Quick Sort

LA home
Computing
Algorithms
Bioinformatics
FP, λ
Logic, π
MML
Prog.Langs

Algorithms
glossary
Sorting
Insertion
Quick
Merge
Heap
Dutch N.F.
Radix

Quick sort partitions the array into two sections, the first of "small" elements and the second of "large" elements. It then sorts the small and large elements separately.

Ideally, partitioning would use the *median* of the given values, but the median can only be found by scanning the whole array and this would slow the algorithm down. In that case the two partitions would be of equal size; In the simplest versions of quick sort an arbitrary element, typically the first element is used as an estimate (guess) of the median.

```
quicksort(int a[], int lo, int hi)
/* sort a[lo..hi] */
 { int left, right, median, temp;

   if( hi > lo ) /* i.e. at least 2 elements, then */
    { left=lo; right=hi;
      median=a[lo];  /* NB. just an estimate! */

      while(right >= left) /* partition a[lo..hi] */
      /* a[lo..left-1]<=median and a[right+1..hi]>=median */
       { while(a[left] < median) left++;
         /* a[left] >= median */

         while(a[right] > median) right--;
         /* a[left] >= median >= a[right] */

         if(left > right) break;

         //swap:
         temp=a[left]; a[left]=a[right]; a[right]=temp;
         left++; right--
       }
      /* a[lo..left-1]<=median and a[right+1..hi]>=median
         and left > right */

      quicksort(a, lo, right);// divide and conquer
      quicksort(a, left,  hi);
    }
 }/*quicksort*/


function quick(a, N)
/* sort a[1..N],  N.B. 1 to N */
 { quicksort(a, 1, N); }
```

Change the data in the HTML FORM below, click `go', and experiment:

input: 6 2 3 1 5 4

go

output:

trace:

Complexity

Time

In the *best case*, the partitions are of equal size at each recursive call, and there are then $\log_2(N)$ *levels* of recursive calls. The whole array is scanned at each *level* of calls, so the total work done is O(N*log(N)).

The *average* time complexity is also O(N*log(N)).

The *worst case* time complexity is $O(N^2)$. This occurs when the estimate of the median is systematically always poor, e.g. on already sorted data, but this is very unlikely to happen by chance.

Space

As coded above the best- and average-case space-complexity is O(log(N)), for the stack-space used.

The worst-case space-complexity is O(N), but it can be limited to O(log(N)) if the code is modified so that the *smaller* half of the array is sorted first (and an explicit stack, or the tail-recursion optimisation, used).

In that case, the best-case space-complexity becomes O(1)
[-- Andrew Clausen '05], "`gcc -O2` does tail-recursion optimization, but `-O1` doesn't."

Stability

Quick sort is not stable.

Testing

It is *very easy* to make errors when programming Quick sort. The basic idea is simple but the details of the manipulation of the

"pointers" hi, lo, left, right, are very easily messed up - this is the voice of bitter experience!

Notes

- C. A. R. Hoare, *Algorithm 63, Partition*; *Algorithm 64, Quicksort*, p321; *Algorithm 65: FIND*, Comm. of the ACM, **4** p321-322, 1961.
  C. A. R. Hoare, *Quicksort*, Comp. J. **5**(1) p10-15 1962.
  Tony Hoare published quick sort in 1961. (See the [bib'].)
- The particular coding above is after N. Wirth, *Algorithms and Data Structures*, Prentice-Hall 1986.
- Partitioning has other applications, e.g. to find the k^{th} largest or smallest element of an array (e.g. median when k=N/2) without completely sorting the array.
- Quick sort can be made faster, i.e. its constant of proportionality reduced, by various techniques:
  - Use a simpler sort, e.g. insertion sort, when the section of the array to be sorted is "small".
  - Use a better estimate for the median, e.g. median of three.
  - Implement a non-recursive version with an explicit stack. In this case push the *larger* partition on the stack while the smaller section is sorted, because this limits the stack requirements to $O(\log_2(N))$.
- See the Dutch National Flag problem for the possibilities of a 3-way partition and quick-sort.
