$$T_k(n,s) \le T_k(n,b) + T_{k+1}\left(\frac{n}{b}, \frac{s}{b}\right) + O(n)$$
$$Q_k(n,s) \le Q_{k-1}\left(\frac{n}{b}, \frac{s}{b}\right) + O(1)$$

$\Rightarrow$ union $O(\log^{\overbrace{*\cdots*}^{k}} s)$ amortized
find $O(k)$

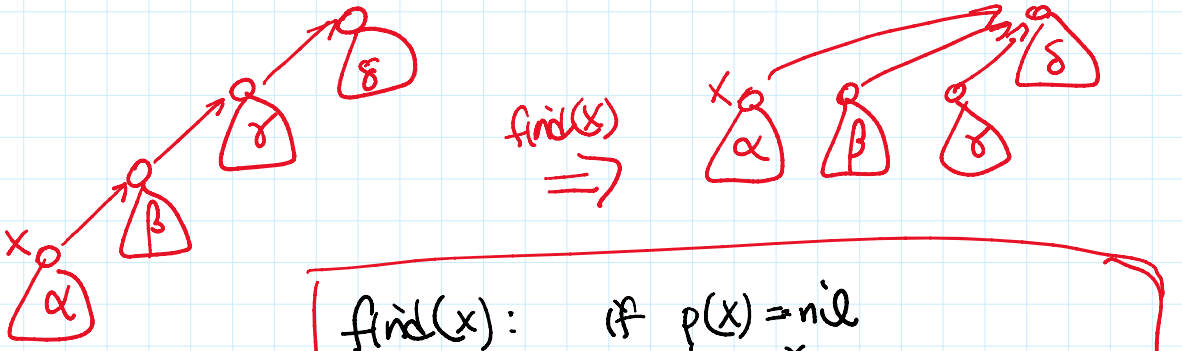
Choose $k = \alpha(n)$ $\Rightarrow$ $O(\alpha(n))$

↑ inverse Ackermann

Better Method 4: Tarjan '75

back to tree method 2
use weighted union heuristic for union
+ path compression for find

} Very simple to implement!!



find(x): if $p(x) = nil$
return x
else return $p(x) = find(p(x))$

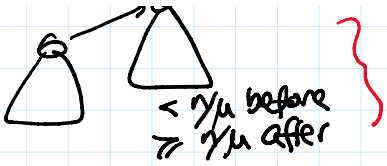
Obs 1 size(p(x)) $\ge$ 2 size(x)
& size(p(x)) only increases

Obs 2 # nodes with size(x) $\ge \frac{n}{\mu}$ is $O(\mu)$.

Pf:

} charge $\le \frac{n}{\mu}$ nodes to x

FT:



charge 1/ n/μ nodes to ^

(each node sends charge to ≤ 1 node)

□

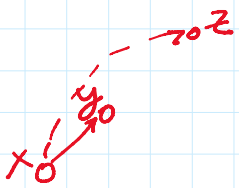
Amortized Analysis by Recurrence: (variant of Seidel-Sharir '05)

Consider interval $I = [\frac{n}{\mu}, s\frac{n}{\mu})$. (Initially: $\mu = n$, $s = n$)

Say x is in I if size(x) $\in$ I.

By obs 2, # nodes in I = $O(\mu)$.

Let m = # finds whose answer is in I.



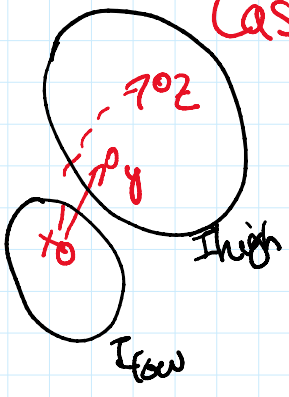
To bound $T(\mu, s, m)$ = # parent changes (i.e. changing p(x) from y to z) with $x, y, z \in I$.

Divide I into $I_{low} = [\frac{n}{\mu}, b\frac{n}{\mu})$, $I_{high} = [b\frac{n}{\mu}, s\frac{n}{\mu}) = [\frac{n}{\mu/b}, \frac{s/b}{\mu/b}\frac{n}{\mu/b})$

Case 1. $x, y, z \in I_{low} \Rightarrow T(\mu, b, m_{low})$

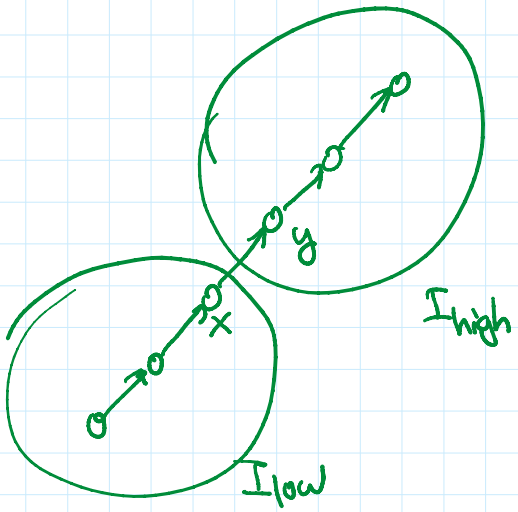
Case 2. $x, y, z \in I_{high} \Rightarrow T(\frac{\mu}{b}, \frac{s}{b}, m_{high})$

Case 3. $x \in I_{low}$, $y, z \in I_{high}$



for each find, this occurs ≤ once

$\Rightarrow O(m_{high})$



Case 4. $x, y \in I_{low}$, $z \in I_{high}$.

for fixed x, this occurs $\le$ once

$\Rightarrow O(\mu)$.

$$T(\mu, s, m) \le T(\mu, b, m_{low}) + T\left(\frac{\mu}{b} \cdot \frac{s}{b}, m_{high}\right) + O(\mu + m_{high})$$

for some $m_{low} + m_{high} = m$.

$b = s/2$: $T(\mu, s, m) \le T(\mu, \frac{s}{2}, m_{low}) + O(\mu + m_{high})$

$\Rightarrow O(\mu \log s + m)$

$b = \log s$: $T(\mu, s, m) \le T(\mu, \log s, m_{low}) + O(\mu + m_{high})$

$\Rightarrow O(\mu \log^* s + m)$

$\vdots$

$O(\mu \log^{*\cdots*} s + km)$ (k stars)

$\Rightarrow$ $O(\alpha(n))$ amort.

Rmks: lower bd $\Omega(\alpha(n))$ for pointer-machine model (Tarjan '80?)

Gabow-Tarjan '8?: $O(1)$ in the offline case (seq of unions is given in advance) by table lookup

INTEGERS

Hashing

Problem ("Dictionary")

Store set S of n integers in $[U] = \{0, 1, \ldots, U-1\}$ in data structure to support

membership query: given y, is $y \in S$?
(if yes, return ptr to elem)

& insert/delete

reduces to pred search $\Rightarrow$ $O(\log n)$ time

but faster?

Method 0

bit vector

$O(1)$ query time
$O(1)$ update time

but $O(U)$ space & preproc time
↑
large

Method 1

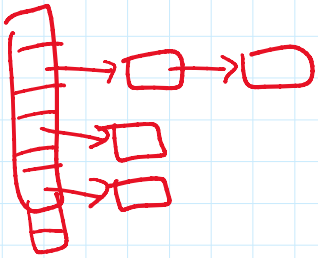
hash table with chaining

Pick <u>hash fn</u> $h: [U] \to [m]$ for some $m \ll U$.

Store array $A[0 \ldots m-1]$

where $A[i]$ = list of all $x \in S$ with $h(x) = i$

("bucket")



query(y): do linear search in $A[h(y)]$

Ex: $h(x) = x \bmod m$

if input is rand, unif. distrib,

each bucket has $\sim \frac{n}{m}$ elems "on average"

choose $m \approx n \Rightarrow$ $O(1)$ "average" query time

$O(n)$ space

but don't want to assume input is random!

key idea [Carter, Wegman '77]

- <u>randomize</u> choice of hash fn h.

<u>Ex</u> Assume U is prime.

Pick rand $a \in [U] - \{0\}$.
$b \in [U]$

Define $h_{a,b} : [U] \to [m]$

$$h_{a,b}(x) = ((ax+b) \bmod U) \bmod m$$

$O(1)$ time

Prop For any fixed $x, y \in [U]$, $x \neq y$,

$$\Pr_{a,b}[h_{a,b}(x) = h_{a,b}(y)] \leq O\left(\frac{1}{m}\right)$$

called universality