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
Čeština
Deutsch
فارسی
Français
Íslenska
日本語
Polski
Português
Русский
Српски / srpski
Türkçe
中文
Edit links

Introsort

From Wikipedia, the free encyclopedia

Introsort or **introspective sort** is a hybrid sorting algorithm that provides both fast average performance and (asymptotically) optimal worst-case performance. It begins with quicksort and switches to heapsort when the recursion depth exceeds a level based on (the logarithm of) the number of elements being sorted. This combines the good parts of both algorithms, with practical performance comparable to quicksort on typical data sets and worst-case O(n log n) runtime due to the heap sort. Since both algorithms it uses are comparison sorts, it too is a comparison sort.

Introsort

Class	Sorting algorithm
Data structure	Array
Worst case performance	O(n log n)
Average case performance	O(n log n)

Introsort was invented by David Musser in Musser (1997), in which he also introduced introselect, a hybrid selection algorithm based on quickselect (a variant of quicksort), which falls back to median of medians and thus provides worst-case linear complexity, which is optimal. Both algorithms were introduced with the purpose of providing generic algorithms for the C++ Standard Library which had both fast average performance and optimal worst-case performance, thus allowing the performance requirements to be tightened.[1]

Pseudocode [edit]

If a heapsort implementation and partitioning functions of the type discussed in the quicksort article are available, the introsort can be described succinctly as

```
procedure sort(A : array):
    let maxdepth = ⌊log(length(A))⌋ × 2
    introsort(A, maxdepth)

procedure introsort(A, maxdepth):
    n ← length(A)
    if n ≤ 1:
        return  // base case
    else if maxdepth = 0:
        heapsort(A)
    else:
        p ← partition(A, p)  // assume this function does pivot selection
        introsort(A[0:p], maxdepth - 1)
        introsort(A[p+1:n], maxdepth - 1)
```

The factor two in the maximum depth is arbitrary; it can be tuned for practical performance. $A[i{:}j]$ denotes the array slice of items i to j.

Analysis [edit]

In quicksort, one of the critical operations is choosing the pivot: the element around which the list is partitioned. The simplest pivot selection algorithm is to take the first or the last element of the list as the pivot, causing poor behavior for the case of sorted or nearly sorted input. Niklaus Wirth's variant uses the middle element to prevent these occurrences, degenerating to O(n^2) for contrived sequences. The median-of-3 pivot selection algorithm takes the median of the first, middle, and last elements of the list; however, even though this performs well on many real-world inputs, it is still possible to contrive a *median-of-3 killer* list that will cause dramatic slowdown of a quicksort based on this pivot selection technique. Such inputs could potentially be exploited by

an aggressor, for example by sending such a list to an Internet server for sorting as a denial of service attack.

Musser reported that on a median-of-3 killer sequence of 100,000 elements, introsort's running time was 1/200 that of median-of-3 quicksort. Musser also considered the effect on caches of Sedgewick's delayed small sorting, where small ranges are sorted at the end in a single pass of insertion sort. He reported that it could double the number of cache misses, but that its performance with double-ended queues was significantly better and should be retained for template libraries, in part because the gain in other cases from doing the sorts immediately was not great.

Implementations [edit]

Introsort or some variant is used in a number of standard library sort functions, including some C++ sort implementations.

The June 2000 SGI C++ Standard Template Library stl_algo.h implementation of unstable sort uses the Musser introsort approach with the recursion depth to switch to heapsort passed as a parameter, median-of-3 pivot selection and the Knuth final insertion sort pass. The element threshold for switching to the simple insertion sort was 16.

The GNU Standard C++ library uses a hybrid sorting algorithm: first introsort is performed, to a maximum depth given by $2 \times \log_2 n$, where n is the number of elements, followed by an insertion sort on the result.[2]

The Microsoft .NET Framework Class Library, starting from version 4.5 (2012), uses Introsort instead of simple QuickSort.[3]

References [edit]

1. ^ "Generic Algorithms", David Musser
2. ^ libstdc++ Documentation: Sorting Algorithms
3. ^ http://msdn.microsoft.com/en-us/library/6tf1f0bc(v=vs.110).aspx

- Musser, David R. (1997). "Introspective Sorting and Selection Algorithms". *Software: Practice and Experience* (Wiley) **27** (8): 983–993. doi:10.1002/(SICI)1097-024X(199708)27:8<983::AID-SPE117>3.0.CO;2-# (inactive 2015-03-14). edit
- Niklaus Wirth. "Algorithms and Data Structures". Prentice-Hall, Inc., 1985. ISBN 0-13-022005-1.

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
Exchange sorts	Bubble sort · Cocktail sort · Odd–even sort · Comb sort · Gnome sort · Quicksort · Stooge sort · 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: Comparison sorts