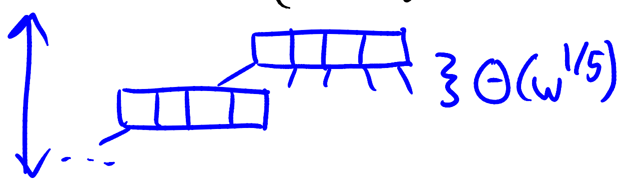
Lecture 13

Fusion trees [Fredman & 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]
$O(\log_w n + \lg\lg n)$]

Top-level idea:

— B-tree with branching factor $\Theta(w^{1/5})$
$\Rightarrow$ height $= \Theta(\log_w n)$
$= \Theta\left(\frac{\lg n}{\lg w}\right)$



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

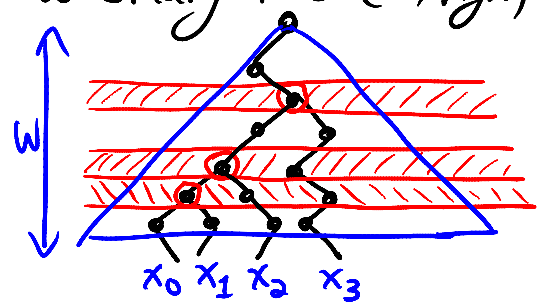
Fusion-tree node:

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

$$\boxed{\text{Note: } \min\{\underbrace{\log_w n}_{\text{fusion}}, \underbrace{\lg w}_{\text{van Emde Boas}}\} \le \sqrt{\lg n}}$$

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)

- call these important bits $b_0 < b_1 < \cdots < b_{r-1}$, $r < k = O(w^{1/5})$

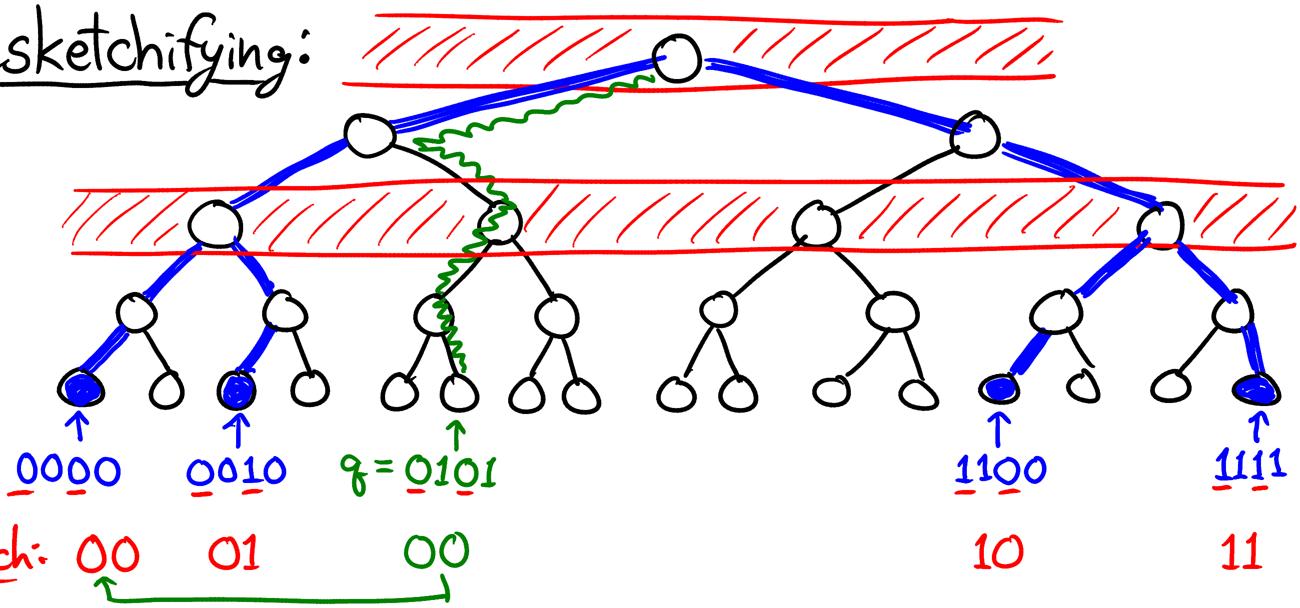
(perfect-)sketch(x) = extract bits $b_0, b_1, \ldots, b_r$ from word x
i.e. r-bit vector whose ith entry = b_ith bit of x
$\Rightarrow$ sketch(x_0) $< \cdots <$ sketch(x_k) ~ all different & same order
& all packable into a word: $k \cdot r = O(w^{2/5})$ bits

- 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), …, sketch(x_{k-1})

- 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:



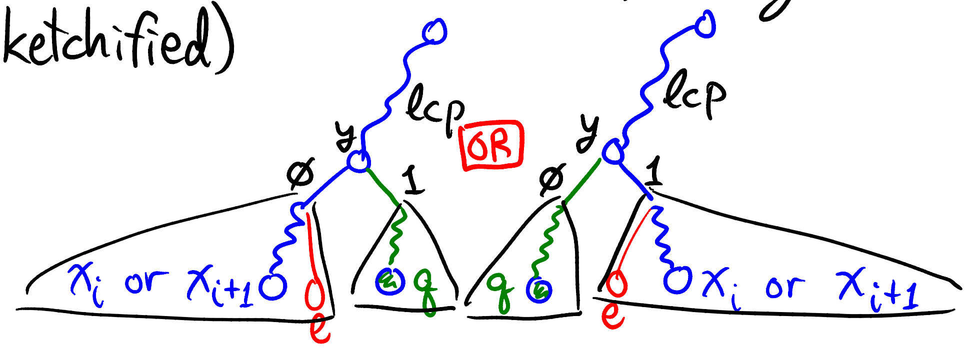
- suppose $\text{sketch}(x_i) \le \text{sketch}(q) < \text{sketch}(x_{i+1})$
- compute longest common prefix = lowest common ancestor between q and (x_i or x_{i+1}) — take longest/lowest
  nonsketch

= node y where q fell off the "right paths" to x_i's

$\Rightarrow$ 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 $= \begin{cases} y100\cdots0 \\ y011\cdots1 \end{cases}$

- predecessor & successor of q among x_i's
= predecessor & successor of sketch(e) among sketch(x_i)'s
(desketchified)



Parallel comparison:

- $\text{sketch(node)} = 1\,\text{sketch}(x_0)\,1\,\text{sketch}(x_1) \cdots 1\,\text{sketch}(x_{k-1})$
- $\text{sketch}(q)^k = 0\,\text{sketch}(q)\;0\,\text{sketch}(q) \cdots 0\,\text{sketch}(q)$

  $= \text{sketch}(q) \cdot 000001\;000001 \cdots 000001$
- difference $= \binom{1}{0}{*}{*}{*}{*}{*}\;\binom{1}{0}{*}{*}{*}{*}{*} \cdots \binom{1}{0}{*}{*}{*}{*}{*}$
- AND with $100000\;100000 \cdots 100000$

  $\rightarrow \binom{1}{0}00000\;\binom{1}{0}00000 \cdots \binom{1}{0}00000$

Idea: extra 1's in sketch(node) protect from underflow

- remains 1 if $\text{sketch}(q) \le \text{sketch}(x_i)$
- becomes 0 if $\text{sketch}(q) > \text{sketch}(x_i)$

$\Rightarrow \binom{1}{0}$ bits are 0 0 0 0 1 1 1 1

where sketch(q) fits → ← most significant 1 bit

Index of most significant 1 bit: $000\boxed{1}0110 \mapsto 4$

$\quad\;\; 7\,6\,5\,\boxed{4}\,3\,2\,1\,0$

- AC^0 operation [Andersson, Miltersen, Thorup – TCS 1999]
- instruction on many real CPUs e.g. Pentiums

  (see Linux kernel: include/asm-*/bitops.h)
- LSB possible with $O(1)$ arithmetic & Boolean ops. [Brodnik 1993]
- easy solution with extra space:
  - lookup table on all strings of $\varepsilon \cdot w$ bits

    $\Rightarrow O(2^{\varepsilon \cdot w} \lg w)$ bits $= O(u^{\varepsilon} \lg\lg u)$ bits of space
  - query: $O(1/\varepsilon)$ [or $O(\lg(1/\varepsilon))$] probes to table
- note for next time: cover Fredman & Willard's clever solution

Approximate sketch(x) $\rightsquigarrow$ 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 $x' \cdot m = \left(\sum_{i=0}^{r-1} x_{b_i} 2^{b_i}\right) \cdot \left(\sum_{i=0}^{r-1} 2^{m_j}\right)$

$$= \sum_{i=0}^{r-1} \sum_{j=0}^{r-1} x_{b_i} 2^{b_i + m_j}$$

Claim: for any $b_0, b_1, \dots, b_{r-1}$, can choose $m_0, m_1, \dots, m_{r-1}$ such that

(a) $b_i + m_j$ are all distinct $\Rightarrow$ no collision

(b) $b_0 + m_0 < b_1 + m_1 < \dots < b_{r-1} + m_{r-1}$ $\Rightarrow$ preserve order

(c) $(b_{r-1} + m_{r-1}) - (b_0 + m_0) = O(r^4) = O(w^{4/5})$ $\Rightarrow$ small span

$$\Rightarrow \text{approx-sketch}(x) = \Big((x \cdot m) \underbrace{\text{AND } \sum_{i=0}^{r-1} 2^{b_i + m_i}}_{\text{discard } i \neq j}\Big) \gg (b_0 + m_0)$$

Proof: (1) choose $m'_0, m'_1, \dots, m'_{r-1} < r^3$ such that $b_i + m'_j$ are all distinct modulo r^3 (strong (a))

- pick $m'_0, m'_1, \dots, m'_{t-1}$ by induction
- m'_t must avoid $\underbrace{m'_i}_{t} + \underbrace{b_j}_{r} - \underbrace{b_k}_{r}$ $\forall i, j, k$ $\Rightarrow tr^2 < r^3$ choices

$\Rightarrow$ choice for m'_t exists

(2) let $m_i = m'_i + (w - b_i + i r^3$ rounded down to multiple of $r^3)$

$\equiv m'_i \pmod{r^3}$

$\Rightarrow m_i + b_i$ in r^3 interval after $\left(\frac{w}{r^2} + i\right)$th multiple of r^3

$\Rightarrow \underbrace{m_0 + b_0}_{\approx w} < m_1 + b_1 < \dots < \underbrace{m_{r-1} + b_{r-1}}_{\approx w + r^4}$ $\Rightarrow$ difference $= O(r^4)$