On Adaptive Integer Sorting

Anna Pagh, IT University of Copenhagen
Rasmus Pagh, IT University of Copenhagen
Mikkel Thorup, AT&T Labs

Presentation at ESA, 16/9, 2004

Outline of talk

- Integer sorting
- Adaptive sorting
- Our result
- General idea: Reduction to multisorting
- Analysis

Word RAM model

- Word RAM model:
  - Unit cost operations on w-bit words: bitwise boolean operations, arithmetic operations, shifts.
  - Integers can be used as addresses.
- Is multiplication a constant time operation?

Problem formulation

- Sort n words of w bits on a word RAM.
- In this talk we only consider algorithms using O(n) space.
- We allow multiplication.

History of integer sorting

Reference	Time complexity	Notes
Radix sort: Comrie '29	O(n)	Requires w=O(log n)
Fredman & Willard '90	O(n log n/log log n)	
	$O(n \log^{1/2} n)$	Expected time
Andersson et al. '95	O(n log log n)	Expected time
Han '02	O(n log log n)	
Han & Thorup '02	$O(n (\log \log n)^{1/2})$	Expected time

Which would you rather sort?

- 1, 3, 4, 6, 5, 7, 8, 14, 15, 13, 16, 19, 18, 20
- 8, 3, 20, 1, 14, 18, 4, 7, 15, 7, 6, 19, 5, 16

Adaptive sorting

- Take advantage of existing **order** to sort faster.
- Measure of order:
  - Inversions: $x_i > x_j$, $i < j$ is one inversion.
  - Inv=Number of inversions in the input.
  - $0 \leq Inv \leq n(n-1)/2$
  - Natural & classical way to measure order.
  - Other measures exists, e.g. Runs, Osc,...

Comparison based adaptive sorting

- Finger search trees: O(n log(Inv/n)+n), Guibas et al. '77.
- Linear time for O(n) inversions, and the classical O(n log n) bound in the worst case.
- n log(Inv/n)+O(n) comparisons.
  - Optimal result.
  - Elmasry & Fredman, '03.

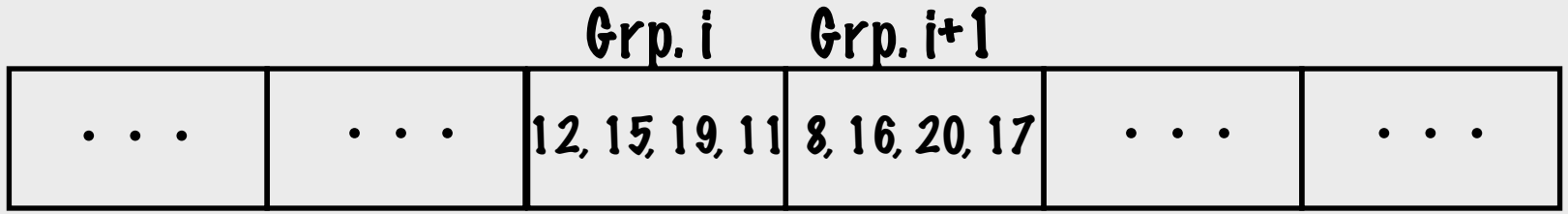
New result

- Sorting n integers, with Inv inversions, on a word RAM in
  - $O(n (\log \log(Inv/n))^{1/2})$ expected time.
  - $O(n \log \log(Inv/n))$ deterministic time.
  - $O(n)$ expected time if $Inv/n < 2^{(\log n)^{1-\epsilon}}$, where ϵ is any positive constant.

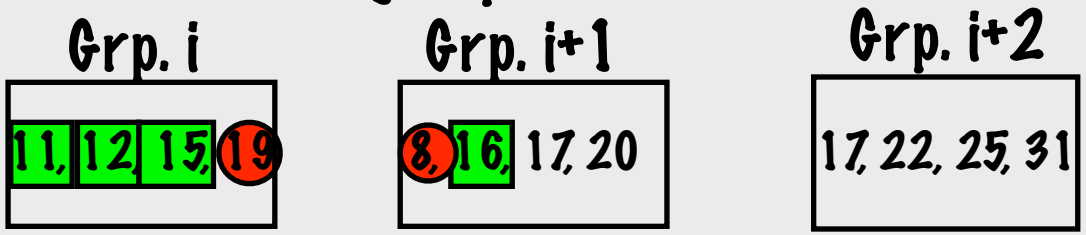
Main reduction

- Multisorting and inversion sorting can be reduced to each other in $O(n)$ time.
  - **Multisorting:** Sort n integers of w bits in groups of size $\leq q$
  - **Inversion sorting**: Sort n integers of w bits with $\mathrm{Inv} \leq qn$.
- Surprisingly, this equivalence has been overlooked for more than 25 years.

Using multisorting to sort



Multisort all groups



Merge the sorted groups:

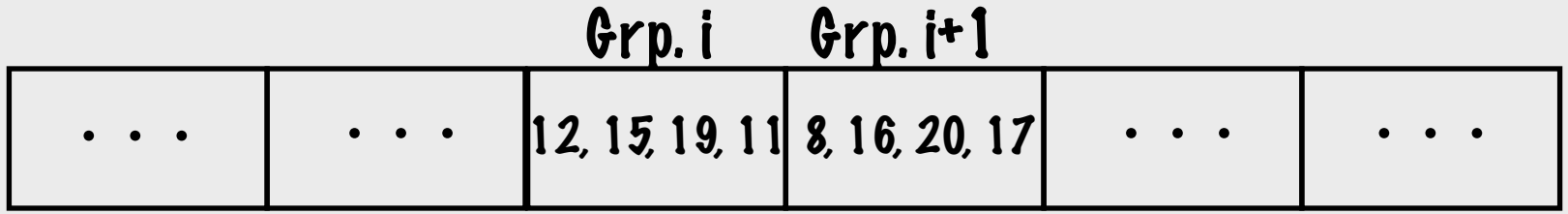
Sort the red integers and merge with the rest.

Analysis of the reduction

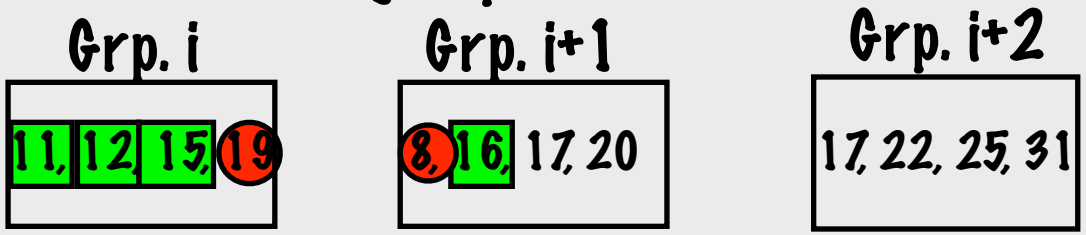
- **Key observation**: There are few **red** integers.

If integer from group i+1 is **red**, then it has inversions with half of group i.

Using multisorting to sort



Multisort all groups



Merge the sorted groups:

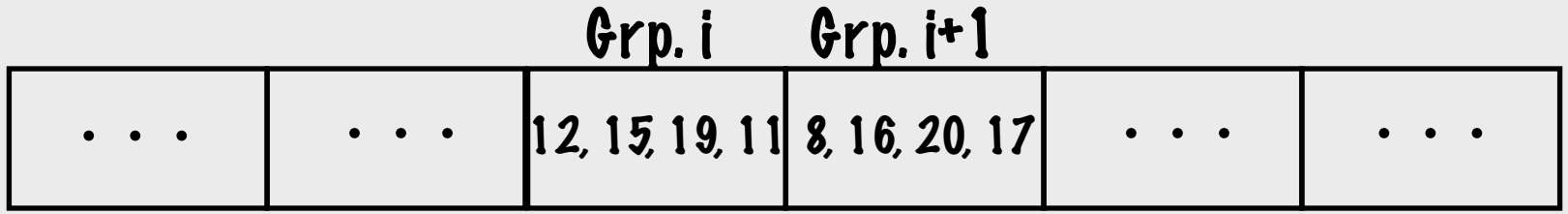
Sort the red integers and merge with the rest.

Analysis of the reduction

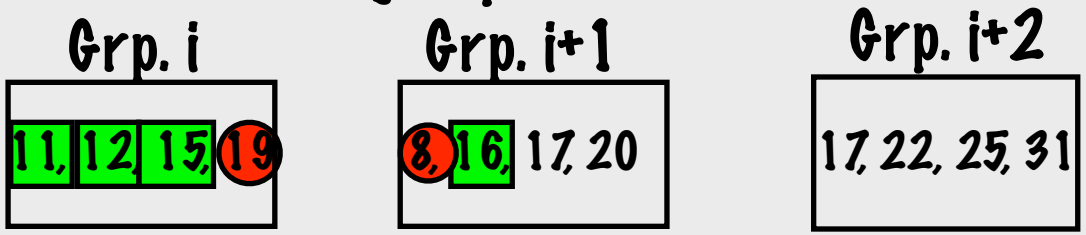
- **Key observation**: There are few red integers.
  - If integer from group i+1 is red, then it has inversions with half of group i.

If integer from group i is red, then it has inversions with at least half of group i+1.

Using multisorting to sort



Multisort all groups



Merge the sorted groups:

Sort the red integers and merge with the rest.

Analysis of the reduction

- Key observation: There are few red integers.
  - If integer from group i+1 is red, then it has inversions with half of group i.
  - If integer from group i is red, then it has inversions with at least half of group i+1.

Choosing group size q log n gives that there are at most 4n/log n red integers.

Time analysis

- Sorting at most 4n/log n red integers: O(n).
- Merging: O(n).
- Multisort n integers in groups of size q log n: O(n log log q) or O(n (log log q)$^{1/2}$) by sorting the groups one by one.
- Multisort in O(n) time when q<$2^{(\log n)^{1-\epsilon}}$ by new algorithm. Uses the fact that several groups are sorted at the same time. Ideas from signature sorting, Andersson et al.

Last slide

- Algorithm assumes q is known. We can run it for increasing value to make it adaptive.
- It is a general reduction, so any improved sorting bound will improve the algorithm.
- The reduction does not use multiplication. Best known bound for sorting not using multiplication is O(n log log n) expected time, Thorup '97.
- More general result, for Osc, in the paper.