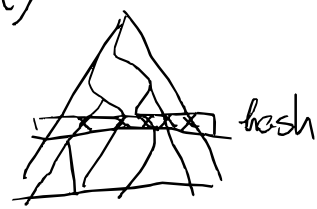
Y-fast tree, fusion tree

Recap: maintain a set of n integers in $[0, u-1]$.
pred(x): return largest number $\leq x$
ins, del

van Emde Boas tree: update & query $O(\log\log u)$
space $O(u)$

x-fast trie: query $O(\log\log u)$
ins/del: $O(\log u)$
space $O(n \log u)$



$O(\log\log u)$ time for binary searching the levels to find the lowest ancestor of query x in the trie.

---

y-fast trie

update, query in $O(\log\log u)$ (amortized) time
space $O(n)$.

structure:
divide the set of integers into $\Theta(\frac{n}{\log u})$

chunks $S_1, \dots, S_k$. $S_1 < S_2 < \cdots < S_k$.

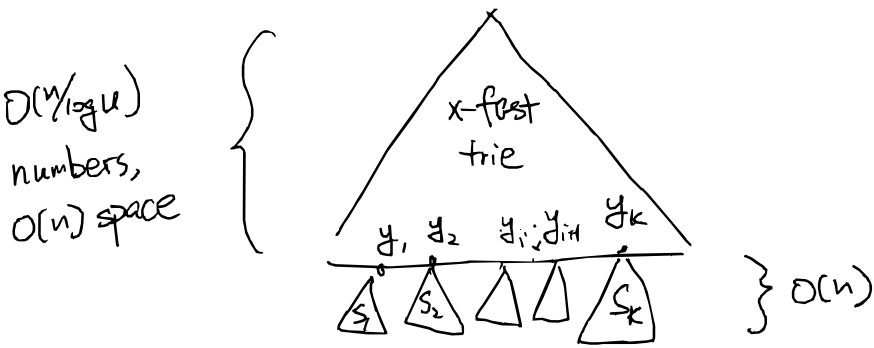
each chunk has between

$\frac{1}{2}\log U$ and $2\log U$

numbers.

Pick any $y_i \in S_i$ as a representative.

- Store $y_1, \dots, y_k$ in a x-fast trie
- store each S_i in a BBST.



Space: $O(n)$

pred(x):

$y_i \leftarrow$ pred(x) in x-fast trie ← $O(\log\log U)$

if $x < \min S_{i+1}$

return pred(x) in S_i ← $O(\log|S_i|) = O(\log\log U)$

else

return pred(x) in S_{i+1}

ins(x):

$y_i = \text{pred}(x)$ in x-fast trie

if $x < \min S_{i+1}$

  ins(x) to S_{i+1}

else

  ins(x) to S_i

$O(\log\log U)$ time

if $|S_i|$ (or $|S_{i+1}|$) $> 2\log U$

  split into two sets of size $\log U$

  ins a new representative into x-fast trie (*)

$O(\log U)$ time if executed.

Lemma: The amortized time of (*) is $O(1)$.

Proof: Consider the time when S_i was created, and all insertions to it afterwards.

Charge the cost of (*) to these insertions, which caused $|S_i|$ to grow from $\log U$ to $2\log U$.

There are $\geq \log U$ such insertions.

Each is charged a cost of $O(1)$.

Every insertion can be charged at most once. □

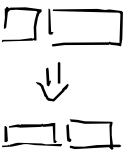
del(x):

pred(x) in x-fast trie to find i s.t. $x \in S_i$

del(x) in S_i

if $|S_i| < \frac{1}{2}\log U$,

$O(\log U)$ time if executed { merge S_i with S_{i+1} (or S_{i-1}) then split if size $> 1.5 \log U$



Similar argument shows $O(1)$ amortized time

Fusion tree

focus on only pred queries, no ins/del

also works if there's ins/del (more complicated)

Let $w = \log U$ be # bits in each int.

Fusion tree: $O(\frac{\log n}{\log w}) = O(\frac{\log n}{\log \log U})$ time,

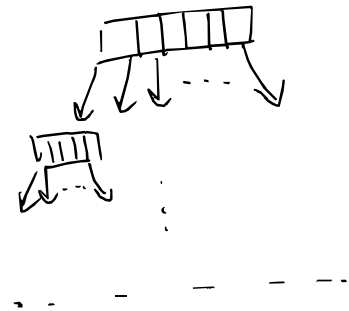
$O(n)$ space.

$\min\{\text{Fusion}, \text{vEB time}\} \leq O(\sqrt{\log n}) \Rightarrow O(n\sqrt{\log n})$ time

algo for sorting n integers.

Build a tree with branching factor $B+1$

$B = w^{1/5}$.



The root stores B "pivots"

$\frac{n}{B+1}, \frac{2n}{B+1}, \ldots, \frac{B}{B+1} \cdot n$-th

largest number, etc.

Each subtree stores numbers between two adjacent pivots. Depth $= O(\log_B n) = O\left(\frac{\log n}{\log w}\right)$.

Key component:

fusion node. stores B pivots $y_1, \ldots, y_B$, s.t. pred(x) among $\{y_1, \ldots, y_B\}$ takes $O(1)$ time.

Imagine $y_1, \ldots, y_B$ in binary trie

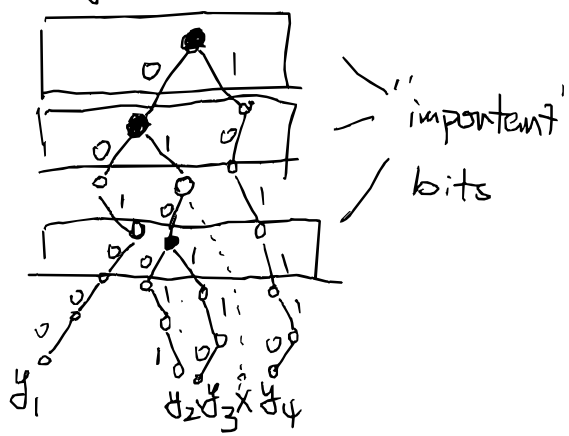
$y_1 = 001000$

$y_2 = 010011$

$y_3 = 010110$

$y_4 = 101110$

B leaves, $B-1$ branching nodes



$sk(y)$ = y restricted to levels with a branch (sketch of y)

$sk(y_1), \ldots, sk(y_B)$, each has $B-1$ bits ($B^2 < w$)

- can compress $|sk(y_1)|sk(y_2)|\cdots|sk(y_B)|$ into $< w$ bits (one word)
- $y_i < y_j$ iff $sk(y_i) < sk(y_j)$
- for query x, make B copies of $sk(x)$

  $|sk(x)|sk(x)|\cdots|sk(x)|$ in one word
- can compare $sk(x)$ with all $sk(y_i)$ at once using bit-ops.

$sk(y_1) = 000$

$sk(y_2) = 010$

$sk(y_3) = 011$

$sk(y_4) = 101$

but $y_3 \le x < y_4$

Suppose

$x = 011011$

$sk(x) = 010$

$sk(y_2) \le sk(x) < sk(y_3)$

$P = 01$

$x' = 011111$

Lemma: suppose $sk(y_i) \le sk(x) < sk(y_{i+1})$.

then either y_i or y_{i+1} has the longest common prefix with x.

$\Leftrightarrow$ lowest ancestor of x in the trie
is an ancestor of either y_i or y_{i+1}.

Suppose x is in the right subtree of p.
Then let $x' = p111\cdots1$,

$y_j \le x < y_{j+1}$ iff $sk(y_j) \le sk(x') < sk(y_{j+1})$

~~left~~

$x' = p00\cdots0$.