Fast Meldable Priority Queues

Gerth S. Brodal
gerth@daimi.aau.dk

BRICS
Computer Science Department
University of Aarhus
Aarhus, Denmark

August 1995

Priority Queue Operations

- MakeQueue
- FindMin(Q)
- Insert(Q, e)
- Meld(Q_1, Q_2)
- DeleteMin(Q)
- Delete(Q, e)*

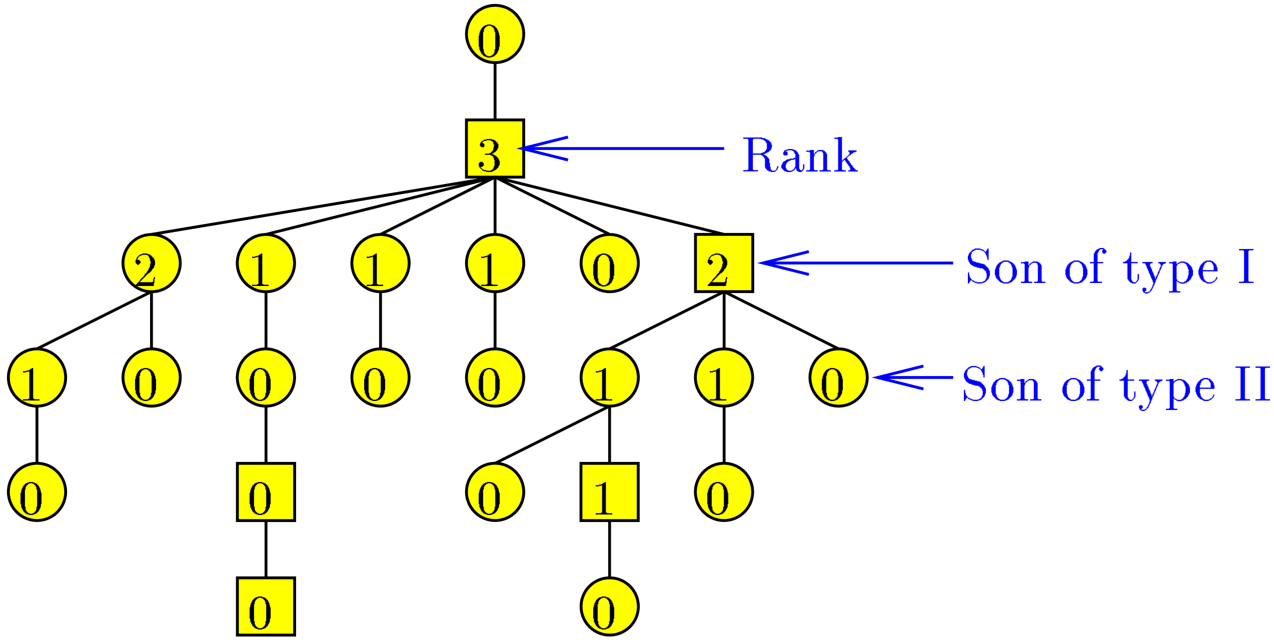
*Assumes that it is known where the element e is stored in Q.

Known And New Time Bounds

	[W64] Heaps	[SS85] Merging Heaps	[DGST88] Relaxed Heaps	[V78] Binomial Queues*	[B95] New Result
FindMin	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
Insert	$O(\log n)$	$O(\log n)$	$O(1)$	$O(1)$	$O(1)$
Meld	$O(n)$	$O(\log^2 n)$	$O(\log n)$	$O(1)$	$O(1)$
Delete(Min)	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(\log n)$

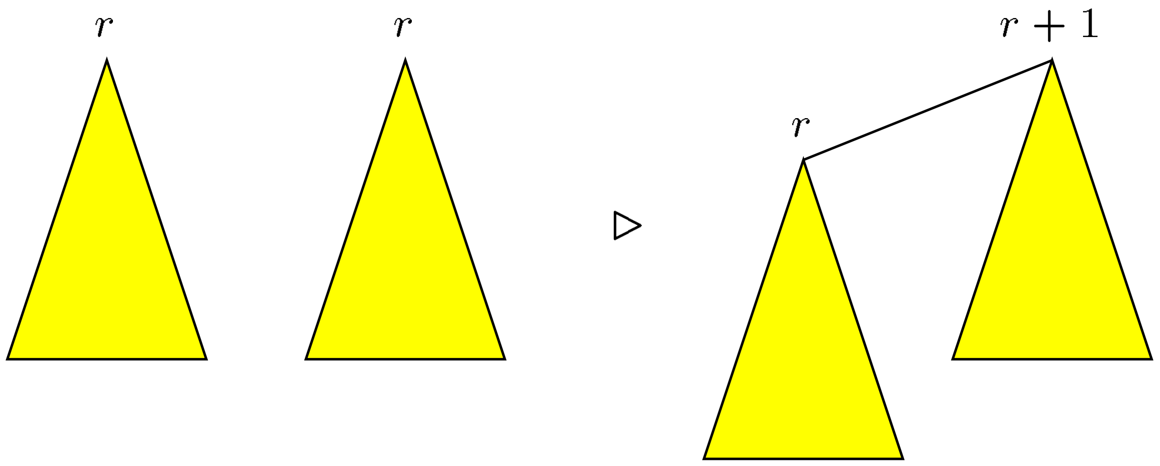
*Amortised bounds

The Data Structure



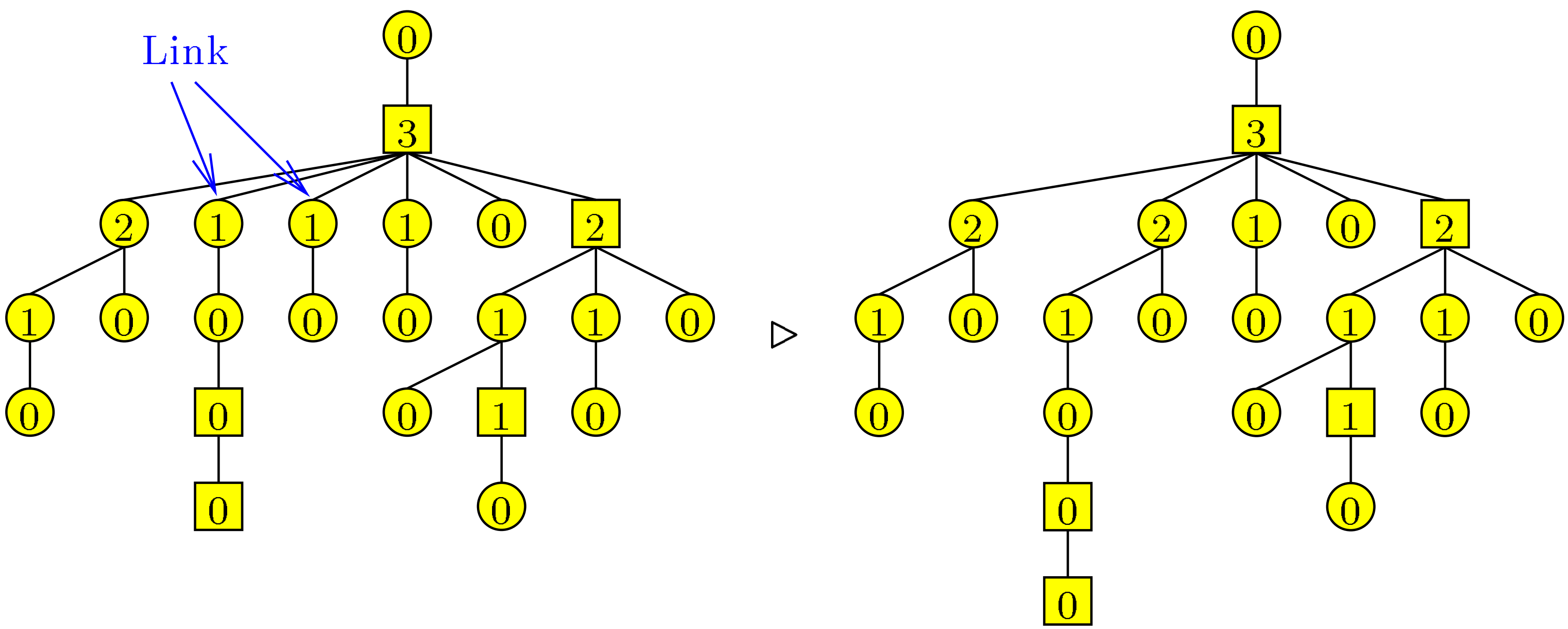
- A priority queue is represented by a heap ordered tree where each node contains an element and has a rank assigned.
- A node of rank r has at most one son of type I and one, two or three sons of type II of rank i for $i = 0, \ldots, r-1$.

Linking

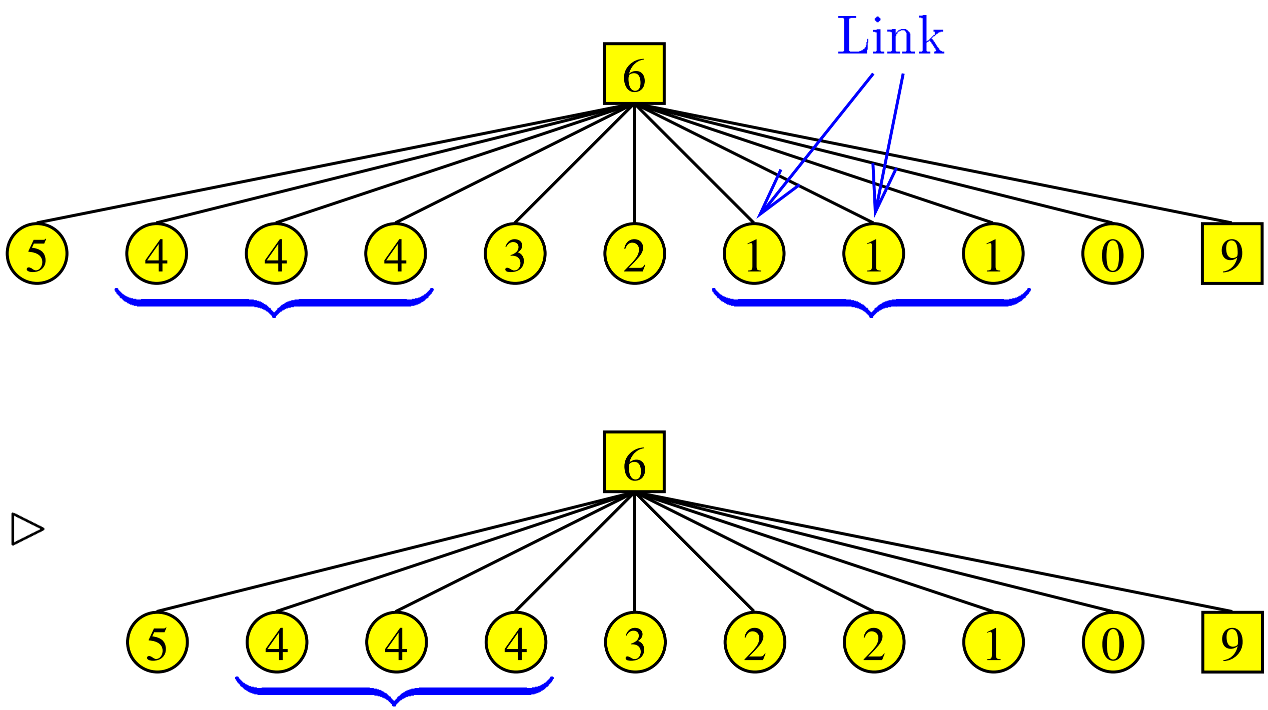


Two trees of equal rank r can be linked to one tree of rank $r+1$ in worst case time O(1).

Linking
Example

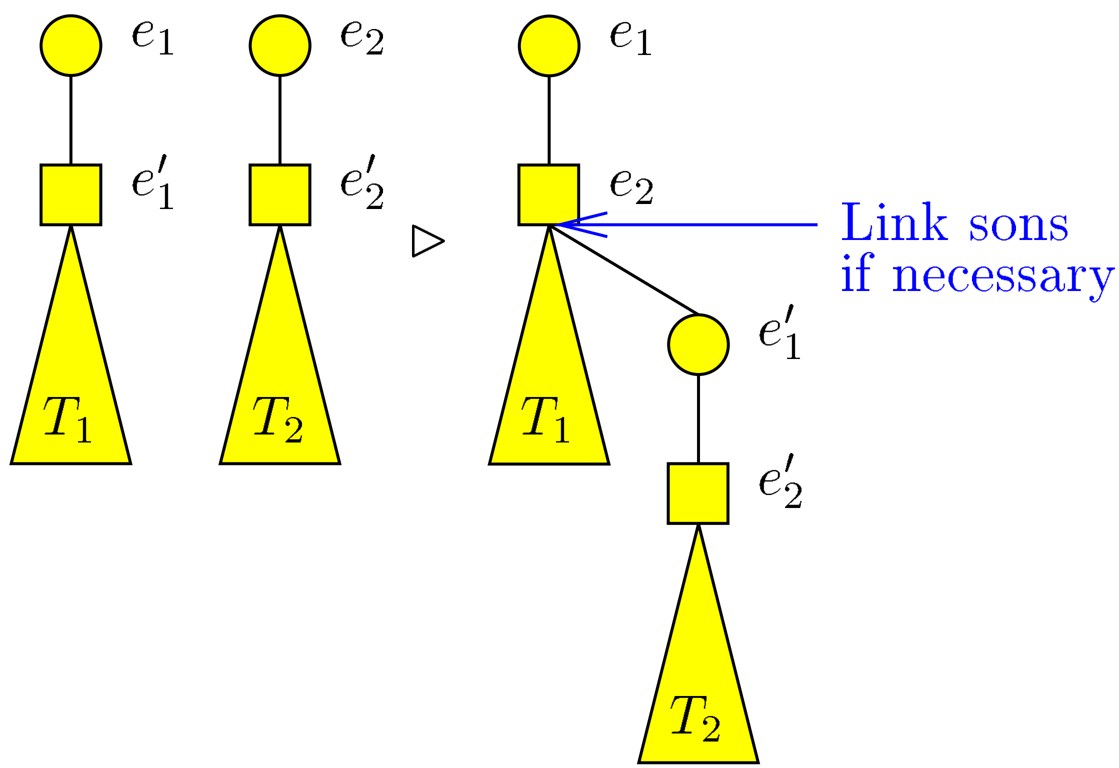


The Invariant



Between two ranks where three sons of type II have equal rank there is a rank of which there only is one son of type II.

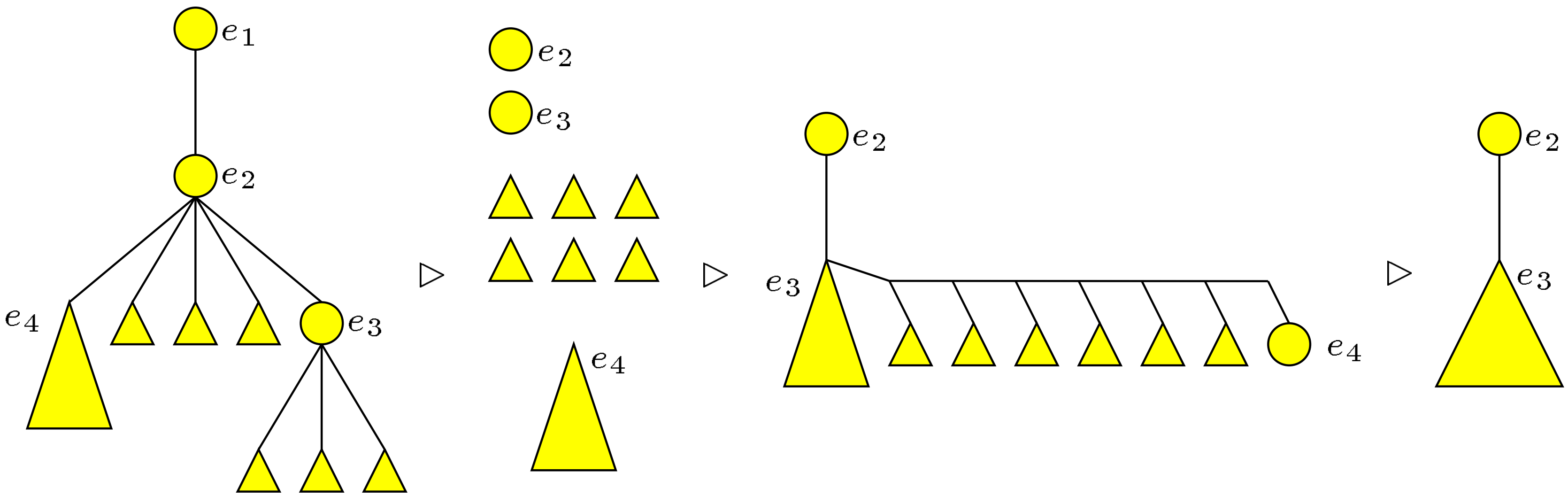
Implementation of MELD



How to perform MELD in worst case time O(1).

The case $e_1 \leq e_2 < e_1' \leq e_2'$.

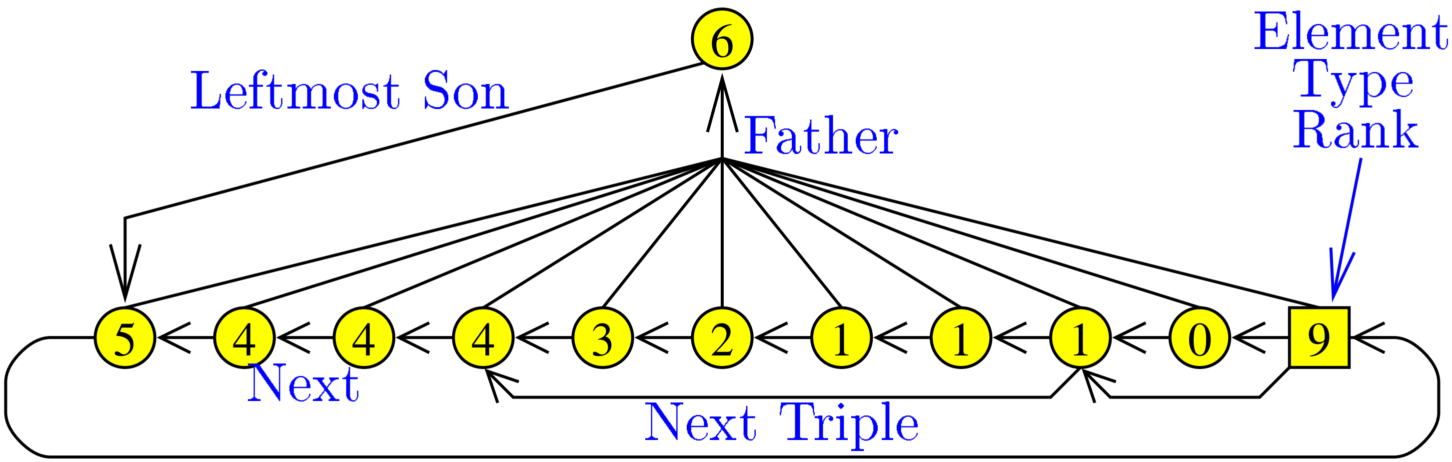
Implementation of DELETEMIN



How to perform DELETEMIN in worst case time $O(\log n)$.

e_1, e_2 and e_3 are the three smallest elements.

Implementation Details



Each node is stored as a record having seven fields.

Optimality

Theorem:

If MELD can be performed in worst case time $o(n)$ then DELETEMIN *cannot* be performed in worst case time $o(\log n)$.

Proof:

For $n = 2^k$ we otherwise by contradiction get

$$\begin{aligned} & \mathrm{T}_{\mathrm{SORTING}}(n) \\ & \quad = n\mathrm{T}_{\mathrm{MAKEQUEUE}} + \sum_{i=0}^{k-1} 2^{k-1-i}\mathrm{T}_{\mathrm{MELD}}(2^i) + \sum_{i=1}^{n} \mathrm{T}_{\mathrm{DELETEMIN}}(i) \\ & \quad = o(n \log n). \end{aligned}$$

□

The Result

We have presented priority queues which

- support MakeQueue, FindMin, Insert and Meld in worst case time O(1),
- support Delete(Min) in worst case time $O(\log n)$,
- require linear space and
- can be implemented on a pointer machine.

Double–Ended Priority Queues

	[ASSS86] Min-Max Heaps	[DW93] Relaxed Min-Max Heaps*	[B93] New Result
Insert	O($\log n$)	O(1)	O(1)
FindMin/FindMax	O(1)	O(1)	O(1)
DeleteMin/DeleteMax	O($\log n$)	O($\log n$)	O($\log n$)
Meld	—	O(1)	O(1)
Delete	—	O($\log n$)	O($\log n$)
DecreaseKey/IncreaseKey	—	O($\log n$)	O($\log n$)

*Amortised bounds

Priority Queues with DecreaseKey

	[W64] Heaps	[DGST88] Relaxed Heaps	[FT84] Fibonacci Queues*	[B95] New Result	[B95b] Recent Result
FindMin	O(1)	O(1)	O(1)	O(1)	O(1)
Insert	O(log n)	O(1)	O(1)	O(1)	O(1)
Meld	O(n)	O(log n)	O(1)	O(1)	O(1)
Delete(Min)	O(log n)	O(log n)	O(log n)	O(log n)	O(log n)
DecreaseKey	O(log n)	O(1)	O(1)	O(log n)	O(1)

*Amortised bounds