Lecture 13

Fusion trees

[Frederman & Willard—JCSS 1993]

\(O(\log_w n)\) time for predecessor/successor
\(O(n)\) space — here, static

[— dynamic via exponential trees:] [Andersson & Thorup—ArXiv:cs.DS/0210006]

\(O(\log_w n + \log \log n)\)

Top-level idea:

— B-tree with branching factor \(\Theta(w^{1/5})\)

\[ \Rightarrow \text{height} = \Theta(\log_w n) = \Theta\left(\frac{\log n}{\log w}\right) \]

— search must visit each node in \(O(1)\) time
— need to find correct branch
— not enough time to read node \((\Theta(w^{1/5})\text{ words})\)

Fusion-tree node:

given \(k\) keys \(x_0 < x_1 < \cdots < x_{k-1}\) \(k = O(w^{1/5})\)
preprocess in \(kO(1)\) time
subject to predecessor/successor in \(O(1)\) time

\[ \text{Note: } \min \{ \log_w n, \frac{\log \log n}{\log w} \} \leq \sqrt{\log n} \]

\text{fusion van Emde Boas}
Distinguishing $k = O(w^{1/5})$ keys:
- View keys $x_0, x_1, \ldots, x_{k-1}$ as binary strings (0/1)
  i.e. root-to-leaf paths in height-$w$ binary tree (left/right)
  $\Rightarrow k-1$ branching nodes
  $\Rightarrow \leq k-1$ levels containing branching nodes
  = bits where $x_0, x_1, \ldots, x_{k-1}$ first differ (first distinct prefix)
  $\Rightarrow$ call these important bits $b_0 < b_1 < \cdots < b_{r-1}$, $r < k = O(w^{1/5})$

**Perfect Sketch** $\text{sketch}(x) = \text{extract bits } b_0, b_1, \ldots, b_r$ from word $x$
  i.e. $r$-bit vector whose $i$th entry = $b_i$th bit of $x$
  $\Rightarrow$ sketch($x_0$) $< \cdots <$ sketch($x_k$) $\sim$ all different & same order
  & all packable into a word: $k \cdot r = O(w^{2/5})$ bits
  $\Rightarrow$ computable in $O(1)$ time as an $AC^0$ operation
  
  [Andersson, Miltersen, Thorup - TCS 1999]

**Idea:** for query $q$, compare sketch($q$) in parallel to sketch($x_0$), $\ldots$, sketch($x_{k-1}$)
  $\sim$ again $AC^0$ operation on $O(1)$ words
  $\Rightarrow$ find where sketch($q$) fits (predecessor/successor) among sketch($x_0$) $< \cdots <$ sketch($x_{k-1}$)
Desketchifying:

\[ \text{Sketch: } 00 \quad 01 \quad 00 \quad 10 \quad 11 \]

- Suppose \( \text{sketch}(x_i) \leq \text{sketch}(q) < \text{sketch}(x_{i+1}) \)
- Compute longest common prefix = lowest common ancestor between \( q \) and \( (x_i \text{ or } x_{i+1}) \) — take longest/lowest

\[ = \text{node } y \text{ where } q \text{ fell off the } \boxed{\text{right paths}} \text{ to } x_i's \]

\[ \Rightarrow \text{correct direction/subtree} = \begin{cases} y1 & \text{if } (y+1)\text{st bit of } q = 0 \\ y0 & \text{if } (y+1)\text{st bit of } q = 1 \end{cases} \]

- Extreme \( e \) in subtree closest to \( q = \begin{cases} y100\ldots0 \\ y011\ldots1 \end{cases} \)

- Predecessor & successor of \( q \) among \( x_i's \)
- Predecessor & successor of sketch(\( e \)) among sketch(\( x_i's \))

(desketchified)
Parallel comparison:
- sketch(node) = 1 sketch(x₀) 1 sketch(x₁) ... 1 sketch(x_{k-1})
- sketch(q)^k = 0 sketch(q) 0 sketch(q) ... 0 sketch(q)
  = sketch(q) ∙ 000001 000001 ... 000001
- difference = (\frac{1}{\delta})***** (\frac{1}{\delta})***** ... (\frac{1}{\delta})*****
- AND with 100000 100000 ... 100000
  \rightarrow (\frac{1}{\delta})00000 (\frac{1}{\delta})00000 ... (\frac{1}{\delta})00000

Idea: extra 1’s in sketch(node) protect from underflow
- remains 1 if sketch(q) ≤ sketch(xᵢ)
- becomes 0 if sketch(q) > sketch(xᵢ)
\Rightarrow (\frac{1}{\delta}) bits are 0 0 0 0 1 1 1 1
where sketch(q) fits \uparrow \text{most significant 1 bit}

Index of most significant 1 bit: 00010110 \Rightarrow 4
76543210
- AC⁰ operation [Andersson, Miltersen, Thorup - TCS 1999]
- instruction on many real CPUs e.g. Pentiums
  (see Linux kernel: include/asm-*/*bitops.h)
- possible with O(1) arithmetic & Boolean ops. [Brodin 1993]
- easy solution with extra space:
  - lookup table on all strings of ε·w bits
    \Rightarrow O(2^{ε·w} \log w) bits = O(ε·\log w) bits of space
  - query: O(1/ε) [or O(\log (1/ε))] probes to table
\underline{- note for next time: cover Fredman & Willard's clever solution}
Approximate sketch \( x \) \( \Rightarrow \) word RAM
- don't need sketch to pack \( b_i \) bits right next to each other
- can be spread out in predictable pattern of length \( O(w^{4/5}) \) independent of \( x \)

Idea: mask important bits: \( x' = x \text{ AND } \sum_{i=0}^{r-1} 2^{b_i} \)
& multiply \( x'.m = \left( \sum_{i=0}^{r-1} x_{b_i} 2^{b_i} \right) \left( \sum_{j=0}^{r-1} 2^{m_j} \right) \)
\[ = \sum_{i=0}^{r-1} \sum_{j=0}^{r-1} x_{b_i} 2^{b_i + m_j} \]

Claim: for any \( b_0, b_1, \ldots, b_{r-1} \), can choose \( m_0, m_1, \ldots, m_{r-1} \) such that \( b_i + m_j \) are all distinct \( \Rightarrow \) no collision

- \( b_0 + m_0 < b_1 + m_1 < \ldots < b_{r-1} + m_{r-1} \Rightarrow \) preserve order
- \( (b_{r-1} + m_{r-1}) - (b_0 + m_0) = O(r^4) = O(w^{4/5}) \Rightarrow \) small span

\[ \Rightarrow \text{approx-sketch}(x) = \left( (x.m) \text{ AND } \sum_{i=0}^{r-1} 2^{b_i + m_i} \right) \sum_{i=0}^{r-1} (b_i + m_i) \]

Proof: ① choose \( m'_0, m'_1, \ldots, m'_{r-1} < r^3 \) such that \( b_i + m'_j \) are all distinct \( \text{modulo } r^3 \) (strong @)
- pick \( m'_0, m'_1, \ldots, m'_{r-1} \) by induction
- \( m'_t \) must avoid \( m'_t + b_j - b_k \) \( \forall i, j, k \)

\[ \Rightarrow \text{choice for } m'_t \text{ exists} \]

② let \( m_i = m'_i + (w - b_i + ir^3) \text{ rounded down to multiple of } r^3 \)
\[ \equiv m'_i \text{ (mod } r^3) \]
\[ \Rightarrow m_i + b_i \text{ in } r^3 \text{ interval after } \left( \frac{w}{r^3} + i \right) \text{th multiple of } r^3 \]
\[ \Rightarrow m_0 + b_0 < m_1 + b_1 < \ldots < m_{r-1} + b_{r-1} \]
\[ \approx w \]
\[ \approx w + r^4 \Rightarrow \text{difference } = O(r^4) \]