Introselect

From Wikipedia, the free encyclopedia

In computer science, **introselect** (short for "introspective selection") is a selection algorithm that is a hybrid of quickselect and median of medians which has fast average performance and optimal worst-case performance. Introselect is related to the introsort sorting algorithm: these are analogous refinements of the basic quickselect and quicksort algorithms, in that they both start with the quick algorithm, which has good average performance and low overhead, but fall back to an optimal worst-case algorithm (with higher overhead) if the quick algorithm does not progress rapidly enough. Both algorithms were introduced by David Musser in (Musser 1997), 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] However, in most C++ Standard Library implementations, another "introselect" algorithm is used, which combines quickselect and heapselect, and has a worst-case running time of $O(n \log n)$.

Introselect

Class	**Selection algorithm**
Data structure	Array
Worst-case performance	O(n)
Best-case performance	O(n)

Algorithms

Introsort achieves practical performance comparable to quicksort while preserving $O(n \log n)$ worst-case behavior by creating a hybrid of quicksort and heapsort. Introsort starts with quicksort, so it achieves performance similar to quicksort if quicksort works, and falls back to heapsort (which has optimal worst-case performance) if quicksort does not progress quickly enough. Similarly, introselect combines quickselect with median of medians to achieve worst-case linear selection with performance similar to quickselect.

Introselect works by optimistically starting out with quickselect and only switching to a worst-case linear-time selection algorithm (the Blum-Floyd-Pratt-Rivest-Tarjan median of medians algorithm) if it recurses too many times without making sufficient progress. The switching strategy is the main technical content of the algorithm. Simply limiting the recursion to constant depth is not good enough, since this would make the algorithm switch on all sufficiently large lists. Musser discusses a couple of simple approaches:

- Keep track of the list of sizes of the subpartitions processed so far. If at any point k recursive calls have been made without halving the list size, for some small positive k, switch to the worst-case linear algorithm.
- Sum the size of all partitions generated so far. If this exceeds the list size times some small positive constant k, switch to the worst-case linear algorithm. This sum is easy to track in a single scalar variable.

Both approaches limit the recursion depth to $k \lceil \log n \rceil = O(\log n)$ and the total running time to $O(n)$.

The paper suggested that more research on introselect was forthcoming, but the author retired in 2007 without having published any such further research.

References

1. "Generic Algorithms (http://www.cs.rpi.edu/~musser/gp/algorithms.html)", David Musser

- Musser, David R. (1997). "Introspective Sorting and Selection Algorithms" (http://www.cs.rpi.edu/~musser/gp/introsort.ps). *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 2017-01-15).