Algorithms and Data Structures

Gerth Stølting Brodal

MADALGO
CENTER FOR MASSIVE DATA ALGORITHMICS

Computer Science Day, Department of Computer Science, Aarhus University, May 25, 2012

People

Faculty

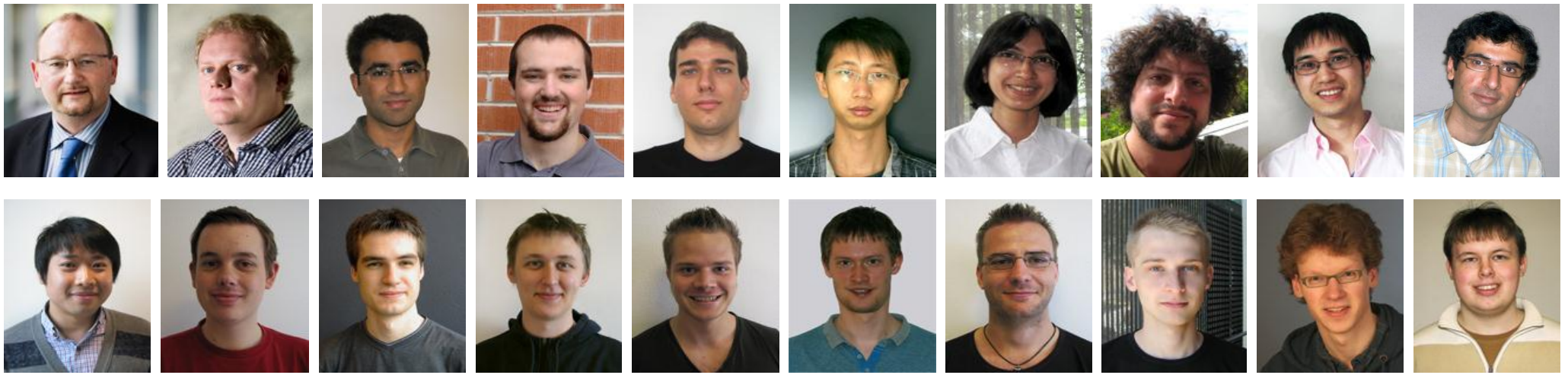
Lars Arge
Gerth Stølting Brodal
Peyman Afshani

PhD Students

Jungwoo Yang
Freek van Walderveen
Casper Kejlberg-Rasmussen
Jesper Erenskjold Moeslund
Jesper Asbjørn Sindahl Nielsen
Jakob Truelsen
Morten Revsbæk
Mark Greve
Kasper Green Larsen
Lasse Kosetski Deleuran

Researchers

Brody Sandel
Constantinos Tsirogiannis
Wei Yu
Cicimol Alexander
Elad Verbin
Qin Zhang
Hossein Jowhari



Strict Fibonacci Heaps

Gerth Stølting Brodal

Aarhus University

MADALGO
CENTER FOR MASSIVE DATA ALGORITHMICS

George Lagogiannis

Agricultural University of Athens

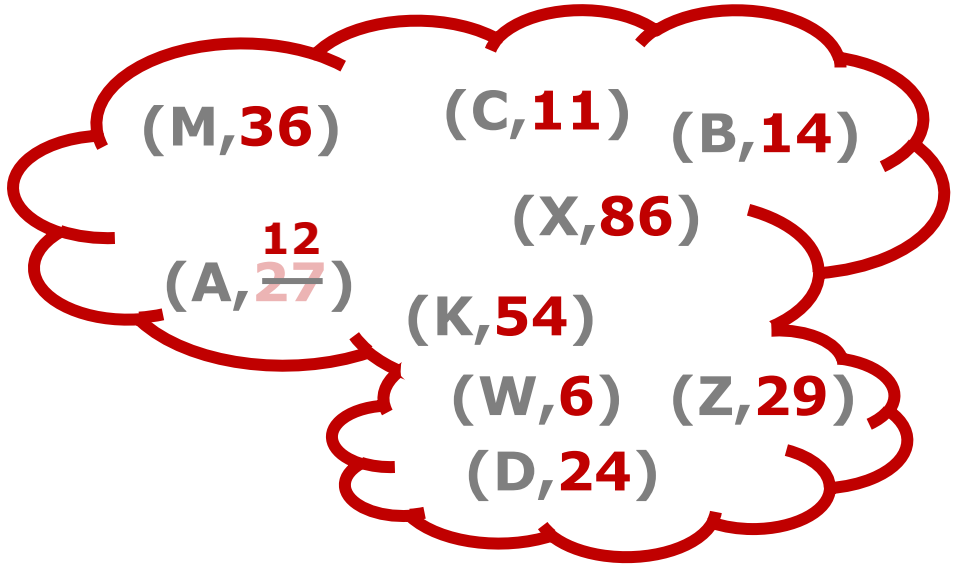
Robert Endre Tarjan

Princeton University & HP

44th Annual ACM Symposium on Theory of Computing, New Yorker Hotel, New York, May 22, 2012

The Problem — Priority Queues

- INSERT(value, key)
- FINDMIN
- DELETEMIN / DELETE(&value)
- MELD(Q_1,Q_2)
- DECREASEKEY(&value, Δ)

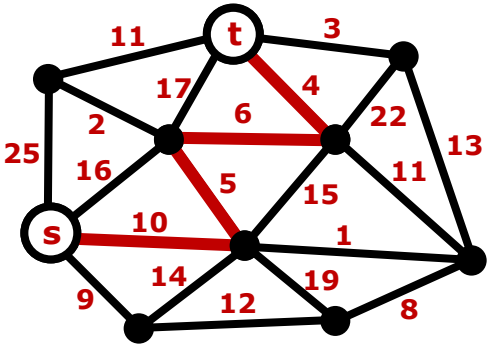


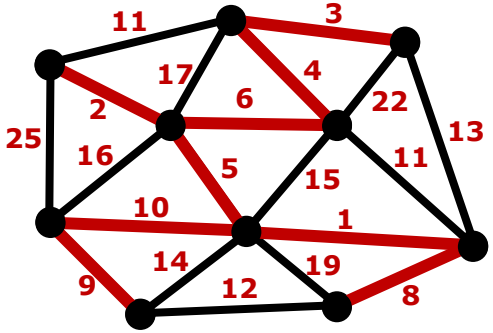
Applications

- **S**hortest **P**ath
  Dijkstra (1956)

- **M**inimum **S**panning **T**ree
  Borůvka (1926)
  Jarník (1930)

  (n node, m edges)





INSERT/DELETEMIN
$(m+n)\cdot\log n$

↓ Fredman, Tarjan 1984

+ DECREASEKEY

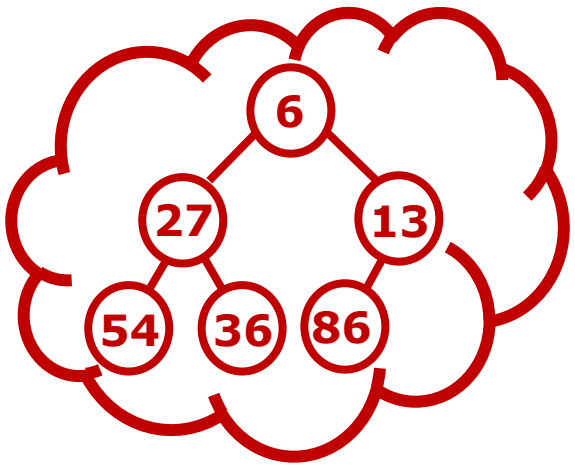
$m+n\cdot\log n$

MST only ↓ Fredman, Tarjan 1984

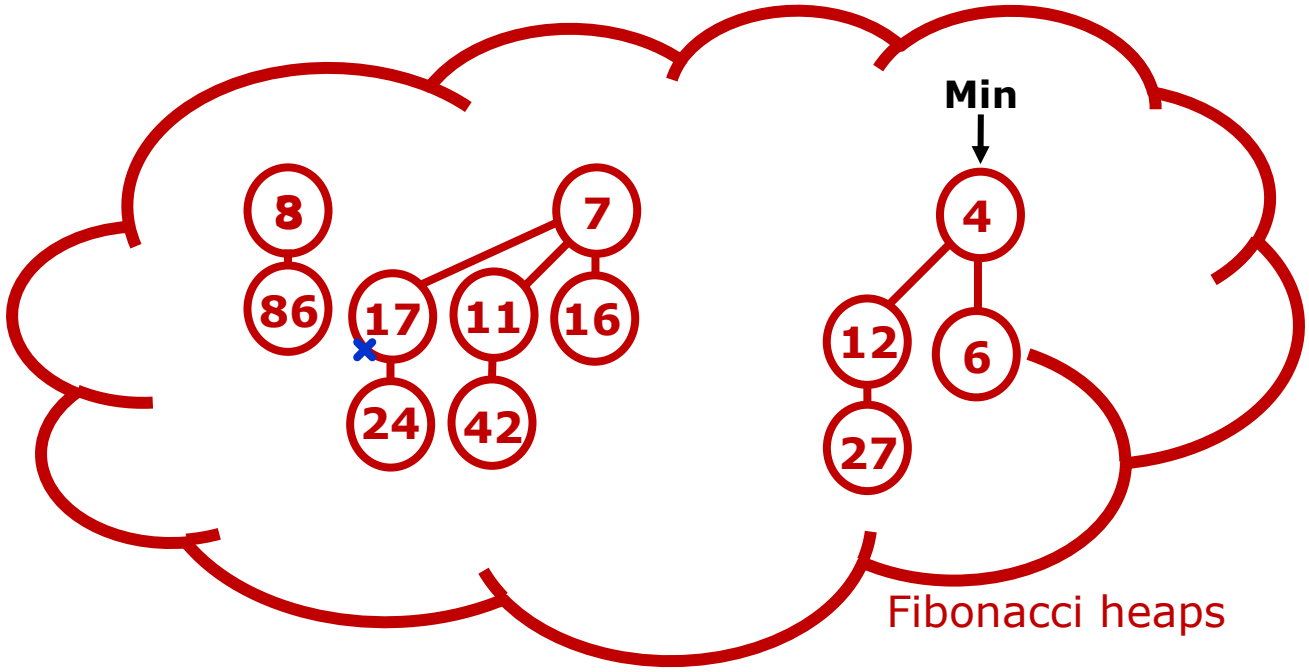
+ DECREASEKEY

$m\cdot\beta(m,n)$

History



Binary heaps



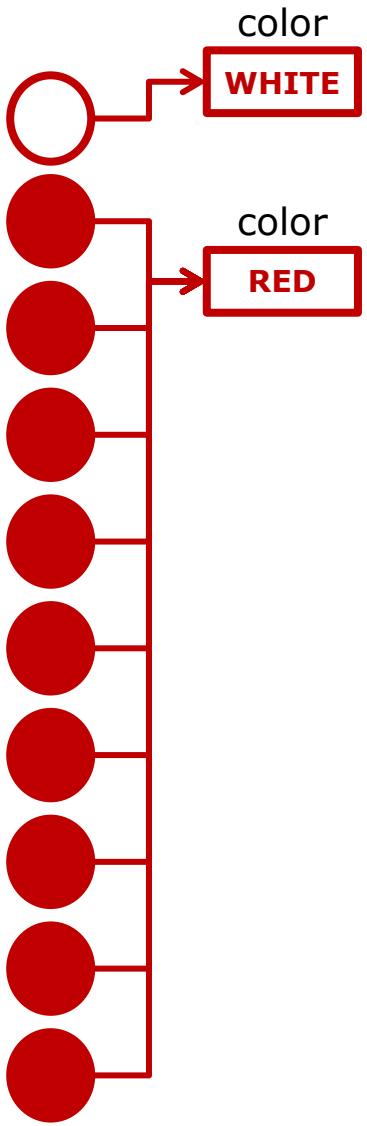
Fibonacci heaps

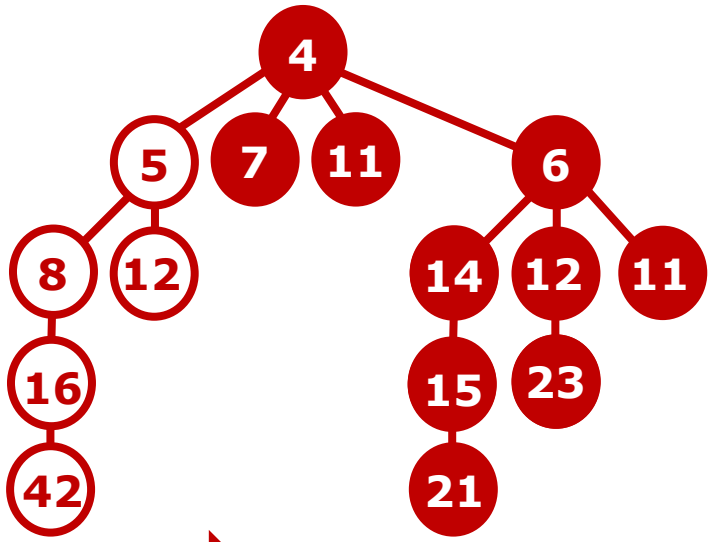
	Binary heaps	Binomial queues		Fibonacci heaps		Run-relaxed heaps	one tree		
	Williams 1964	Vuillemin 1978		Fredman Tarjan 1984		Driscoll et al. 1988	Brodal 1995	Brodal 1996	STOC 2012
Insert	log n	log n	1	1	**1**	1	1	1	**1**
FindMin	1	1	1	1	**1**	1	1	1	**1**
Delete	log n	log n	log n	n	**log n**	log n	log n	log n	**log n**
Meld	-	log n	1	1	**1**	log n	1	1	**1**
DecreaseKey	log n	log n	log n	n	**1**	1	log n	1	**1**

Amortized complexity (Tarjan 1983)

Arrays | Pointers only

Ideas





MELD ⇨ smallest tree **red** + link

Invariants

1. **white** nodes share one color record
2. **white** children left of **red**
3. root is **red**

Intuition

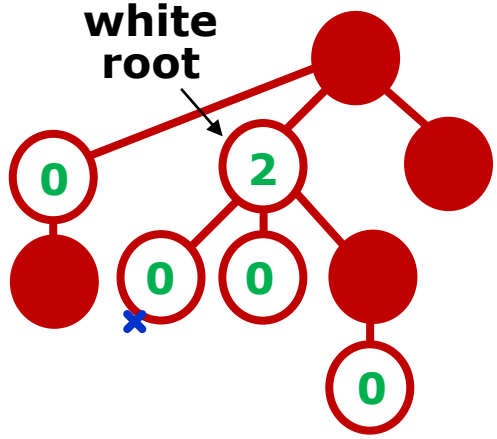
1. Only maintain structure for **white** nodes
2. **Red** non-root nodes never increase degree
3. DELETE: O(1) **red** nodes → **white**

Definitions

1. **white** node **rank** = #**white** children
2. DECREASEKEY: cut + **mark** parent (if white)

Invariants

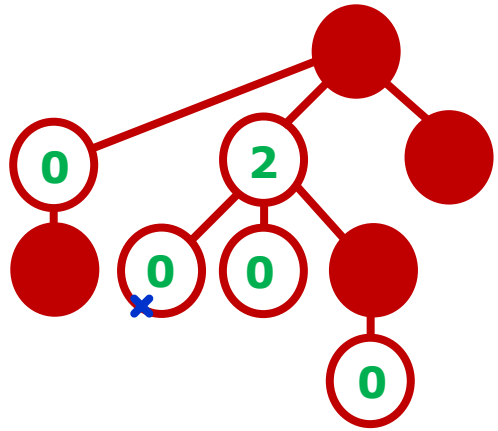
1. i'th rightmost **white** child of a **white** node:
   rank + #**marks** ≥ i - 1
2. #**marks** + #**white roots** = O(log n)



Theorem

max rank
O(log n)

Priority Queue Operations



- FINDMIN = return root
- INSERT = make single **red** node + MELD
- DELETE = DECREASEKEY to -∞ + DELETEMIN
- MELD = color smaller tree **red** + link + O(1) transformations
  root degree +1
- DECREASEKEY = cut + link with root + O(1) transformations
  root degree +1; white root (+1); marks (+1)
- DELETEMIN = cut root + find new root + O(log n) link
  root degree +O(log n); white root +O(log n)
  + O(log n) transformations

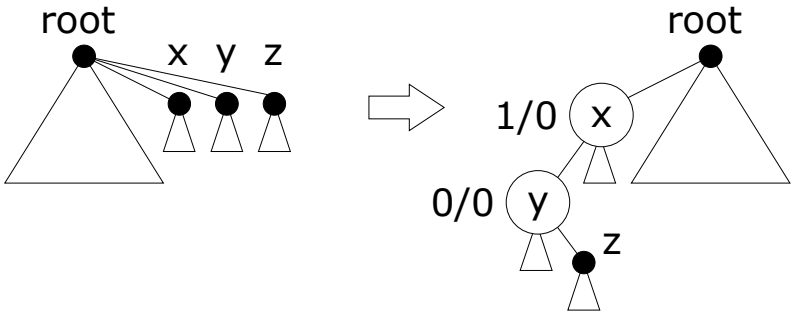
Invariant

R = α·log (3/2 · #**white** nodes + #**red** nodes) + β
- degree **unmarked white** nodes ≤ **R**
- degree **red** and **marked white** nodes ≤ **R** – 1

Transformations

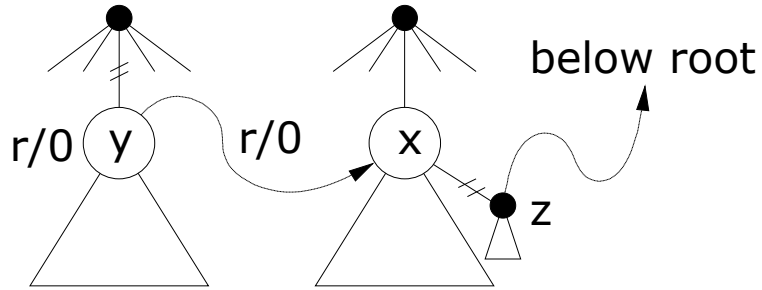
Root degree reduction

Converts two red nodes to white; reduces the root degree by two; creates one new white root



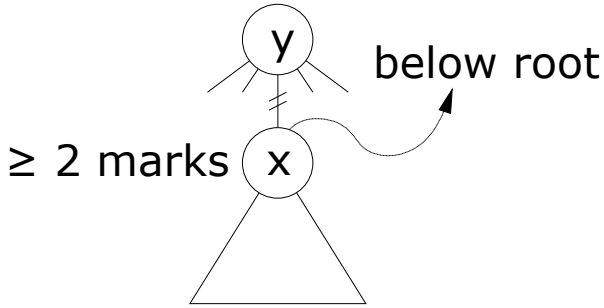
White root reduction

Two white roots of rank r is replaced by one rank r+1; increases root degree by one



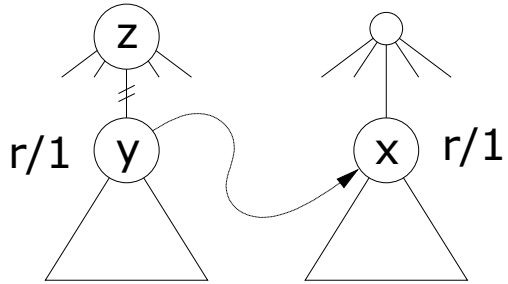
One node mark reduction

White node with ≥ 2 marks becomes a white root (unmarked); increases the root degree by one



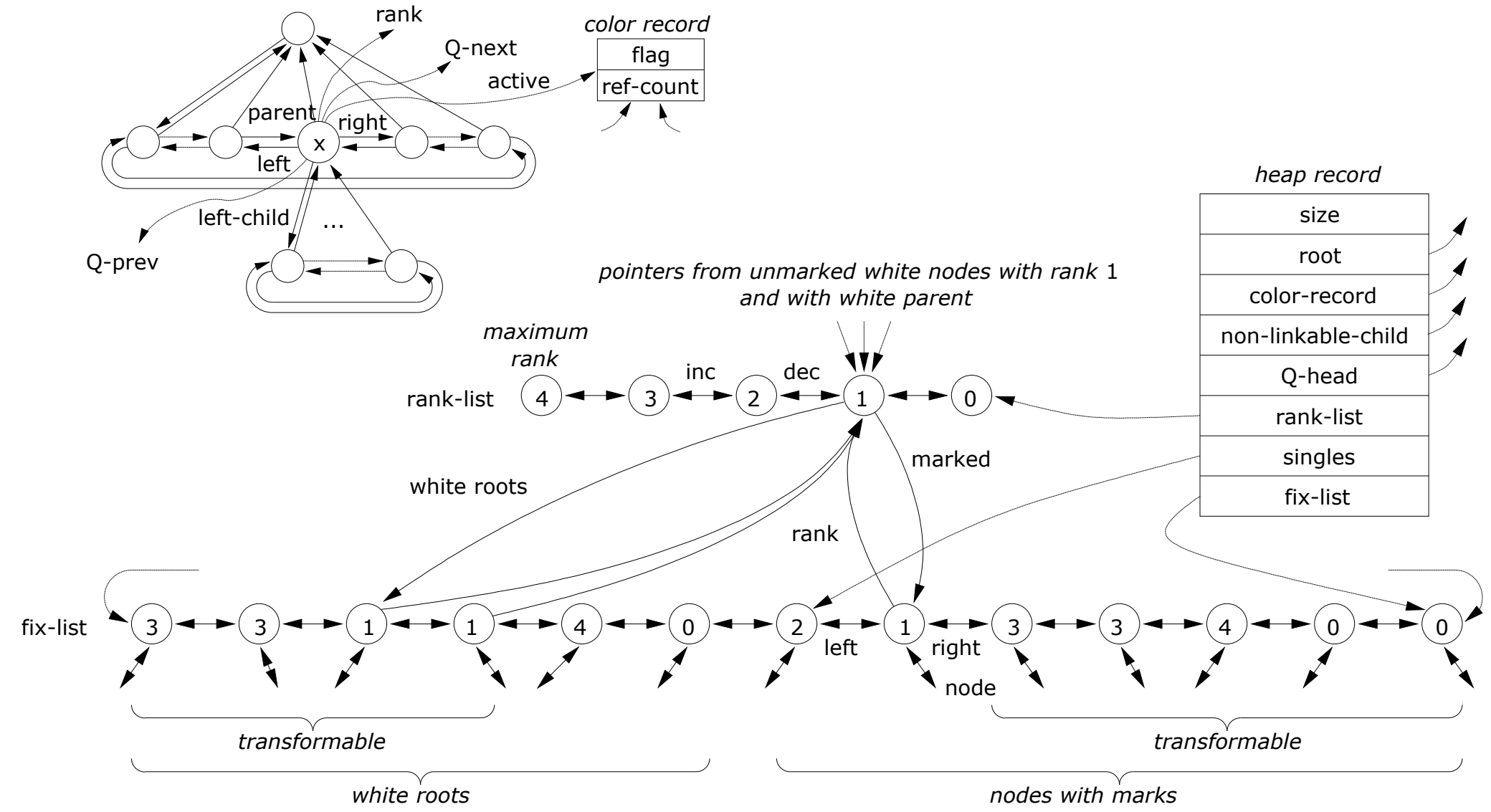
Two node mark reduction

Two nodes of equal rank r with 1 mark become unmarked; one parent one more mark



r / m = rang / #marks

Representation



	Williams 1964	Vuillemin 1978		Fredman Tarjan 1984		Driscoll et al. 1988	Brodal 1995	Brodal 1996	STOC 2012
Insert	log n	log n	1	1	**1**	1	1	1	**1**
FindMin	1	1	1	1	**1**	1	1	1	**1**
Delete	log n	log n	log n	n	**log n**	log n	log n	log n	**log n**
Meld	-	log n	1	1	**1**	log n	1	1	**1**
DecreaseKey	log n	log n	log n	n	**1**	1	log n	1	**1**

Pointers only

Thank You