

Finger Search Trees with Constant Insertion Time

Gerth Stølting Brodal



Max-Planck-Institut für Informatik

Saarbrücken

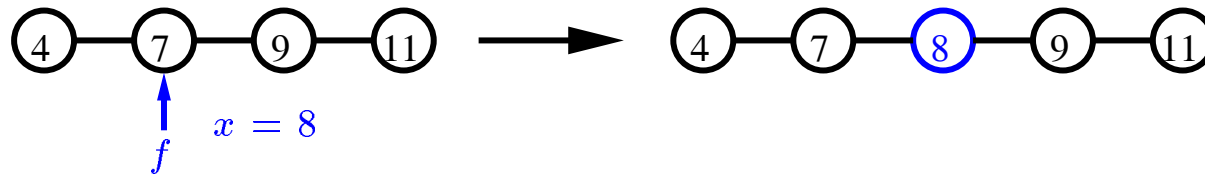
Germany

January 1998

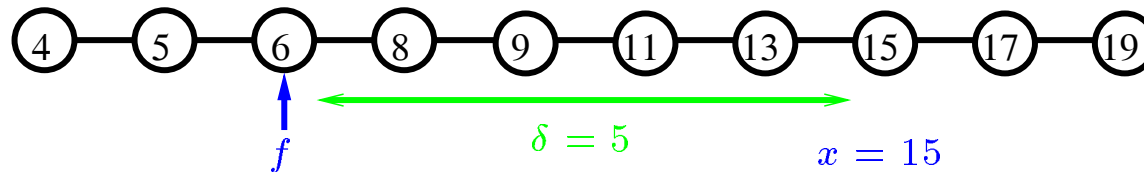
The problem

Maintain a **sorted list** of elements under the operations

- **INSERT**(f, x), insert element x to the right of **finger** f



- **SEARCH**(f, x), search for element x in the list starting at **finger** f

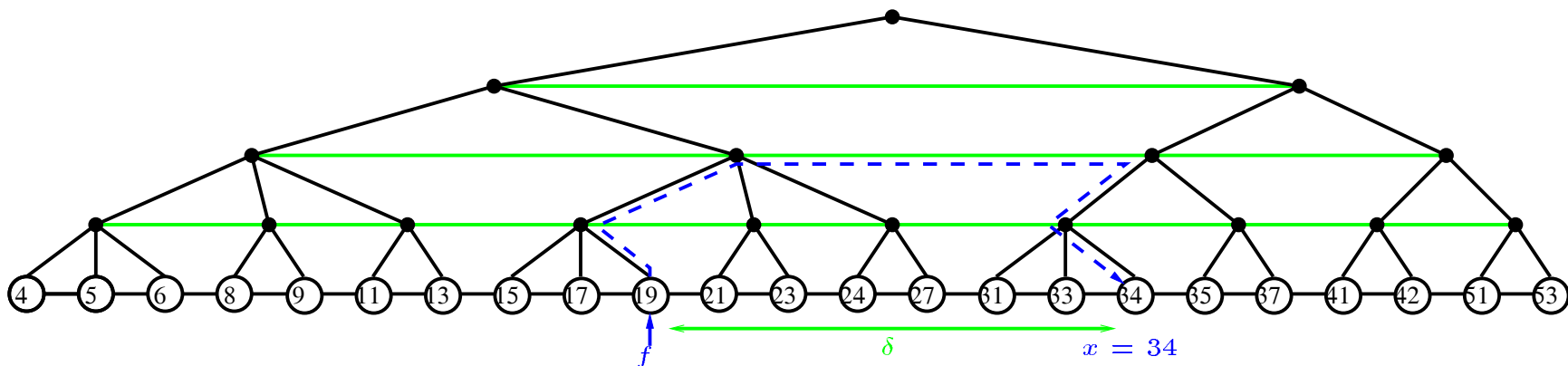


δ = the distance (rank difference) between f and x

A simple finger search tree

Brown, Tarjan '80

Represent the sorted list by a level linked (2,3)-tree



INSERT : worst-case $O(\log n)$ time, **amortized** $O(1)$ time

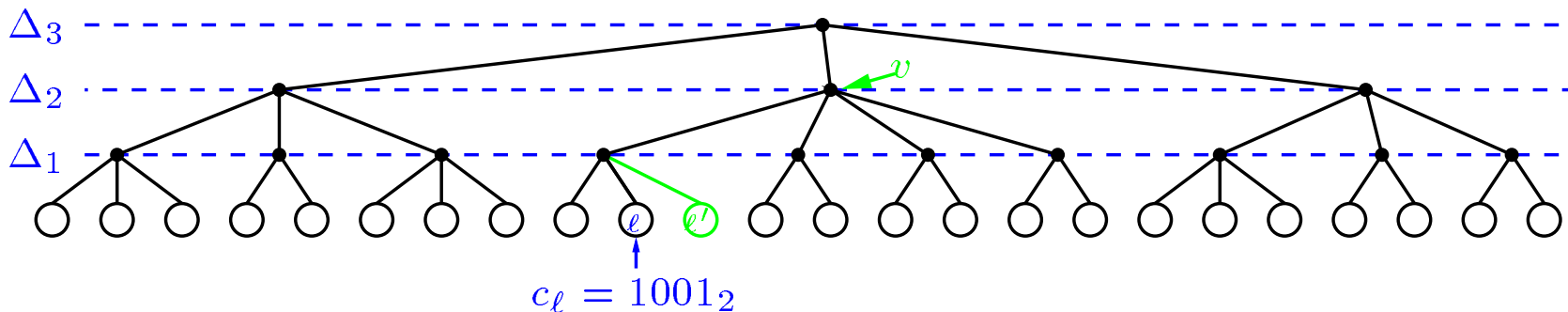
SEARCH : worst-case $O(\log \delta)$ time

Question: Can the insertion time be made worst-case $O(1)$?

Known results

	INSERT	SEARCH	
Search trees	$\log n$	$\log n$	
Levcopoulus, Overmars '88	1	$\log n$	
Brown, Tarjan '80	1	$\log \delta$	– amortized insertions
Dietz, Raman '94	1	$\log \delta$	– requires the RAM
Harel, Lucker '79	$\log^* n$	$\log \delta$	
Guibas <i>et al.</i> '77	1	$\log \delta$	– $O(1)$ fixed fingers
Brodal '98	1	$\log \delta$	– pointer machine

A new algorithm for splitting nodes in search trees



- To each leaf l is associated a binary counter c_l
- All internal nodes of height d have degree $\geq \Delta_d$ (except for the root)

INSERT(l, l')

- | | |
|--------------------------------------------------------------------|-------------------|
| (1) $c_l := c_l + 1$ | $c_l = 1010_2$ |
| (2) $c_{l'} := c_l$ | $c_{l'} = 1010_2$ |
| (3) Find $d : c_l \bmod 2^d = 2^{d-1}$ | $d = 2$ |
| (4) Split the d th ancestor of l (if degree $\geq 2\Delta_d$) | v |

Main lemma

Lemma If $\Delta_d \geq 2^{2^d} - 1$, then all nodes of height d have degree $\leq 2^{2 \cdot 2^d} \Delta_d$

Proof Let v be an internal node of height d , and ℓ any leaf

Define potentials

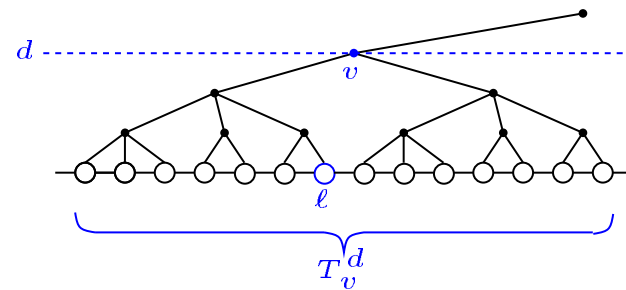
$$\Phi_\ell^d = 2^{2^d - 1 - ((c_\ell + 2^{d-1}) \bmod 2^d)}$$

$$\Phi_v^d = \sum_{\ell \in T_v^d} \Phi_\ell^d$$

Invariant

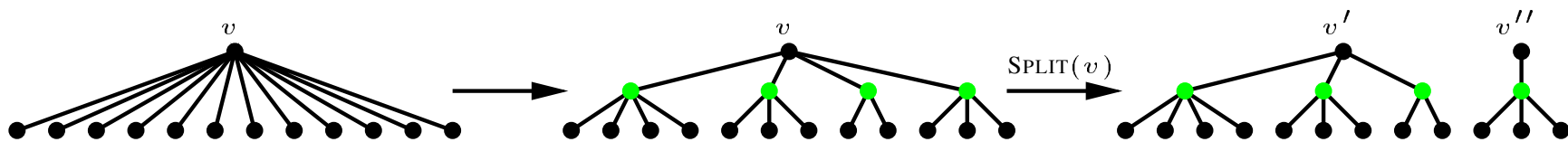
$$\Phi_v^d \leq 2^{2 \cdot 2^d} \prod_{i=1}^d \Delta_i$$

The lemma follows from $|T_v^d| \geq \prod_{i=1}^d \Delta_i$ □



Splitting nodes of large degree

Split nodes incrementally in advance by introducing **intermediate** nodes of degree $\Delta_d \dots 2\Delta_d - 1$



$\text{SPLIT}(v) \equiv$ move one intermediate node.

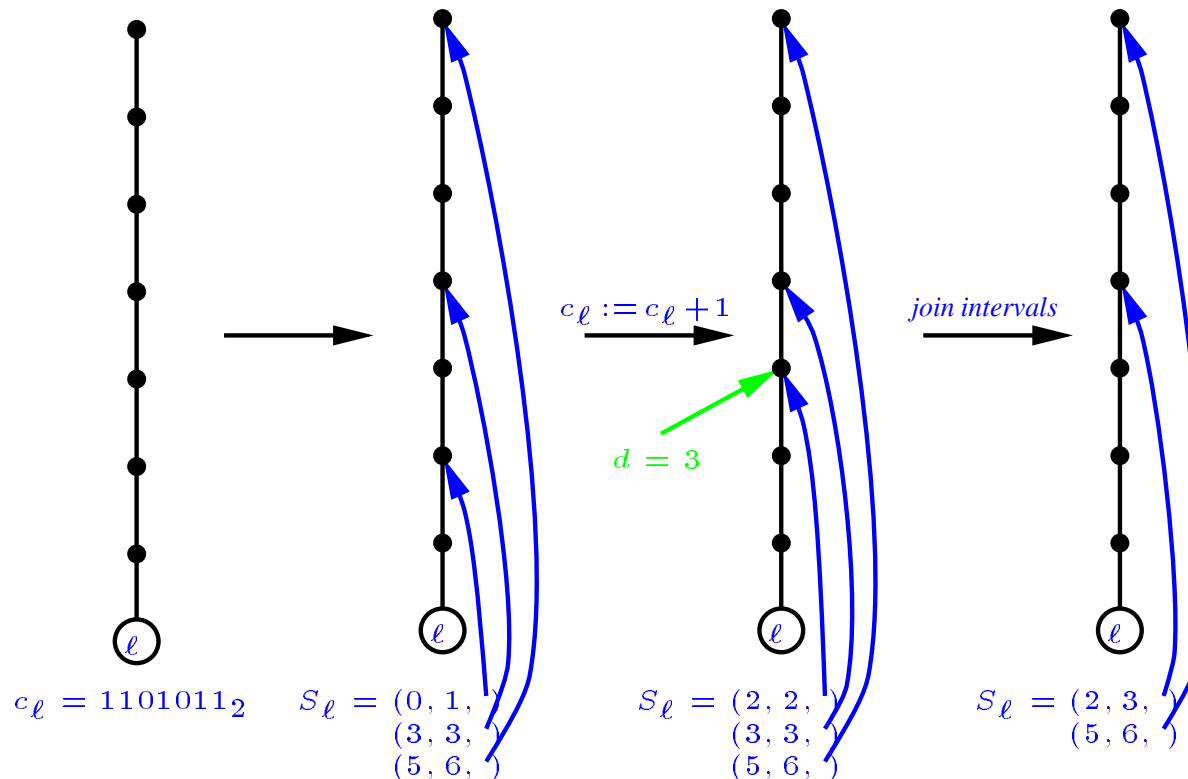
Finger searches

- (1) Level link the tree
- (2) Represent the children of each internal/intermediate node by the search trees of [Levcopoulos, Overmars '88](#)

\Rightarrow finger searches take $O(\log \delta)$ time

Finding level d ancestors

- (1) Represent each counter c_ℓ by a **stack** S_ℓ of intervals of 1's
 - (2) With each interval (i, j) in S_ℓ store a pointer to the $j + 1$ st ancestor of ℓ
- ⇒ in worst-case $O(1)$ time S_ℓ can be updated and the d th ancestor found



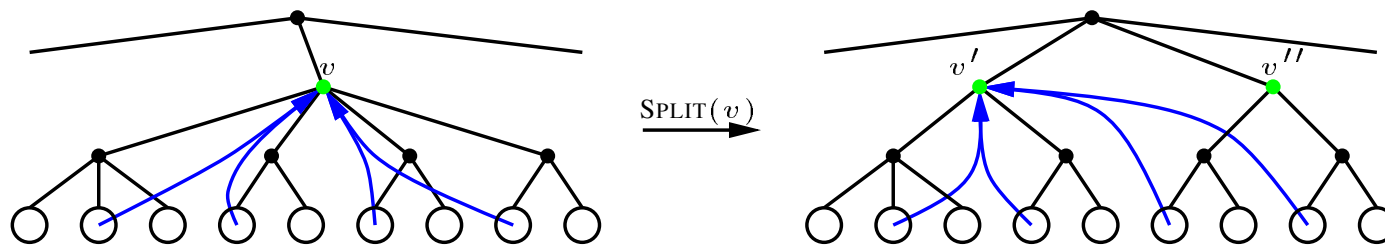
Copying S_ℓ stacks in $O(1)$ time

Let the S_ℓ stacks be **functionally** implemented

$\Rightarrow S_{\ell'} := S_\ell$ takes worst-case $O(1)$ time

Pointers to wrong subtrees

Splitting a node v can make S_ℓ stacks contain pointers to wrong subtrees !
But the algorithm still works — nodes of height d have **degree** $\leq 2^3 \cdot 2^d \Delta_d$



Conclusion and open problems

Theorem

A pointer based implementation of finger search trees exists with worst-case time bounds

Insert : $O(1)$

Search : $O(\log \delta)$

Delete : $O(\log^* n)$

The data structure requires linear space

Open problems

- **DELETE** in worst-case $O(1)$ time too
- Make other splitting based data structures worst-case
e.g. the full persistence technique of [Driscoll *et al.* '89](#).