September 29, 2023 10:50 AM

Problem
Pred search for integers in [U]

What we know: $O(\log n)$ query/update, $O(n)$ space

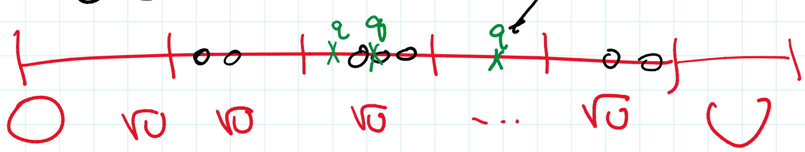
(or $O(1)$ query time, $O(U)$ space, $O(U)$ update time)

van Emde Boas '75: $\boxed{O(\log\log U)}$ query time & update time (expected)
$O(n)$ space

vEB Trees

idea - multi-way D&C ← $\sqrt{U}$-way

divide [U] into $\sqrt{U}$ chunks of length $\sqrt{U}$



store min & max for each chunk ← univ size $\sqrt{U}$

recurse inside each chunk ← excluding min & max

let $D = \{$(indices of) all nonempty chunks$\}$ ← univ size $\sqrt{U}$

store D in dictionary / hashing

recurse in D.

Query(q):

$i =$ index of chunk containing q ← $O(1)$ time by int division

$O(1)$ time by hashing → if $i \in D$,

recurse inside chunk (also check min/max)

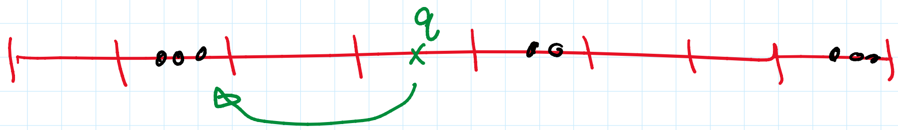
if $i \notin D$,

by hashing

if $i \notin D$,

recurse in D

& return max of pred chunk



$$Q(U) = 1 \cdot Q(\sqrt{U}) + O(1)$$

let $U = 2^{\ell}$,

$\sqrt{U} = 2^{\ell/2}$

$$Q'(\ell) = Q'(\ell/2) + O(1)$$

$$\Rightarrow O(\log \ell) = \boxed{O(\log\log U)}$$

insert(x):

i = index of chunk containing x

if $i \in D$,

recurse inside chunk & update min/max

if $i \notin D$,

insert i to D,

set chunk's min/max to x.

$$I(U) = I(\sqrt{U}) + O(1)$$

$$\Rightarrow \boxed{O(\log\log U)}$$ ← expected by hashing

Delete: similar

Space: $O(n \log\log U)$ space

but can be reduced to $O(n)$

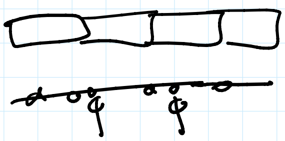
trick - divide sorted list into $O(\frac{n}{b})$ blocks of $O(b)$ elems

store min/max per block in vEB tree

$\Rightarrow$ space $O(n + \frac{n}{b}\log\log U) \Rightarrow \boxed{O(n)}$

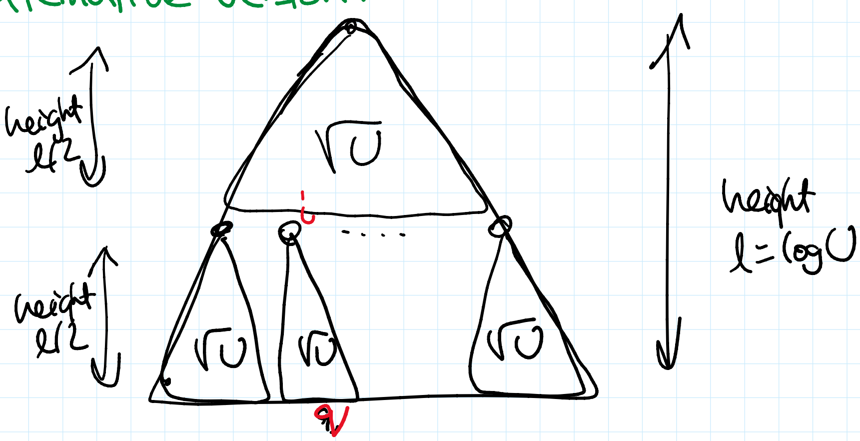
query/update time $O(\log\log U + b) \Rightarrow \boxed{O(\log\log U)}$

linear search



choose $b = \log\log U$

Alternative version:



query → binary search over path of length $\ell = \log U$

Rmk: static:

Beame, Fich '02 $\quad O\left(\frac{\log\log U}{\log\log\log U}\right)$ query time

but $O(n^{1+\varepsilon})$ space

matching lower bds

Rmk on Model of Computation: Word RAM

- RAM on w-bit word
- assume standard ops on w-bit words in $O(1)$ time

  (add, mult, div, bitwise and/or)
- assume $w \geq \log n$ (indices, ptrs fit in word)
- assume $w \geq \log U$ (input numbers fit in word)

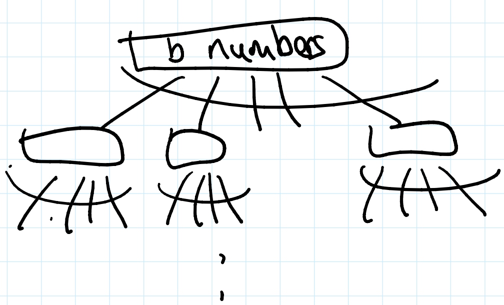
Next: Fusion Tree (Fredman, Willard '90)

static, $O(n)$ space,

query time $\boxed{O\left(\frac{\log n}{\log w}\right)} \leq O\left(\frac{\log n}{\log\log n}\right)$

beats $\log n$, regardless of U !!

basic idea - deg-b search tree, with $b \approx \sqrt{w}$



height $\log_b n = \frac{\log n}{\log w}$

how to search among b numbers in $O(1)$ time?

Compress b numbers in one word

e.g. $\{8, 10, 14, 15\} = \{\overbrace{0100}^{\log U}, 0110, 1110, 1111\}$

idea - trie