

ARRAY-PARTITION

I. Language Agnostic Overview

This code is part of the Quicksort algorithm, one of the fastest and most efficient sorting algorithms in use today. Quicksort is extremely common, used explicitly in many programs and implicitly in others, since it is part of the Java libraries. Since it is so frequently used, correctness is required, and proving this correctness for all situations is worthwhile.

This specific implementation is from Saswat Anand, Corina S. Pasareanu, and Willem Visser. Symbolic execution with abstract subsumption checking. *SPIN Workshop on Model Checking of Software*, 2006. This was in turn implemented from: Tomas Ball. A theory of predicate-complete test coverage and generation. Technical report, Microsoft Research, 2004.

II. Java Specific Overview

This class divides an array into upper and lower halves, using the first element (`a[0]`) as a pivot. It does this by moving all elements whose value is larger than the pivot to after all the elements whose value is smaller than the pivot. Ideally, this pivot will be the median value, and after partitioning the array would be evenly divided between low and high values. It should also be noted that the algorithm uses an in-place partitioning scheme to save memory.

As the name indicates, this class is implemented with an array rather than a linked list or tree of any kind. This implementation has several advantages and disadvantages, and a future study may wish to consider the same algorithm with different underlying data structures.

The invariant is realized through a basic assert and function.

partition

- a. This is the partitioning method which divides the array into upper and lower parts. The dividing line is set at the value of the first element.
- b. The preconditions verify that the underlying array is not null, that there are at least three elements in the array and that the count of the number of elements accurately reflects the size of the array. The postcondition verifies that the array is “weakly” sorted, see the discussion of the invariant for thorough explanation.

isWeaklySorted

- a) This invariant verifies that the array, having been partitioned, is divided into two halves. The first half must be less than or equal to the pivot (`a[0]`) and the second half must be strictly greater than the pivot.

b) As it is used in other invariants, this method has no invariants of its own.

Metrics:

Lines of code for implementation: 13

Methods for implementation: 1

Lines of code for invariants: 22

Methods for invariants: 2

References:

Wikipedia's entry on in-place partitioning: http://en.wikipedia.org/wiki/Quicksort#Version_with_in-place_partition

Data Structures and Algorithm Analysis in Java by Mark Allen Weiss. Section 7.7.2: “Partitioning Strategy.” pgs 266 – 268.