

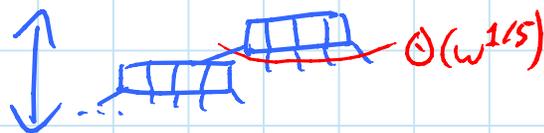
Fusion trees: [Fredman & Willard - JCSS 1993]

- store n w -bit integers - here, statically
- $O(\log_w n)$ time for predecessor/successor
- $O(n)$ space
- dynamic version via exponential trees:
 $O(\log_w n + \lg \lg n)$ updates [Andersson & Thorup
 - JACM 2007]

Consequence: $\min \{ \underbrace{\log_w n}_{\text{fusion}}, \underbrace{\lg w}_{\text{van Emde Boas}} \} \leq \sqrt{\lg n}$
 upper bound for predecessor problem

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

$$\Rightarrow \text{height} = \Theta(\log_w n) \\ = \Theta(\lg n / \lg w)$$



- search must visit a node in $O(1)$ time
- not enough time to read the node
 $(w^{1/5}$ w -bit words) to figure out which child

Fusion-tree node:

- store $k = O(w^{1/5})$ keys $x_0 < x_1 < \dots < x_{k-1}$
- $O(1)$ time for predecessor/successor
- $k^{O(1)}$ preprocessing

Distinguishing $k = O(w^{1/5})$ keys:

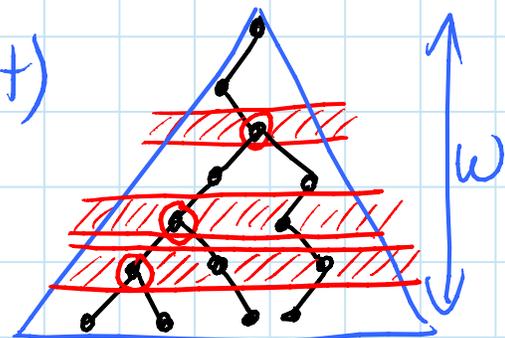
- view keys x_0, x_1, \dots, x_{k-1} as binary strings (0/1)
i.e. root-to-leaf paths in
height- w binary tree (left/right)

\Rightarrow $k-1$ branching nodes \odot

$\Rightarrow \leq k-1$ levels |||||

containing branching nodes

i.e. bits where x_0, x_1, \dots, x_{k-1} first differ
(first distinct prefix)



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

(perfect) sketch(x) = extract bits b_0, b_1, \dots, b_{r-1} from x
i.e. r -bit vector whose i th bit = b_i th bit of word x

\Rightarrow $\text{sketch}(x_0) < \text{sketch}(x_1) < \dots < \text{sketch}(x_{k-1})$

& can pack (fuse) into one word: $k \cdot r = O(w^{2/5})$ bits

- computable in $O(1)$ time as AC^0 operation

[Andersson, Miltersen, Thorup - TCS 1999]

- we'll see a cool way to compute approximate sketch using multiplication & standard ops.

Node search: for query q , compare $\text{sketch}(q)$

in parallel to $\text{sketch}(x_0), \dots, \text{sketch}(x_{k-1})$

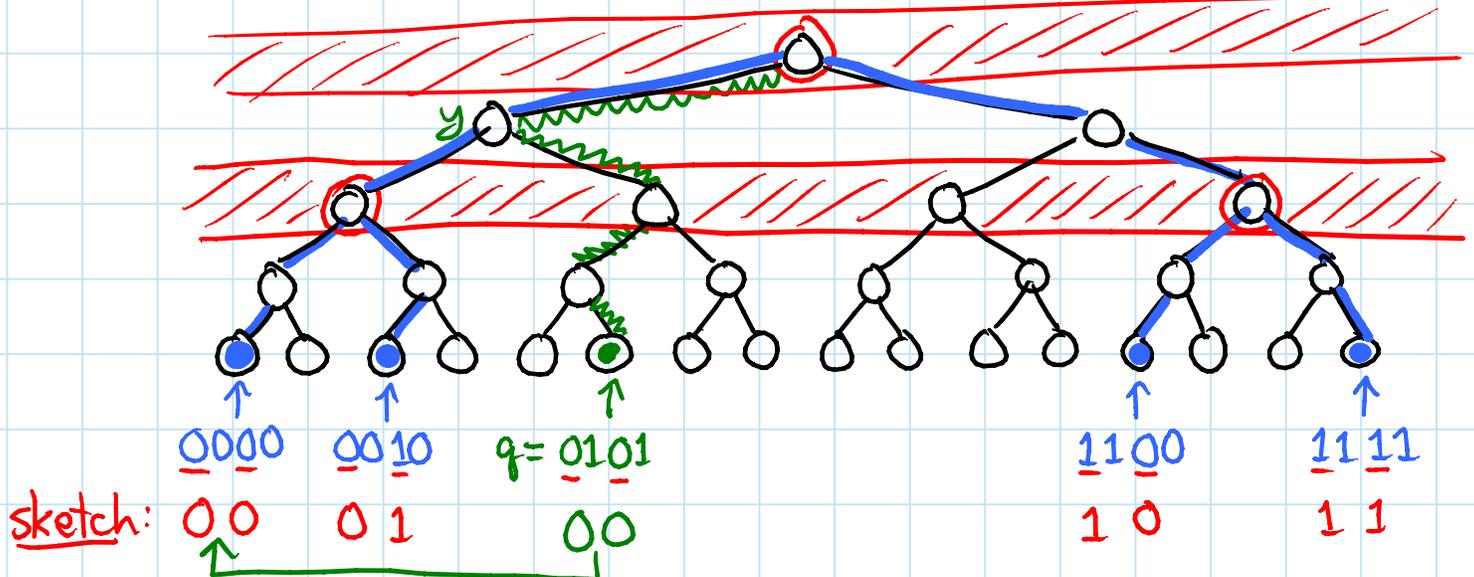
- again AC^0 operation on $O(1)$ words

& we'll see a nice way with standard ops.

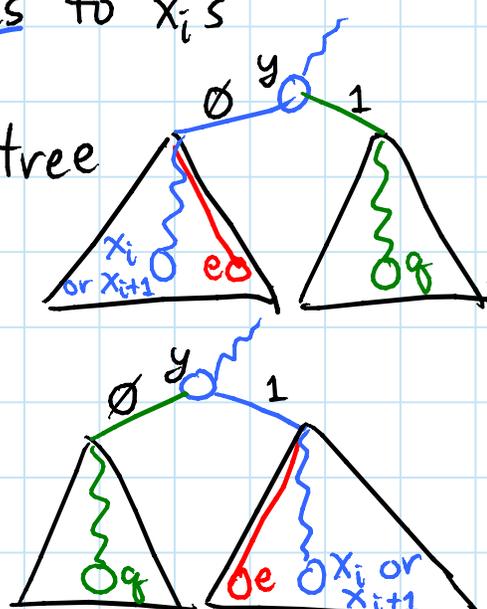
\Rightarrow find where $\text{sketch}(q)$ fits among $\text{sketch}(x_0) < \dots < \text{sketch}(x_{k-1})$

- want where q fits among $x_0 < \dots < x_{k-1}$

Desketchifying:



- suppose $\text{sketch}(x_i) \leq \text{sketch}(q) < \text{sketch}(x_{i+1})$
- longest common prefix = lowest common ancestor between q & (either x_i or x_{i+1})
- nonsketch* \rightarrow *whichever's longest/lowest*
- = node y where q fell off paths to x_i 's
- if $|y|+1$ st bit of q is 1:
 - nearest x_i is in $y0$ subtree
 - nearest extreme in that subtree is $e = y011\dots 1$



- else: $e = y100\dots 0$

- predecessor & successor of q among x_i 's
- = predecessor & successor of $\text{sketch}(e)$ among $\text{sketch}(x_i)$'s (desketchified via sketch^{-1})

Approximate sketch(x): on word RAM

- don't need sketch to pack b_i bits consecutively
- can 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 $x' \cdot m = \left(\sum_{i=0}^{r-1} x_{b_i} 2^{b_i} \right) \cdot \left(\sum_{j=0}^{r-1} 2^{m_j} \right)$
$$= \sum_{i=0}^{r-1} \sum_{j=0}^{r-1} x_{b_i} 2^{b_i + m_j}$$

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

- (a) $b_i + m_j$ are all distinct (no collision)
- (b) $b_0 + m_0 < \dots < b_{r-1} + m_{r-1}$ (preserve order)
- (c) $(b_{r-1} + m_{r-1}) - (b_0 + m_0) = O(r^4) = O(w^{4/5})$ (small)

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

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 $\underbrace{m'_i + b_j}_{t} - \underbrace{b_k}_{r} \forall i, j, k$
 $\Rightarrow tr^2 < r^3$ choices

\Rightarrow choice for m'_t exists

② let $m_i = m'_i + \underbrace{(w - b_i + i r^3)}_{\text{to make nonnegative}}$ rounded down to mult. of r^3
 $\equiv m'_i \pmod{r^3}$

$\Rightarrow m_i + b_i$ in r^3 interval after $(\lfloor \frac{w}{r^3} \rfloor + i) \cdot r^3$

$\Rightarrow \underbrace{m_0 + b_0}_{\approx w} < \dots < \underbrace{m_{r-1} + b_{r-1}}_{\approx w + r^4} \Rightarrow \text{diff.} = O(r^4)$ (b) (c) \square

Parallel comparison: → protect from underflow

- sketch(node) = $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ sketch(x₀) ... 1 sketch(x_{k-1})

- sketch(q)^k = $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$ sketch(q) ... $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$ sketch(q)
 = sketch(q) · $\begin{pmatrix} 0 & 00001 & \dots & 0 & 00001 \end{pmatrix}$

- difference = $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ ***** ... $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ *****

- AND with $\begin{pmatrix} 1 & 00000 & \dots & 1 & 00000 \end{pmatrix}$

→ $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ 00000 ... $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ 00000

$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ if sketch(q) ≤ sketch(x_i)
 $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$ if sketch(q) > sketch(x_i)

⇒ these bits look like 0000111
 where sketch(q) fits ↗ ↖

need index of most sig. 1 bit

- multiply with $\begin{pmatrix} 0 & 00001 & \dots & 0 & 00001 \end{pmatrix}$

→ #1's #1's to right last 1
 desired

- or if you prefer:

Index of most significant 1 bit: 00010110 ⇨ 4
 76543210

- AC⁰ operation [Andersson, Miltersen, Thorup 1999]

- instruction on most modern CPUs

(see Linux kernel: include/asm-*/bitops.h)

- computable in O(1) using fusion tricks

[Fredman & Willard 1993]