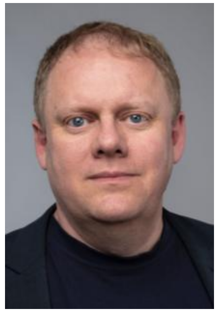
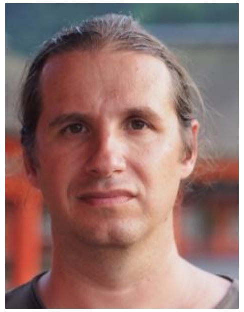
External-Memory Priority Queues with Optimal Insertions



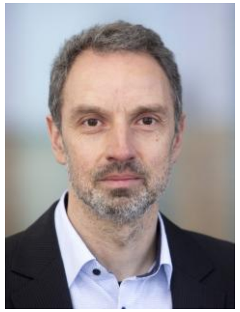
Gerth Stølting Brodal
Aarhus University

Michael Goodrich
University of California, Irvine



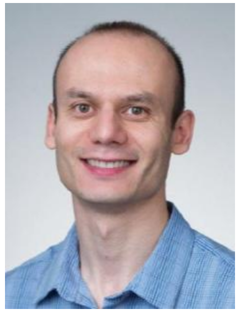
John Iacono
Université libre de Bruxelles

Jared Lo
University of Hawai'i at Mānoa



Ulrich Meyer
Goethe University Frankfurt am Main

Victor Pagan
University of Hawai'i at Mānoa



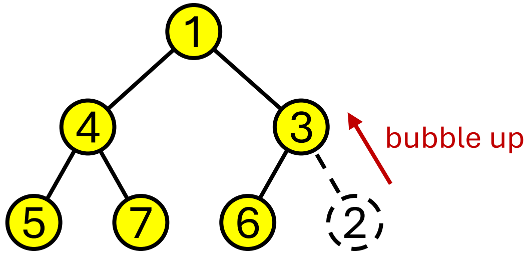
Nodari Sitchinava
University of Hawai'i at Mānoa

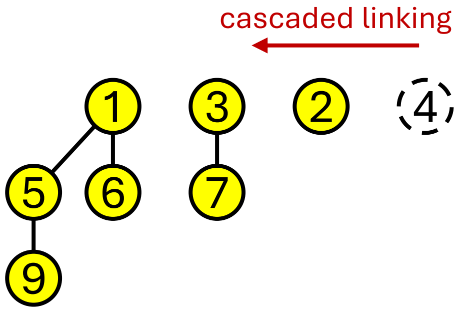
Rolf Svenning
Aarhus University

33rd Annual European Symposium on Algorithms (ESA), September 15, 2025, Warsaw, Poland

Priority Queues

	INSERT(e)	DELETEMIN()
Binary heap (Williams 1964)	O(log N)	O(log N)
Binomial queue (Vuillemin 1978)	$O_A(1)$	O(log N)



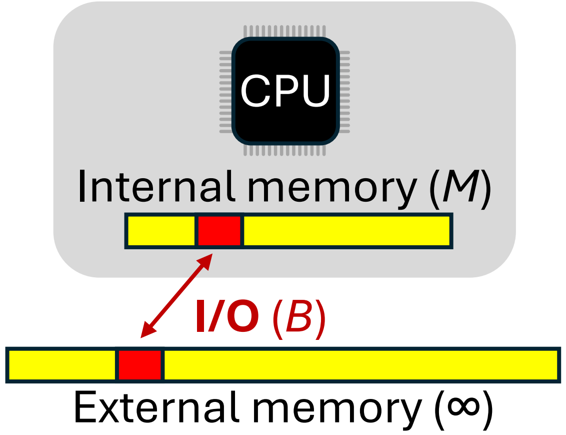


Can we achieve a similar result in external memory ?

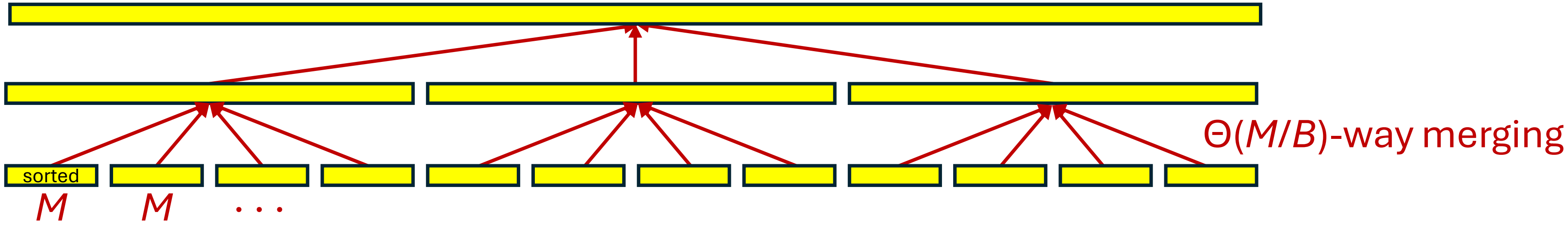
Sorting = N × INSERT + N × DELETEMIN ⇒ INSERT **or** DELETEMIN $\Omega(\log N)$ comparisons

# INSERT ≥ # DELETEMIN

External Memory

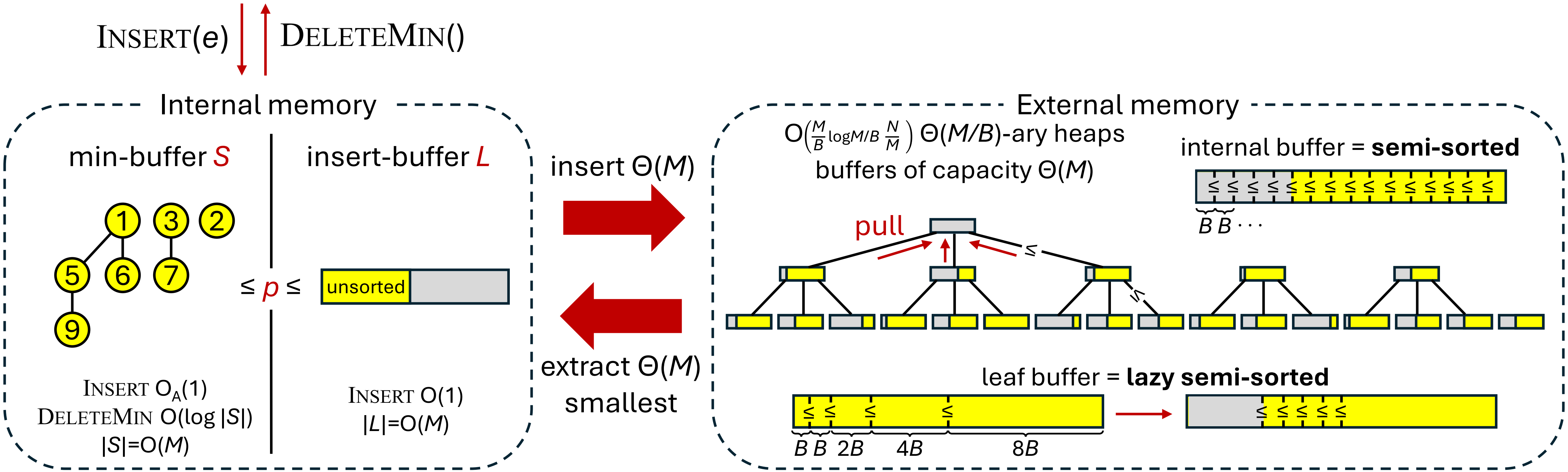


- External-memory model (Aggarwal, Vitter 1988)
- **Sorting** $O\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$ I/Os and $O(N\cdot\log N)$ comparisons



- **Priority queue** $O_A\left(\frac{1}{B}\log_{M/B}\frac{N}{B}\right)$ I/Os and $O_A(\log N)$ comparisons $\left\{\begin{array}{l}\textsc{Insert}\\ \textsc{DeleteMin}\end{array}\right.$
  - **Buffer tree** (Arge 1995)
  - **Θ(*M*/*B*)-ary heap** (Kumar, Schwabe 1996; Fadel, Jakobsen, Katajainen, Teuhola 1997)
  - **Merging sorted lists** (Brengel, Crauser, Ferragina, Meyer 1999; Brodal, Katajainen 1998)
- **This talk** $\textsc{Insert}$ $O_A\left(\frac{1}{B}\right)$ I/Os and $O_A(1)$ comparisons

New External-Memory Priority Queue



INSERT(e) DELETEMIN()

Internal memory

min-buffer S

$\leq p \leq$

insert-buffer L

unsorted

INSERT $O_A(1)$
DELETEMIN $O(\log |S|)$
$|S| = O(M)$

INSERT $O(1)$
$|L| = O(M)$

insert $\Theta(M)$

extract $\Theta(M)$ smallest

External memory

$O\left(\frac{M}{B}\log_{M/B}\frac{N}{M}\right)$ $\Theta(M/B)$-ary heaps
buffers of capacity $\Theta(M)$

pull

internal buffer = **semi-sorted**

$\underbrace{}_{B}\underbrace{}_{B}\cdots$

leaf buffer = **lazy semi-sorted**

B B $2B$ $4B$ $8B$

Lemma Total $O\left(\frac{N/M}{M/B} + \frac{\#\,\text{deletions}}{M}\cdot\log_{M/B}\frac{N}{B}\right)$ pulls

Lemma Pulling M elements from M/B semi-sorted lists : $O(M/B)$ I/Os and $O\left(M\cdot\log_2\frac{M}{B}\right)$ comparisons

Theorem Total $O\left(\frac{N}{B} + \frac{\#\,\text{deletions}}{B}\cdot\log_{M/B}\frac{N}{B}\right)$ I/Os and $O(N + \#\,\text{deletions}\cdot\log_2 N)$ comparisons

Summary

- **New optimal external memory priority queue**

	I/Os	Comparisons
INSERT	$\mathrm{O_A}\left(\frac{1}{B}\right)$	$\mathrm{O_A}(1)$
DELETEMIN	$\mathrm{O_A}\left(\frac{1}{B}\log_{M/B}\frac{N}{B}\right)$	$\mathrm{O_A}(\log N)$

- **Open problems**
  - Our data structure is inherently amortized – can it be made **worst-case** ?
  - Structure inherently cache-aware – is there a **cache-oblivious** data structure with matching I/O and comparison bounds ?