Purely Functional Worst Case Constant Time Catenable Sorted Lists

Gerth Stølting Brodal
University of Aarhus

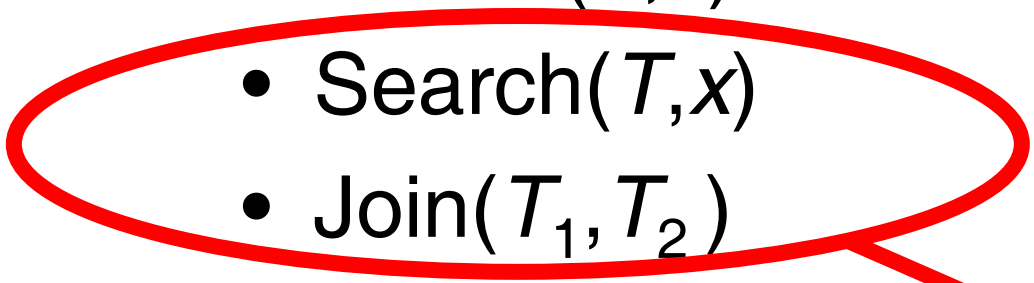
Joint work with

Christos Makris Kostas Tsichlas
University of Patras

ESA'06, Zürich, Switzerland, September 13, 2006

Catenable Sorted Lists

- Insert(T,x)
- Delete(T,x)
- Search(T,x)
- Join(T_1, T_2)
- Split(T,x)



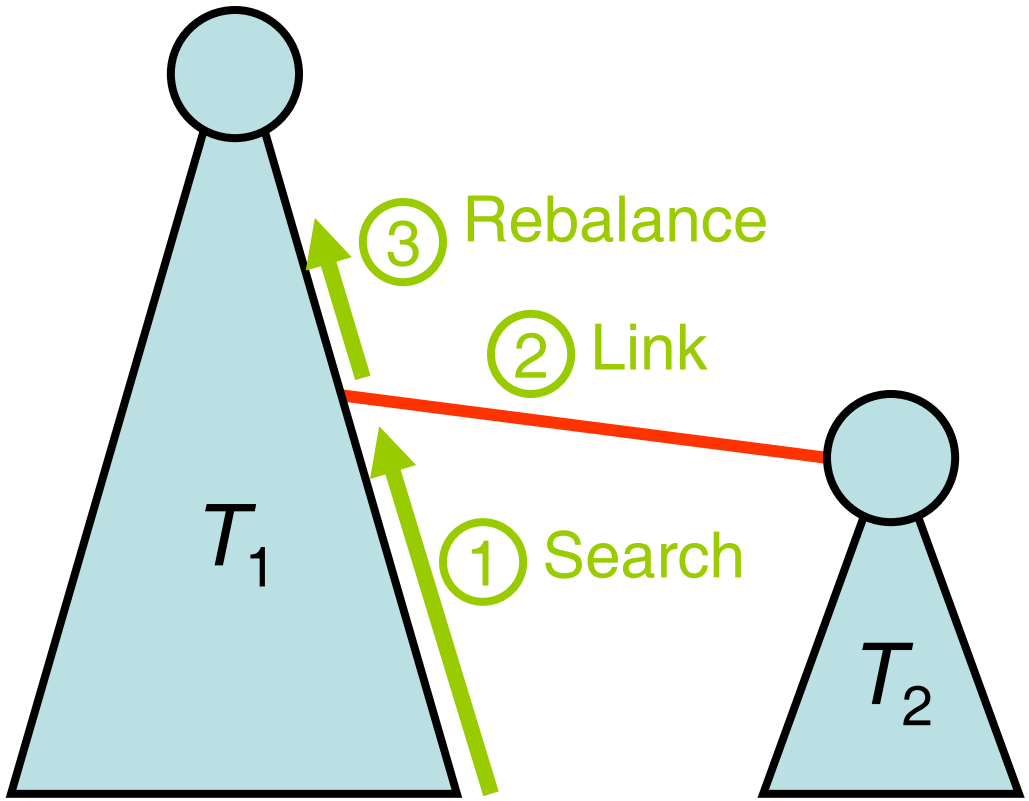
This talk

T_1 = 2,5,7,8,10,11 T_2 = 13,17,19,34,58,79

Join(T_1, T_2) = 2,5,7,8,10,11,13,17,19,34,58,79

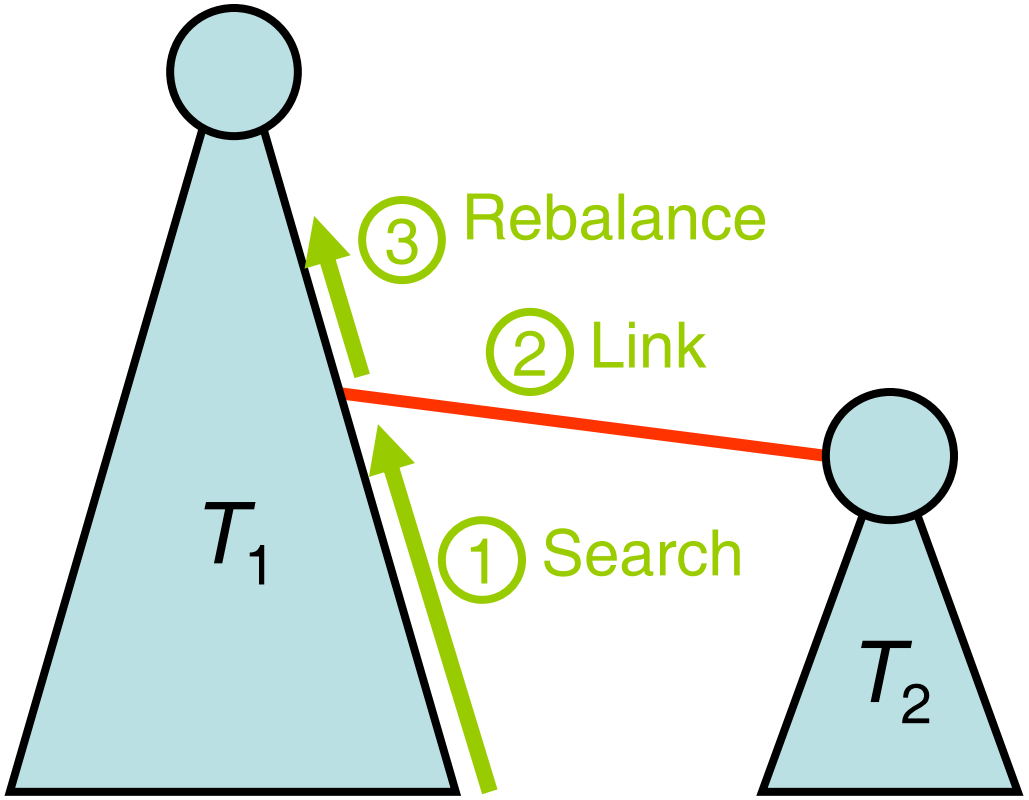
Catenable Sorted Lists: Search Trees

(2,4-trees, red-black trees…)



Catenable Sorted Lists: Search Trees

(2,4-trees, red-black trees…)



1. Worst-case O(log $|T_2|$)
   or O(loglog $|T_2|$)
   or amortized O(1)
2. Worst-case O(1)
3. Worst-case O(log $|T_1|$)
   or amortized O(1)
   or worst-case O(1)

Catenable Sorted Lists

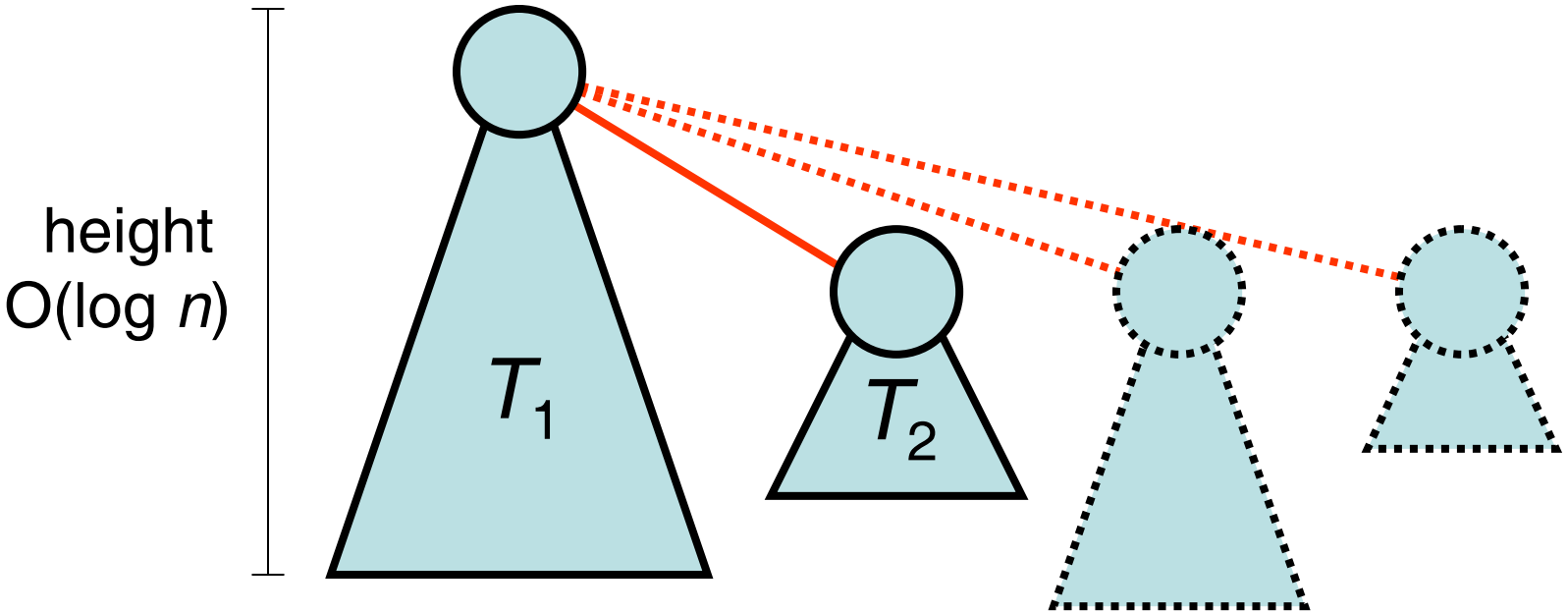
	Search trees	Kaplan Tarjan'96	This talk
Search	log n	log n	log n
Insert/Delete	log n	log n	log n
Join	log n 1*	loglog n_s	1
Split	log n	log n_s	-

$n = |T|$ $n_s = \min(|T_1|,|T_2|)$ * amortized

O(1) Join and O(log *n*) Search

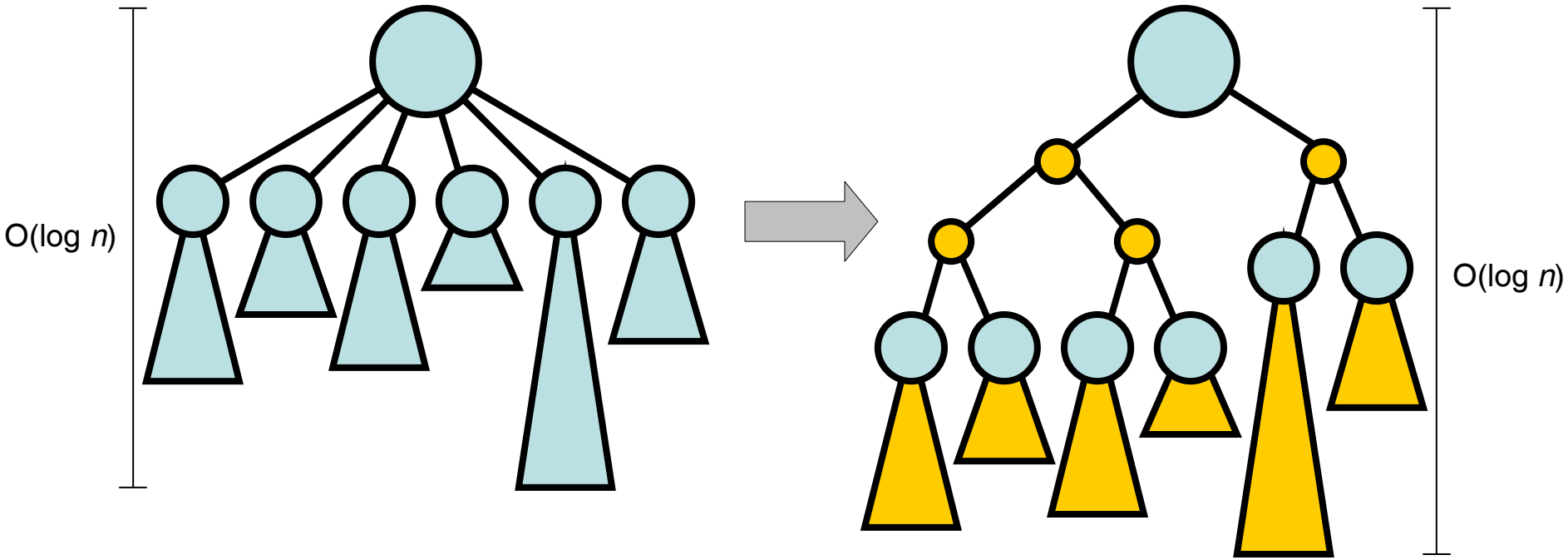
The ideas….

Idea 1: Linking By Size



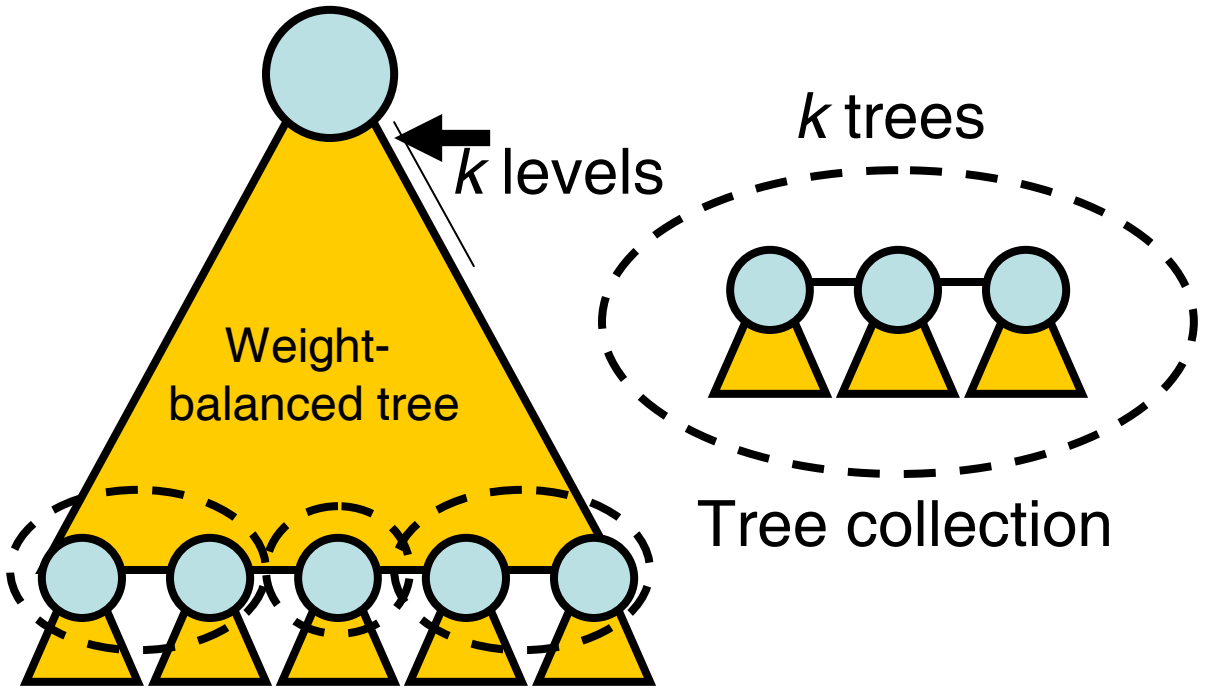
Problem: Nodes of arbitrary degree

Idea 2: Represent Nodes by Weight-Balanced Trees

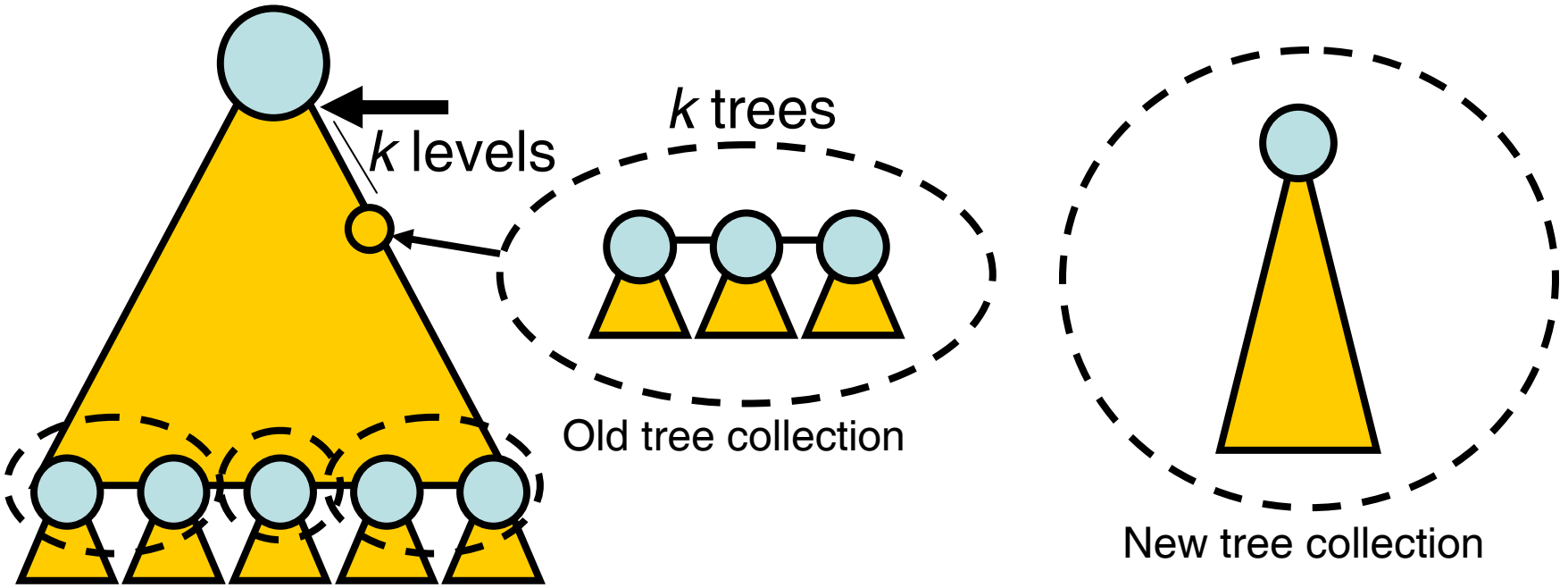


Problem: Does not support linking by size in O(1) time

Idea 3: Tree-Collections and Lazy Join



Idea 3 (*cont.*): Lazy Join

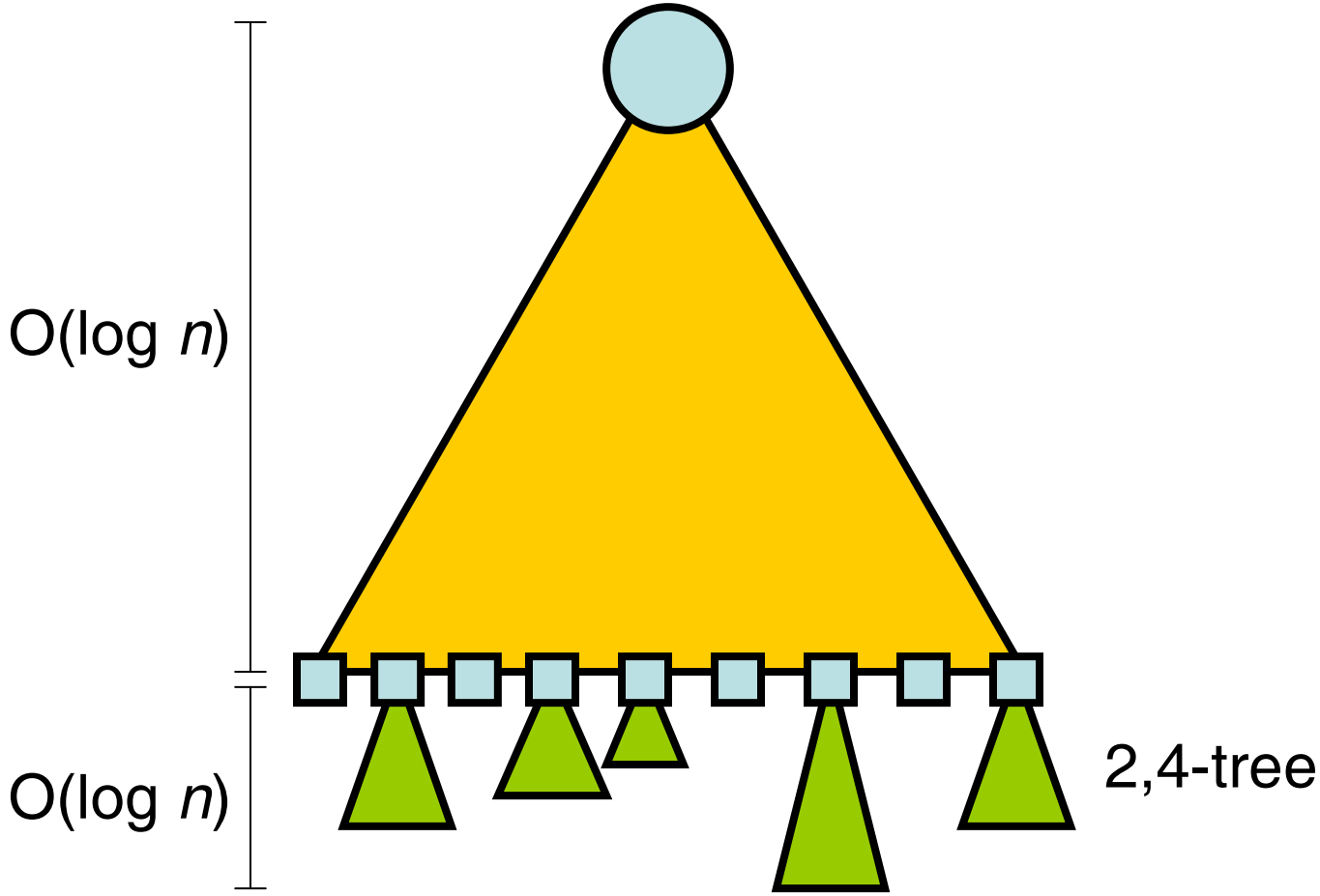


Weight balanced trees with top-down rebalancing
[Bent, Sleator, Tarjan 1985]

Catenable Sorted Lists

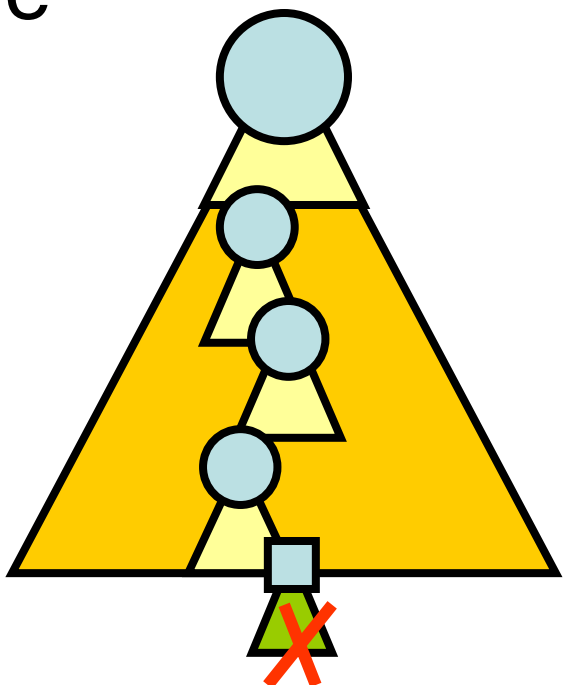
Operation	Worst-case time
Search	O(log *n*)
Join	O(1)

Supporting Insertions: Buckets



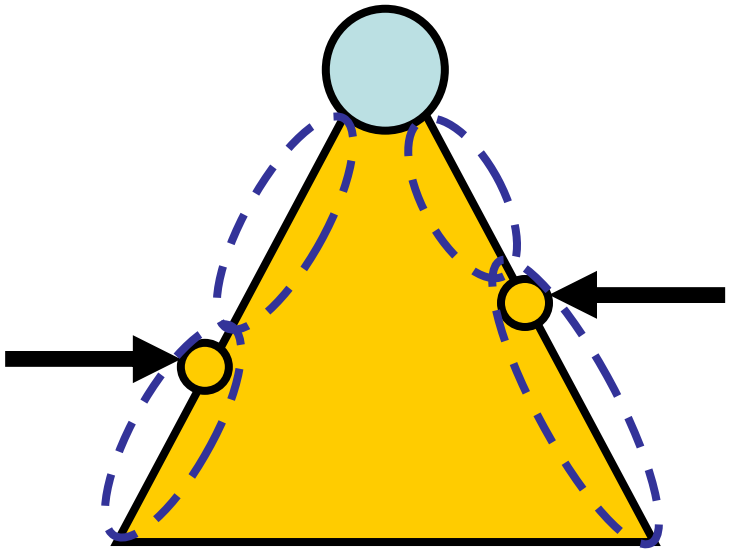
Supporting Deletions: The ideas…

- Delete leaf
- Rebalance weight-balanced trees on root-to-leaf path
- Join O(1) weight-balanced trees in O(log *n*) time



Functional Catenable Sorte Lists

- Functional setting = no side effects allowed
- Data structures are automatically persistent
- Idea: Represent spines by functional catenable lists [Kaplan, Tarjan 1995]



Conclusion

	Search trees	This talk ♪
Search	log *n*	log *n*
Insert/Delete	log *n*	log *n*
Join	log *n* 1*	1
Split	log *n*	-

* amortized ♪ purely functional