1D1020 2018-09-14 #6

Mergesort and Quicksort
Key components of IT-infrastructure

Divide and Conquer
- Divide large problems in smaller managable problems
- Merge results

Time complexity recurrence relation

Mergesort (John von Neumann)
- Halve array
- Recursively sort halves
- Merge the halves

Precondition: must be true before
Postcondition: must be true after

assert statement (-ea to enable in compiler)

Insertion sort (N^2) Mergesort ($N \lg N$)

In-place uses $O(1)$ (often $\leq c \log N$) auxiliary memory.

In-place merge (Konrod 1969)

Improvements: -Use insertion sort on ~10 elements at a time
-Check first/last element to see if arrays are sorted
-Eliminate copying to auxiliary array

Bottom-up Mergesort

- Pass through and merge sub-arrays of size 1, 2, 4, ...
- Less memory usage but ~10% slower due to caching.

Time complexity of merge sort

- $N \lg N$ Best worst case possible (For comparison based sorting)
- Mergesort is optimal with regard to comparisons. (Not with regard to memory usage)

Sorting not based on comparison may be faster
- Radix, tries

Lower bound can be reduced if the algorithm can

- Exploit knowledge of how the input is ordered
  Insertion sort is $O(N)$ for partially sorted arrays
- Exploit the distribution of key values
  3-way quicksort only needs $O(N)$ comparisons for limited # of keys.
- Representation of the keys
  Radix sort does not compare keys

Mergesort comparisons

Comparable - natural order
Comparator - alternative order
- must define a total order.

Stability
- A stable sort maintains the relative order of elements with equal values.

- Merge sort is stable if we take from left array when equal
- Insertion sort is stable
- Selection sort is not stable
- Shellsort is not stable