Worst-Case Efficient Priority Queues

Gerth Stølting Brodal

gerth@brics.dk

BRICS
Computer Science Department
University of Aarhus
Aarhus, Denmark

January 1996

Priority Queue Operations

- $\textsc{MakeQueue}$
- $\textsc{FindMin}(Q)$
- $\textsc{Insert}(Q, e)$
- $\textsc{Meld}(Q_1, Q_2)$
- $\textsc{DeleteMin}(Q)$
- $\textsc{Delete}(Q, e)^*$
- $\textsc{DecreaseKey}(Q, e, e')^*, \quad e' \leq e$

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

Frame Work

- Elements can only be compared.
- RAM model.
- Extendible arrays

  — achievable by array doubling and incremental copying.
- Goal: Good worst-case performance.

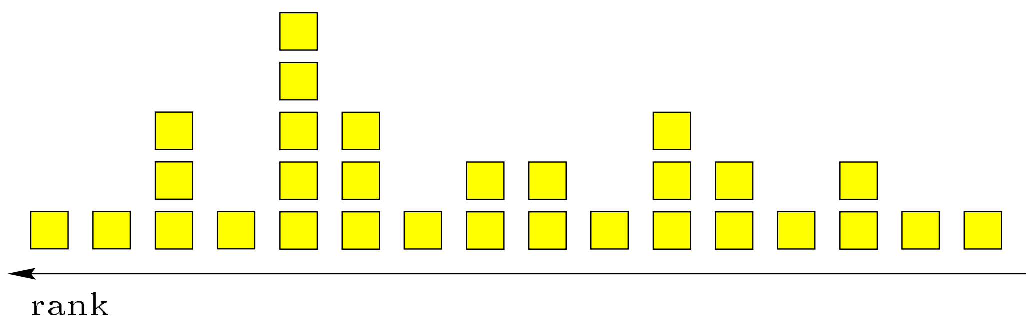
Known and New Time Bounds

	Williams	Driscoll Gabow Shrairman Tarjan	Brodal	Fredman Tarjan	Brodal
	1964	1988	1995	1984	1996
	Heaps	Relaxed Heaps		Fibonacci Queues*	
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	$O(\log n)$	$O(1)$	$O(\log n)$	$O(1)$	$O(1)$

*Amortized bounds

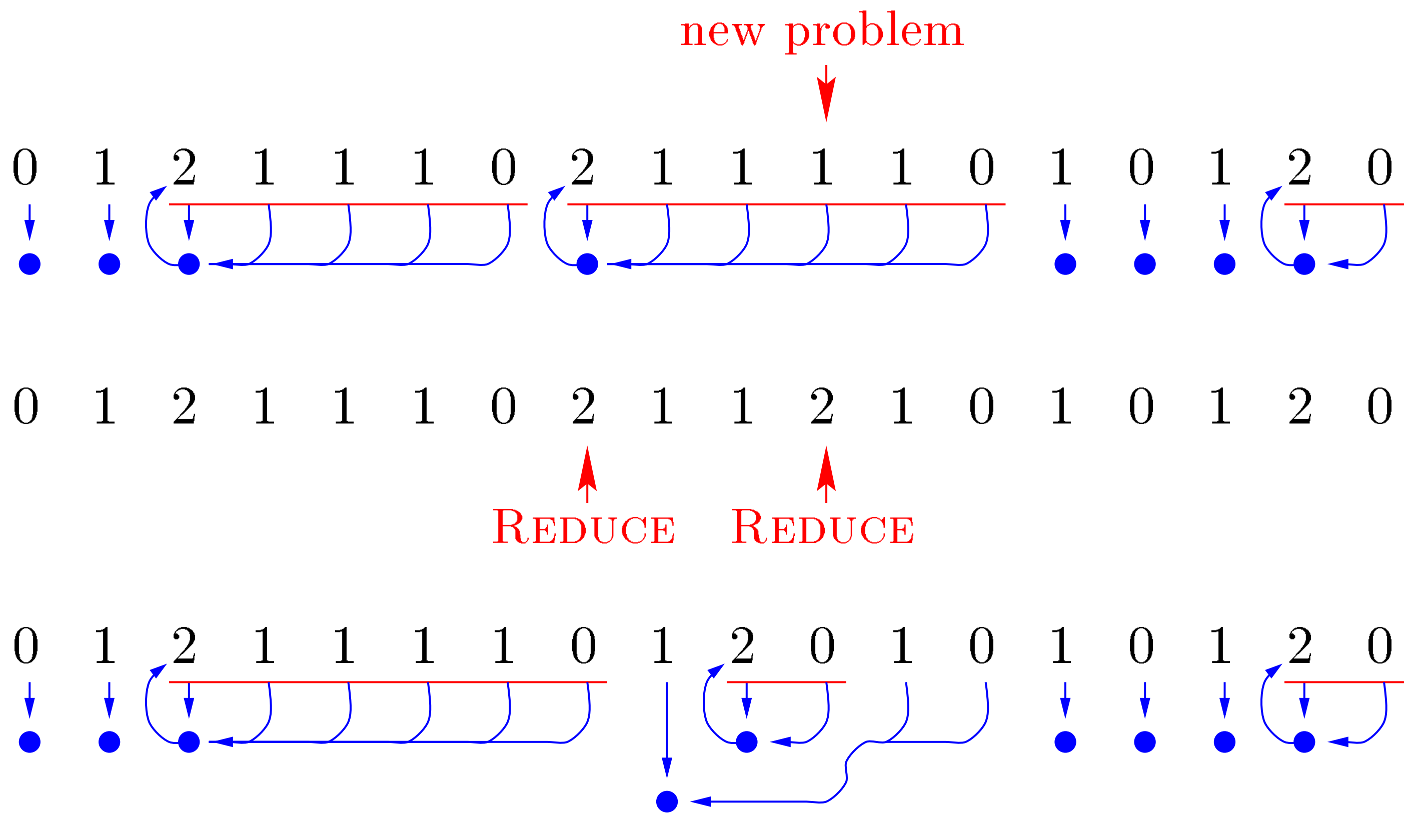
Guides

- We have a set of integer ranked problems.
- We can replace two rank r problems by a rank $r+1$ problem, Reduce(r).
- When a new problem is introduced, we can perform O(1) Reduce operations.
- Goal: Bound the number of problems of each rank by a constant.



Guides

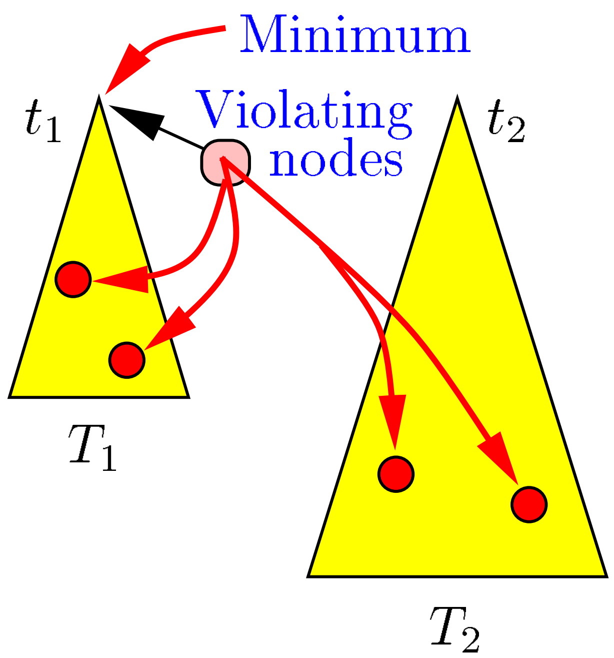
- By performing two Reduce operations for each new problem, the number of problems of each rank can be bounded by two.



The Priority Queue Data Structure

- A priority queue is represented by two trees T_1 and T_2.
- Each node holds one element.
- The minimum element is always at the root of T_1.
- A node larger than its parent is a good node. Otherwise it is a violating node — because it violates heap order.

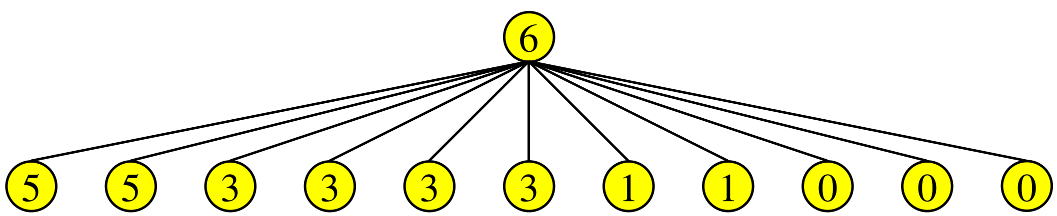
The Basic Idea



- A set of violating nodes is associated to each node.
- The violating nodes associated to a node are larger than the node.
- A violating node belongs to exactly one set.

The Sons of a Node

- Each node has a rank.
- The rank of a node is less than the rank of its father.
- All nodes except for the roots have a brother of equal rank.
- Leaves have rank zero.
- A node of rank $r \geq 1$ has at least two sons of rank $r - 1$.
- At most O(1) brothers can have equal rank.



- The ranks and degrees are $O(\log n)$.
- Linking and unlinking can be done in constant time.

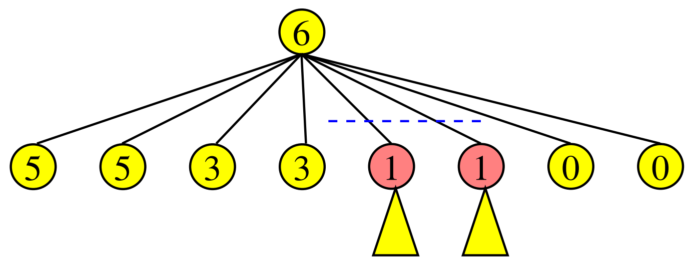
The Application of Guides

We use five guides to

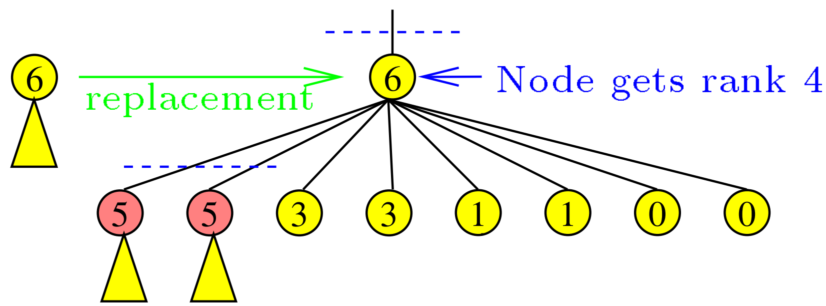
- restrict the number of sons at t_1,
- restrict the number of sons at t_2,
- guarantee the existence of sons at t_1,
- guarantee the existence of sons at t_2,
- restrict the number of violations at t_1.

How to Reduce The Number of Violating Nodes

- The two brothers are cut off.

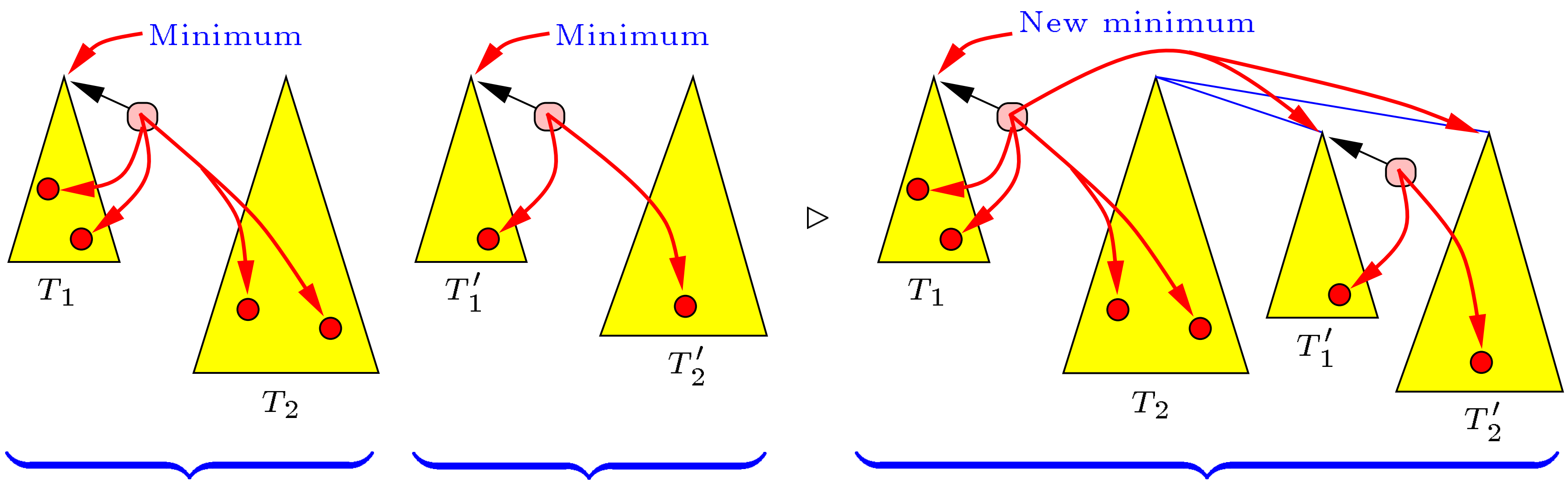


- The two brothers and the father are cut off and replaced by a son of t_1. The replacement becomes a new violating node.



The cut off nodes are made good sons of the root t_1.

The Basic Idea of MELD



At most two new violating nodes are created.

The Result

Priority queues exist that

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

Open Problems

- Simplify the data structure!
- Eliminate the requirement for arrays.