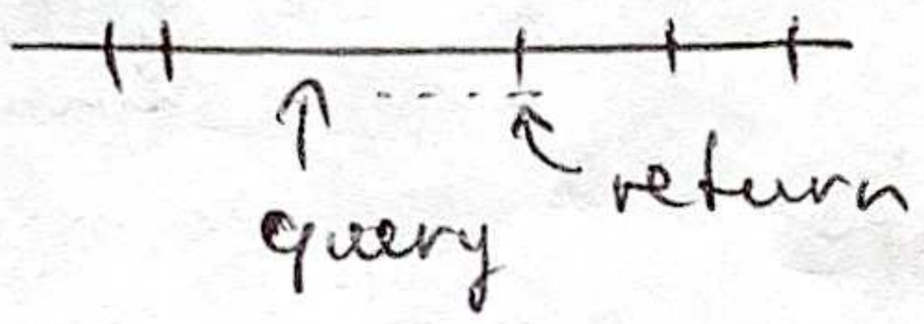
6.046 (2015) Van Emde Boas Trees 

lecture 4

- series of improved DS's
- Insert, Delete, Successor
- Space

size of universe $U = u$

**Goal**: maintain $n$ elements among $\{0, 1, \ldots, u-1\}$ subject to Insert, Delete, Successor

Successor: 

, predecessor is symmetric

in $O(\lg \lg u)$ time

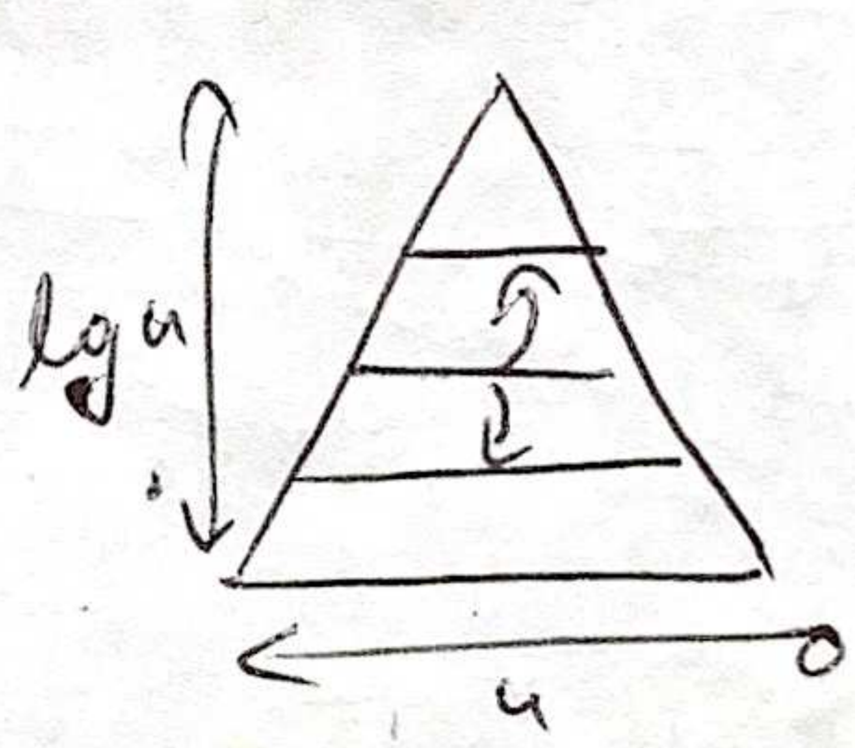
my comment [redacted] $O(n^{\lg n})$ ← size of universe in radix sort

$\lg \lg 2^{64} = \lg 64 = 6$

if $u = n^{O(1)}$ or $n^{\lg^{O(1)} n}$, then $\lg \lg u = O(\lg \lg n)$ [redacted]

- application: network router: ip range → send to port y

mark beginnings of IP ranges and use predecessor




Where might $O(\lg \lg u)$ come from?

⇒ binary search on levels of tree

$T(k) = T(\frac{k}{2}) + O(1) = O(\lg k)$

$T(\lg u) = T(\frac{\lg u}{2}) + O(1) = O(\lg \lg u)$

$T'(u) = T'(\sqrt{u}) + O(1) = O(\lg \lg u)$

intuition

take a problem of size $u$
split into problems of size $\sqrt{u}$
and recurse on one of them

(1) Bit vector = array of size $u$

0 = absent  1 = present

array  $u = 16$, $n = 4$

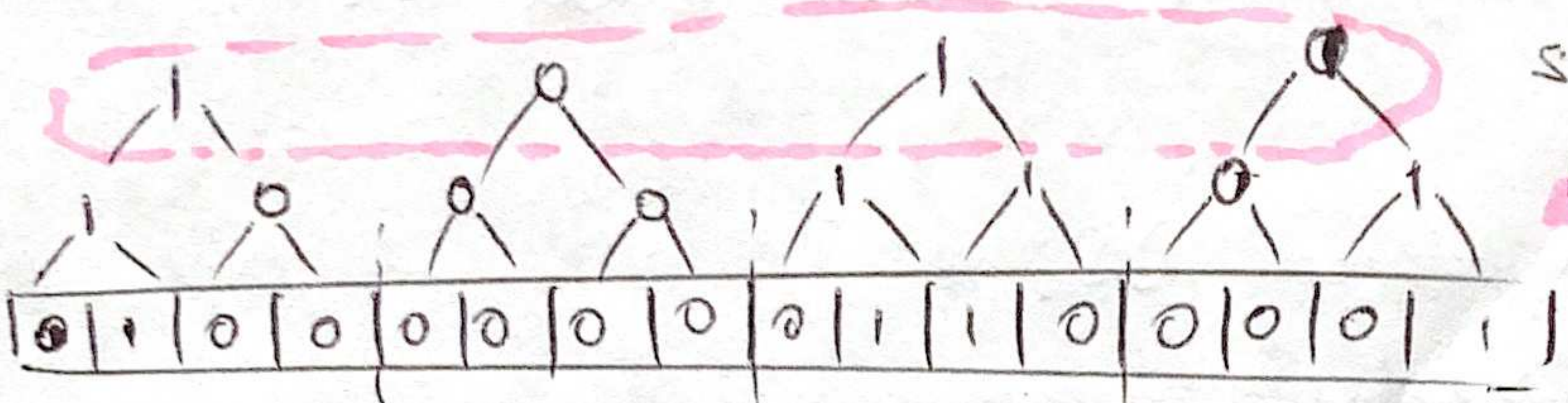
| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |

Insert / Delete : $O(1)$

Successor : $O(u)$

or

or



summary vector
↑
my comment: recursion [redacted] see vEB [redacted] tree. Summary has all info about "cluster bit vector" similar [redacted] idea in ideal non-rand skip lists balancing

(2) split the universe into $\sqrt{u}$ clusters & of size $\sqrt{u}$.

my comment

| skip list | vEB |
|---|---|
| lg(n) levels, each $\lg_{\lg n}\sqrt{n}$ | repeated square rooting |
| $2 \leftarrow 4 \leftarrow 8 \leftarrow 16$ | $2 \leftarrow 4 \leftarrow 16$ |

**Insert: O(1)**
check / change bits on 2 levels

**Successor (x):**

$O(\sqrt{u})$ {
- look in x's cluster
- look for next 1 in the summary vector
- look for first 1 in that cluster

**Terminology:**

if $x = i\sqrt{u} + j$    i: cluster number

$0 \le j < \sqrt{u}$    j: position within that cluster

ex

$x = 9$    $u = 16$, $n = 4$

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |

(clusters 0, 1, 2)

$high(x) = \lfloor x/\sqrt{u} \rfloor$   integer division → i

$low(x) = x \bmod \sqrt{u}$   remainder → j

$index(i,j) = i\sqrt{u} + j$   use i and j to go back to x ← x representation

$9 = 2 \cdot \sqrt{16} + 1$

go two clusters in the summary vector, then multiply by the size of each cluster and continue with 0, 1.

**Why high/low?** O(lg(u)) long

$x = 9$   $\underbrace{1\,0}_{high(x)}\underbrace{0\,1}_{low(x)}$

high(x) = 2 → high half of the bits

low(x) = 1 → low half of the bits

divide the bit vector of 9 in 2 halfs and get high half and low half.

very efficient ←

(3) recurse : V = size u — vEB

— V. clusters [i] = size $\sqrt{u}$  $0 \le i \le \sqrt{u}-1$ — vEB

— V. summary = size $\sqrt{u}$ — vEB

**Insert (V, x)**

low(x) becomes x in the next Insert call.

my comment

Insert (V. cluster [high(x)], low(x))

Insert (V. summary, high(x)) ⇐ updates which (sub) clusters are not empty, recursively for each cluster that is recursed on

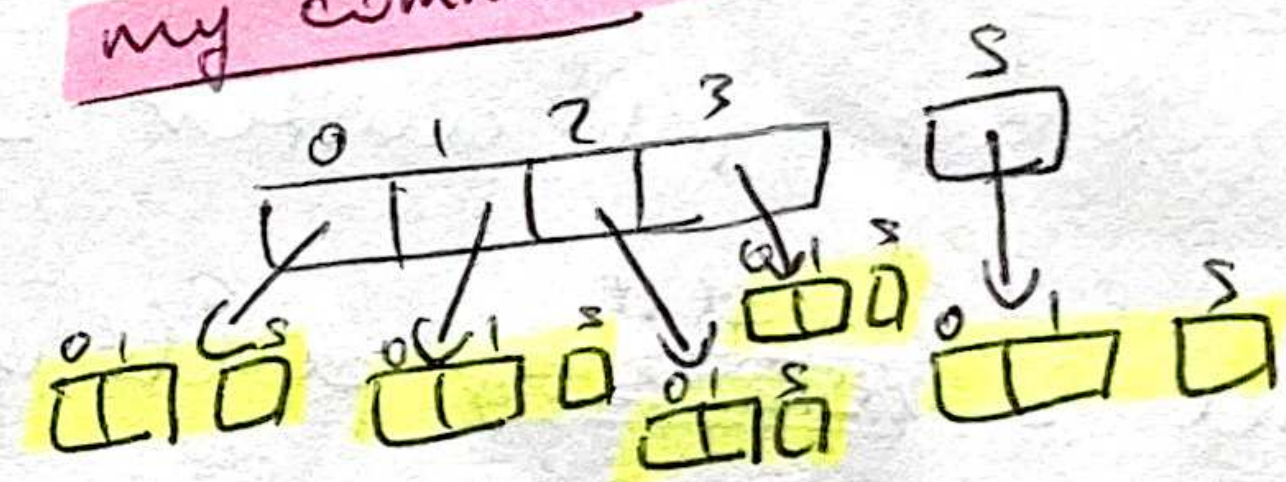
high(x), low(x)

$T(u) = 2T(\sqrt{u}) + O(1)$

$T'(\lg u) = 2T'(\frac{\lg u}{2}) + O(1)$

$= O(\lg u)$

my comment



**Successor (V, x)**

i = high(x)

j = Successor (V. cluster [i], low(x))

if $j = \infty$

  i = Successor (V. summary, i)

  j = Successor (V. cluster [i], $-\infty$)

return index(i, j)

$O((\lg u)^{\lg 3})$

(4) DS augmentation
store min and max

→ every vEB structure (clusters & summaries) knows its min and max.

**Insert (V, x)**

if x < V. min :

  V. min = x

if x > V. max

  V. max = x

Insert ...

Insert ...

still O(lg u) so far

**Successor (V, x)**

if x < V. min
return V. min

i = high(x)

if low(x) < V. cluster [i]. max :

  j = Successor (V. cluster [i], low(x))

else :

  i = Successor (V. summary, high(x))

  j = V. cluster [i]. min

return index (i, j)

only 1 recursion on $\sqrt{u}$

$O(\lg \lg u)$

(5) don't store min recursively (equivalent to lazy propagation, never move down)

if vEB structure is empty, set V.min = x and stop.
V.max = x

Insert (V, x)

if V.min = None : // inserting onto an empty structure

V.min = V.max = x

return

if x < V.min : swap x ⟺ V.min // every item except min is recursively inserted

if x > V.max : V.max = x

if V.cluster [high(x)].min = None : // if cluster is empty need to [redacted] update summary

Insert (V.summary, high(x))

Insert (V.cluster [high(x)], low(x))

still 2 Insert [redacted] calls in worst case

But if Insert (V.summary, high(x)) is called
V.cluster [high(x)] was empty

Thus [redacted] Insert (V.cluster [high(x)], low(x)) will only update the min and max of V.cluster [high(x)]

⇒ in each case only 1 recursive call!

⇒ $O(\lg \lg u)$

6.046 (2015)
Lecture 4



**Delete (V, x)**

if x = V.min:
    i = V.summary.min
    if i = None:
        V.min = V.max = None
        return // min deleted
    x = V.min = index (i, V.cluster[i].min)
    ← deleting min, but if is not the only elt; actual deletion in following code

my comment
V.min is not in V.summary!
triggers ② in the preceding Delete call

① Delete (V.cluster[high(x)], low(x)) // undo Insert of new x in sub trees!

my comment

② if V.cluster[high(x)].min = None:
    Delete (V.summary, high(x))

if x = V.max: // at this point max was just deleted
    if V.summary.max = None:
        V.max = V.min
    else:
        i = V.summary.max
        V.max = index (i, V.cluster[i].max)

---

if ② leads to a recursive call of Delete, it must be true that in ① the last item in V.cluster[high(x)] was deleted, which is V.cluster[high(x)].min, in constant time

⇒ $O(\lg \lg u)$

lower bound

$\Omega(\lg \lg u)$ for $u = n^{\lg^{O(1)} n}$ & space $O(n$ [redacted] $\text{poly} \lg n)$

⑥ Space is $O(u)$

- only store non-empty clusters

⇒ V.cluster = hash table, not array

[redacted] run time bounds to expectation bounds!

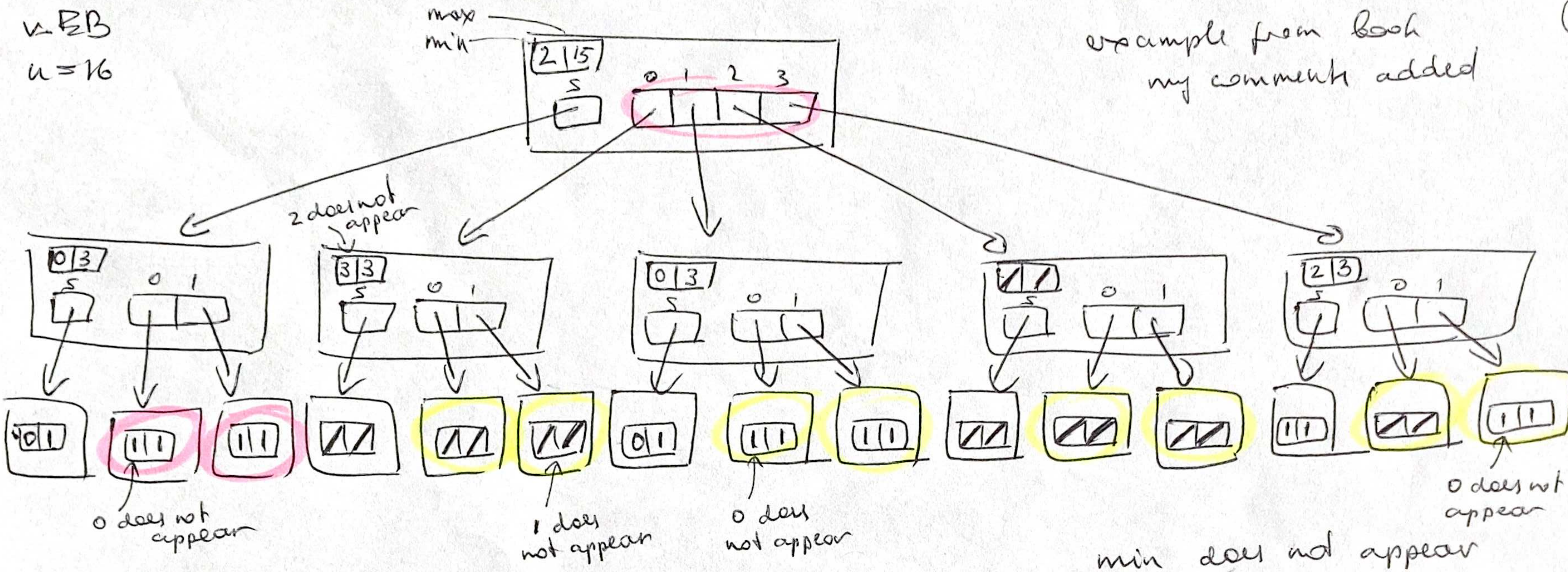
but space goes down

- $O(n \lg \lg u)$ space, $\xrightarrow{\text{fix}}$ $O(n)$

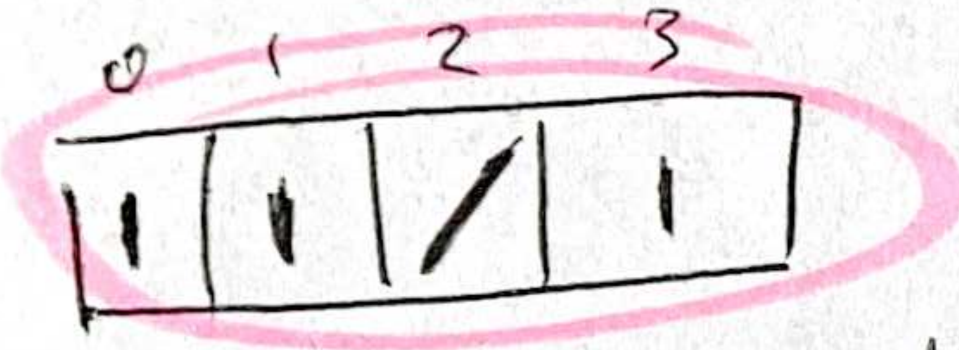
vEB
$u=16$

example from book
my comments added

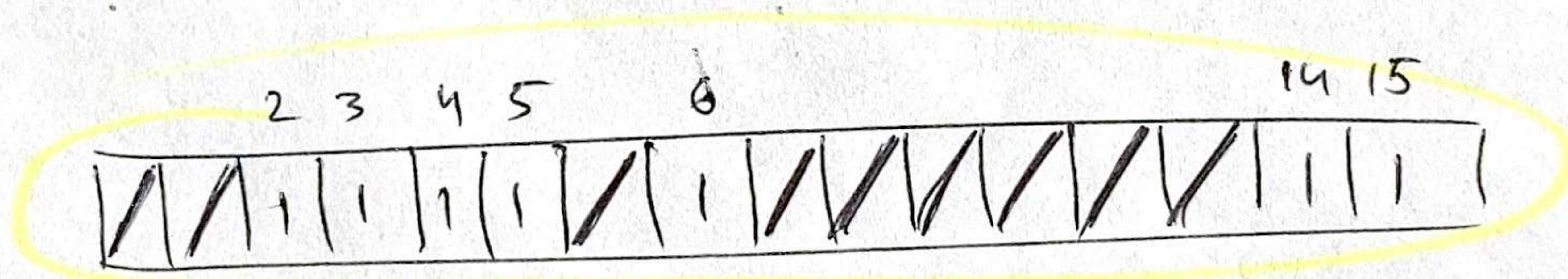
(4)




min does not appear in any subtree (see Insert).




upper level clusters are the bit vector for upper level summary




each pointer ~~[illegible]~~ points to same-size vEB at the next level.

bit vector: 16 → 4 of 4 → each 2 of 2

256 → 16 of 16 → each 4 of 4 → each 2 of 2

4 clusters of 4
each has 2 clusters of 2