Today: Integers & van Emde Boas
- models: word RAM & cell probe
- predecessor problem
- van Emde Boas DS
- y-fast trees

Models for integer data structures:
- word = w-bit integer \( \in \{0, 1, \ldots, w-1\} \)
  * all elements: inputs, outputs, ...
- transdichotomous RAM (Random Access Machine):
  - memory = array of \( S \) words
  - operations read/write \( O(1) \) words
  - words serve as pointers

\( \Rightarrow w \geq \log S \)
- in particular \( w \geq \log n \)
- word RAM: transdichotomous RAM
  with C-style operations: [], +, -, *, /, %, <, >, &,
  standard model
- cell probe: count # memory reads & writes
- computation is free
- unrealistic
- useful for lower bounds

\[ 2^w \]

bridging two worlds: machine/problem

\[ \text{BST} \]

\[ \text{transdichotomous RAM} \]

\[ \text{word RAM} \]

\[ \text{cell probe} \]

\[ \text{pointer machine} \]

\[ \text{STRONG} \]

\[ \text{WEAK} \]
Predecessor problem: maintain set \( S \) of \( n \) words subject to:
- insert \((x \in U)\)
- delete \((x \in S)\)
- \( \text{predecessor}(x \in U) \): max\{yes | y < x\}
- \( \text{successor}(x \in U) \): min\{yes | y > x\}

- harder than dictionaries/hashing
- comparison model \( \Rightarrow \) BST: \( \Theta(lg n) \)/op. optimal

- word RAM:
  - van Emde Boas: \( O(lg w) \)/op. \( \Theta(w) \) space
  - \([\text{FOCS} 1975; \text{IPL} 1977]\)
    \( \Theta(lg w) \) w.h.p. \( \Theta(n) \) space
  - \( y \)-fast trees: \( O(lg w) \) w.h.p. \( \Theta(n) \) space
    \([\text{Willard- IPL} 1983]\)

- fusion trees:
  - \( O(lg \sqrt{n}) \) w.h.p. \( \Theta(n) \) space
    \([\text{Fredman & Willard-JCSS 1993}; \text{Raman-ESA 1996}\)]

- \( \min: \) \( O(\sqrt{lg n}) \) w.h.p. \( \Theta(n) \) space

- cell probe lower bound: \( \Omega(\min\{lg w, \frac{lg lg n}{lg lg w}\}) \)
  \([\text{Pătraşcu & Thorup - STOC 2006 & SODA 2007}]\)

\( \Rightarrow \) vEB optimal for \( w = O(poly lg n) \) & fusion trees optimal for \( w = 2^{\Omega(lg n)} \)

- pointer machine, word specified by pointer:
  - van Emde Boas: \( O(lg lg w) \)/op. \( \Theta(w) \) space
  - lower bound: \( \Omega(lg lg w) \)/op. \( \Omega(w) \) space

\([\text{Mehlhorn, Näher, Alt - SICOMP 1988}]\)
van Emde Boas: (Peter) (reinterpreted by Bender & Farach-Colton)

- idea: \( T(u) = T(\sqrt{u}) + O(1) \)
  \( = O(\log \log u) \)

- split universe \( \mathcal{U} \) into \( \sqrt{u} \) clusters, each size \( \sqrt{u} \)

- hierarchical coordinates: word \( x = \langle c, i \rangle \)
  - \( c = x \mod \sqrt{u} = \) cluster containing \( x \)
  - \( i = x \div \sqrt{u} = x's \) index within cluster

- integer division \( \mod \)
  \(- x = c \sqrt{u} + i \Rightarrow O(1)-time \) conversion

- binary perspective:
  - split bits in half
  - \( c = \) high order \( = x \gg w/2 \)
  - \( i = \) low order \( = x \& (1 << w/2) - 1 \)
  - \( x = (c << w/2) | i \)

- recursive vEB \( V \) of size \( u \):
  - \( V.cluster[i] = \text{vEB of size } \sqrt{u} \) for \( 0 \leq i < \sqrt{u} \)
  - \( V.summary = \text{vEB of size } \sqrt{u} \& w' = w/2 \)
    - stores which clusters \( c \) are nonempty
  - \( V.min = \) minimum element in \( V \), not stored recursively
    \( \text{OR None if } V \text{ is empty} \)
  - \( V.max = \) (copy of) max. element in \( V \)
**Successor** \((V, x=\langle c, i \rangle)\):
- if \(x < V\).min: return \(V\).min  
- if \(i < V\).cluster[\(c\)].max:
  return \(\langle c, \text{Successor}(V\text{.cluster}[c], i) \rangle\)
- else: \(c' = \text{Successor}(V\text{.summary}, c)\)
  return \(\langle c', V\text{.cluster}[c']\text{.min} \rangle\)

\(O(1)\)

**Insert** \((V, x=\langle c, i \rangle)\):
- if \(V\).min = None: \(V\).min = \(V\).max = \(x\); return
- if \(x < V\).min: swap \(x \leftrightarrow V\).min
- if \(x > V\).max: \(V\).max = \(x\)
- if \(V\).cluster[\(c\)].min = None:
  Insert \((V\text{.summary}, c)\) \(\Rightarrow\) next call is \(O(1)\)
- Insert \((V\text{.cluster}[c], i)\)

**Delete** \((V, x=\langle c, i \rangle)\):
- if \(x = V\).min:
  - \(c = V\text{.summary}\).min
  - if \(c = \text{None}\): \(V\).min = \(\text{None}\); return \((\text{now empty})\)
  - \(x = V\).min = \(\langle c, i = V\text{.cluster}[c]\text{.min} \rangle\)
  Delete \((V\text{.cluster}[c], i)\)
- if \(V\).cluster[\(c\)].min = None: \(\text{(empty now)}\)
  Delete \((V\text{.summary}, c)\) \(\Rightarrow\) previous call \(O(1)\)
- if \(V\).summary.min = None: \(V\).max = \(V\).min
- else: \(c' = V\text{.summary}\).max
  \(V\).max = \(\langle c', V\text{.cluster}[c']\text{.max} \rangle\)
Tree view: expand recursion

Summary

Summary bits

Clusters

Bit vector

- node = OR of children
- path from leaf \( x \) to root is monotone
- could binary search for 0\(\rightarrow\)1 transition
- max/min of last 0's left/right sibling is predecessor/successor of \( x \) (if \( \neq 5 \))
- store sorted linked list on elements to find successor/predecessor
- query in \( O(\lg \lg \lg u) \) ~ roughly same as above

- even in pointer machine & \( O(u \lg \lg \lg u) \) space: node stores linked list of pointers to ancestor of height \( 2^i \) for \( i = 0, 1, ..., \lg u \)

- but updating these bits costs \( \Theta(\lg u) / \text{op.} \)
- vEB's not-storing-min reduces to \( \Theta(\lg w) \)
- again possible on pointer machine with \( O(u \lg \lg u) \) space [vEBKZ77]
Indirection: (trick from [Willard - IPL 1983])
- take $O(\log w)$ query, $O(w)$ update DS such as "simple" tree above
- reduce update to $O(\log w)$:
  split $n$ elements into chunks of size $O(w)$

\[
\begin{array}{c}
\Theta(n/w) \\
\Theta(w) \quad \Theta(w) \ldots \quad \Theta(w)
\end{array}
\] \Rightarrow \begin{array}{c}
\{ O(\log w) \text{ query} \\
\{ O(w) \text{ update} \\
\{ O(\log w) \text{ via BSTs}
\end{array}

- query: query top $\rightarrow O(\log w)$
  query bottom $\rightarrow O(\log w)$
- update: update bottom $\rightarrow O(\log w)$
  split & possibly merge with neighbor to keep chunks $O(w)$ size
  $\Rightarrow$ update top $\rightarrow O(w)$, charged to $O(w)$ updates in chunk
  $\Rightarrow O(\log w)$ query & amortized update

- top structure can actually use $u' = u/\Theta(w)$: bottoms can guarantee separation $\Omega(w)$ between representatives
  $\Rightarrow \Theta(u)$ space $\sim$ on pointer machine!
- similar trick, splitting $u$ directly instead of $n$, applied to stratified trees in [VEB-IPL1977]
Saving space: [Vladimír Čunát – S.M. thesis 2011?]
- don’t store empty clusters in vEB
  \[\Rightarrow V.\text{clusters} = \text{hash table}\]
- \(\Theta(1)\) w.h.p. e.g. via dynamic perfect hashing
- space = \(O(\# \text{nonempty “child” clusters} + 1)\)
- charge each table entry to \(\min\) in \(\text{child}\)
- Insert cuts up element into \(O(\log w)\) \(\min\) fields
  \[\Rightarrow O(n \log w)\text{ space}\]
  - tight in worst case \(\Rightarrow \text{not} \ O(n)\!\)!
- \(O(n)\) space via indirection as above

\(\alpha\)-fast trees: [Willard – IPL 1983]
- don’t store \(\emptyset\)s in simple tree view
- store hash table of root-to-1 paths
  \[\text{per length viewed in binary; } 0 = \text{left}, 1 = \text{right}\]
  i.e. prefixes of elements in \(S\)
- \(O(\log w)\) query via binary search as before
- \(\Theta(w)\) update as before
- \(\Theta(nw)\) space

\(\gamma\)-fast trees: [Willard – IPL 1983]
= \(\alpha\)-fast trees
+ indirection as above
- \(O(\log w)\) query still
- \(O(\log w)\) amortized update
- \(O(n)\) space