Fully Persistent B-Trees

Gerth Stølting Brodal
Konstantinos Tsakalidis
Aarhus University, Denmark

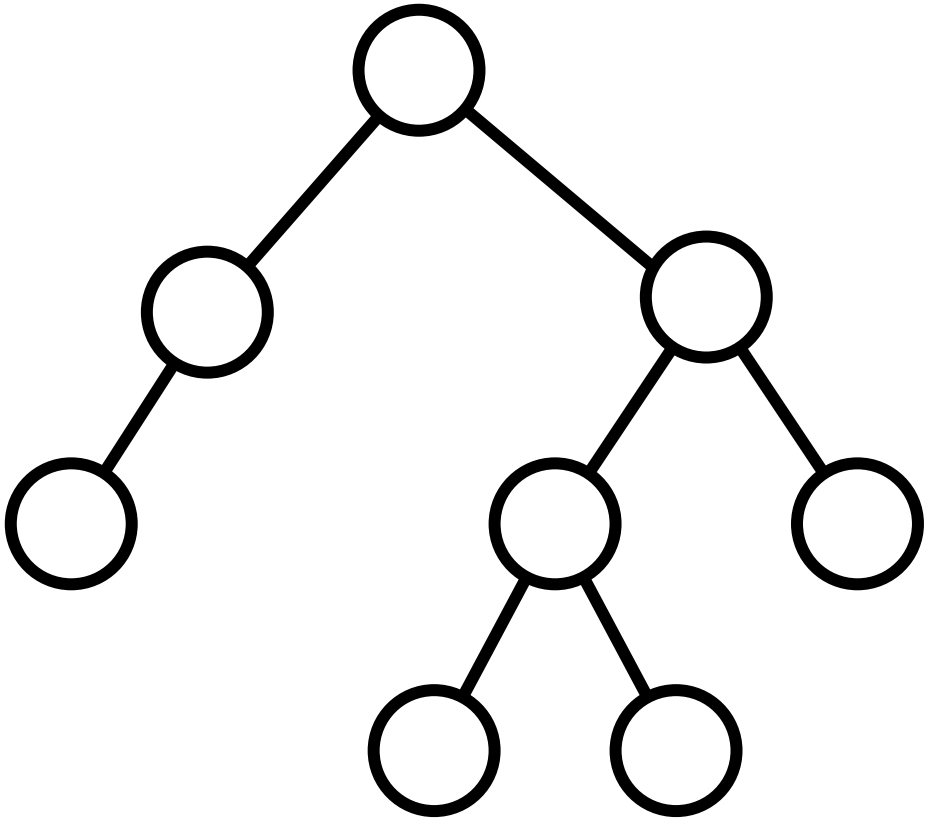
MADALGO
CENTER FOR MASSIVE DATA ALGORITHMICS

Spyros Sioutas
Ionian University, Corfu, Greece

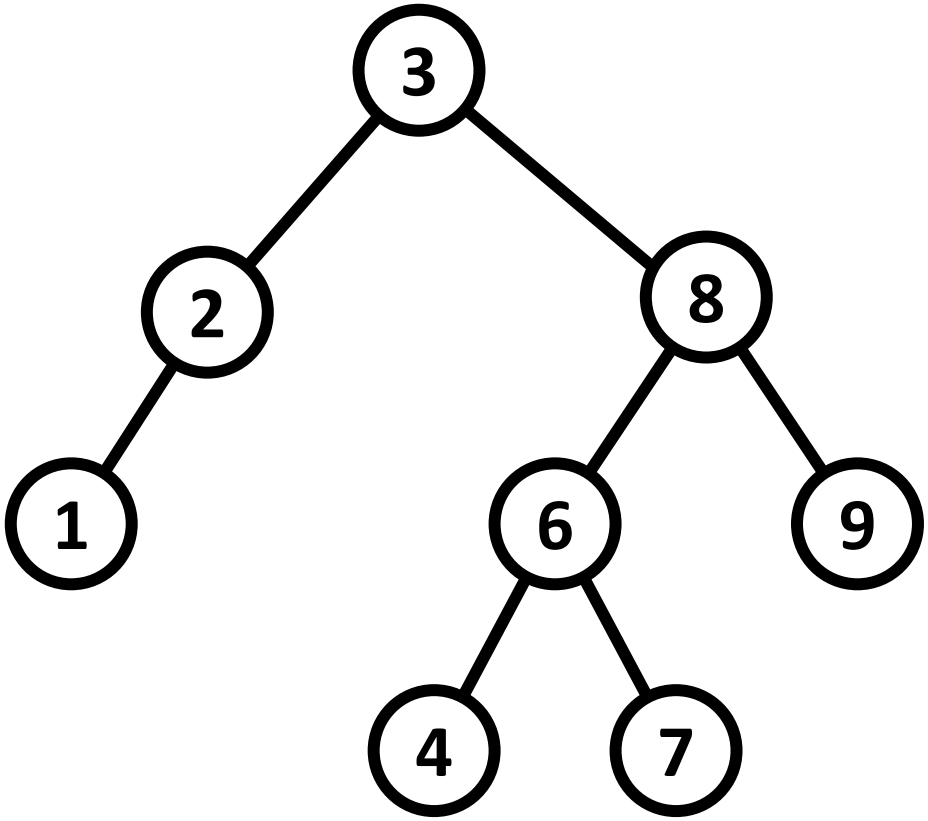
Kostas Tsichlas
Aristotle University of Thessaloniki, Greece

23rd Annual ACM-SIAM Symposium on Discrete Algorithms, Kyoto, Japan, January 18, 2012

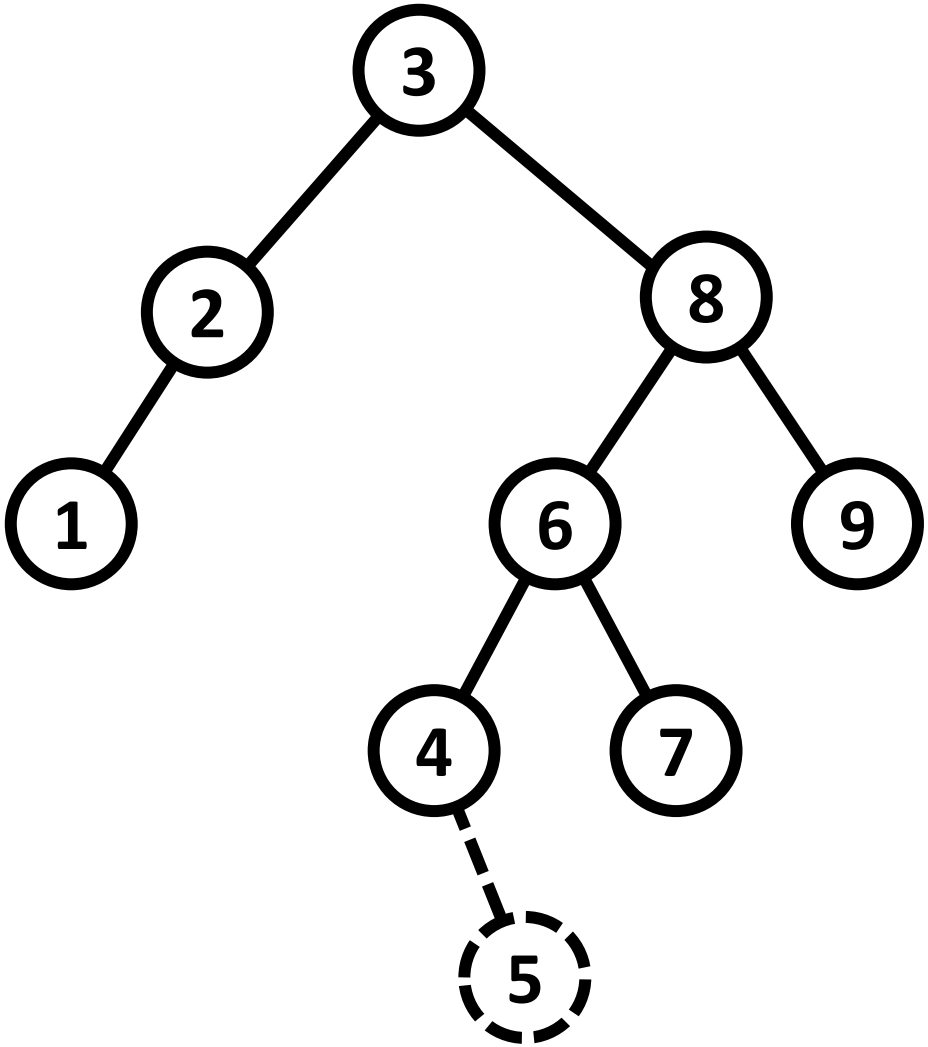
Binary Tree



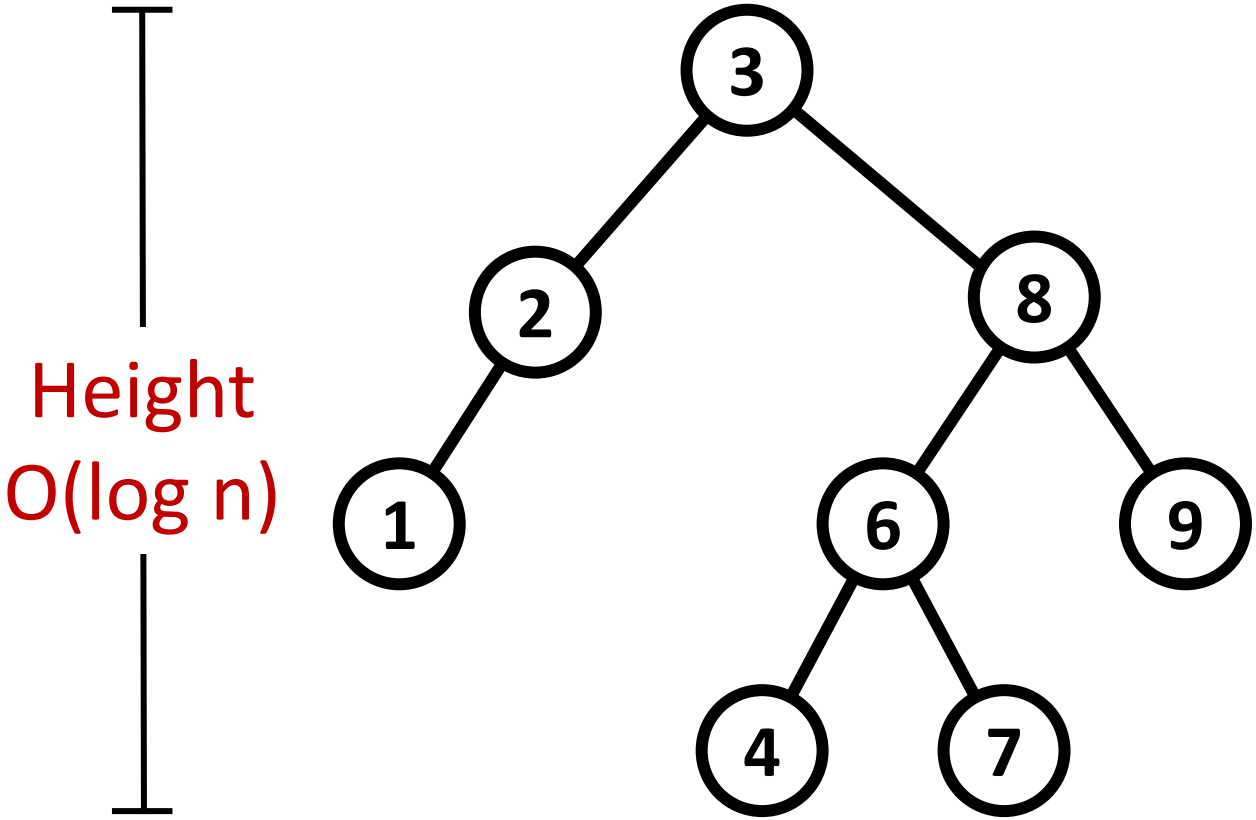
Binary Search Tree



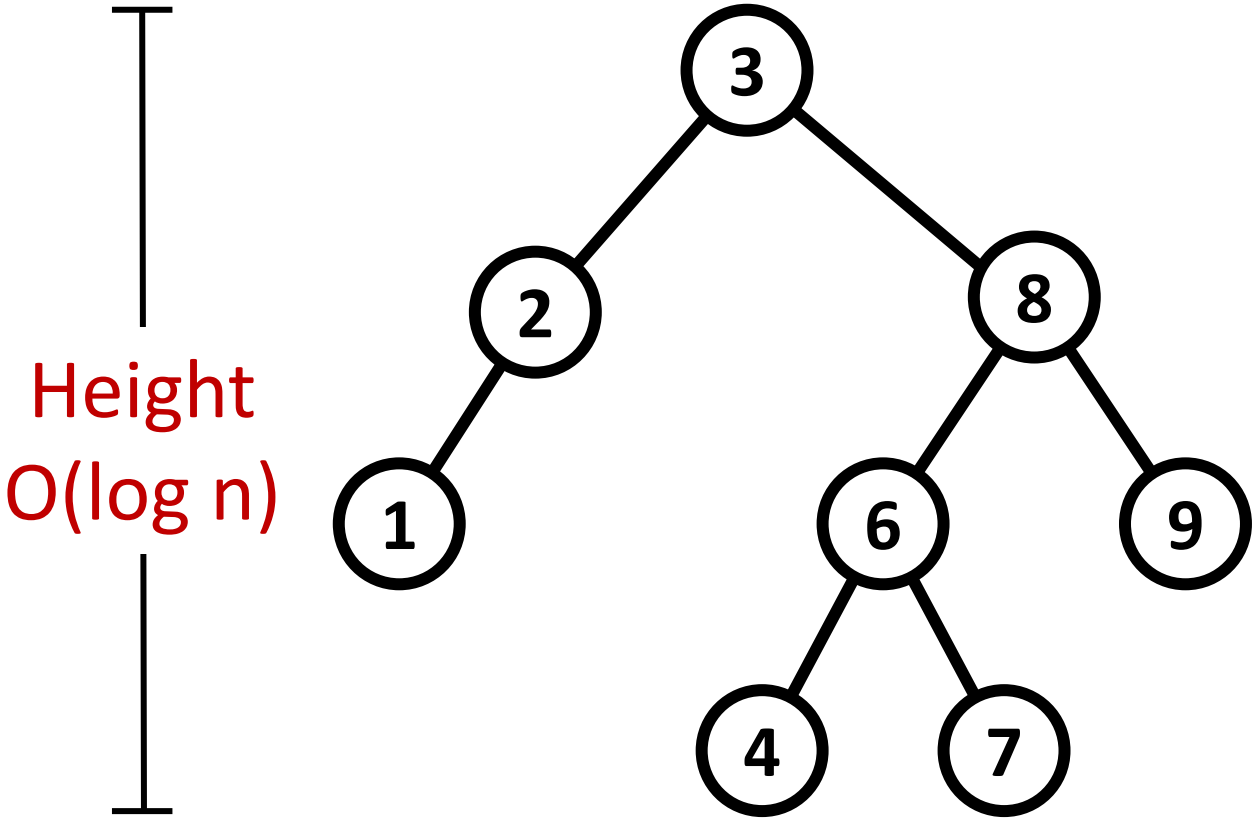
Insert(5)



Balanced Binary Search Tree

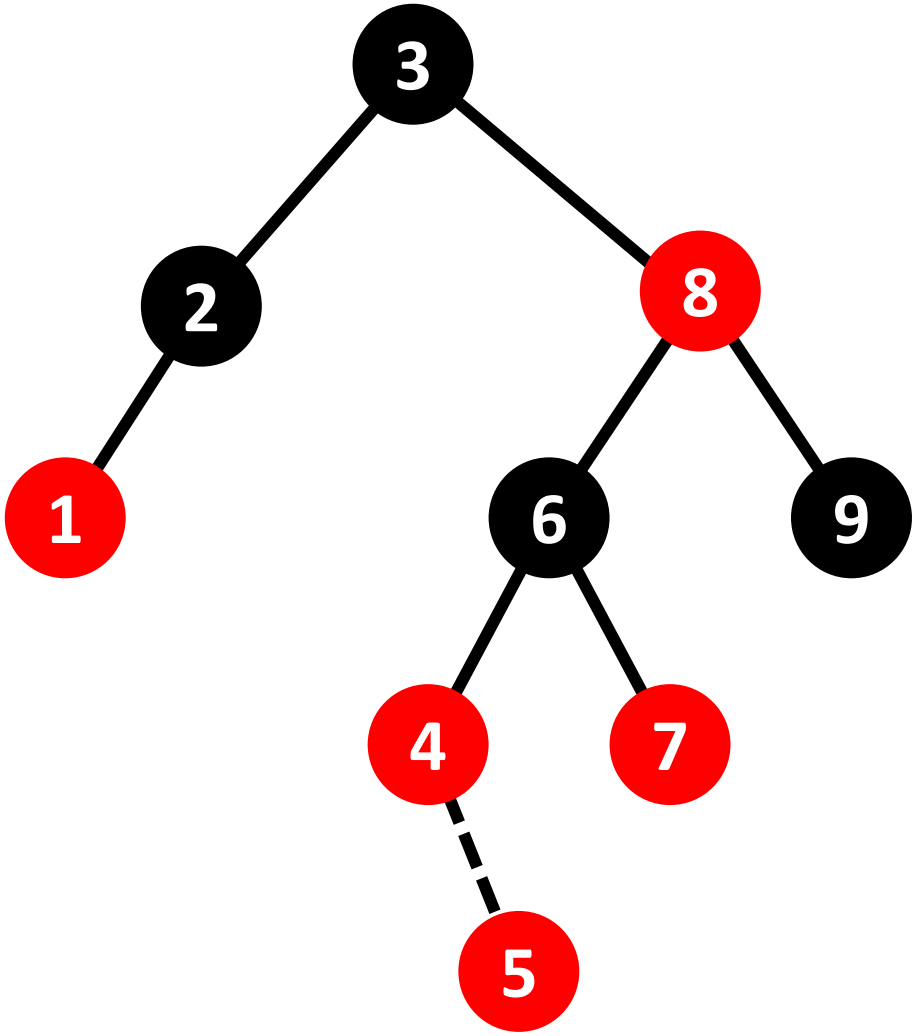


Report(2,7)



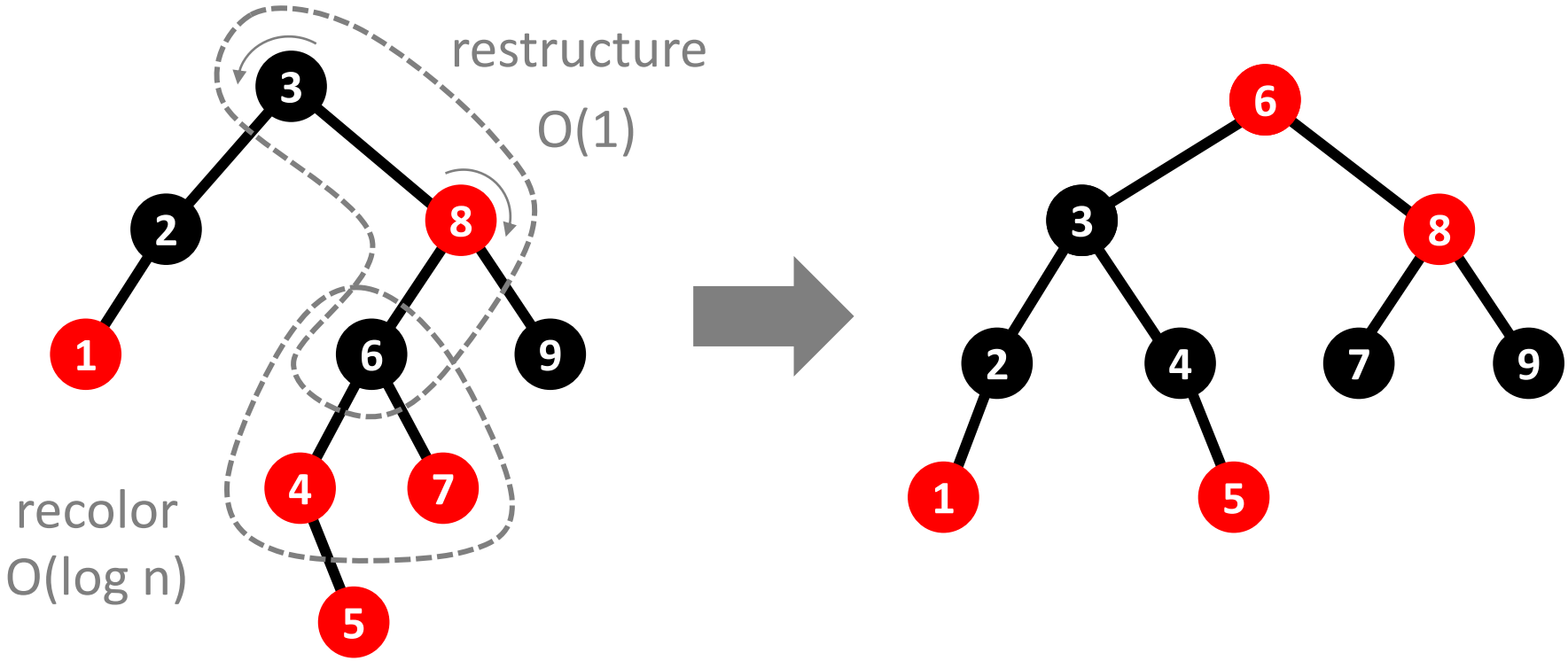
Time O(log n + t)

Red-Black Tree



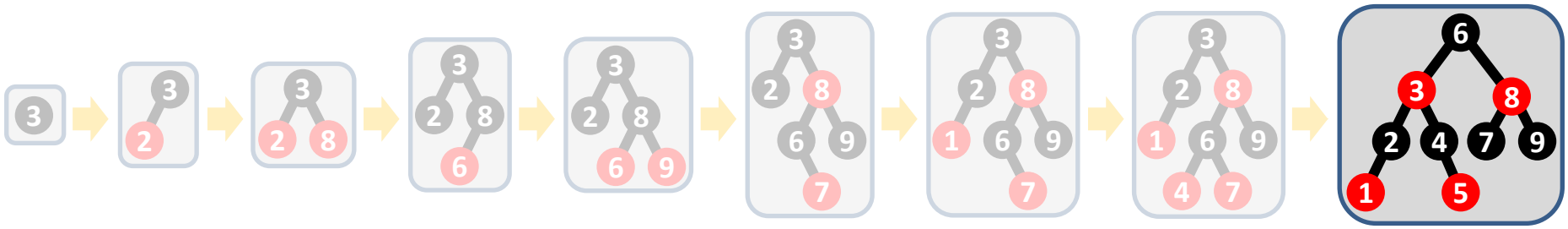
Bayer 1972

Red-Black Tree - rebalancing



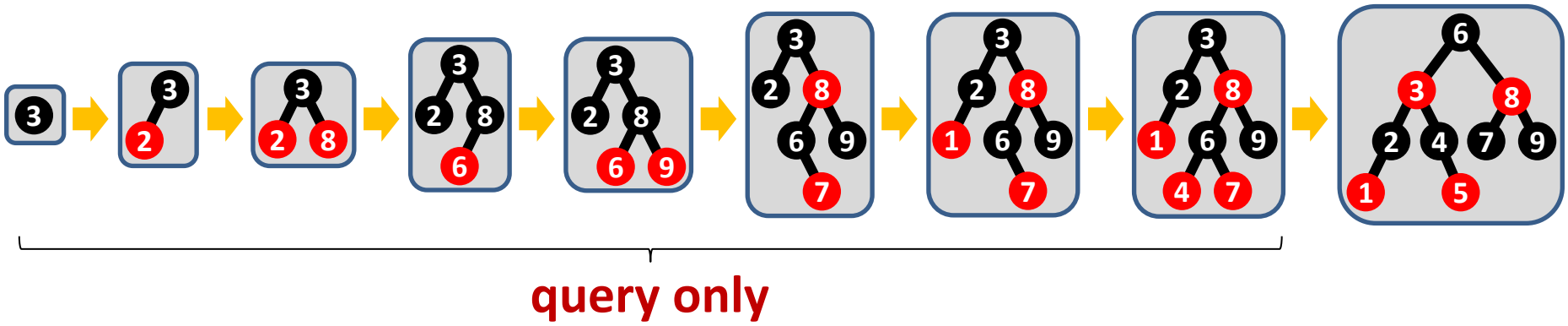
Bayer 1972

Ephemeral Red-Black Tree



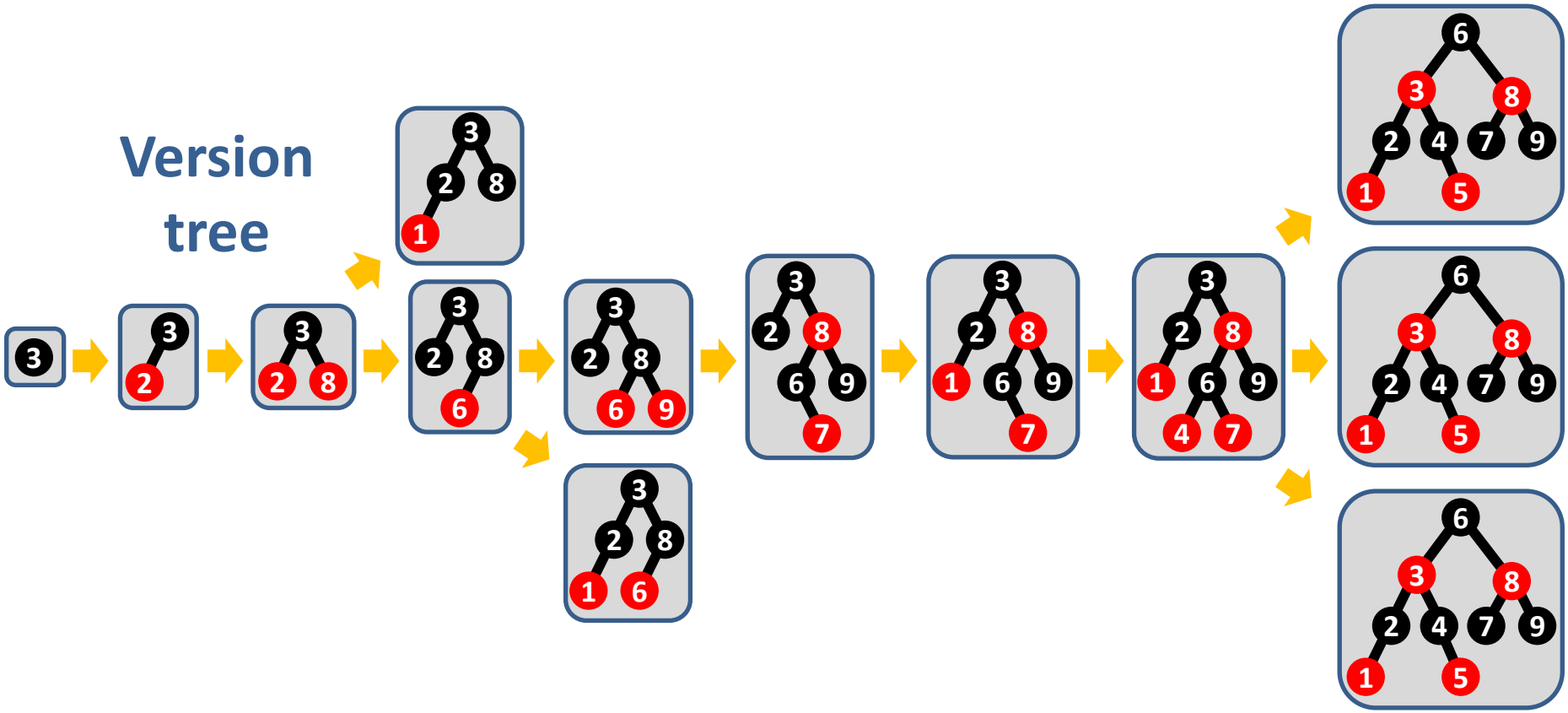
Driscoll, Sarnak, Sleator, Tarjan 1986

Partially Persistent Red-Black Tree



Driscoll, Sarnak, Sleator, Tarjan 1986

Fully Persistent Red-Black Tree



Driscoll, Sarnak, Sleator, Tarjan 1986

Fully Persistent Red-Black Tree

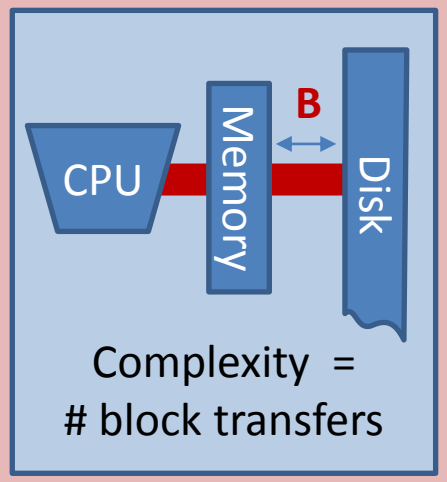
Contributions of Driscoll et al.

1. **Red**-**black** trees with O(1) modifications per update (displacement paths)
2. General technique to make pointer based data structures fully persistent

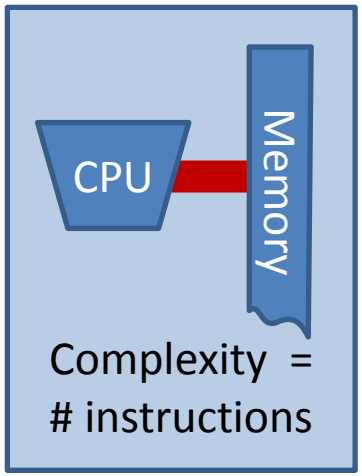
Expensive to record updates performing **many modifications**

Driscoll, Sarnak, Sleator, Tarjan 1986

IO model



RAM model



Binary Search Trees

Red-Black Trees
Bayer 1972

B-Trees
Bayer 1972

Maintaining Order in a List
Dietz, Sleator 1987

Red-Black Trees w. Displacement Paths
Driscoll et al. 1986

Incremental B-Trees

IO Efficient Node Splitting

Full Persistence: Node splitting
Driscoll et al. 1986

Fully Persistent B-Trees

Fully Persistent Balanced Binary Search Trees
Driscoll et al. 1986

Results

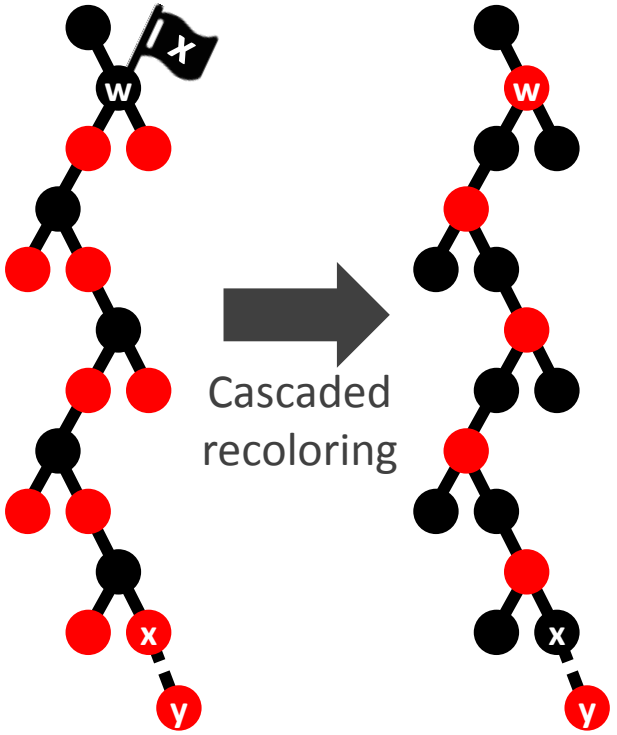
Incremental B-tree with range searches in $O(\log_B n + t/B)$ IOs, updates in $O(\log_B n)$ IOs with O(1) key and pointer modifications, O(n/B) space

Any emphemeral pointer based IO data structure with O(1) indegree can be made fully persistent with O(1) overhead per block access and $O(\log_2 B)$ IOs amortized overhead per field modification

Fully persistent B-tree with updates in $O(\log_B n + \log_2 B)$ IOs amortized, range searches in $O(\log_B n + t/B)$ IOs, O(m/B) space

	Persistent B-tree result	Query	Update
Partial	Arge, Danner, Teh 2003	$\log_B n + t/B$	$\log_B n$
Full	Lanka, Mays 1991	$(\log_B n + t/B) \cdot \log_B m$	$\log_B n \cdot \log_B m$
	This talk	$\log_B n + t/B$	$\log_B n + \log_2 B$

Incremental Search Tree Updates

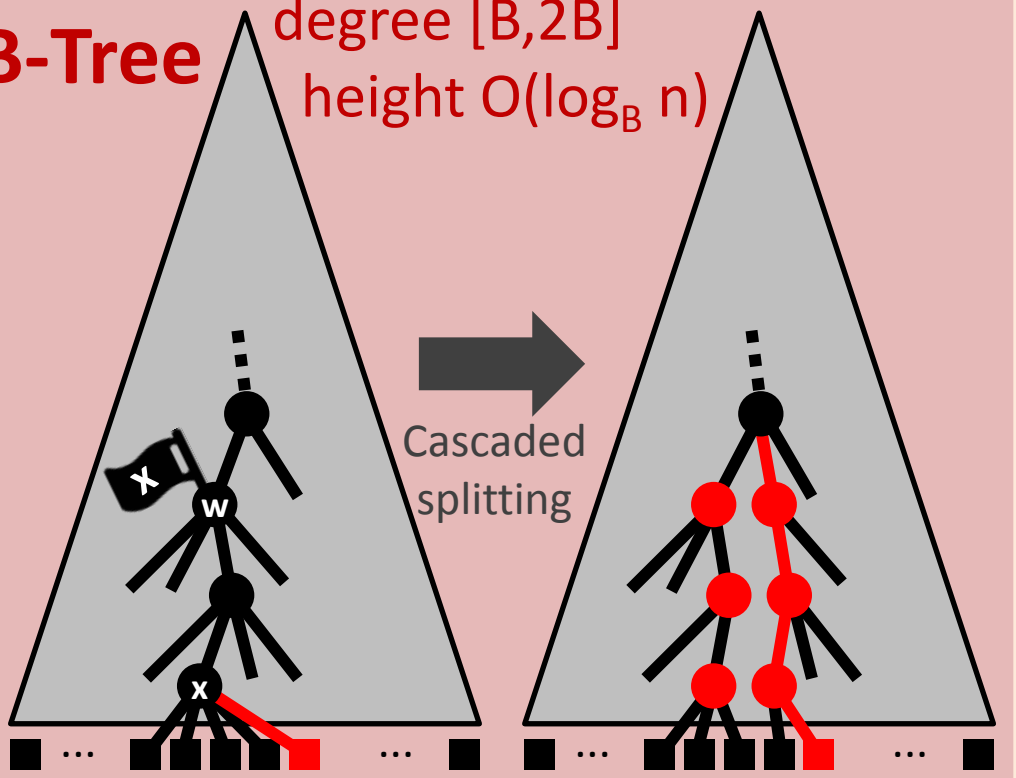


Displacement path w→x
≡ flip all colors on path + all red siblings (flag w with x)

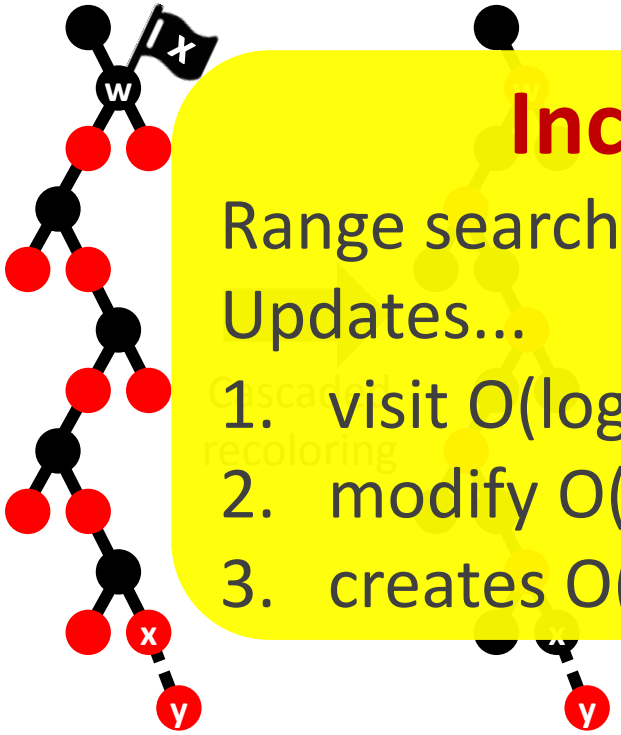
Driscoll et al. 1986

B-Tree

degree [B,2B]
height $O(\log_B n)$

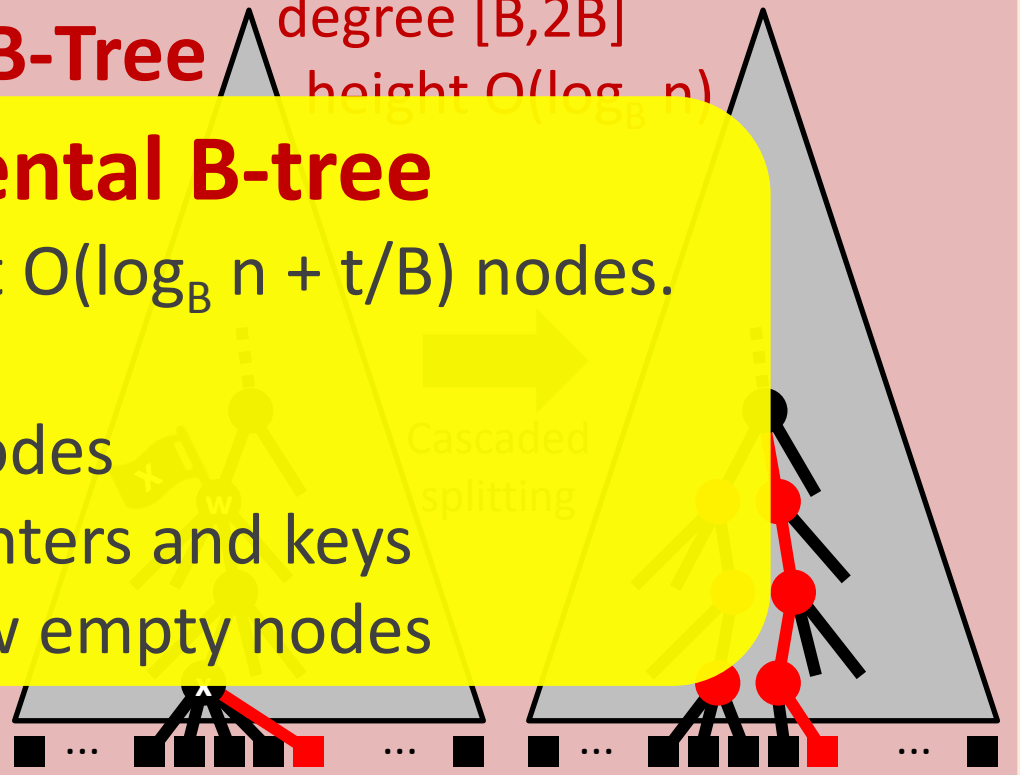


Incremental Search Tree Updates



B-Tree degree [B,2B]

height $O(\log_B n)$



Incremental B-tree

Range searches visit $O(\log_B n + t/B)$ nodes.

Updates...

1. visit $O(\log_B n)$ nodes
2. modify O(1) pointers and keys
3. creates O(1) new empty nodes

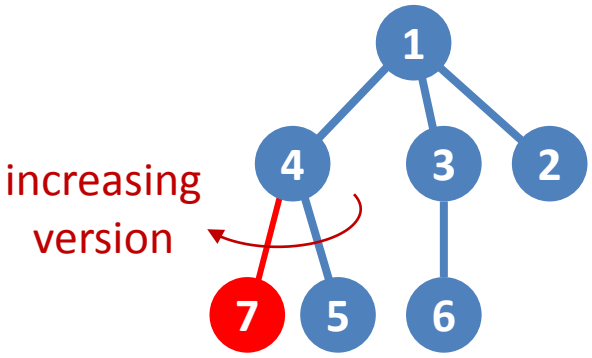
Displacement path w→x

≡ flip all colors on path + all red siblings (flag w with x)

Driscoll et al. 1986

Full persistence : Node Splitting Technique

Driscoll, Sarnak, Sleator, Tarjan 1986



Version tree (numbers = version ids)

preorder traversal

Version list
(order maintenance data structure)

Dietz, Sleator 1987

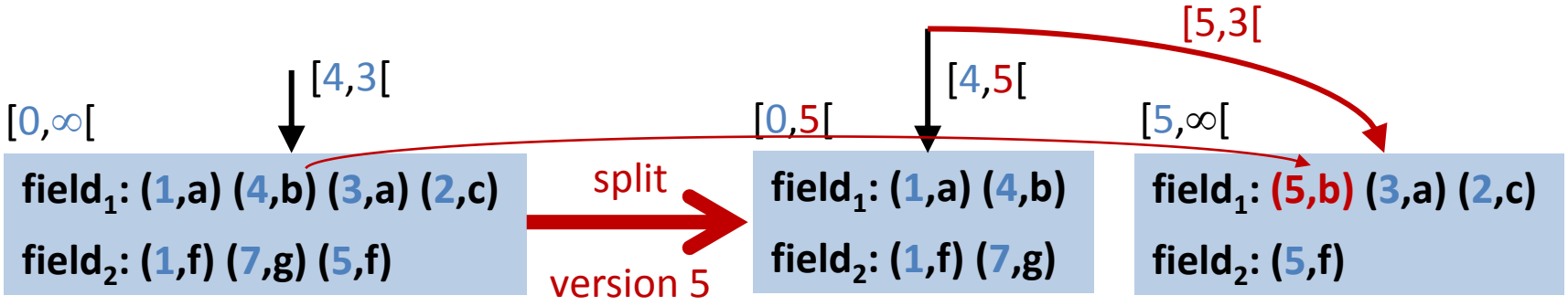
Fat node

Queries = binary search among versions

field: (1,x) (7,z) (5,x) (6,y) (2,x)

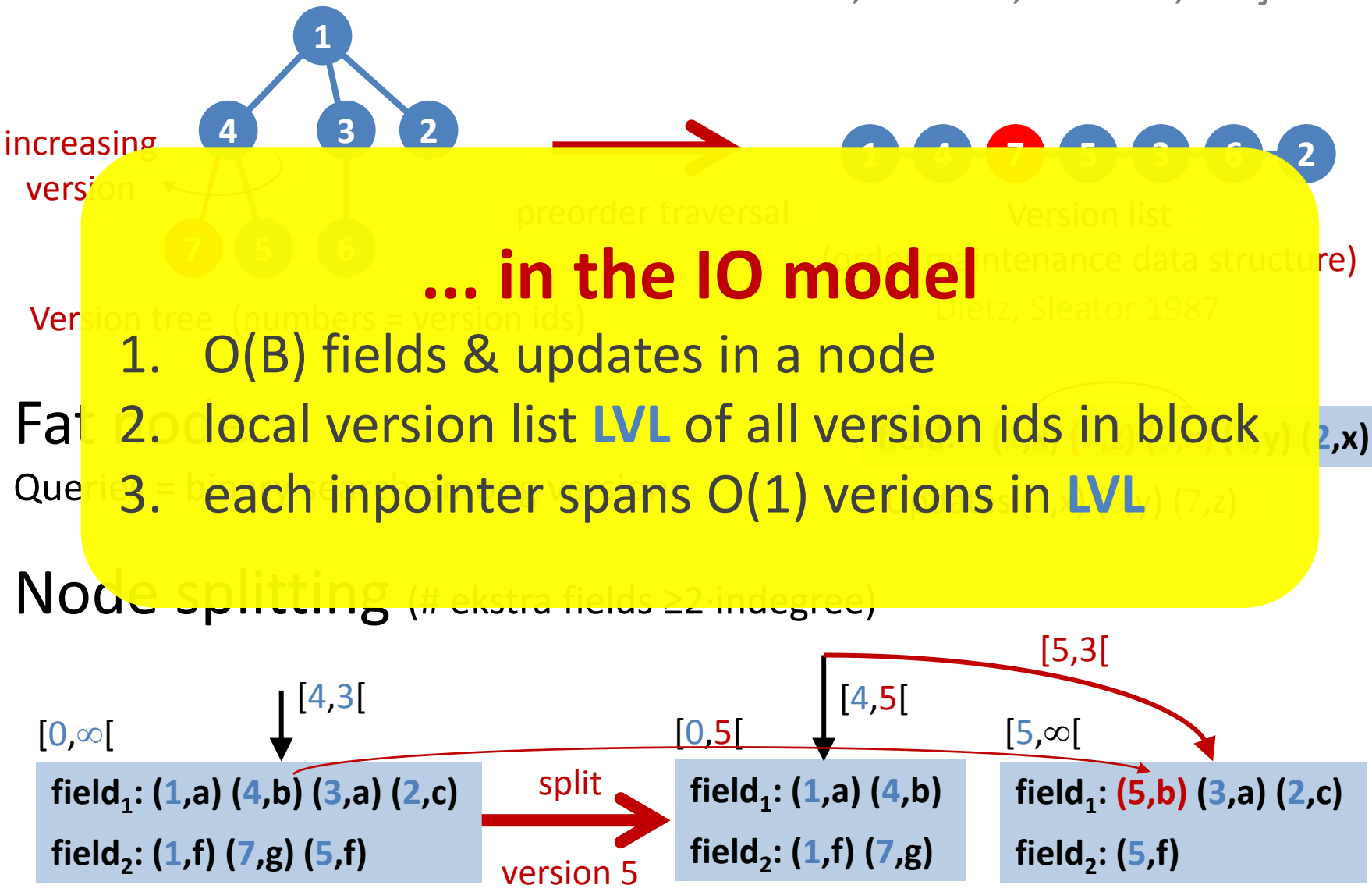
Updates (1,x) (6,y) (7,z)

Node splitting (# ekstra fields ≥2·indegree)



Full persistence : Node Splitting Technique

Driscoll, Sarnak, Sleator, Tarjan 1986



Main result

Fully persistent B-tree with updates in $O(\log_B n + \cancel{\log_2 B})$?
IOs, searches in $O(\log_B n + t/B)$ IOs, using space $O(m/B)$

Thank You