Optimal Time-Space Trade-Offs for Non-Comparison-Based Sorting

Rasmus Pagh
BRICS, University of Aarhus, Denmark

joint work with

Jakob Pagter
CRYPTOMAThIC, Denmark

The problem

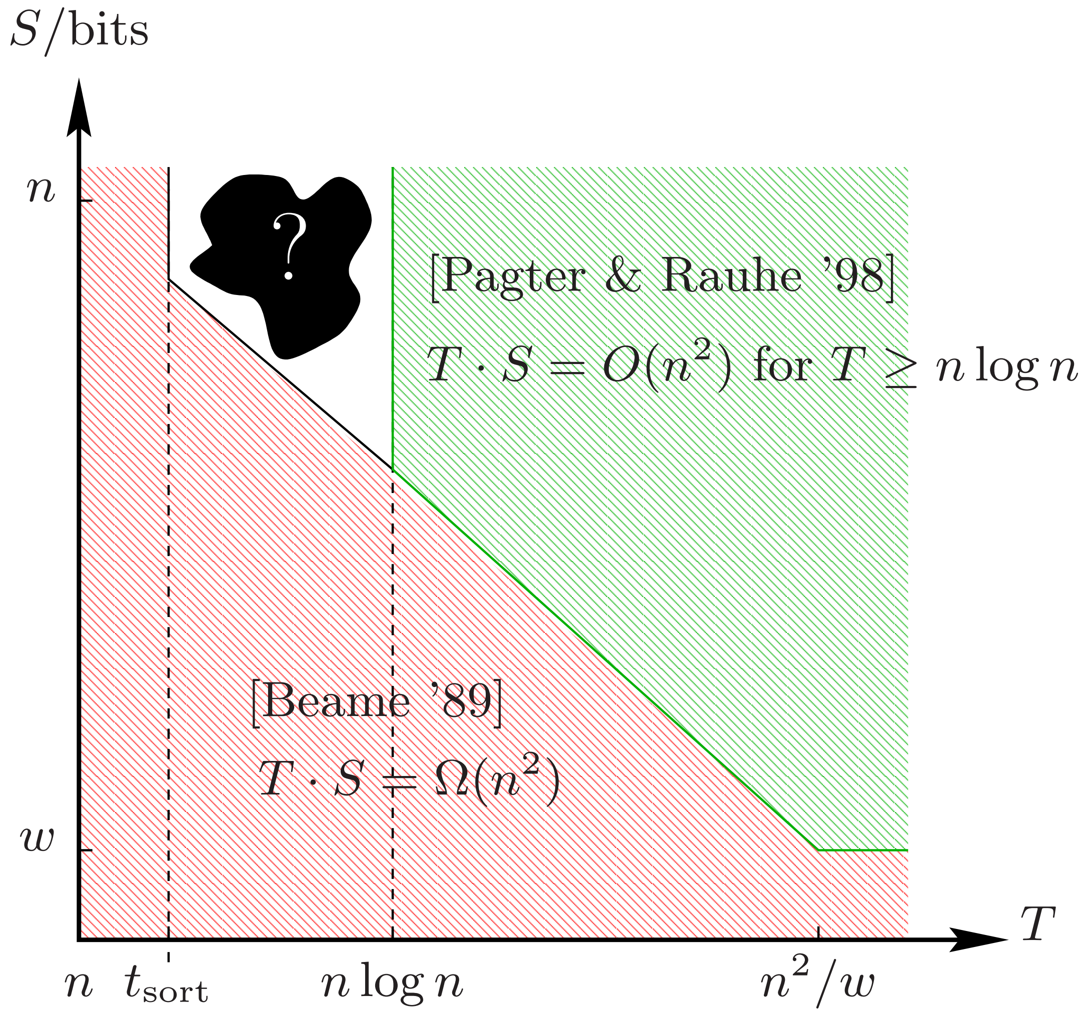
Sort a list of n integers of w bits.

- Unit cost RAM with word size w.
- Read-only input, write-only output.

Goal:

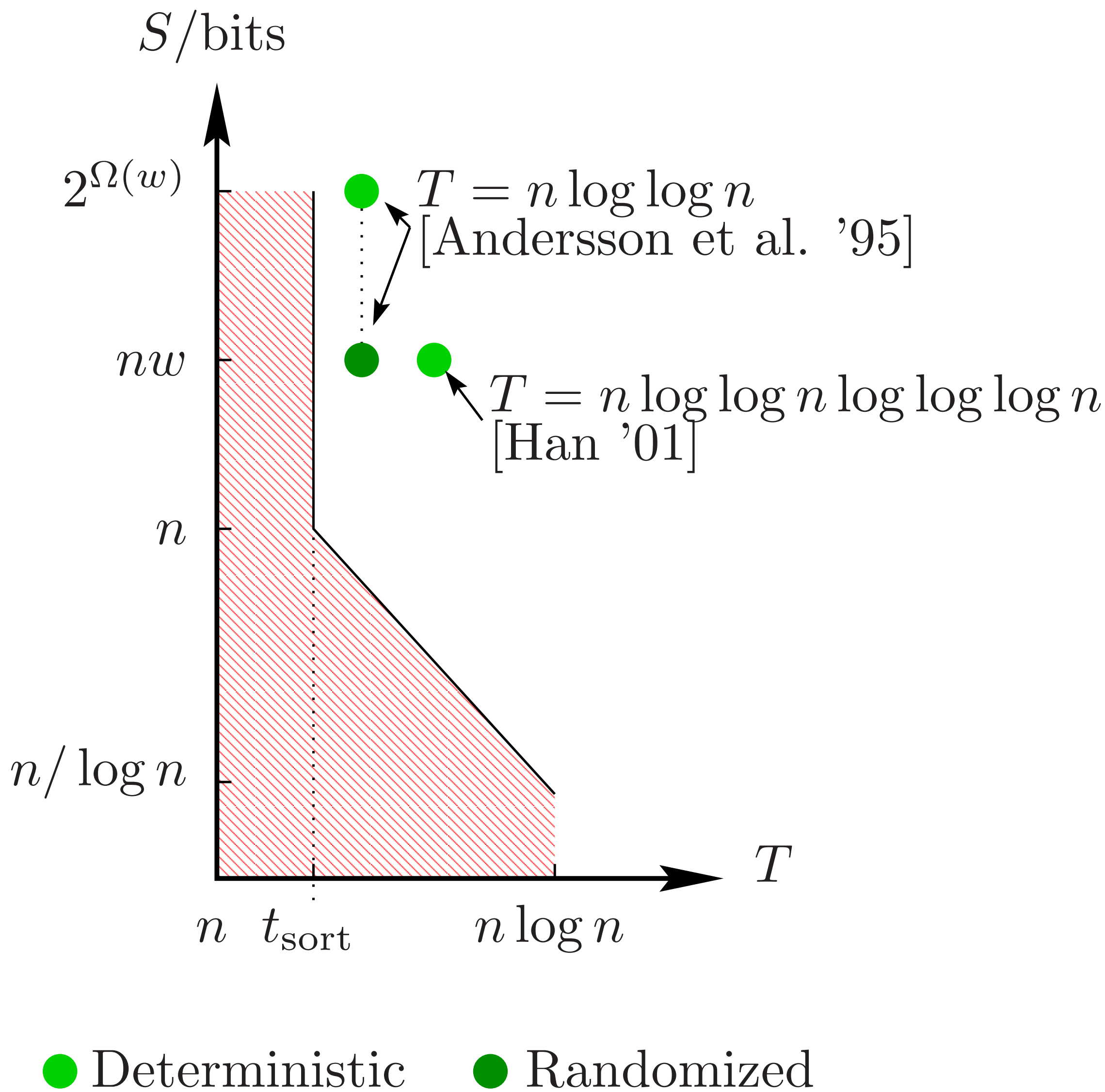
- Determine the time-space trade-off, i.e., the range of possible combinations of time and *work*space usage.

Known time-space trade-off results for sorting



t_{sort}: The time complexity of sorting

Best results on sorting in time $o(n \log n)$



Our results (vanilla versions)

THEOREM.
Sorting in time $T(n) \geq n \log^* n \Rightarrow$
Sorting in time $T(n)$ *w.h.p. and space* $S = O(\frac{n^2}{T(n)} + n^\epsilon w)$.

The proof of the theorem uses a reduction of [Thorup '96]:

Sorting in time $n\,t(n)$ and space $s(n) \Rightarrow$ Monotone priority queue with time $t(n)$/op and space $s(n) + O(nw)$ bits.

MAIN LEMMA.
A monotone priority queue using time $t(n) \geq \log^* n$ *and space* $n^{O(1)} w$
$\Rightarrow$ *Sorting in time* $O(n\,t(n))$ *using space* $S = O(\frac{n}{t(n)} + n^\epsilon w)$.
(The reduction does not introduce randomization.)

Consequences

Beame's sorting lower bound can be met:

- For $T(n) \geq n \log\log n$, w.h.p., by a randomized algorithm.
- For $T(n) \geq n\ \log\log n\ \log\log\log n$ deterministically.

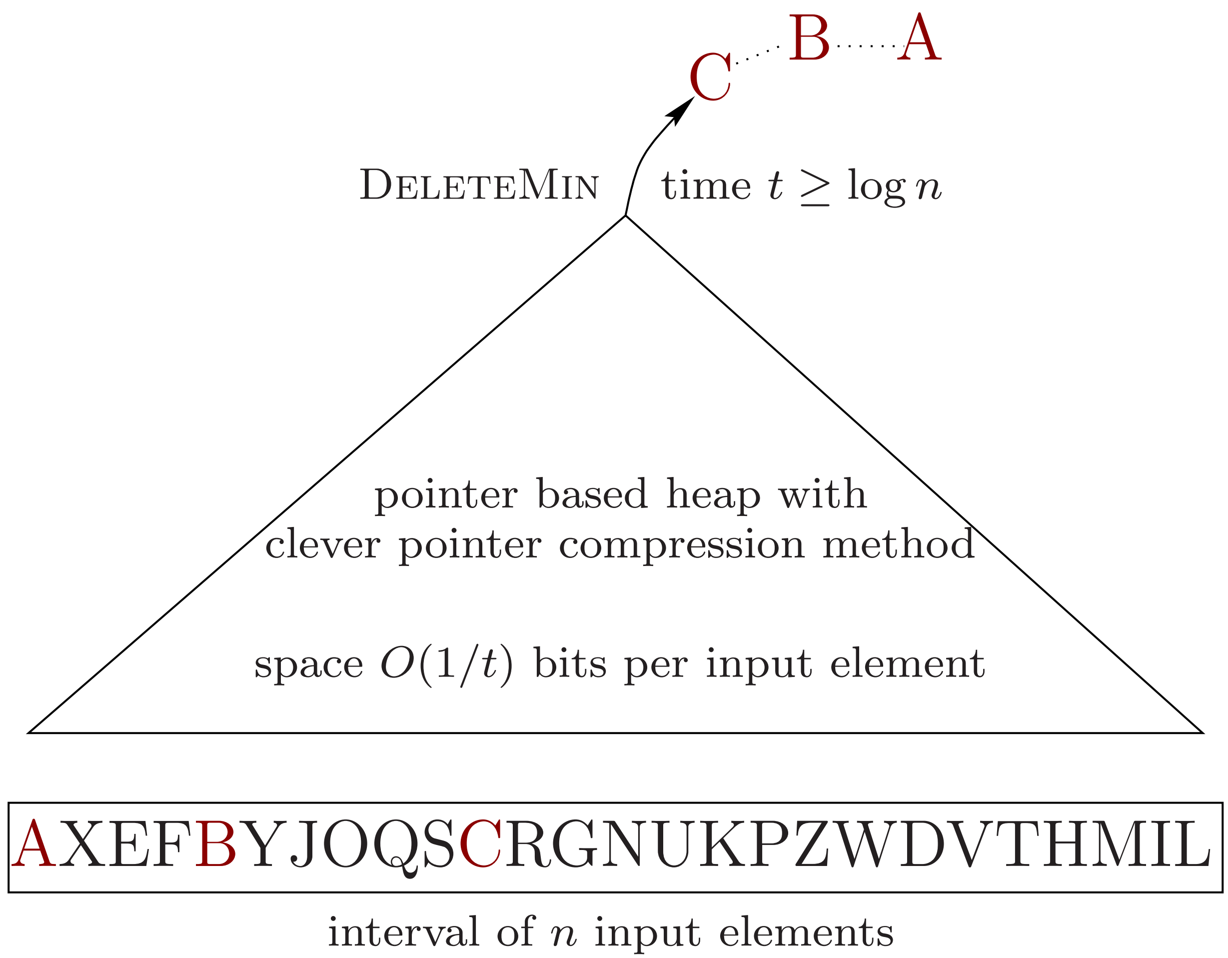
Technical content of this talk

The rest of the talk outlines our proof of the main lemma in the special case where:

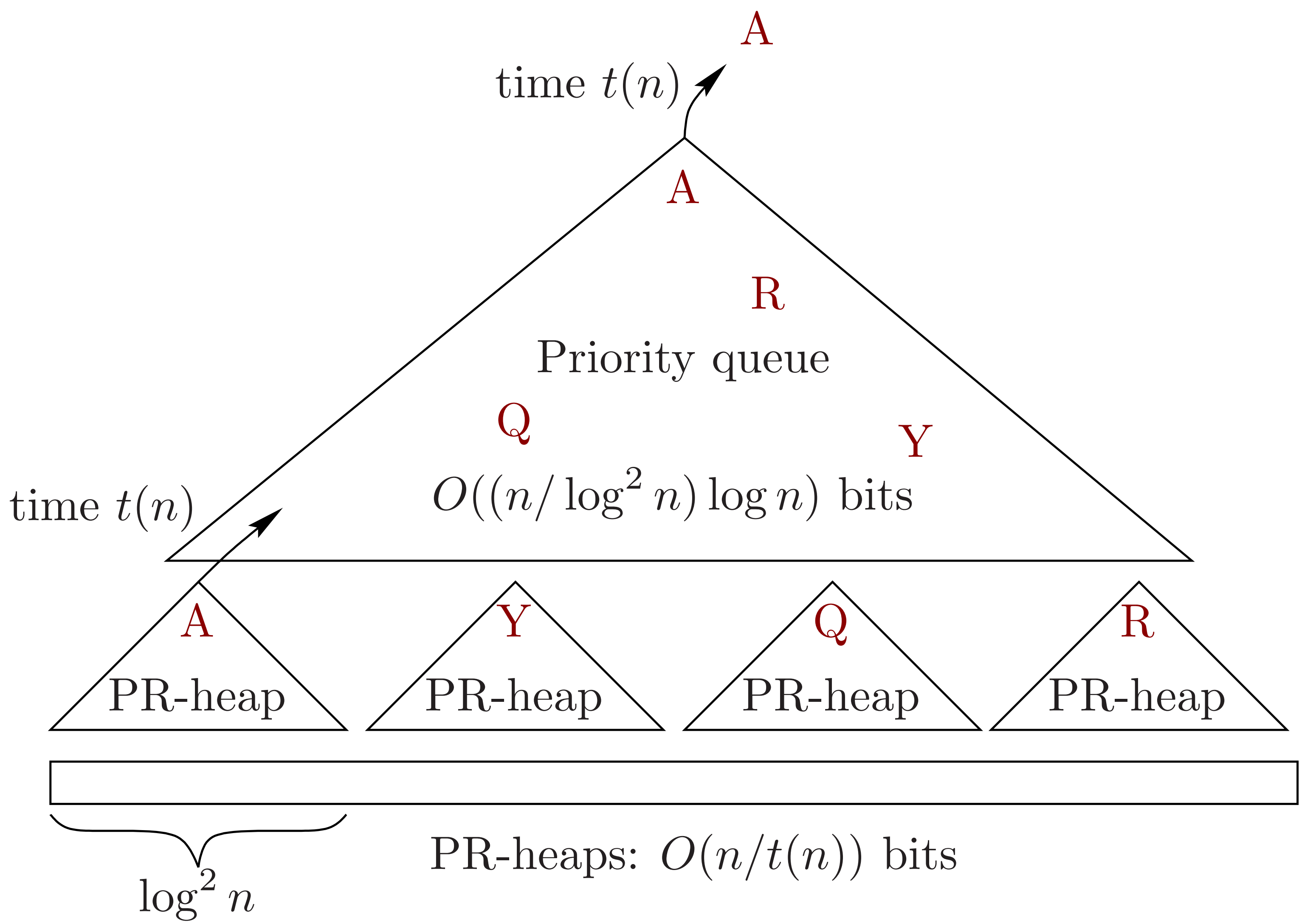
- the priority queue uses $O(n \log n)$ bits of space, and
- $t(n) \geq \log \log \log n$.

The Pagter-Rauhe heap

We will use the comparison-based Pagter-Rauhe heap which reports the elements from an interval, in sorted order, using optimal space.



The case $t(n) \geq \log\log n$



Data structure summary

The case $t(n) \geq \log \log n$:

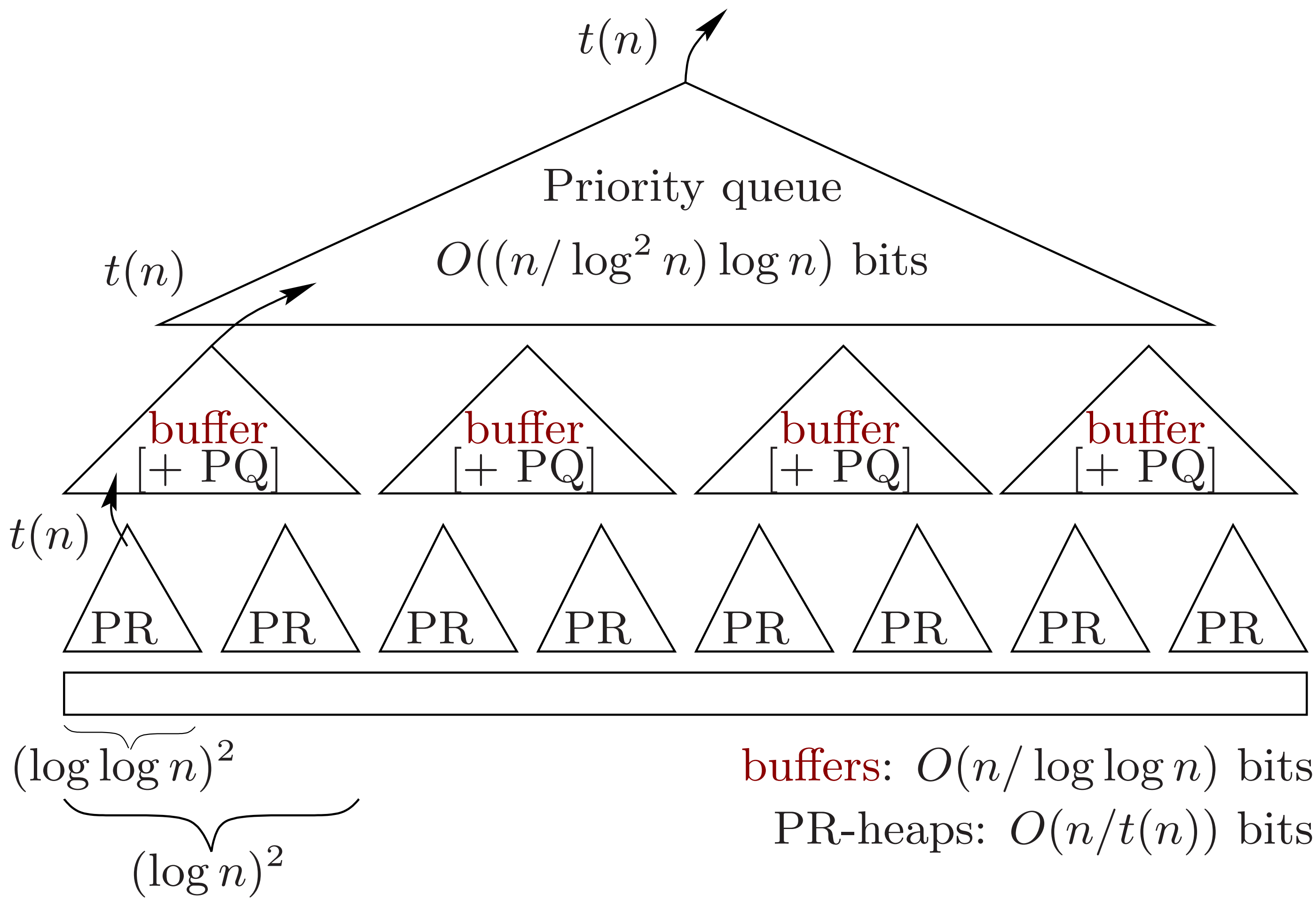
- A PR-heap reports elements from each "small" interval.
- A fast priority queue on top always contains the smallest remaining element from each interval.

Data structure summary

The case $t(n) \geq \log\log\log n$:

- A PR-heap reports elements from each *very* small interval.
- For each small interval of PR-heaps we have a buffer with pointers to the smallest remaining elements.
- When the buffer runs out, it is filled using a *temporary* fast priority queue. This costs time $t(n)$ per element in the buffer.
- A fast priority queue on top always contains the smallest remaining element from each buffer.

The case $t(n) \geq \log\log\log n$



buffers: $O(n/\log\log n)$ bits

PR-heaps: $O(n/t(n))$ bits

The general case

Handling greater speeds:

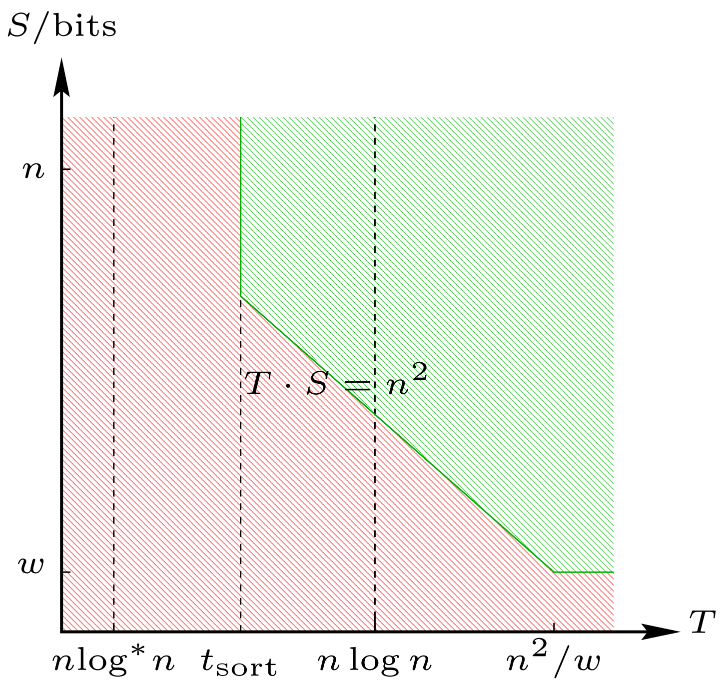
- Use more levels of buffers.
- Use constant-time priority queues (Q-heaps) for very small sets.

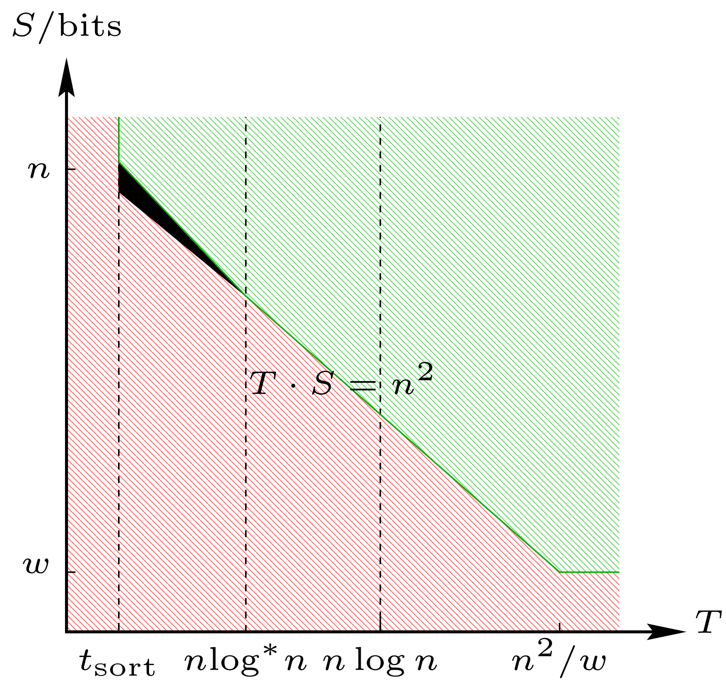
Improving dependence on word length:

- Replace the top priority queue by $O(1)$ levels of buffers of size n^ϵ.

Conclusion

The time-space trade-off of randomized sorting is almost completely understood. Depending on the time complexity of sorting, we have one of these situations:





Open problems

Open problems

- Give deterministic upper bounds.
- Close the small gap for time $o(n \lg^* n)$
  – or show a lower bound on the time for sorting!
- Give better upper bounds in special cases, e.g., $w = O(\log n)$.