:

(**1** **2** 4 5 8) → (**1** **2** 4 5 8)

(1 **2** **4** 5 8) → (1 **2** **4** 5 8)

(1 2 **4** **5** 8) → (1 2 **4** **5** 8)

(1 2 4 **5** **8**) → (1 2 4 **5** **8**)

Finally, the array is sorted, and the algorithm can terminate.

Implementation

Pseudocode implementation

The algorithm can be expressed as: procedure bubbleSort(A : list of sortable items) do swapped = false for each i in 1 to length(A) - 1 inclusive do: if A[i-1] > A[i] then swap(A[i-1], A[i]) swapped = true end if end for while swapped end procedure

Optimizing bubble sort

The bubble sort algorithm can be easily optimized by observing that the largest elements are placed in their final position in the first passes. Or, more generally, after every pass, all elements after the last swap are sorted, and do not need to be checked again. This not only allows us to skip over a lot of the elements, but also skip tracking of the "swapped" variable. This results in about a worst case 50% improvement in iteration count, but no improvement in swap counts.

To accomplish this in pseudocode we write the following: procedure bubbleSort(A : list of sortable items) n = length(A) do newn = 0 for (i = 0; i < n-1; i++) do: if A[i] > A[i+1] then swap(A[i], A[i+1]) newn = i + 1 end if end for n = newn while n > 1 end procedure

Here is the Nested Loop Method:

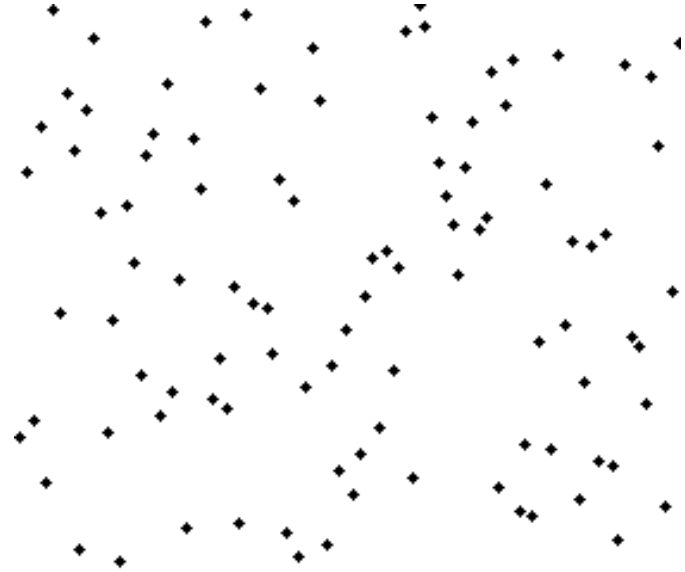
procedure bubbleSort(A : list of sortable items) n = length(A) for (i = 0; i < n; i++) /* back through the area bringing smallest remaining element to position i */ for (j = n-1; j > i; j--) if A[j-1] > A[j] then swap(A[j-1], A[j]) end if end for end for end procedure

Another method for optimizing the bubble sort is the double bubble sort, also known as the 'Bubble Bobble' sort, named after the 1986 arcade game, Bubble Bobble. [3]

In practice

Although bubble sort is one of the simplest sorting algorithms to understand and implement, its $O(n^2)$ complexity means it is far too inefficient for use on lists having more than a few elements. Even among simple $O(n^2)$ sorting algorithms, algorithms like insertion sort are usually considerably more efficient.

Due to its simplicity, bubble sort is often used to introduce the concept of an algorithm, or a sorting algorithm, to introductory computer science students. However, some researchers such as Owen Astrachan have gone to great lengths to disparage bubble sort and its continued popularity in computer science education, recommending that it no longer even be taught.[1]



A bubble sort, a sorting algorithm that continuously steps through a list, swapping items until they appear in the correct order.

The Jargon file, which famously calls bogosort "the archetypical perversely awful algorithm", also calls bubble sort "the generic **bad** algorithm".[4] Donald Knuth, in his famous book *The Art of Computer Programming*, concluded that "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", some of which he then discusses.[2]

Bubble sort is asymptotically equivalent in running time to insertion sort in the worst case, but the two algorithms differ greatly in the number of swaps necessary. Experimental results such as those of Astrachan have also shown that insertion sort performs considerably better even on random lists. For these reasons many modern algorithm textbooks avoid using the bubble sort algorithm in favor of insertion sort.

Bubble sort also interacts poorly with modern CPU hardware. It requires at least twice as many writes as insertion sort, twice as many cache misses, and asymptotically more branch mispredictions. Experiments by Astrachan sorting strings in Java show bubble sort to be roughly 5 times slower than insertion sort and 40% slower than selection sort.[1]

In computer graphics it is popular for its capability to detect a very small error (like swap of just two elements) in almost sorted array and fix it in with just linear complexity (2n). It is used for example in a polygon filling algorithm, where bounding lines are sorted by their x coordinate at specific scan line (a line parallel to x axis) and with incrementing y their order changes (two elements are swapped) only at intersections of two lines.

Variations

- Odd-even sort is a parallel version of bubble sort, for message passing systems.
- In some cases, the sort works from right to left (the opposite direction), which is more appropriate for partially sorted lists, or lists with unsorted items added to the end.

Notes

[1] Owen Astrachan. Bubble Sort: An Archaeological Algorithmic Analysis. SIGCSE 2003 Hannan Akhtar . (pdf) (http://www.cs.duke.edu/~ola/papers/bubble.pdf)

[2] Donald Knuth. *The Art of Computer Programming*, Volume 3: *Sorting and Searching*, Second Edition. Addison-Wesley, 1998. ISBN 0-201-89685-0. Pages 106–110 of section 5.2.2: Sorting by Exchanging.

[3] http://www.cs.sunysb.edu/~algorith/implement/handbook/distrib/handbook/algs/4/411b.sort.c.html

[4] http://www.jargon.net/jargonfile/b/bogo-sort.html

References

- Donald Knuth. *The Art of Computer Programming*, Volume 3: *Sorting and Searching*, Third Edition. Addison-Wesley, 1997. ISBN 0-201-89685-0. Pages 106–110 of section 5.2.2: Sorting by Exchanging.
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*, Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Problem 2-2, pg.38.
- Sorting in the Presence of Branch Prediction and Caches (https://www.cs.tcd.ie/publications/tech-reports/reports.05/TCD-CS-2005-57.pdf)

External links

- Bubble Sort implemented in 34 languages (http://codecodex.com/wiki/Bubble_sort)
- Softpanorama bubblesort page (http://www.softpanorama.org/Algorithms/Sorting/bubblesort.shtml)
- Illustrated explanation of Bubble sort (http://www.algolist.com/Bubble_sort)
- Pictorial step by step procedure of Bubble sort (http://electrofriends.com/source-codes/software-programs/c/sorting-programs/program-to-sort-the-numbers-using-bubble-sort/)
- Animated Sorting Algorithms: Bubble Sort (http://www.sorting-algorithms.com/bubble-sort) – graphical demonstration and discussion of bubble sort
- Lafore's Bubble Sort (http://lecture.ecc.u-tokyo.ac.jp/~ueda/JavaApplet/BubbleSort.html) (Java applet animation)
- A colored graphical Java applet (http://coderaptors.com/?BubbleSort) which allows experimentation with initial state and shows statistics

Article Sources and Contributors

Bubble sort *Source*: http://en.wikipedia.org/w/index.php?oldid=406324409 *Contributors*: 4wajzkd02, Abu adam, Adam Zivner, Adamuu, AdmN, Adrian.hawryluk, Agorf, Ahy1, Alansohn, Alexius08, Alfie66, Allen3, Ambassador1, Amrish deep, Andre Engels, Andrejj, Andres, Anti-Nationalist, Arvindn, AxelBoldt, Beetstra, Billaaa, Black Falcon, Blaisorblade, Booyabazooka, Bpapa2, BrianWilloughby, BrokenSegue, Bubble snipe, Buddhikaeport, CJLL Wright, Cacophony, Carey Evans, Centrx, Ceros, Charles yiming, Cl2ha, Cloudrunner, Clx321, Conversion script, Copysan, CountingPine, Crashmatrix, Cwolfsheep, DVdm, Daniel Brockman, Dantheox, Dcoetzee, DevastatorIIC, DivideByZero14, Djh2400, Drjan, Dudesleeper, E2eamon, EEMIV, ESkog, Eequor, Encyclopediamonitor, Everard Proudfoot, Foobar, Fredrik, FvdP, GLmathgrant, Garas, Gargaj, Garyzx, Gdavidp, Giftlite, Glrx, Grendelkhan, Hannan1212, Happypal, Hari, Hashar, Hate123veteran, Hefo, Hertz1888, I dream of horses, IanOsgood, Intel4004, Isis, J.delanoy, Jafet, Jdb67usa, Jellyworld, Jeltz, Jerazol, Jerk111, Jeronimo, Jevy1234, Jmallios, Jmartinezot, Jni, Josecgomez, Josh Kehn, Jossi, Justin W Smith, Kaare, Kan8eDie, Kizor, Klrste, Knutux, Kragen, Krithin, Kuru, L Kensington, LOL, Lailsonbm, LarsMarius, Lee Daniel Crocker, Liao, Looxix, Louis Kyu Won Ryu, Mahanthi1, Mantipula, Marc van Leeuwen, Materialscientist, Mattbash, Merovingian, Michael Hardy, Mortense, Mshonle, NeilFraser, NellieBly, Newuserwiki, NickShaforostoff, Nixdorf, Nmnogueira, Noisylo65, Nuno Tavares, Octahedron80, Oddity-, Officialhopsof, Ohnoitsjamie, Olaolaola, Oli Filth, Oskar Sigvardsson, Oxaric, Oğuz Ergin, Palit Suchitto, Panzi, Paranomia, PauliKL, Pedro, Pfalstad, Philip Trueman, Prestonmag, Psym, Pt, QFlyer, Quaestor, Quang thai, Quasipalm, Qxz, Qz, RTBarnard, Reedy, Rhanekom, Richie, Ronhjones, Ruolin59, Ryk, SPTWriter, Saifhhasan, Sam Hocevar, Sandyjee, Santhoshreddym, Shearsongs78, Shreevatsa, Silly rabbit, Simeon, SiobhanHansa, Sjakkalle, Sligocki, So God created Manchester, Spammyammy, Spiff, Stebbins, Stephenb, Super-Magician, Swift, THEN WHO WAS PHONE?, Themania, Tide rolls, Timwi, Toxin1000, Two Bananas, Udirock, Uranographer, Userabc, Vdaghan, Vexis, Virus326, Voyevoda, VzjrZ, Waldir, Wangbing7928, Washa rednecks, Who then was a gentleman?, WikHead, Ww, XP1, Ycl6, Yuval madar, Zaphod Beeblebrox, ZeroOne, Ztothefifth, 492 anonymous edits

Image Sources, Licenses and Contributors

File:Bubblesort-edited.png *Source*: http://en.wikipedia.org/w/index.php?title=File:Bubblesort-edited.png *License*: Public Domain *Contributors*: User:Crashmatrix

File:Bubble sort animation.gif *Source*: http://en.wikipedia.org/w/index.php?title=File:Bubble_sort_animation.gif *License*: Creative Commons Attribution-Sharealike 2.5 *Contributors*: Original uploader was Nmnogueira at en.wikipedia

License