Amortized Analysis: Union-Find 25.10.15

We saw it reported in Sedgewick & Wayne's slides that:

- logically treating the collection of sets as a forest
- actually representing the forest as an array (of parent pointers).
- implementing Union-by-rank and path-compression yields $\Theta(\alpha(m,n))$ amortized running time!

(m: # ops, n: # elements)

$\alpha(m,n)$ is "inverse Ackerman's" function which grows so slowly that if m, n are $\leq$ # atoms in universe (10^{80})

then $\alpha(m,n) \leq 4$

I.e., amortized running time of $\Theta(\alpha(m,n))$ is, for all practical purposes, like $\Theta(1)$.

It takes a bit of work to show the $\Theta(\alpha(m,n))$ bound. We will show a different but similar bound.

Defn: $\lg^{(i)} n = \begin{cases} n & \text{if } i = 0 \\ \lg(\lg^{(i-1)} n) & i > 0 \text{ and } \lg^{(i-1)} n > 0 \\ \text{undefined} & i > 0 \text{ and } \lg^{(i-1)} n < 0 \text{ or } \lg^{(i-1)} \text{ is undefined.} \end{cases}$

Defn: $\lg^* n = \min\{ i \geq 0, \lg^{(i)} n \leq 1 \}$

$\lg^* n$ is inverse of repeated exponentiation.

2^{65536} is way more than the number of atoms in the universe (2^{82})

$\lg 2^{65536} = 65536$

$\lg 65536 = 16$

$\lg 16 = 4$

$\lg 4 = 2$

$\lg 2 = 1$

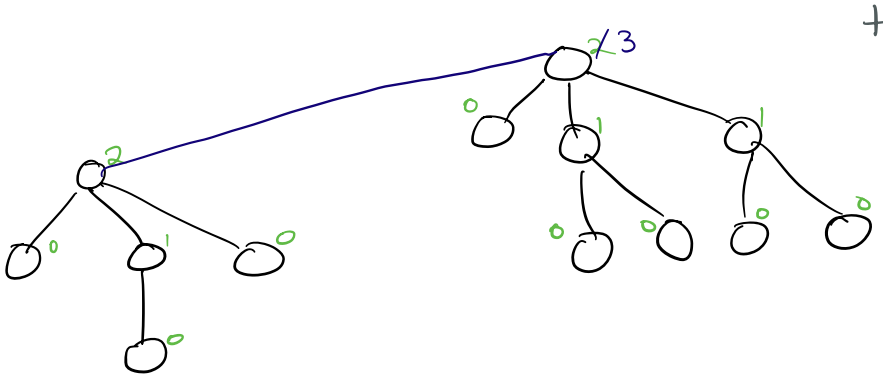
$\lg 1 = 0$

How many times do we apply lg to 2^{65536} till we get to ≤ 1? 5

$\therefore \lg^* 2^{65536} = 5$

Indeed, $\lg^* n \leq 5$ $\forall$ integers we will usually encounter in our work, so proving that an alg runs in $\lg^* n$ amortized time means practically it runs "like constant amortized

time "



- rank of a singleton tree is 0
- rank after union op is
  - same if one tree has smaller rank than other (smaller ranked tree becomes child)
  - inc by 1 if trees have same rank.

Easy to show: (Cormen, Leiserson, Rivest & Stein, Algorithms)

Lemma 22.3 (CLRS) $\forall$ tree roots x,

$size(x) \geq 2^{rank(x)}$

Proof: use induction on number of Link operations, where Link is linking root of lower rank tree to x (x is parent - why?)

Exercise for the student.

Lemma 22.2 (CLRS)

rank(x) starts at 0, can only increase while x is a root, and does not change once

x becomes a child.

ranks increase as you traverse up a tree.

Lemma 22.4 (CLRS)

$\forall$ integer $r \geq 0$, $\exists \leq \frac{n}{2^r}$ nodes of rank r.

Proof: Fix r.

Suppose that, in the course of things, whenever a root X gets rank r, X paints all the nodes in its tree X-coloured

$\therefore$ by Lemma 22.3, $\geq 2^r$ nodes are painted each time a root gets rank r.

Can a node a be painted twice?

No ~ will never paint a node twice.

Since no node can be painted twice,

- there are $\leq n$ painted nodes,
- Each colour-class has size $\geq 2^r$ by Lemma 22.3

$\therefore \exists \leq \frac{n}{2^r}$ colour classes

$\therefore \leq \frac{n}{2^r}$ nodes ever get rank r. ▨

Corollary: $\forall$ nodes have rank $\leq \lfloor \lg n \rfloor$.

Let us consider a sequence of operations...

m' ops:

MakeSet(30), Makeset(·), ... Makeset(50), Union(30, 50), Find(20), ...

replace Union with this. →

Find(30), Find(50), Link(r_1, r_2) ← m is new # ops in sequence

What does that do to m', the number of ops? no more than triples it, to become $m \leq 3m'$.

If we can show the sequence runs in $O(m \lg^* n)$ time, then it runs in $O(m' \lg^* n)$ time.

So we will consider our sequence as being of operations i.e. pull "Find" ops out of the "Union" ops

MakeSet(-), Find(-), Link(-,-)

Theorem: A sequence of m MakeSet, Find, Link ops performed using path compression and union-by-rank runs in worst-case $O(m \lg^* n)$ time.

Proof

- We assess charges to each operation corresponding to actual cost of each operation.

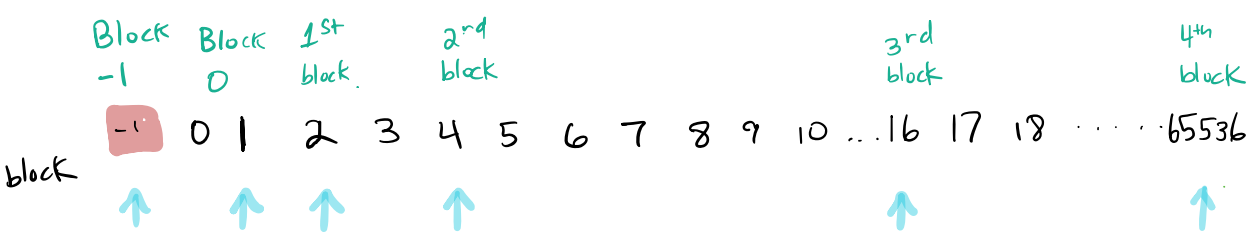
MakeSet – one charge per op'n

Link – one charge per op'n.

For Find:

- note that number of charges = number of parent pointers altered to point to a new parent of ~~same~~ or greater rank than old parent

Ranks will be considered to fall into Blocks



5th block.

65537 ... ... 2^{65536}

$B_{-1} = -1$

$B_0 = 1$

$B_1 = 2$

$B_2 = 4$

$B_3 = 16$

$B_4 = 65536$

Block	
-1	
0	-1+1 ... 1
1	2 ... 2
2	3 ... 4
3	5 ... 16
4	17 ... 65536
5	

Ranks will be considered to fall into Blocks

We will now use an Accounting Method

- all work is charged to the "account" of work done during the course of m operations, Makeset(-), Find(-), Link(-,-) where n of the ops are makeset(-).
- a "charge" will count for any constant amount of work.

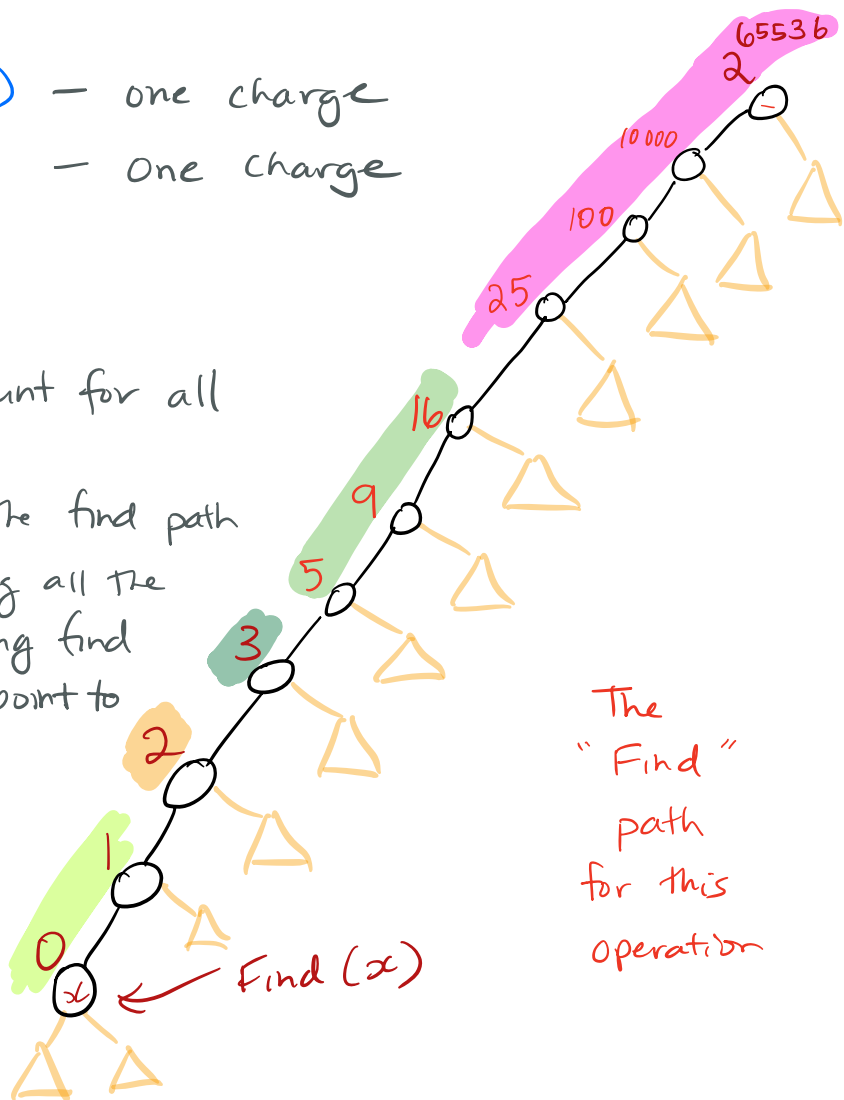
Makeset(-) — one charge

Link(-,-) — one charge

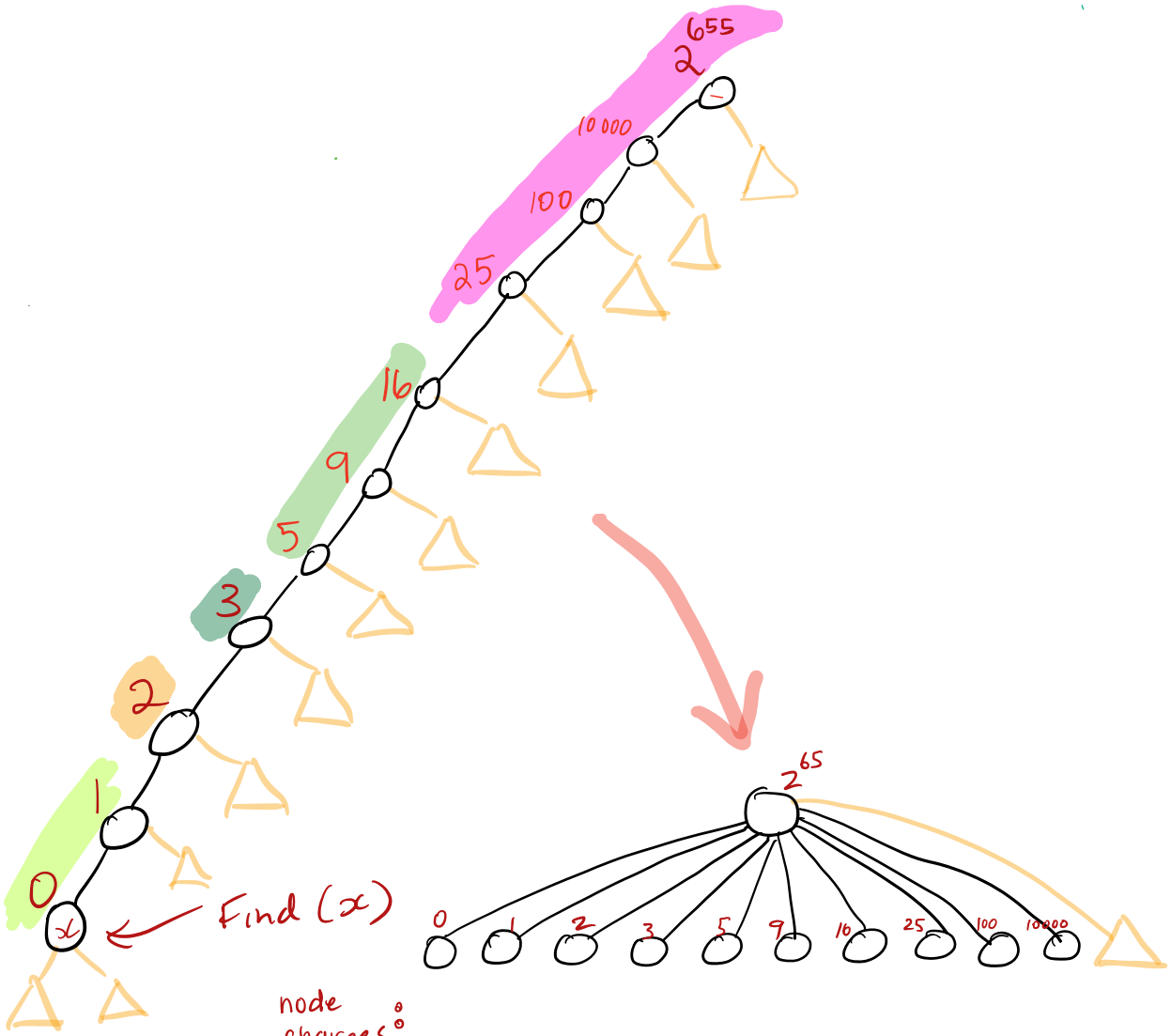
"Find" charges

- need to account for all costs of
  - climbing the find path
  - reassigning all the edges along find path to point to root

$\in O($number of parent pointers reassigned $+ 1)$



The "Find" path for this operation



node charges:

FIND: charges

The parent-pointer reassignment is charged to the node if its
- was not a child of the root
- its parent had rank in same block

Charged to the FIND otherwise.

← How many charges to the FIND? $\leq \lg^* n$

We only charged the FIND part of the work
We now count up all the charges that were attributed to nodes and amortize them over all the operations.

Note that...

- once a node is made a child of another node, its rank is frozen
- for any node x, the "rank of parent of x" is a "strictly" increasing function — each parent-pointer reassign is to a larger rank.
- once "rank of parent of x" is Outside x's block then x will never again accumulate a charge.

$\Rightarrow$ the most charges a node x can accumulate is Size of x's rank block.

$B_j - B_{j-1} - 1$

$\Rightarrow$ adding up over all blocks, and recalling $\exists \leq \frac{n}{2^r}$ nodes of rank r:

$$N(j) \leq \sum_{r=B_{j-1}+1}^{B_j} \frac{n}{2^r}$$

For $j=0$ $\quad N(j) = \frac{n}{2^0} + \frac{n}{2^1} = \frac{3n}{2}$

$= \frac{3n}{2B_0}$

For $j>0$, $N(j) \leq \frac{n}{2^{B_{j-1}+1}} \cdot \sum_{r=0}^{B_j - B_{j-1}} \frac{1}{2^r}$

$\leq 2 \cdot \frac{n}{2^{B_{j-1}+1}} = \frac{n}{2^{B_{j-1}}} = \frac{n}{B_j} \leq \frac{3n}{2B_j}$

$$\Rightarrow \quad N(j) \le \frac{3n}{2B_j} \quad \forall\, j \ge 0.$$

Now we add up over all blocks j, each x in the block can accumulate $\le B_j - B_{j-1} - 1$ charges

Total node charges for all n nodes $\le$

$$\sum_{j=0}^{\lg^* n - 1} \frac{3n}{2B_j} \cdot \left(B_j - B_{j-1} - 1\right)$$

$\frac{3n}{2B_j}$: # nodes

$B_j - B_{j-1} - 1$: max # of times parent ptr reassigned within block

$$\le \sum_{j=0}^{\lg^* n - 1} \frac{3n}{2B_j} \cdot B_j$$

$$\le 3n \cdot \lg^* n .$$

i.e. $O(\lg^* n)$ for each node (each MakeSet op)

The number of ops is – n MakeSet ops

m ops of all kinds (makeset, Link, Find)

$\therefore\ n \le m$

Add the amortization of the node charges to all the MakeSets

	Direct	+ Node Charges	Total
MakeSet	$O(1)$	$O(\lg^* n)$	$O(\lg^* n)$
Link	$O(1)$		$O(1)$
Find	$O(\lg^* n)$		$O(\lg^* n)$

$\Rightarrow$ The amortized running time of the sequence of m union-find operations is $O(\lg^* n)$ per operation.

$N(j) \leq \frac{3n}{2B(j)}$.

($^{\max}$ (number of nodes with rank in j^{th} block))

Let $P(n)$ = number of path charges

$$P(n) \leq \sum_{j=0}^{\lg^* n - 1} \frac{3n}{2B(j)} \left(B(j) - B(j-1)\right)$$

upper bound on number of nodes with rank $\in B(j)$

upper bound on number of ranks the node's parent can have
- each path charge comes with a node's reparenting within the block.

$$\leq \frac{3n}{2} \lg^* n$$

∴ Total # of path charges is $\leq \frac{3}{2} n \lg^* n$

Total # of block charges is $\leq m \lg^* n$

Also, $n \leq m$, so total # charges is $O(m \lg^* n)$

n = # makesets ↑ ops

$n \leq$ # effective link ops

Corollary: A sequence of m MakeSet(_) Union(_,_) Find() operations, n of which are MakeSet(), can be performed on the disjoint-set-forest implementation of Union-Find (using union-by-rank and path compression) in worst case $O(\lg^* n)$ amortized time.

Going back to that mysterious page...

Hence $N(j) \leq \frac{3n}{2B(j)}$

Let $P(n)$ = number of path changes

$$P(n) \leq \sum_{j=0}^{\lg^* n - 1} \frac{3n}{2B(j)} \left(B(j) - B(j-1)\right)$$

# of different blocks

upper bound on number of nodes with rank $\in B(j)$

upper bound on number of ranks the nodes parent can have
- each path change comes with a new parent-rank within the block.

$$\leq \frac{3n}{2} \lg^* n$$

∴ Total # of path changes $ is $\leq \frac{3}{2} n \lg^* n$

Total # of block changes € is $\leq m \lg^* n$

Also, $n \leq m$, so total # charges is $O(m \lg^* n)$

How many blocks can there be for n-element forest?

rank r is in block $\lg^* r$, for

$r = 0, 1, \ldots, \lfloor \lg n \rfloor$ ← $\lfloor \lg n \rfloor$ is maximum rank.

The highest numbered block is

$$\lg^*(\lg n) = \lg^* n - 1.$$