L10 2.5.17

Last time: van Emde-Boas trees

Today: Fusion trees

<u>Model</u> word RAM

- w-bit word
- $O(1)$ op on those

e.g. + - , −, /, &, |

VEB trees fast for small words

Fusion trees are $O(\log_w n)$ → good for long words

$O(\log n / \log w)$

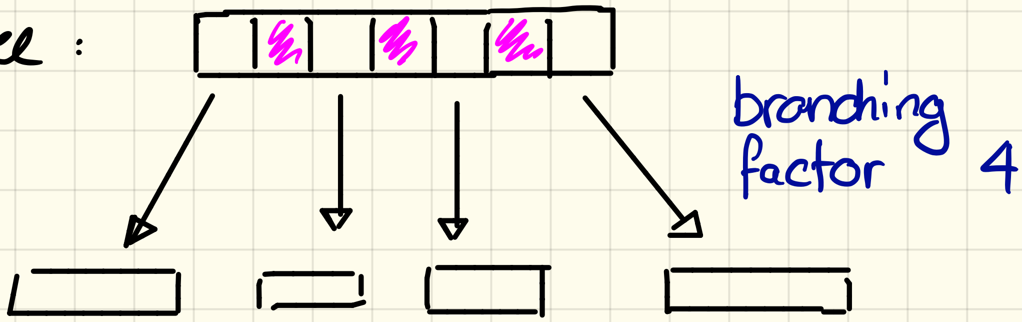
We consider <u>static</u> Fusion trees

Dynamic Fusion trees possible

→ $O(\log_w n + \log\log n)$

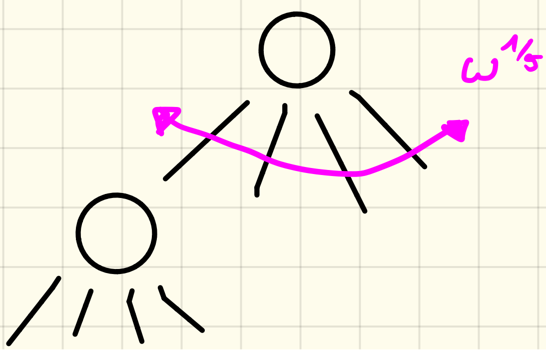
<u>Idea</u> B-tree with branching factor $O(w^{1/5})$

B-tree:



$$h = \log_B(n) = \frac{\log n}{\log B}$$

→ We get a tree of height $O(\log_w n)$



Problem: - each key is w bits

- $w^{1/5}$ keys

w bits

$\Rightarrow$ pack them into $O(1)$ words

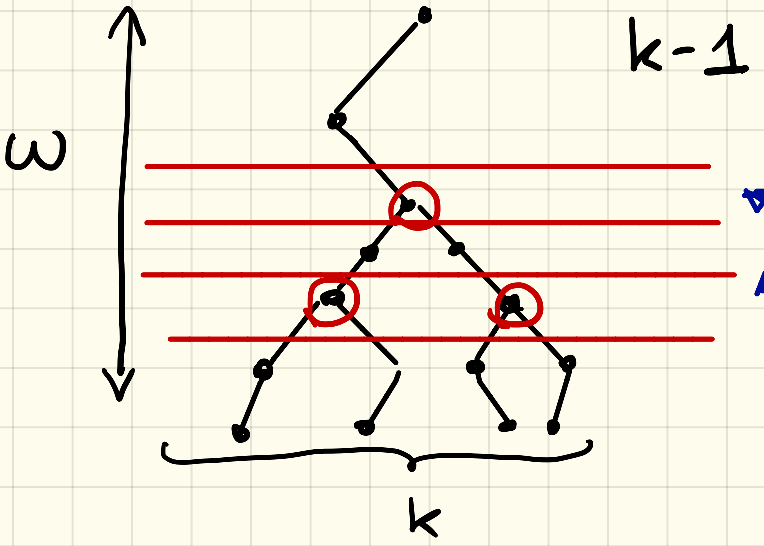
In a node $k = O(w^{1/5})$ keys $x_0 < x_1 < \dots < x_{k-1}$

We want $O(1)$ pred/succ queries on x_i

We allow for poly(k) preprocessing

Distinguishing k keys: (using tries)

$k-1$ branching nodes



$\leq k$ bits where x_0, x_1 diverge

x_1, x_2 diverge

...

x_{k-2}, x_{k-1} "

e.g.

$x = 01\underline{0}0\underline{0}0$

sketch(x) = 00

$\Rightarrow$ $b_0, b_1, \dots, b_{k-1}$ important bits

Sketch of a key x :

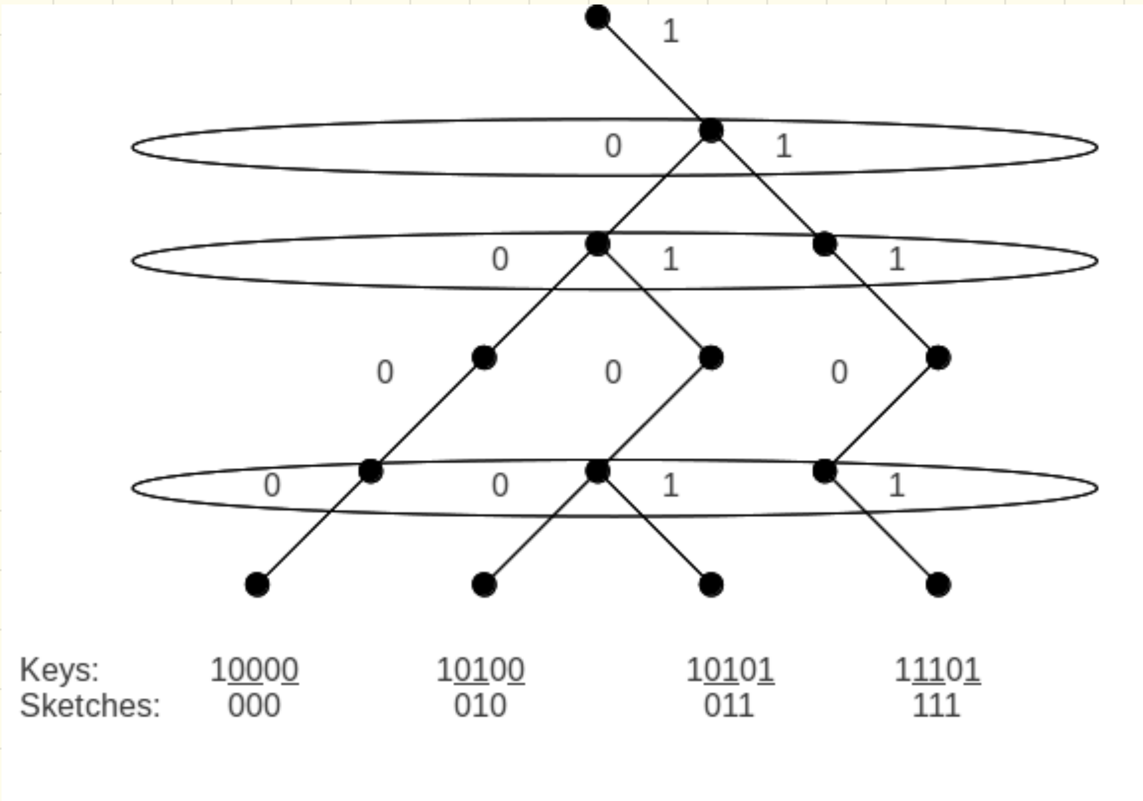
sketch(x) = extract bits $b_0, b_1, \dots, b_{k-1}$ from x, i.e.

$$(x)_{b_0} (x)_{b_1} \dots (x)_{b_{k-1}}$$

To store the order only k bits required:

$$\text{sketch}(x_i) < \text{sketch}(x_{i+1})$$

Wikipedia: Fusion Tree



Problems: - Construction time
- How to sketch
- How to query q:

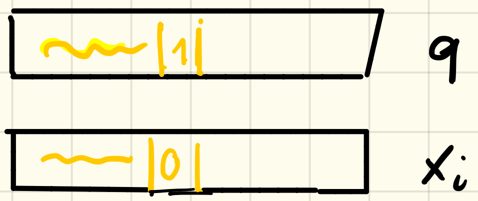
$x_i \leq q \leq x_{i+1} \not\Leftrightarrow \text{sketch}(x_i) \leq \text{sketch}(q) \leq \text{sketch}(x_{i+1})$

Good news: Space is fine

$|\text{sketch}| = \Theta(w^{1/5}) \quad k^2 = O(w^{2/5})$

We will (at the moment) not consider how to search in parallel.
Instead: How to query q in $O(1)$

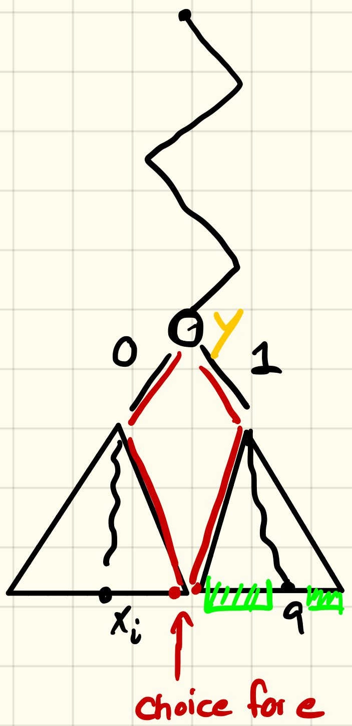
- Suppose we know
  $\text{sketch}(x_i) \leq \text{sketch}(q) \leq \text{sketch}(x_{i+1})$
- Then find LCP = LCA of q & x_i
  or q & x_{i+1} (the longer one)



oldest_bit(q xor x_i)

Pratice : 1 CPU OP

Theory : $O(1)$ word RAM



<u>Claim</u> In subtree of q is no key

<u>pf</u> Othw y would be in the sketch and x_i, x_{i+1} would not be closest in 'sketch'

If first different bit of q & x_i is 1
nearest x is in $y{=}0$ subtree:

- query not for q but for
$$e = y011\ldots1$$

else
$$e = y1000\ldots$$

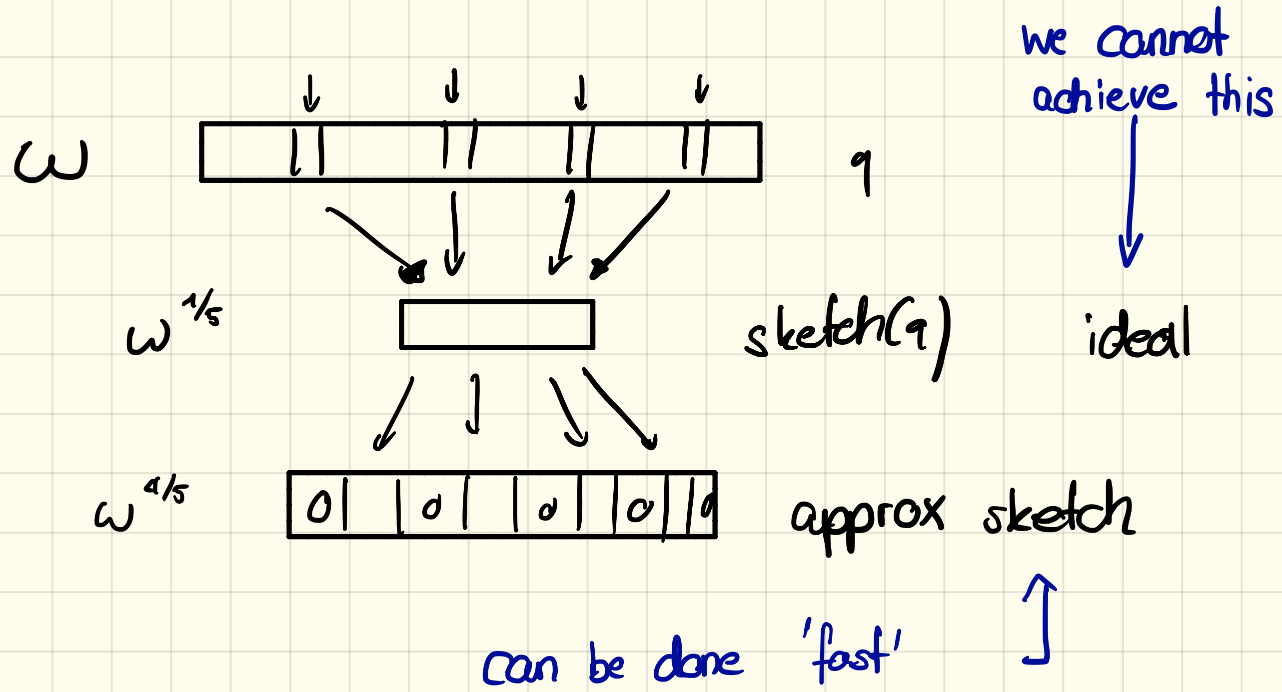
Pred/Succ. of q among x's
= pred/succ of e " x's
= " " " sketch(e) among sketch(x)'s

<u>Lemma</u> $x_i \le e \Rightarrow \text{sketch}(x_i) \le \text{sketch}(e)$
$x_i > e \Rightarrow \text{sketch}(x_i) > e$

Search(q):
- sketch(q)
- find $\text{sketch}(x_i) \le \text{sketch}(q) \le \text{sketch}(x_{i+1})$
- max $\text{Lcp}(x_i, q)$ $\text{lcp}(x_{i+1}, q)$
- compute e
- sketch(e)
- search sketch(e) in sketch(x)'s.

It remains to see how one

- compute sketch
- searches in parallel



We approximate sketch(x)

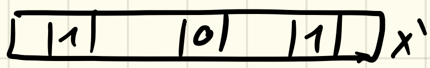
→ don't need to pack b_i bits consec.

→ spread them in a predictable pattern interleaved with 0s

→ preserves order

1) mark important bits

$$x' = x \,\&\, \left(\sum_{i=0}^{r-1} 2^{b_i} \right)$$



$$m = \sum_{j=0}^{r-1} 2^{m_j}$$

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

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

$\exists m$: a) $b_i + m_j$ are all distinct

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

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

$= O(\omega^{4/5})$

Compute $(x' \cdot m) \,\&\, \left(\sum_{i=0}^{r-1} 2^{b_i + m_i} \right)$

pf of claim

1) We build m_j' where $b_i + m_j$ are dist. mod r^3

$m_0', m_1', \dots, m_{t-1}'$

m_t' must $\neq$ $\underbrace{m_i'}_{t} + \underbrace{b_j}_{r} - \underbrace{b_k}_{r}$

there are $(r-1)r^2$ values to avoid.

$\Rightarrow$ a)

2) $m_i \equiv m_i' \bmod r^3$

$m_i + b_i \in [i \cdot r^3, (i+1) \cdot r^3)$

$\Rightarrow$ b) + c)