6.851	Lecture 12	Apr. 3, 2012

TODAY: Fusion trees

- sketch & why it's enough
- approximate sketch via multiplication
- parallel comparison
- most significant set bit

1 year after "cold fusion" debacle

Fusion trees: [Fredman & Willard – STOC 1990 / JCSS 1993]

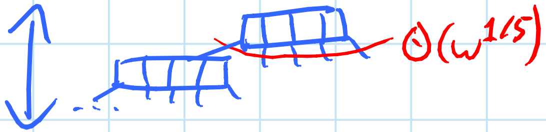
- store n w-bit integers – here, statically
- $O(\log_w n)$ time for predecessor/successor
- $O(n)$ space
- word RAM

$$\boxed{\Rightarrow \text{predecessor} \le \min\{\underbrace{\log_w n}_{\text{fusion}}, \underbrace{\lg w}_{\text{vEB}}\} \le \sqrt{\lg n}}$$

- AC^0 RAM version [Andersson, Miltersen, Thorup – TCS 1999]
  ↳ ops. are constant-depth (unbounded fan) circuits
  ⇒ no multiplication
- dynamic version via exponential trees:
  $O(\log_w n + \lg\lg n)$ deterministic updates
  [Andersson & Thorup – JACM 2007]
- dynamic version via hashing: [Raman – ESA 1996]
  $O(\log_w n)$ expected updates
- **OPEN**: $O(\log_w n)$ w.h.p. updates?

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

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



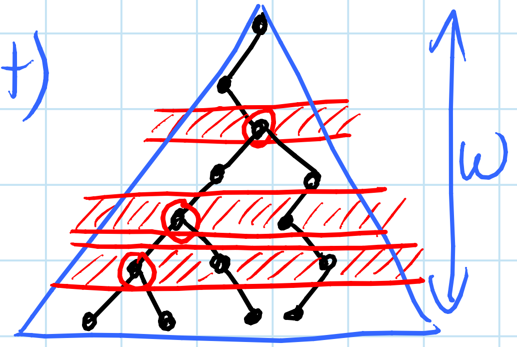
- search must visit a node in $O(1)$ time
- not enough time to read the node ($w^{1/5}$ w-bit words) to figure out which child

Fusion-tree node:

- store $k = O(w^{1/5})$ keys $x_0 < x_1 < \cdots < x_{k-1}$
- $O(1)$ 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)



⇒ $k-1$ branching nodes ⊙

⇒ $\le k-1$ levels ▨ containing branching nodes
  i.e. bits where $x_0, x_1, \dots, x_{k-1}$ first differ
  (first distinct prefix)

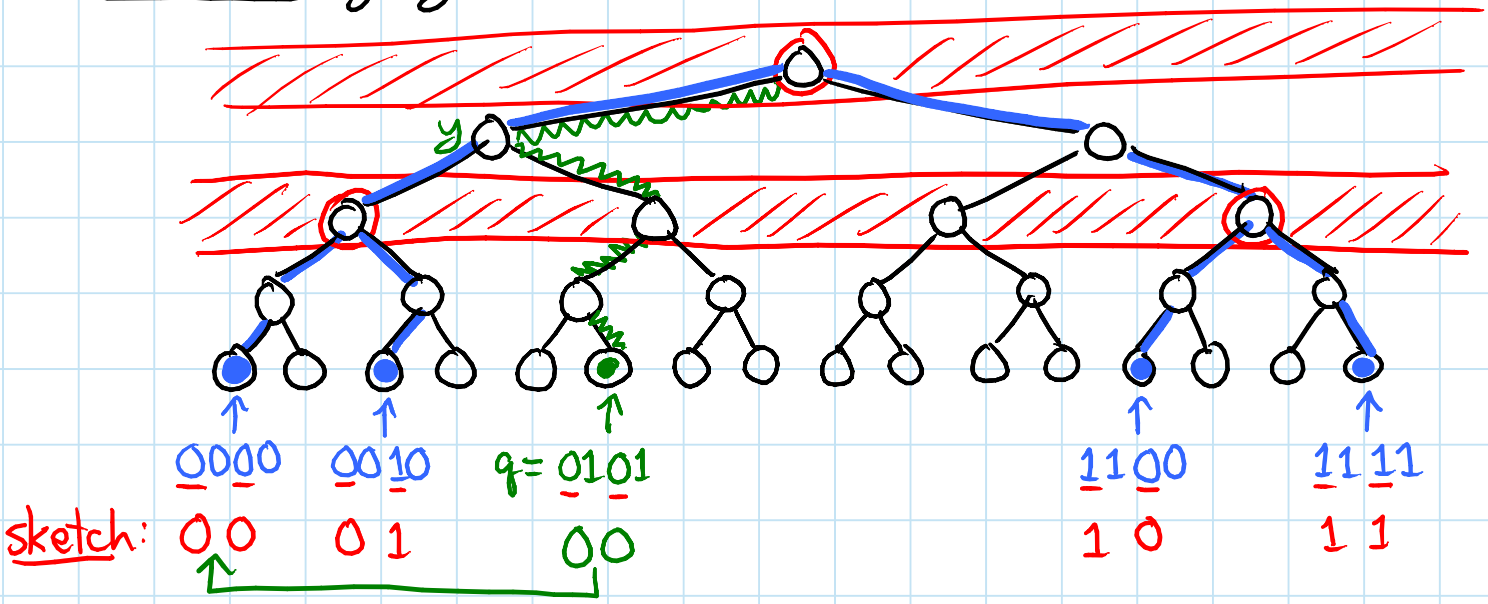
- call these **important bits** $b_0 < b_1 < \dots < b_{r-1}$,
  $r < k = O(w^{1/5})$

(perfect) sketch(x) = extract bits $b_0, b_1, \dots, b_{r-1}$ from x
  i.e. r-bit vector whose ith bit = b_ith bit of word x
  ⇒ $\text{sketch}(x_0) < \text{sketch}(x_1) < \dots < \text{sketch}(x_{k-1})$
  & 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 sketch(q) in parallel to sketch(x_0), …, sketch(x_{k-1})
- again AC^0 operation on $O(1)$ words
  & we'll see a nice way with standard ops.

⇒ find where 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:



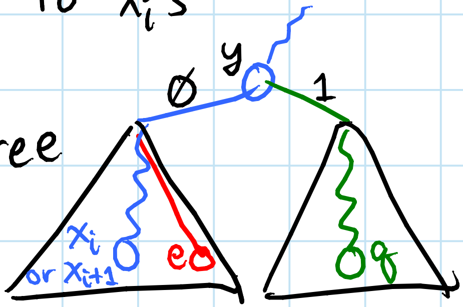
- suppose sketch(x_i) ≤ sketch(q) < sketch(x_{i+1})
- longest common prefix = lowest common ancestor between q & (either x_i or x_{i+1})

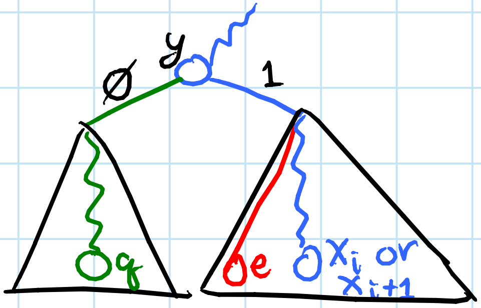
nonsketch

whichever's longest/lowest

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

- if |y|+1st bit of q is 1:
  - nearest x_i is in y0 subtree
  - nearest extreme in that subtree is e=y011⋯1





- else: e=y100⋯0

- predecessor & successor of q among x_i's

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

(in terms of rank i ~ can translate to x_i)

Approximate sketch(x): on word RAM

- don't need sketch to pack b_i bits consecutively
- can spread out in $\underline{\text{predictable}}$ 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 ⓐ $b_i + m_j$ are all distinct (no collision)

ⓑ $b_0 + m_0 < \cdots < b_{r-1} + m_{r-1}$ (preserve order)

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

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

Proof: ① choose $m'_0, m'_1, \dots, m'_{r-1} < r^3$ such that

$b_i + m'_j$ are all distinct $\underline{\text{modulo } r^3}$ (strong ⓐ)

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

$\Rightarrow$ choice for m'_t exists

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

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

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

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

Parallel comparison:

→ protect from underflow

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

  $= \text{sketch}(q) \cdot 0\ 00001\ \cdots\ 0\ 00001$
- difference $= \binom{1}{0}\ *****\ \cdots\ \binom{1}{0}\ *****$
- AND with $1\ 00000\ \cdots\ 1\ 00000$

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

  1 if $\text{sketch}(q) \leq \text{sketch}(x_i)$

  0 if $\text{sketch}(q) > \text{sketch}(x_i)$

⇒ these bits look like $0000\boxed{1}11$

where sketch(q) fits ↗ ↑

need index of most sig. 1 bit

- multiply with $0\ 00001\ \cdots\ 0\ 00001$

  $\rightarrow \underbrace{\boxed{\#\ 1\text{'s}}}_{\text{desired}}\ \cdots\ \boxed{\#1\text{'s to right}}\ \boxed{\text{last } 1}$

⇒ AND with 1111 & shift right to get # 1's

= index of Ø→1 transition

= k − rank in sketch world

- special case of:

Index of most signifigant 1 bit:

$$\begin{array}{cccccccc} 0&0&0&\boxed{1}&0&1&1&0 \\ 7&6&5&\boxed{4}&3&2&1&0 \end{array} \mapsto 4$$

- AC^0 operation [Andersson, Miltersen, Thorup 1999]
- instruction on most modern CPUs

  (see Linux kernel: include/asm-*/bitops.h ;
  GCC: __builtin_clz ; VC++: _BitScanReverse)
- needed during desketchifying (q XOR $x_{i(+1)}$)

Word RAM solution: [Fredman & Willard 1993]

- split word into $\sqrt{w}$ clusters of $\sqrt{w}$ bits each:

$$x = \underbrace{0101}_{\leftarrow\sqrt{w}\rightarrow} \mid \underbrace{0000}_{\leftarrow\sqrt{w}\rightarrow} \mid \underbrace{1000}_{\leftarrow\sqrt{w}\rightarrow} \mid \underbrace{1101}_{\leftarrow\sqrt{w}\rightarrow}$$

$\sqrt{w}$

- similar to van Emde Boas, but no recursion
- identify first nonempty cluster, then first 1 within

① identify nonempty clusters

- AND x with $F = 1000 \quad 1000 \quad 1000 \quad 1000$
  $\rightarrow \quad 0000 \quad 0000 \quad 1000 \quad 1000$
  = which clusters have first bit set
- XOR with x $\rightarrow$ $0101 \quad 0000 \quad 0000 \quad 0101$
  = remaining bits
- subtract F − this: $0{*}{*}{*} \quad 1000 \quad 1000 \quad 0{*}{*}{*}$
  borrow ⇔ nonempty no borrow ⇔ subtract ∅
- AND with F $\rightarrow$ $0000 \quad 1000 \quad 1000 \quad 0000$
- XOR with F $\rightarrow$ $1000 \quad 0000 \quad 0000 \quad 1000$
  nonempty empty
- OR with which clusters have first bit set
  $\rightarrow y = 1000 \quad 0000 \quad 1000 \quad 1000$
  = which clusters are nonempty

② perfect sketch of y → $\underline{1011}$

- $b_i = \sqrt{w} - 1 + i\sqrt{w}$
- use $m_j = w - (\sqrt{w} - 1) - j\sqrt{w} + j$

$\Rightarrow b_i + m_j = w + (i-j)\sqrt{w} + j$ are unique for $0 \le i, j < \sqrt{w}$

& $b_i + m_i = w + i$

$\Rightarrow$ bits $w, w+1, \dots, w+\sqrt{w}-1$ of $y \cdot m$ (shifted right w) form perfect-sketch(y)

③ find first 1 bit in sketch(y) = first nonempty cluster c

- use parallel comparison to find rank among:

$$\left\{\begin{array}{c}0001\\0010\\0100\\1000\end{array}\right\} \sqrt{w} \text{ powers of } 2$$

- fits: $\sqrt{w} \cdot (\sqrt{w} + 1) < 2w$ bits

④ find first 1 bit d in identified cluster c

- shift right $c \cdot \sqrt{w}$ & AND with 1111 to obtain cluster
- use parallel comparison as in ③

⑤ answer $= c\sqrt{w} + d$

6.851 Advanced Data Structures
Spring 2012