



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

[فارسی](#)  
[Српски / srpski](#)  
[粵語](#)  
[中文](#)

[Edit links](#)

[Create account](#) [Log in](#)

Article [Talk](#)

[Read](#) [Edit](#) [View history](#)

# Bead sort

From Wikipedia, the free encyclopedia

**Bead sort** is a **natural sorting algorithm**, developed by **Joshua J. Arulanandham**, **Cristian S. Calude** and **Michael J. Dinneen** in 2002, and published in The Bulletin of the **European Association for Theoretical Computer Science**. Both **digital** and **analog** hardware **implementations** of bead sort can achieve a sorting time of  $O(n)$ ; however, the implementation of this algorithm tends to be significantly slower in **software** and can only be used to sort lists of **positive integers**. Also, it would seem that even in the best case, the algorithm requires  $O(n^2)$  space.

**Contents** [\[hide\]](#)

- [1 Algorithm overview](#)
- [2 Complexity](#)
- [3 References and notes](#)
- [4 External links](#)

## Algorithm overview [\[edit\]](#)

The bead sort operation can be compared to the manner in which beads slide on parallel poles, such as on an **abacus**. However, each pole may have a distinct number of beads. Initially, it may be helpful to imagine the beads suspended on vertical poles. In Step 1, such an arrangement is displayed using  $n=5$  rows of beads on  $m=4$  vertical poles. The numbers to the right of each row indicate the number that the row in question represents; rows 1 and 2 are representing the positive integer 3 (because they each contain three beads) while the top row represents the positive integer 2 (as it only contains two beads).<sup>[1]</sup>

If we then allow the beads to fall, the rows now represent the same integers in sorted order. Row 1 contains the largest number in the set, while row  $n$  contains the smallest. If the above-mentioned convention of rows containing a series of beads on poles 1.. $k$  and leaving poles  $k+1$ .. $m$  empty has been followed, it will continue to be the case here.

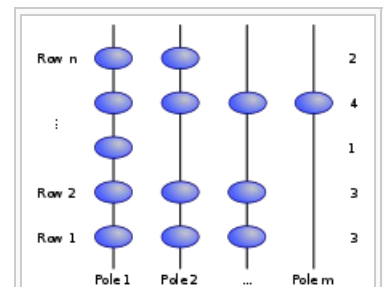
The action of allowing the beads to "fall" in our physical example has allowed the larger values from the higher rows to propagate to the lower rows. If the value represented by row  $a$  is smaller than the value contained in row  $a+1$ , some of the beads from row  $a+1$  will fall into row  $a$ ; this is certain to happen, as row  $a$  does not contain beads in those positions to stop the beads from row  $a+1$  from falling.

The mechanism underlying bead sort is similar to that behind **counting sort**; the number of beads on each pole corresponds to the number of elements with value equal or greater than the index of that pole.

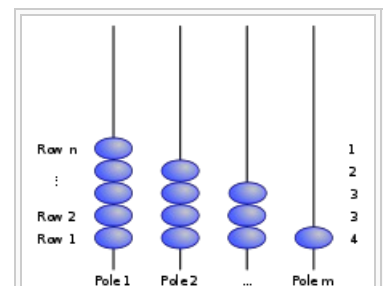
## Complexity [\[edit\]](#)

Bead sort can be implemented with four general levels of complexity, among others:

- $O(1)$ : The beads are all moved simultaneously in the same time unit, as would be the case with the simple physical example above. This is an abstract complexity, and cannot be implemented in practice.
- $O(\sqrt{n})$ : In a realistic physical model that uses gravity, the time it takes to let the beads fall is proportional to the square root of the maximum height, which is proportional to  $n$ .
- $O(n)$ : The beads are moved one row at a time. This is the case used in the analog and **digital hardware** solutions.
- $O(S)$ , where  $S$  is the sum of the integers in the input set: Each bead is moved individually. This is the case



Step 1: Suspended beads on vertical poles.



Step 2: The beads have been allowed to fall.


when bead sort is implemented without a mechanism to assist in finding empty spaces below the beads, such as in software implementations.

Like the [Pigeonhole sort](#), bead sort is unusual in that it can perform faster than  $O(n\log n)$ , the fastest performance possible for a [comparison sort](#). This is possible because the key for a bead sort is always a positive integer and bead sort exploits its structure.

### References and notes [\[edit\]](#)

- <sup>^</sup> By convention, a row representing the positive integer *k* should have beads on poles 1..*k* and poles *k*+1..*m* should be empty. This is not a strict requirement, but will most likely simplify implementation.

### External links [\[edit\]](#)

- The Bead Sort paper  PDF (114 KiB)
- Bead Sort in MGS [↗](#), a visualization of a bead sort implemented in the [MGS](#) [↗](#) programming language
- Bead Sort on MathWorld [↗](#)

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

Categories: [Sorting algorithms](#)