Project
Algorithm Engineering

Jens K. Mueller

jkm@informatik.uni-jena.de

Department of Mathematics and Computer Science
Friedrich-Schiller-University Jena

Wednesday 25th June, 2014

Algorithmic Improvements

Comparison-Based Sorting Algorithms

Running Time Complexity

case algorithm	worst	average	best
Insertion sort	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n)$
Mergesort	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n \log n)$
Quicksort	$\Theta(n^2)$	$\Theta(n \log n)$	$\Theta(n \log n)$

Comparison-Based Sorting Algorithms

Space Complexity

case algorithm	worst	average	best
Insertion sort	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Mergesort	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$
Quicksort	$\Theta(\log n)/\Theta(n)$	$\Theta(\log n)$	$\Theta(\log n)$

Quicksort

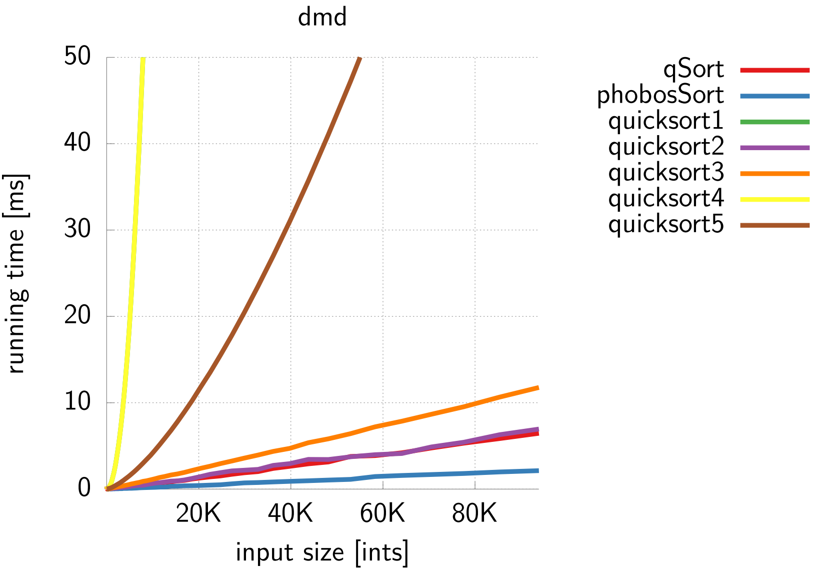
Quicksort sketch

1. Choose pivot element
2. Partition
3. Recurse

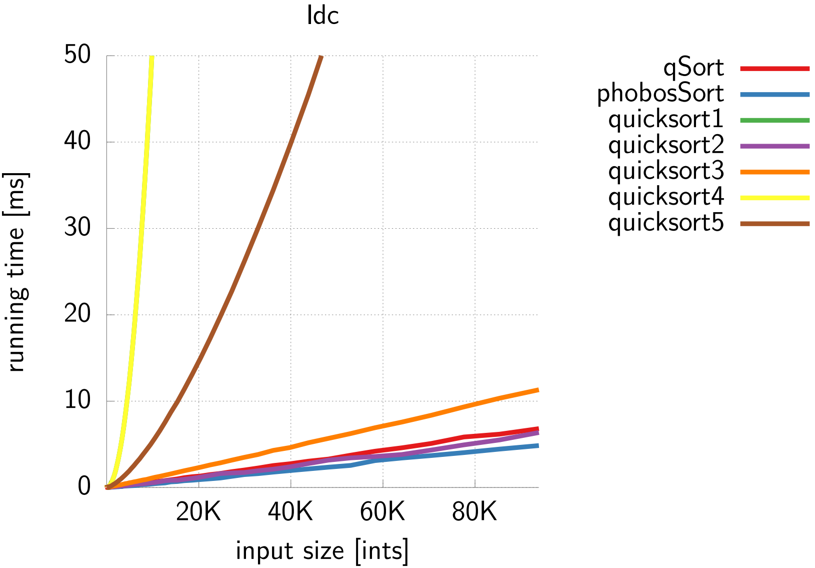
Algorithmic improvements

- Median of three
- Random pivot
- Insertion sort for small input
- Fat partition

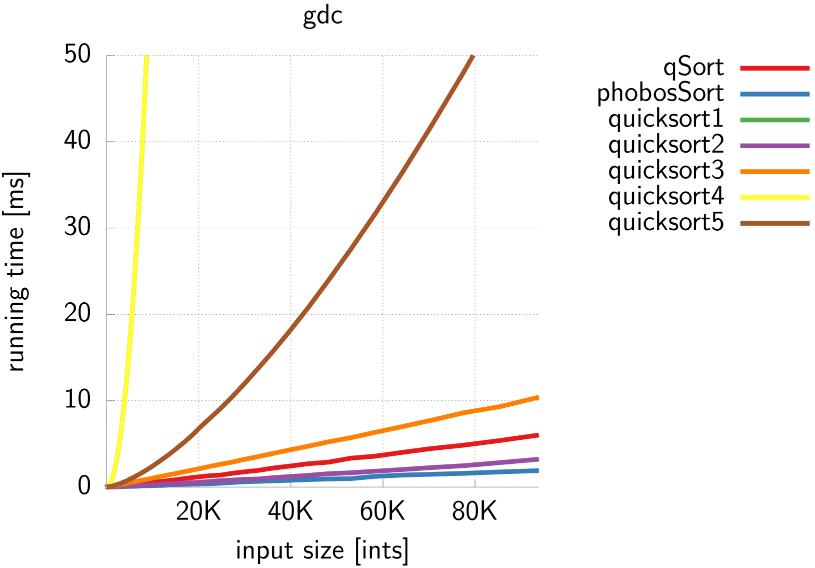
Quicksort on Sorted Input



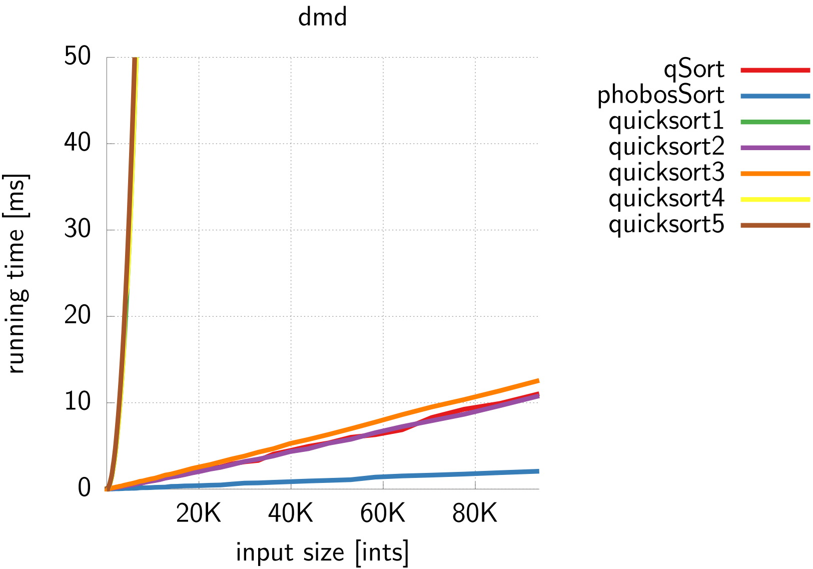
Quicksort on Sorted Input (cont.)



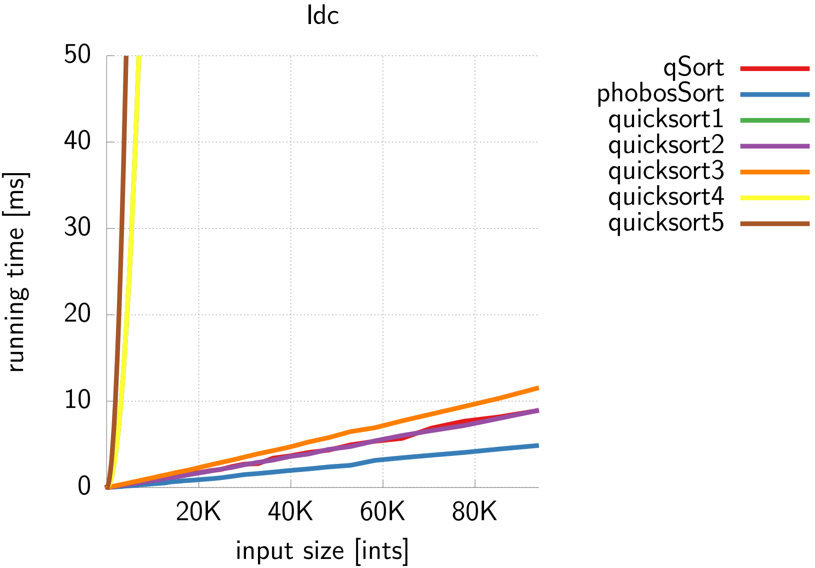
Quicksort on Sorted Input (cont.)



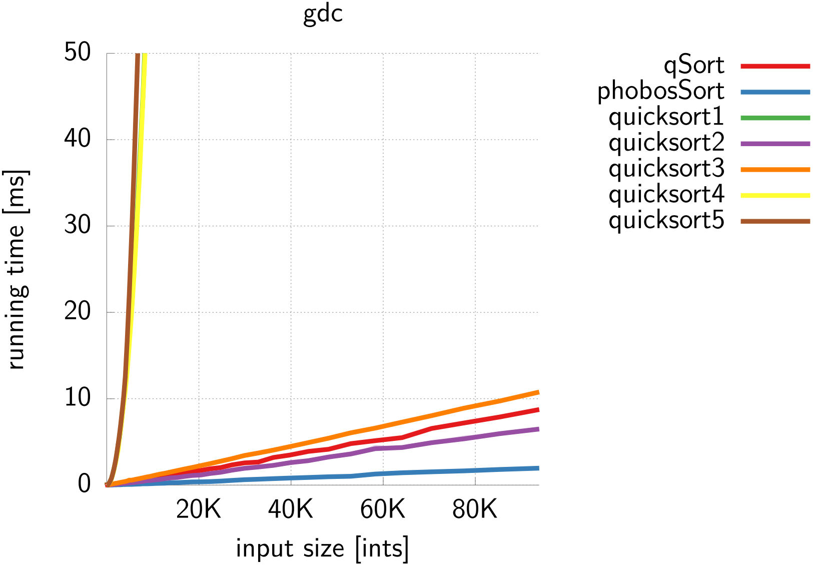
Quicksort on Reverse Sorted Input



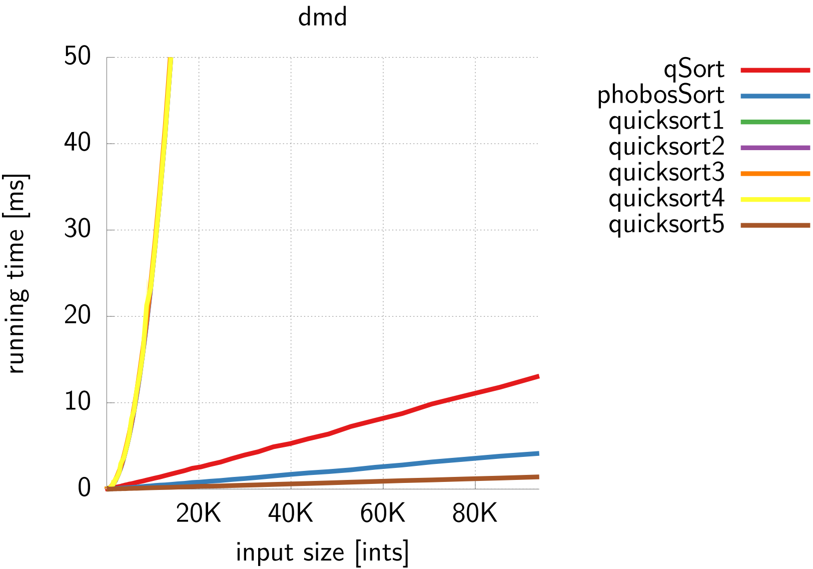
Quicksort on Reverse Sorted Input (cont.)



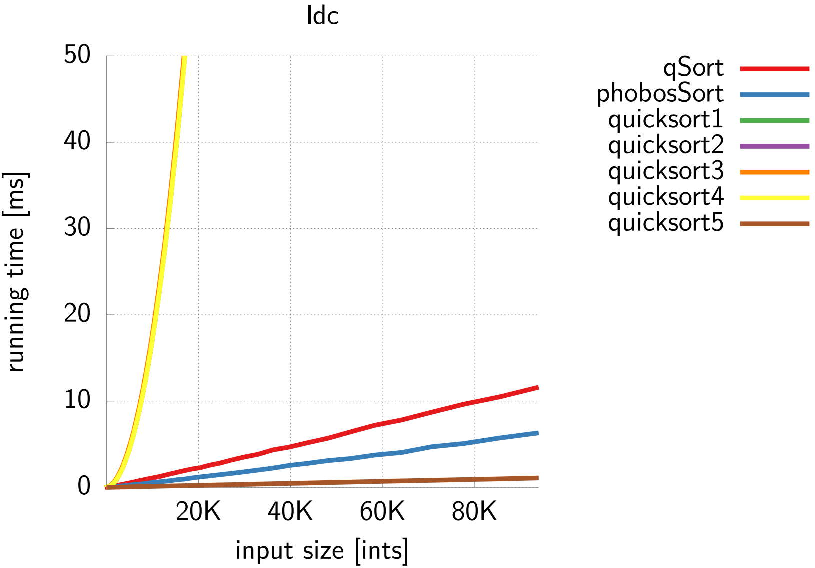
Quicksort on Reverse Sorted Input (cont.)



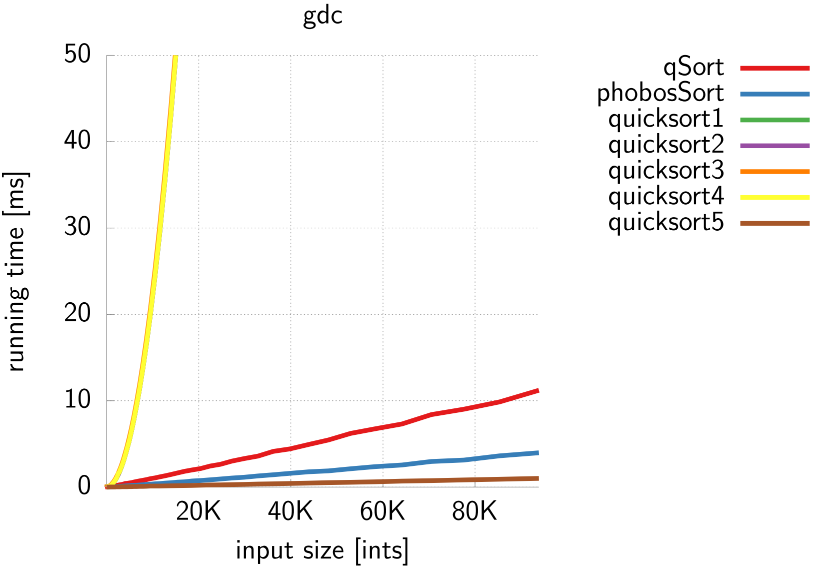
Quicksort on Three Repeated Elements



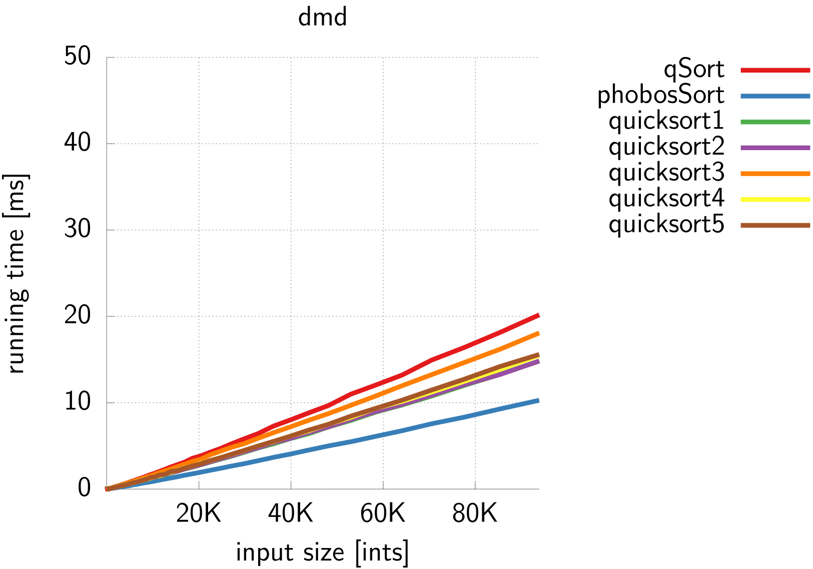
Quicksort on Three Repeated Elements (cont.)



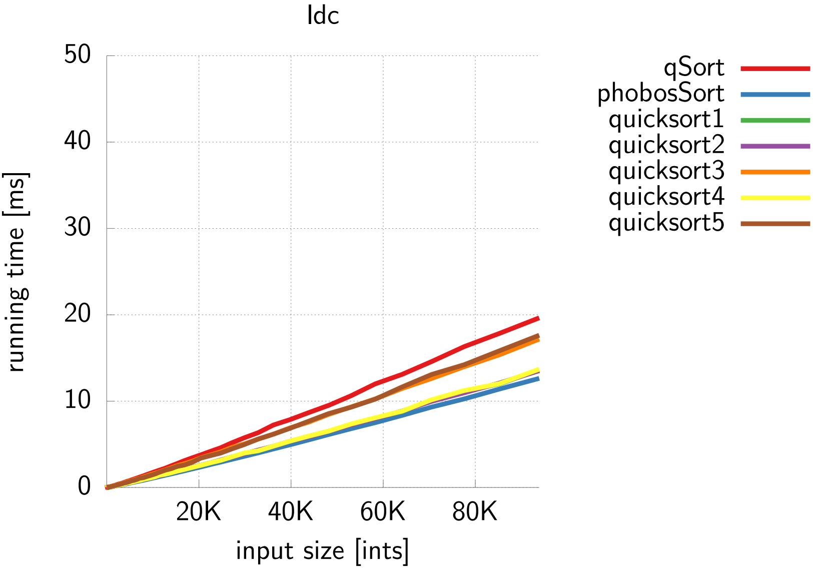
Quicksort on Three Repeated Elements (cont.)



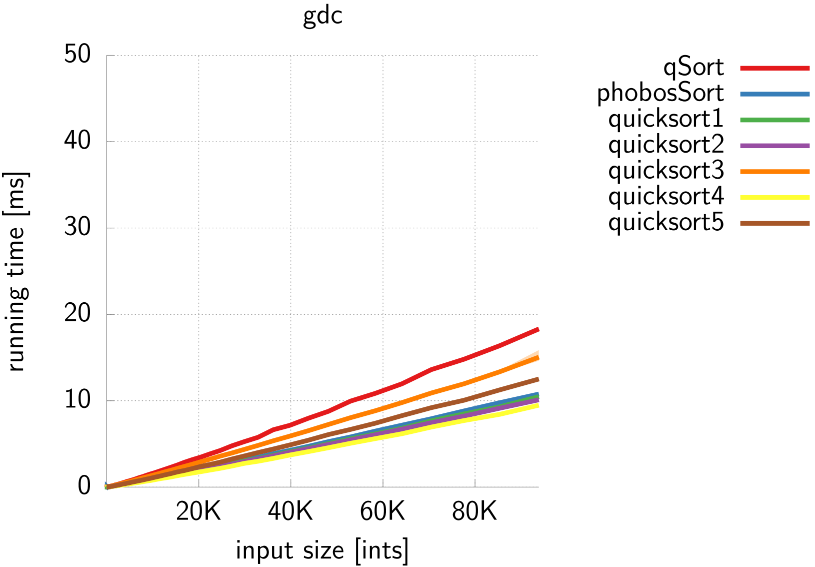
Quicksort on Random Input



Quicksort on Random Input (cont.)



Quicksort on Random Input (cont.)



Summary

Critical improvements

- Better pivot and fat partition to counter act worst cases
- Insertion sort on small sizes

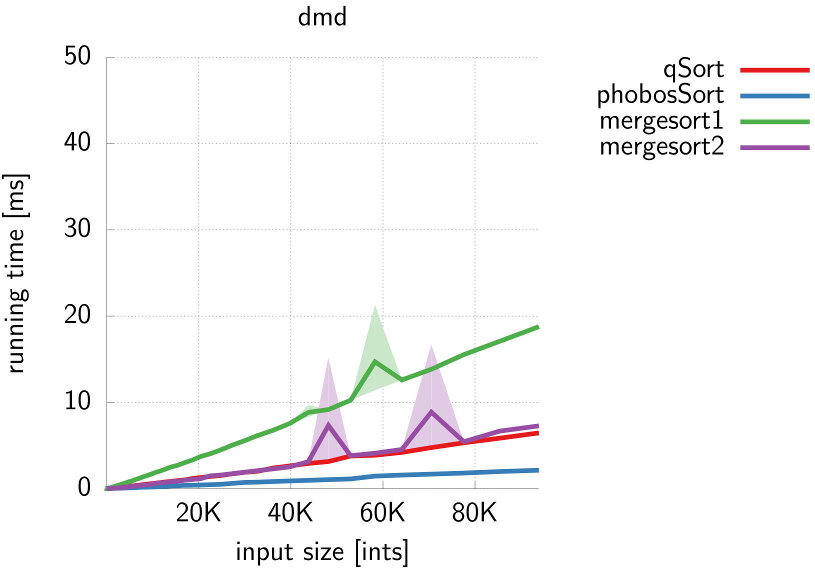
Mergesort

1. Half input
2. Recurse
3. Merge sorted halfs

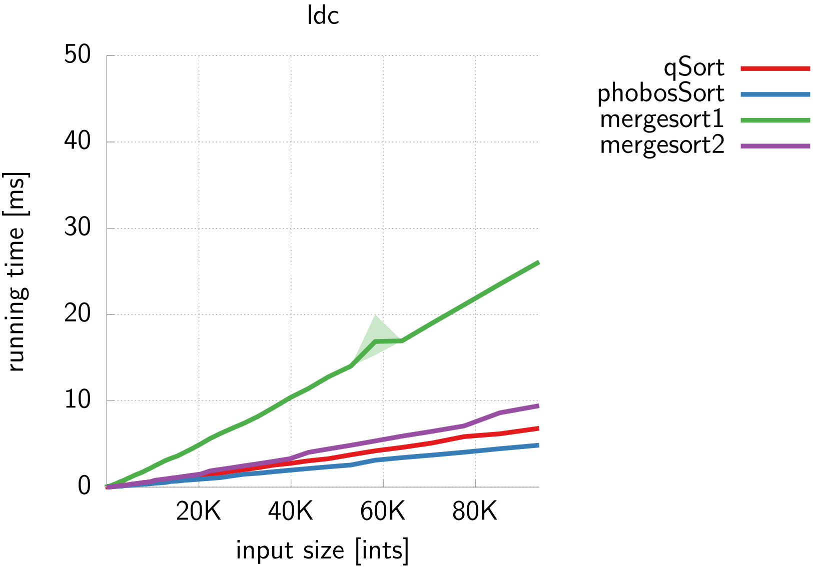
Algorithmic improvements

- Allocate additional memory once
- Improve merge
- Insertion sort on small sizes

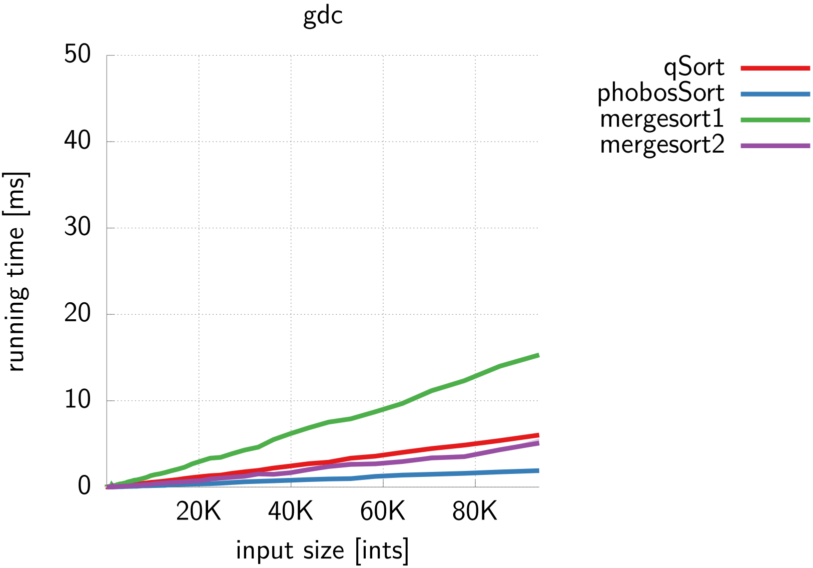
Mergesort on Sorted Input



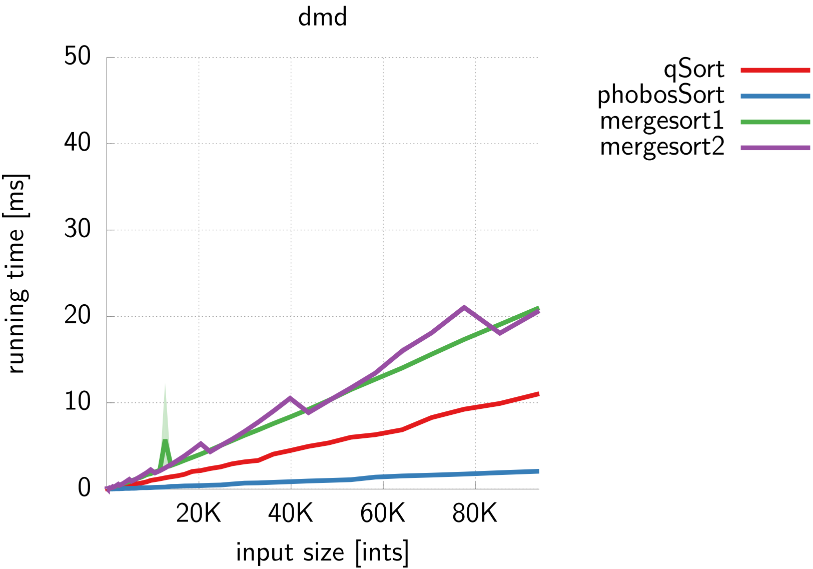
Mergesort on Sorted Input (cont.)



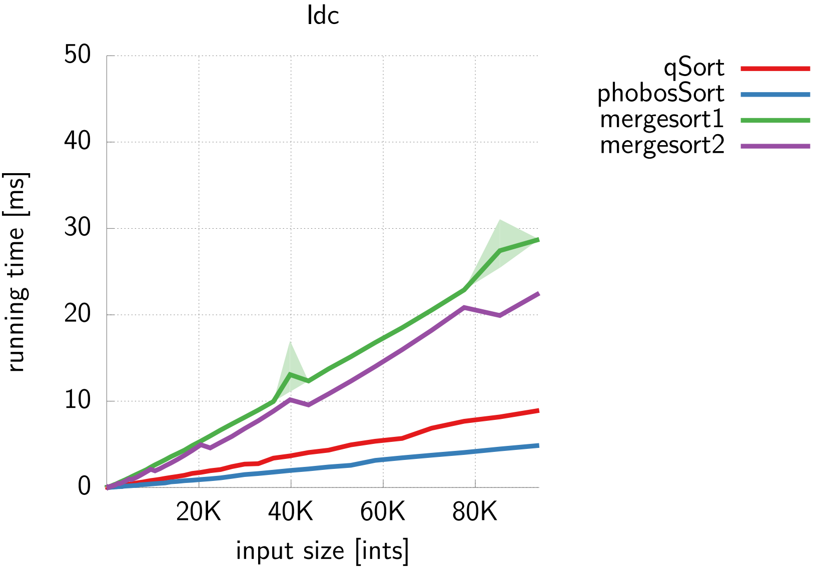
Mergesort on Sorted Input (cont.)



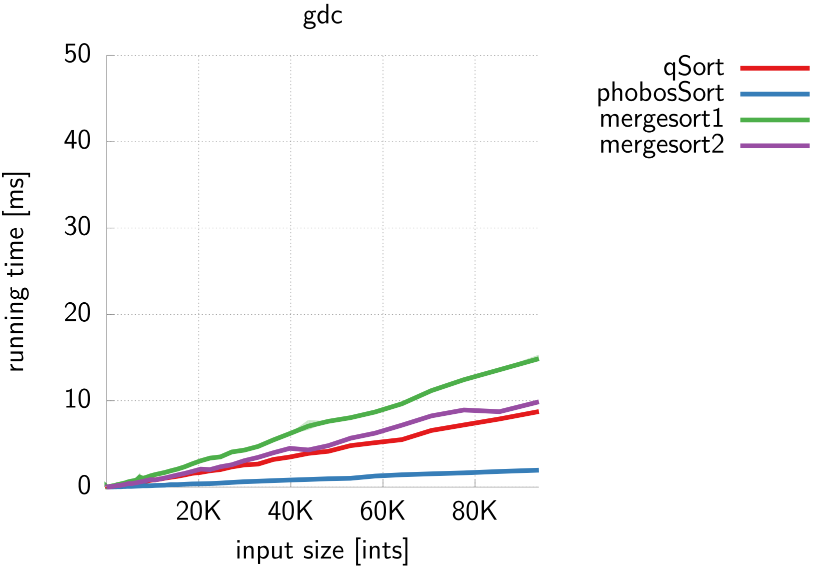
Mergesort on Reverse Sorted Input



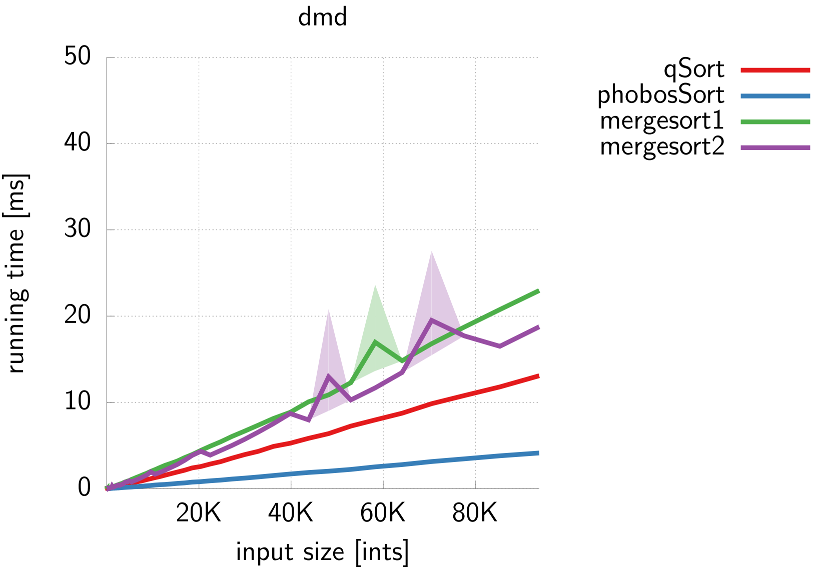
Mergesort on Reverse Sorted Input (cont.)



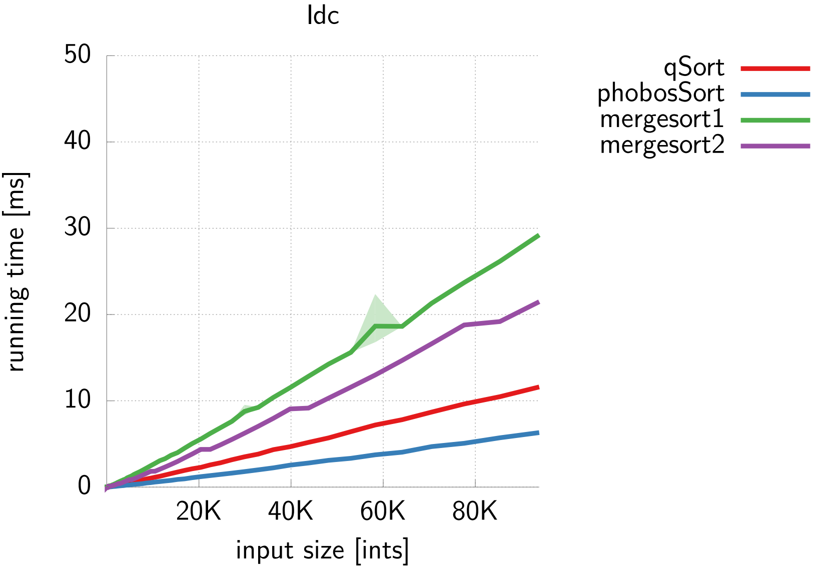
Mergesort on Reverse Sorted Input (cont.)



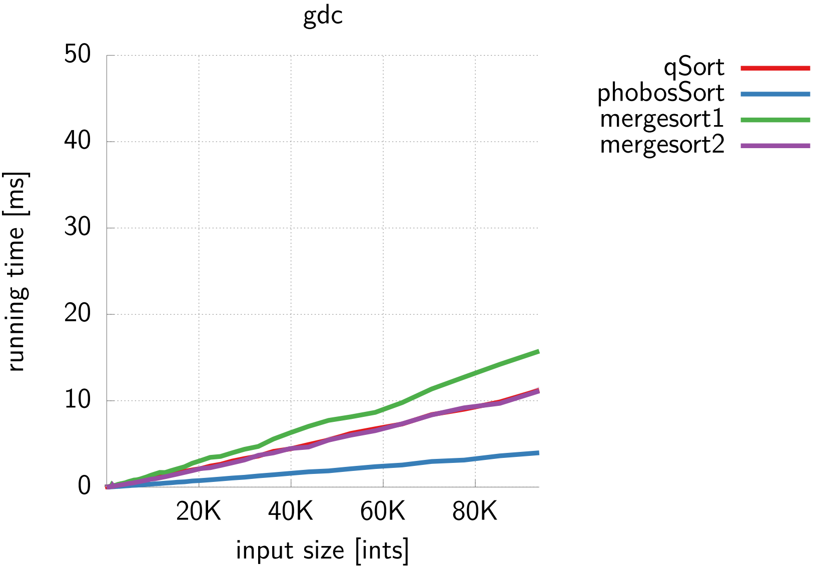
Mergesort on Three Repeated Elements



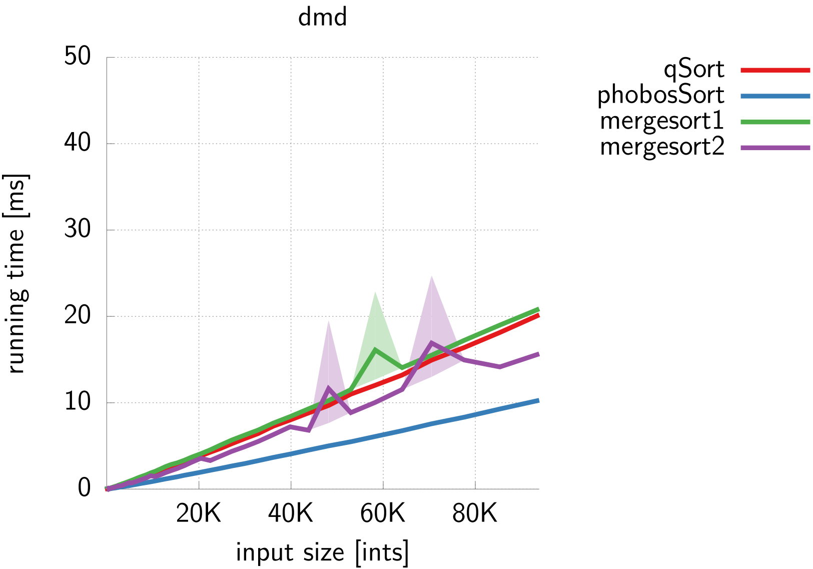
Mergesort on Three Repeated Elements (cont.)



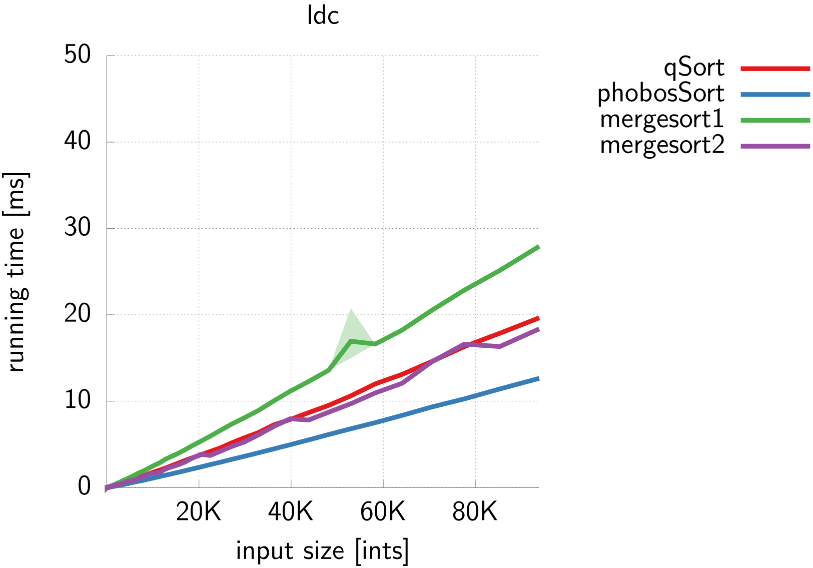
Mergesort on Three Repeated Elements (cont.)



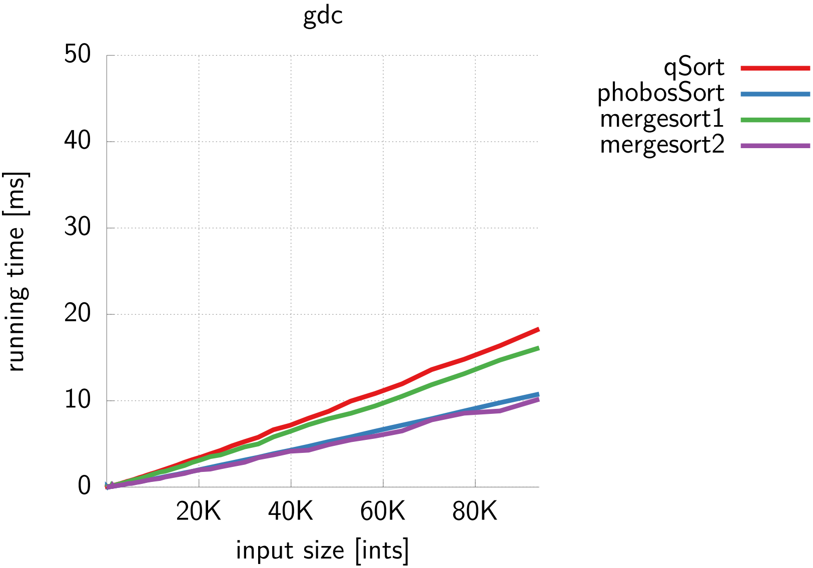
Mergesort on Random Input



Mergesort on Random Input (cont.)



Mergesort on Random Input (cont.)



Summary

Mergesort improvements

- Linear additional memory
- Improve merge further

Insertion Sort

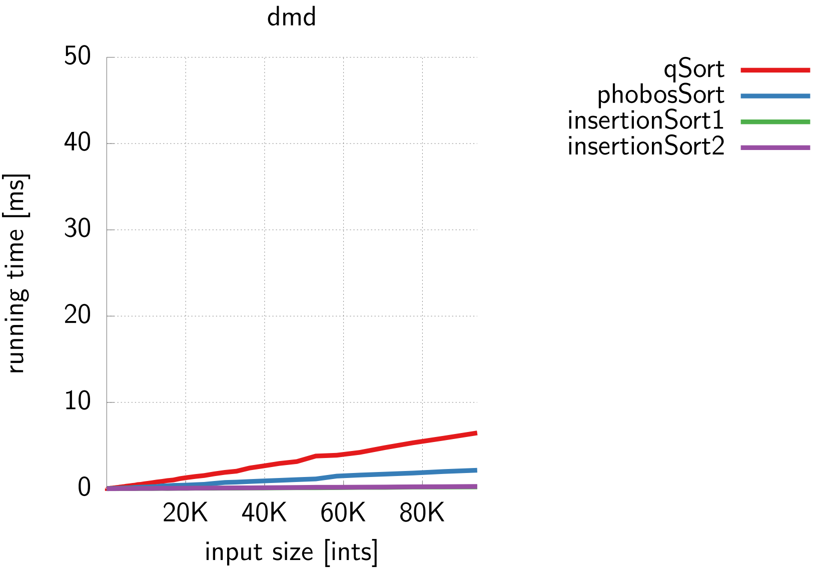
Sketch

1. Assume first i elements are sorted
2. Insert new element $i + 1$ in sorted manner

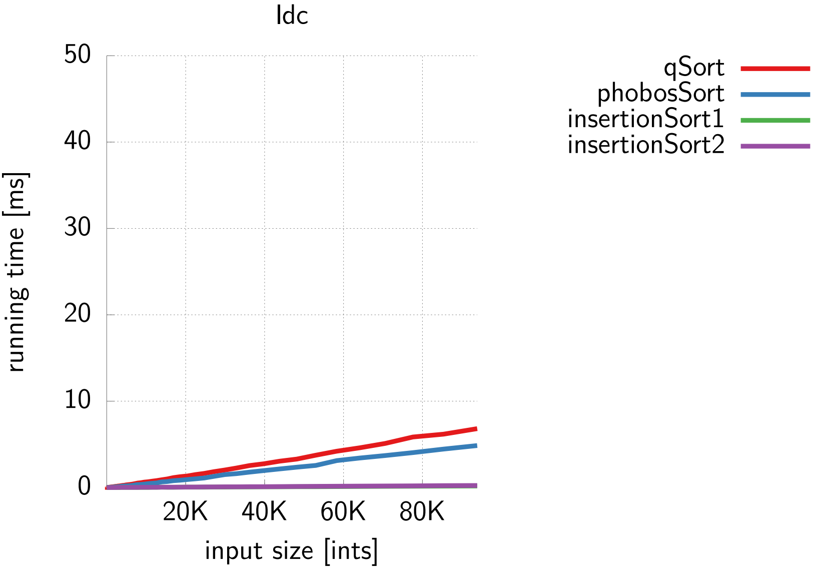
Algorithmic improvements

- Use less memory accesses

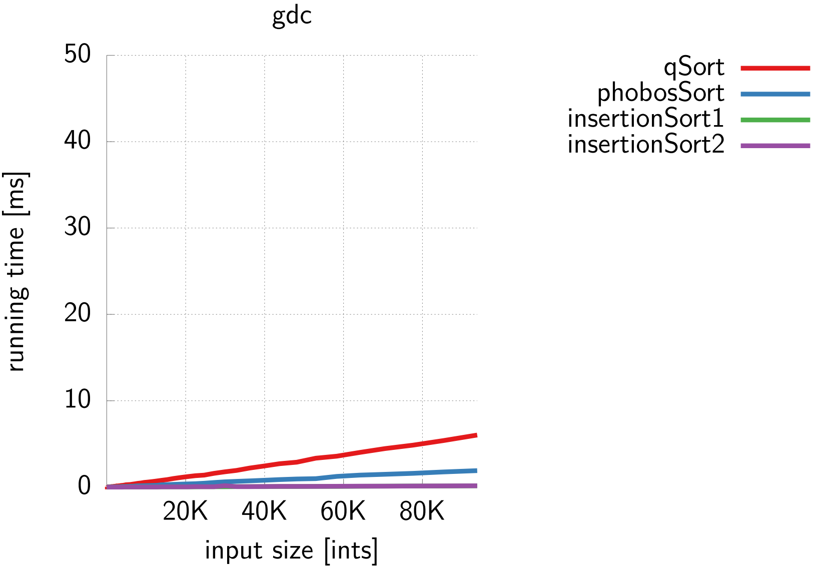
Insertion Sort on Sorted Input



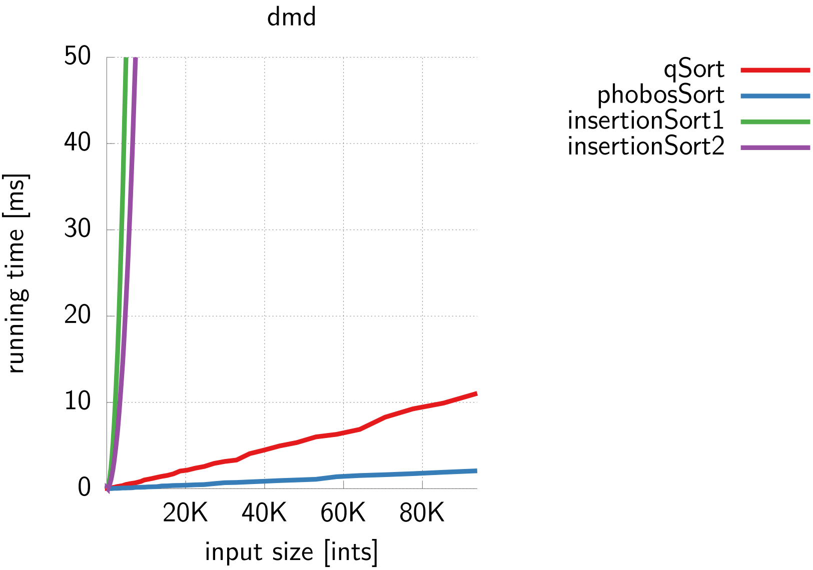
Insertion Sort on Sorted Input (cont.)



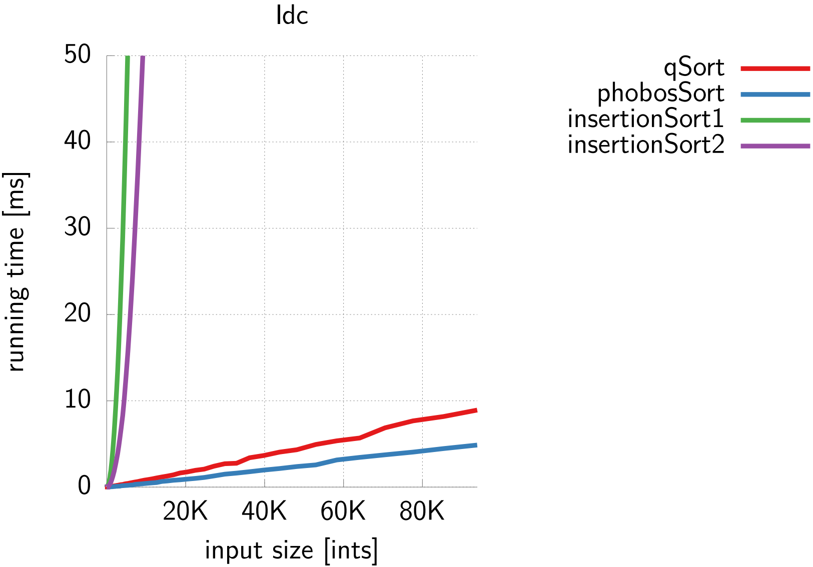
Insertion Sort on Sorted Input (cont.)



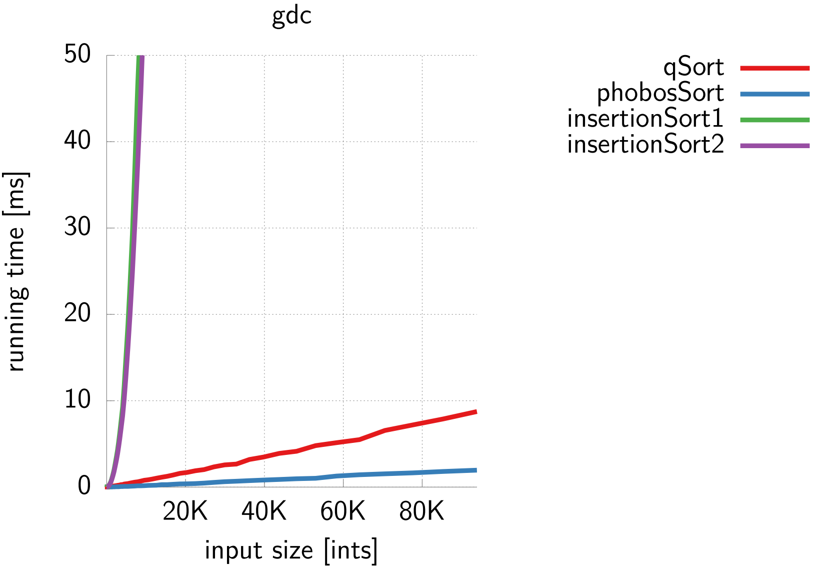
Insertion Sort on Reverse Sorted Input



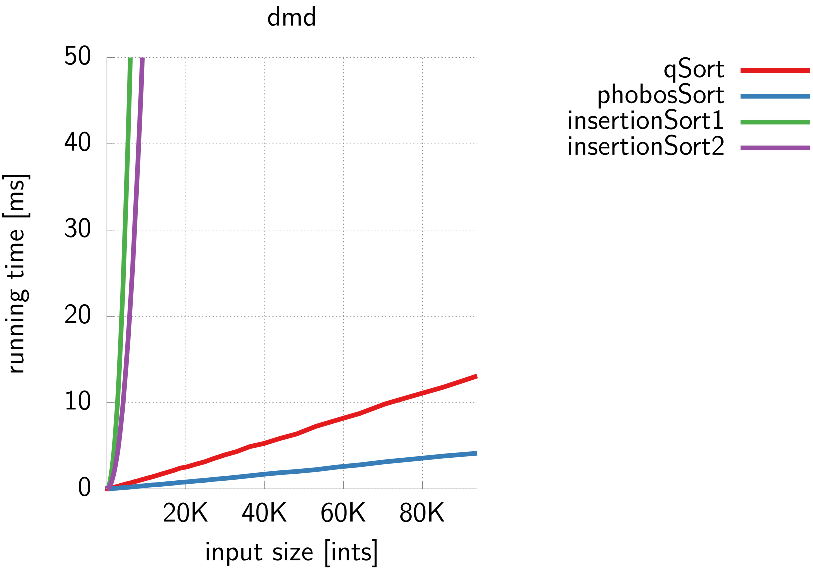
Insertion Sort on Reverse Sorted Input (cont.)



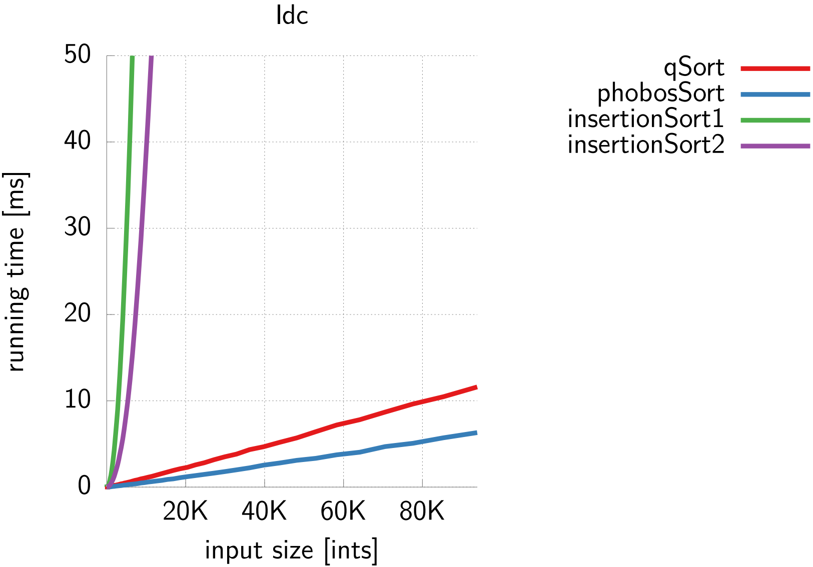
Insertion Sort on Reverse Sorted Input (cont.)



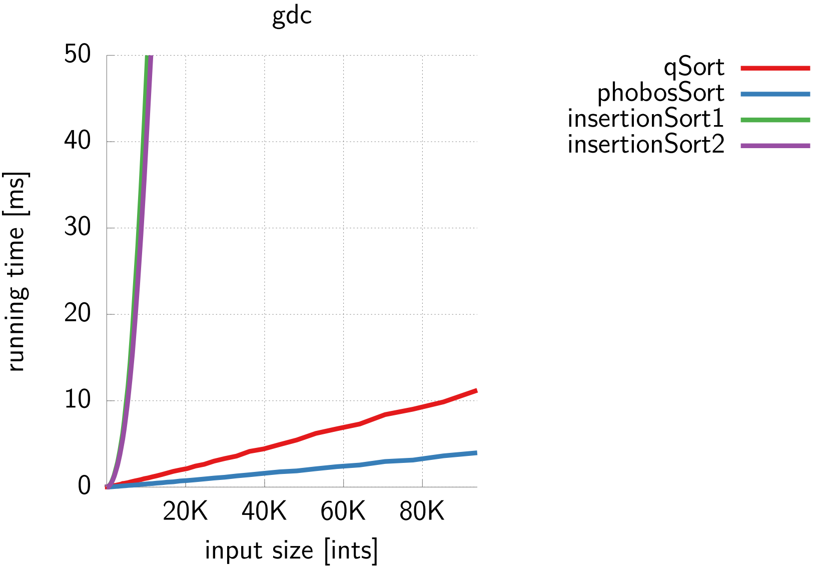
Insertion Sort on Three Repeated Elements



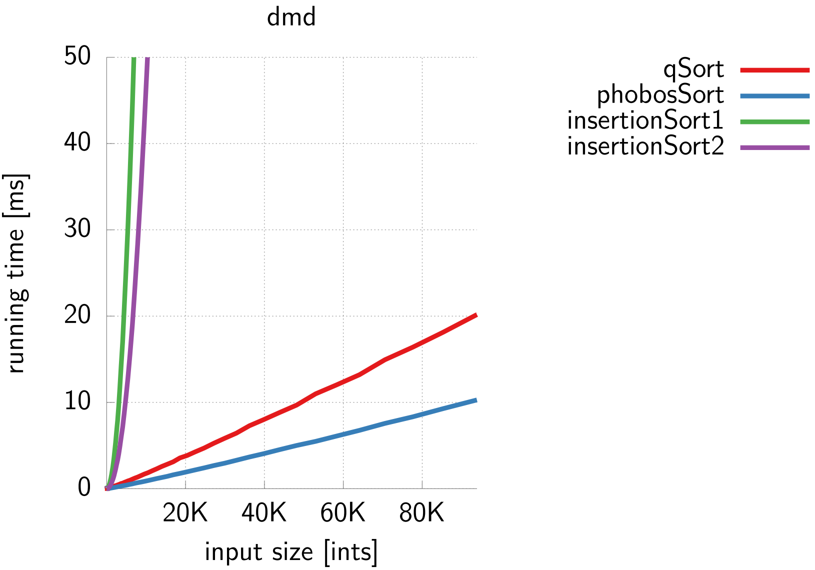
Insertion Sort on Three Repeated Elements (cont.)



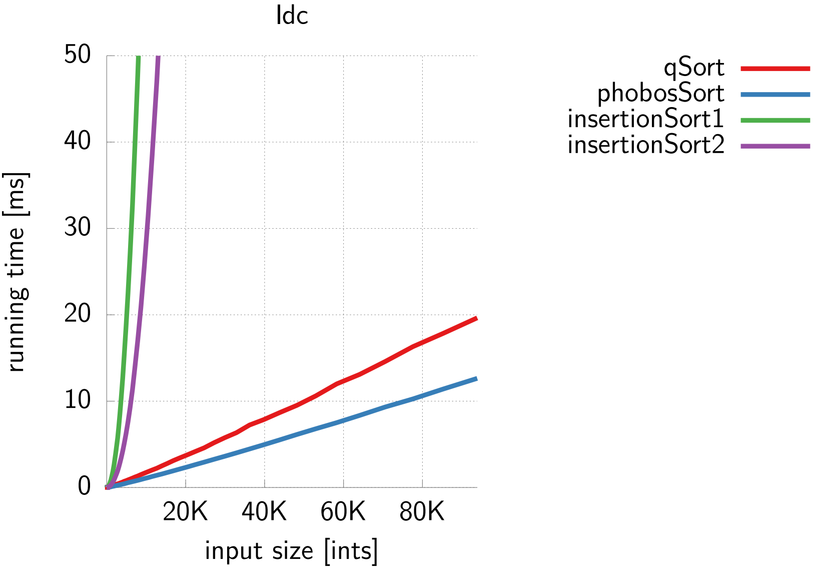
Insertion Sort on Three Repeated Elements (cont.)



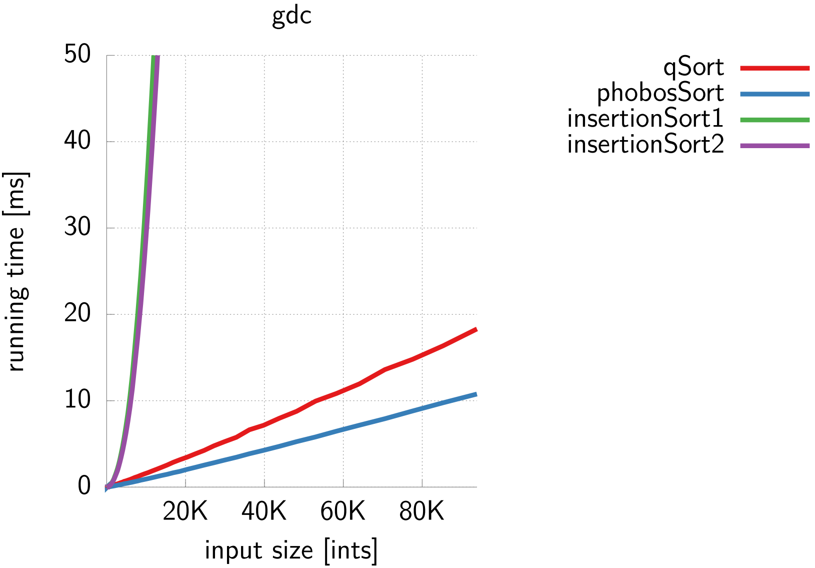
Insertion Sort on Random Input



Insertion Sort on Random Input (cont.)



Insertion Sort on Random Input (cont.)



Summary

- Adjust algorithm to real input (if possible)
- Worst cases may not be representative
- Improvement may introduce overhead
- Constants usually differ for different algorithms
- Average case analysis usually difficult
- On small instances non-optimal algorithm may be better
- Many small instances vs. few big instances

Tension between practice (specialize, complex) and theory (generalize, simplified)