Finger Search
Searching in a sorted array

2 3 5 7 8 11 13 14 15 17 18 20 24 25 26 28 29 31 33 34

Finger

Binary-search(13)

time $O(\log n)$

Exponential-search(13)

time $O(\log d)$

$2^0$ $2^1$ $2^2$
# Dynamic Finger Search

<table>
<thead>
<tr>
<th></th>
<th>Search</th>
<th>Insert/Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>No fingers</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Red-black, AVL, 2-4-trees, ...</td>
<td>O(log (n))</td>
<td>O(log (n))</td>
</tr>
<tr>
<td><strong>O(1) fixed fingers</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Guibas et al. 1977, ....</td>
<td>O(log (d))</td>
<td>O(1)</td>
</tr>
<tr>
<td><strong>Each node a finger</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Level-linked (2,4)-trees</td>
<td>O(log (d))</td>
<td></td>
</tr>
<tr>
<td>Randomized Skip lists</td>
<td>O(log (d)) exp.</td>
<td>O(1) exp.</td>
</tr>
<tr>
<td>Treaps</td>
<td>O(log (d)) exp.</td>
<td>O(1) exp.</td>
</tr>
<tr>
<td>Brodal, Lagogiannis, Makris,</td>
<td>O(log (d))</td>
<td>O(1)</td>
</tr>
<tr>
<td>Tsakalidis, Tsichlas 2003</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Level-Linked (2,4)-trees


Updates  Split nodes of degree >4, fusion nodes of degree <2
Search   Search up + top-down search

Potential $\Phi = 2 \cdot \# \text{ degree-4} + \# \text{ degree-2}$
Randomized Skip Lists


**Insertion**  Increase pile to next level with pr. = 1/2

**Height**    O(log n) expected with high probability

**Pointer**   Horizontally spans O(1) exp. piles one level below

**Finger**    Remember nodes on search path
Treaps – Randomized Binary Search Trees


- Each element random priority
- Search tree wrt element
- Heap order wrt priority
- Height $O(\log n)$ expected
- Insert & deletion rotations $O(1)$ expected time
- **Search** Go up to LCA, and search down – concurrently follow excess path to find next LCA candidate
  Search path $O(\log d)$ expected
Application: Binary Merging


- Merging sorted lists $L_1$ and $L_2$ / finger search trees

  \[
  \sum \log(d_i) = |L_1| \log \left( \frac{|L_2| + |L_1|}{|L_1|} \right)
  \]

- Merging leaf lists in an arbitrary binary tree $O(n \cdot \log n)$

  **Proof** Induction $O(\log n!)$

  
  \[
  O(\log n_1! + \log n_2! + n_1 \cdot \log ((n_1 + n_2)/n_1)) \\
  = O(\log n_1! + \log n_2! + n_1 \cdot \log \left( \frac{n_1 + n_2}{n_1} \right)) \\
  = O(\log n_1! + \log n_2! + (n_1 + n_2) \cdot \log (n_1 + n_2) - \log n_1! - \log n_2!) = O(\log (n_1 + n_2)!) 
  \]
Maximal Pairs with Bounded Gap


- Build suffix tree (ST) & make it binary
- Create leaf lists at each node
- Right-maximal pairs = ST nodes
- Find maximal pairs = finger search at ST nodes

$O(n \cdot \log n + k)$