L9 25. April 17

Topic : INTEGERS

- Word RAM
- Predecessor Problem:

$|U| = 2^w$

w size of word
e.g. $w = 64$

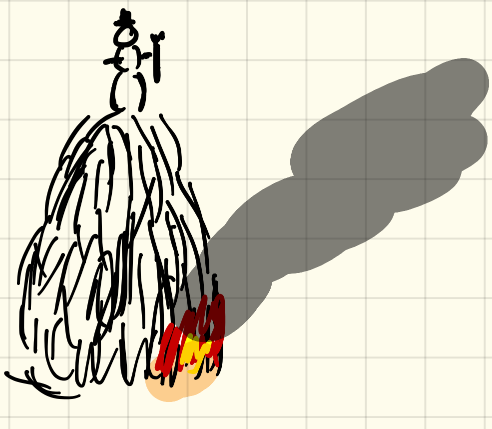
$w \geq \log n$

- Maintain set S of n words
- Insert$(x \in U)$
- Delete$(x \in S)$
- Predecessor$(x \in U)$
  $= \max \{ y \in S \mid y \leq x \}$

In comparison model $\Theta(\log n)$

VEB in short	Operation	Space
van Emde Boas tree	$O(\log w)$	$O(U)$
+ hashing	$O(\log w)$ w.h.p.	$\Theta(n)$
y-fast trees	$O(\log w)$ w.h.p.	$\Theta(n)$
fusion tree	$O(\log n / \log w)$ w.h.p.	$\Theta(n)$
[static dynamic		

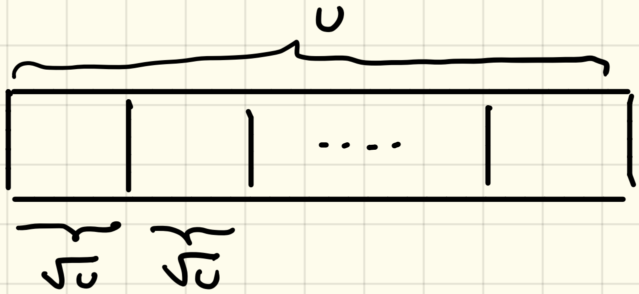
$$\min\left(\log w, \ \log n / \log w\right) \le \sqrt{\log n}$$



VEB complexity: $T(U) = T(\sqrt{U}) + \mathcal{O}(1)$

$\leadsto$ aka bin. search on ω

split universe into $\sqrt{U}$ clusters of size $\sqrt{U}$



word $x = \langle c, i \rangle$, $c = \lfloor x / \sqrt{U} \rfloor$, $i = x \% \sqrt{U}$

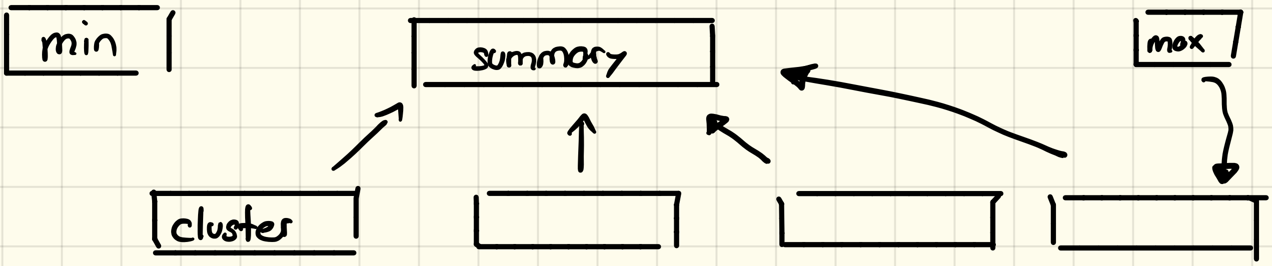
$\rightarrow \sqrt{}$ expensive, thus:

$c = x >> (\omega/2)$

$i = x \,\&\, ((1 << (\omega/2))^{x} - 1)$

$x = (c << (\omega/2)) \,|\, i$

A VEB tree of size U consists of three components:



- v. cluster[i] = VEB of size $\sqrt{U}$, $0 \le i \le \sqrt{U}$
- v. summary = VEB of size $\sqrt{U}$
  - → Allows to check if clusters are empty.
- v. min = minimum element in V
  - → Is not stored elsewhere!
- v. max = maximum element in V

```
Successor(v, x = <c,i>)
  if x < v.min return v.min

else if i < v.cluster[c].max   // answer in cluster c
      return <c, successor(v.cluster[c], i)>
  else   // answer in next non-empty cluster
      c' = successor(v.summary, c)
      return <c', v.cluster[c'].min>

Insert(v, x = <c,i>)
  if v.min = none  v.min = v.max = x; return;
  if x < v.min swap x <-> v.min
  if x > v.max  v.max = x
  if v.cluster[c].min = none
      Insert(v.summary, c)  // if invoked next call
                            //    of is of type
  Insert(v.cluster[c], i)
```

Delete(v, x = <c, i>)

if x = v.min && x = v.max : v.min = v.max = none ; return ;

if x = v.min :
    c = v.summary.min ; i = cluster[c].min ;
    v.min = <c, i>

Delete(v. cluster [c] , i)

if v. cluster[c]. min = None

    Delete (v. summary , c)

if v. summary. min = None

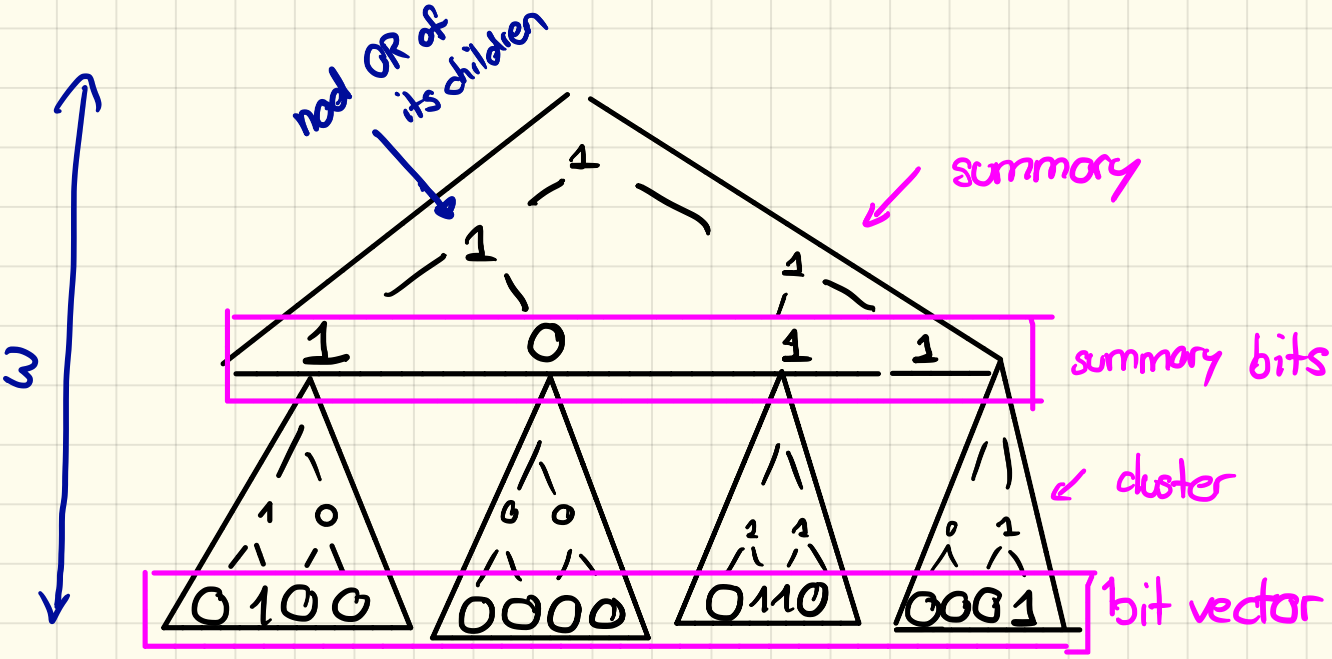
    v. max = v.min

else

    c' = v. summary.max

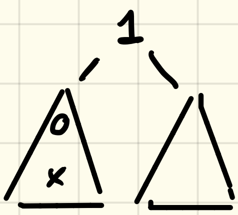
    v.max = <c', v. cluster[c']. max>

Tree view – expanded recursion (w/o min/max)



Update = $\boxed{\Theta(w)}$

Query = bin. search for transition on path leaf → root



$\Theta(\log w)$

Each tree stores min/max, linked list of 1s

Space $\Theta(U)$ space

How to save space:

- do not store empty clusters in VEB

  v. clusters = hash table (FKS perfect hashing)

  space = $O(\#\text{nonempty clusters} + 1)$

- charge each table entry to min in its child

space $O(n \cdot \log w)$

$\rightarrow$ $O(n)$ with INDIRECTION

X- fast tree 1977

- take tree view
- store hash table of each level 1's positions

$$\left.\begin{array}{ll}\text{Query} & \Theta(\log w) \\ \text{Updates} & O(w)\end{array}\right\} \text{w.h.p.}$$

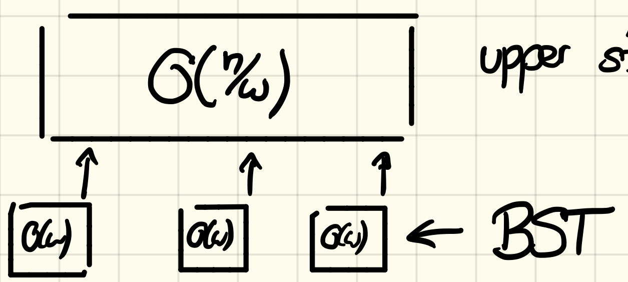
Space $O(n \cdot w)$

Y- fast tree

- x-fast tree + Indirection

$$\left.\begin{array}{ll}\text{Query} & O(\log w) \\ \text{Updates} & O(\log w) \text{ amortized}\end{array}\right\} \text{w.h.p.}$$

Space $O(n)$



Query: top $O(\log w)$ bottom $O(\log w)$

Updates $O(\log w)$ bottom

$O(w)$ top in worst case

amortized to $O(w)$ changes

Space $O(n/w \cdot w + n) = O(n)$