UNION BY SIZE ANALYSIS

ARUN KONAGURTHU
FIT3155

Using union-by-size, any root node's height has the following relationship with the size (number of elements) in/of the tree:

root node ← height[r]

$$\text{Size}[r] \geq 2^{\text{height}[r]}$$

This can be proved inductively:

Base case: $\text{size}[r] = 1$, $\text{height}[r] = 0$;
$\Rightarrow 2^{\text{height}[r]} = 2^0 = 1$

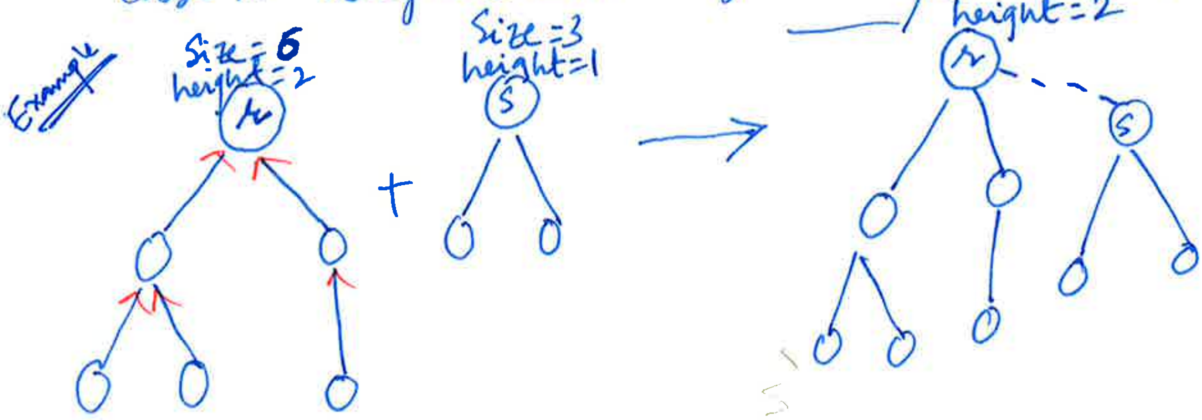
Singleton tree (basecase)
(r)
size=1
height=0

Inductive case:
Assume statement is true for/after 'k' union operations.

$$\text{Size}[r] \geq 2^{\text{height}[r]} \quad \text{(assumption)}$$

Let another tree rooted at 's' is unioned/merged with the current tree rooted at 'r'.

Case 1: $\text{height}[r] > \text{height}[s]$



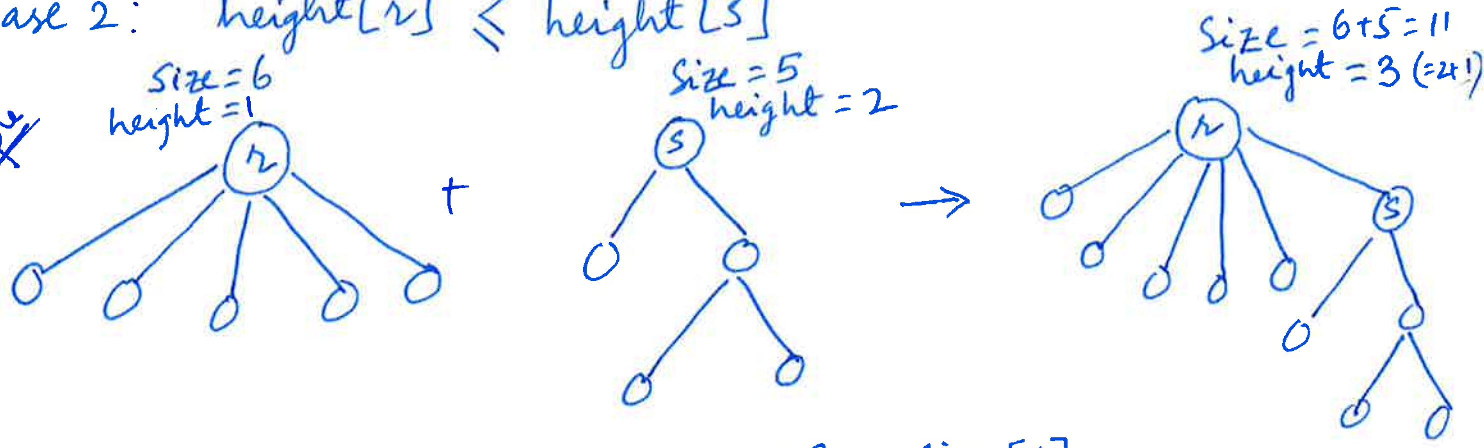
$$\text{(new) Size}[r] \geq \text{(old) Size}[r]$$
$$\geq 2^{\text{(old) height}[r]} \quad \text{(Inductive step)}$$

But $\text{(new) height}[r] = \text{(old) height}[r]$

$$\Rightarrow \text{(new) Size}[r] \geq 2^{\text{(new) height}[r]} \quad \blacksquare$$

Case 2: height[r] $\leq$ height[s]

Example



$$\text{(new) size}[r] = \text{(old) size}[r] + \text{size}[s]$$
$$\geq 2 \times \text{size}[s]$$
$$\geq 2 \times 2^{\text{height}[s]} \quad \text{(Inductive step)}$$
$$\geq 2^{\text{height}[s]+1} \quad \text{height}[s]+1 = \text{(new) height}[r]$$

$$\Rightarrow \text{(new) size}[r] \geq 2^{\text{(new) height}[r]} \quad \blacksquare$$

So, this proves that for any tree under union-by-size

$$\text{size}[r] \geq 2^{\text{height}[r]}$$

But in worst-case size[r] is bounded by 'n' (Total number of elems)

$$n \geq 2^{\text{height}[r]} \quad \text{(worst case)}$$

$$\log_2 n \geq \text{height}[r]$$

$\Rightarrow$ Find (any element) is $O(\text{height}[r]) = O(\log_2 n)$

Union (two elements) involves 2 Find operations + $O(1)$ effort

$$= O(\log_2 n)$$

Union by rank (height) Analysis

(WITHOUT PATH COMPRESSION)

OBSERVATION 1:

For any element 'x' $\boxed{\text{not}}$ = root node,
rank[x] < rank[parent[x]]

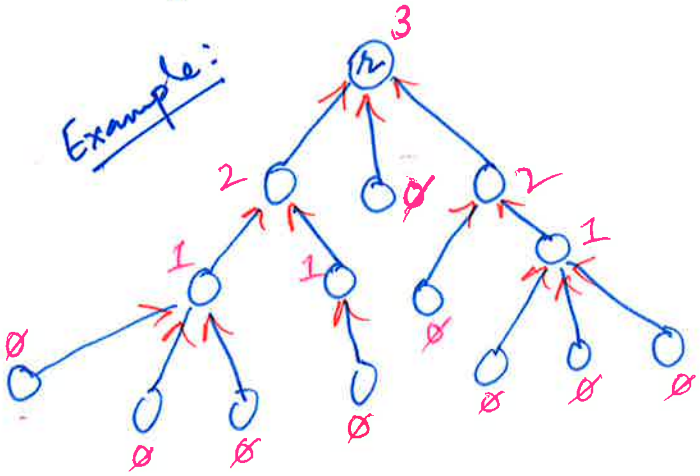
OBSERVATION 2:

For any element 'x' $\boxed{\text{not}}$ = root, rank[x] does NOT change under further union operations.

OBSERVATION 3:

- For any path from a node 'x' to root, the ranks of nodes along that path is a strictly increasing sequence.

Example:



ranks in red

Lemma 1: Any node with rank 'k' has $\geq 2^k$ nodes in its (SUB) tree.

Inductive Proof:

Base case: when $k=0$ (tree with singleton (root) node)
$2^0 = 1$ (TRUE).

Inductive case: Assume true for some rank $k-1$.
A node of rank k is created only when merging 2 trees with rank $k-1$ each.
But size of each rank $= k-1$ tree has $\geq 2^{k-1}$ nodes (Inductive step)
$\Rightarrow$ size of (rank $=k$) tree $\geq 2 \times 2^{k-1} = 2^k$ ■

COROLLARY 1: From OBSERVATION 1 and LEMMA 1, the heighest rank for any tree (under union-by-rank) operations is $\leq \log_2 n$ (where n is the total # of elems)

Lemma 2: (AKA "rank lemma")

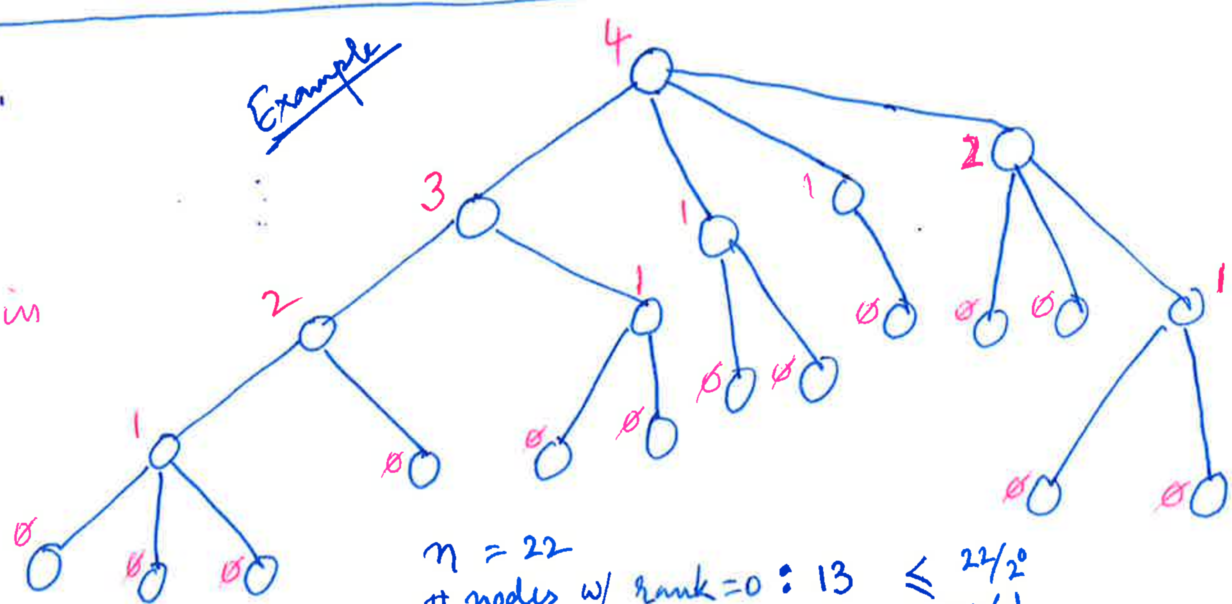
For any rank $k \geq 0$, there are $\leq \frac{n}{2^k}$ nodes in the tree with rank $= k$.

Proof:

① From Lemma 1, ~~any~~ each element in the tree with rank $= k$ has $\geq 2^k$ elements in its (sub)tree.

② Different ~~nodes~~ elements/nodes with same rank cannot have common descendants (from Observation 1)

∴ From the above 2 points, if we only have a total of 'n' elements, and each rank $= k$ ~~element~~ node has AT LEAST 2^k elements (from ①), and they do not share common descendants (from ②), then it follows that there are AT MOST $\frac{n}{2^k}$ such elements/nodes ■



ranks in red

$n = 22$

\# nodes w/ rank $= 0$: 13 $\leq 22/2^0$

\# nodes w/ rank $= 1$: 5 $\leq 22/2^1$

\# nodes w/ rank $= 2$: 2 $\leq 22/2^2$

\# nodes w/ rank $= 3$: 1 $\leq 22/2^3$

\# nodes w/ rank $= 4$: 1 $\leq 22/2^4$

Hopcroft - Ullman's Theorem

~~Theorem~~

statement: Starting from a fully disjoint set of n elements ~~Uniq~~ for any sequence of $m \geq n$ Find/Union operations using (1) Union by rank/height & (2) path compression, the Complexity for this sequence of m operations is $O(m \underbrace{\log^* n}_{})$.

↳ Iterated Logarithm function.

Before proving:

Definition of Iterated Log function ($\log^*$)

$$\log^* n = \begin{cases} 0 & \text{if } n \leq 1 \\ 1 + \log^*(\log n) & \text{otherwise.} \end{cases}$$

By this definition

$n = 1$, $\log^*(1) = 0$

$n = 2$, $\log^*(2) = 1$

$n \in [3, 4]$, $\log^*(n) = 2$

$n \in [5, 16]$, $\log^*(n) = 3$

$n \in [17, 65536]$, $\log^*(n) = 4$

$n \in [65537, 2^{65536}]$, $\log^*(n) = 5$

SIDE NOTE: # of atoms in the universe estimated $< 2^{270}$:)

BOTTOM LINE: $\log^*(n)$ is an <u>extremely</u> slowly growing function.

Hopcroft-Ullman's theorem proof (union-by-rank with Path compression)

① Divide up ranks into the following "RANK BLOCKS"

BLOCK 1 : Rank $k = \{1\}$
BLOCK 2 : Rank $k = \{2\}$
BLOCK 3 : Ranks $k = \{3, 4\}$
Block 4 : Ranks $k = \{5, 6, \dots 16\}$
BLOCK 5 : Ranks $k = \{17, 18, \dots, 2^{16}\}$
BLOCK 6 : Ranks $k = \{2^{16}+1, 2^{16}+2, \dots, 2^{65536}\}$
⋮ ⋮

There are only $\log^*(n)$ possible rank blocks for a given 'n'

② The Same properties/observations that were discussed for union-by-rank (without path compression; see pages 3-4) also holds here (with path compression)

OBSERVATION 1: If node 'x' ≠ root, rank[x] < rank[parent[x]]
OBSERVATION 2: If node 'x' ≠ root, rank[x] does NOT change.
OBSERVATION 3: Ranks along any path from 'x' to root, follows a strictly increasing sequence
OBSERVATION 4: Any node with rank = k has $\geq 2^k$ nodes in its (SUB) tree.
OBSERVATION 5: Highest rank of a node is $\leq \log_2 n$
OBERVATION 6: For any integer $k \geq 0$, there are $\leq \frac{n}{2^k}$ nodes with rank k.

③ Running Time of Find/Union is bounded by number of parent pointers traversed/followed. Two cases arise:

Case 1: rank(parent[x]) is in a higher RANK BLOCK than rank(x)

Case 2: rank(parent[x] is in the same RANK BLOCK as rank[x].

$\boxed{\text{Case 1}}$: rank[parent[x]] is in a bigger rank block than rank[x]

Since there are at most $\log^*(n)$ rank blocks, there can only be $O(\log^*(n))$ nodes that satisfy this property. For m find/union operations, this becomes $O(m \log^* n)$.

$\boxed{\text{Case 2}}$: rank[parent[x]] is in the same rank block as rank[x]

↳ Fix some arbitrary rank block: $\{k+1, k+2, \ldots, 2^k\}$

↳ The number of times a node in this rank block gets visited before ~~#~~ its parent goes into a different/next rank block (over union operations) is $\leq 2^k$

↳ But, using "rank lemma" (Page 4), the total # of elements/nodes in the tree whose rank lies in this rank block is bounded by:

$$\frac{n}{2^{k+1}} + \frac{n}{2^{k+2}} + \frac{n}{2^{k+3}} + \cdots + \frac{n}{2^{2^k}} \leq \frac{n}{2^k}$$

↳ $\Rightarrow$ Total visits to nodes whose ranks lie in this rank block is $\leq \frac{n}{2^k} \times 2^k \Rightarrow \leq n$

↳ But we only have at most $O(\log^* n)$ rank blocks

↳ Total work for this case $O(n \log^* n)$

SUMMING UP

Total work covering cases 1 & 2 above is:

$$O(m \log^* n) + O(n \log^* n)$$

[note $m \geq n$ in this theorem]

$$= O(m \log^* n)$$

NOT EXAMINABLE (but worth knowing)

Tarjan (another brilliant contributor to Computer Science) showed that, Hopcroft-Ullman's bound is in fact loose, and that it can be shown that any 'm' sequence of find/union operation grows as

~~O(m log~~ $$O(m\,\underbrace{\alpha(n)})$$

where $\alpha(n)$ is the Inverse Ackerman function

Compare this with Hopcroft-Ullman's bound of

$$O(m\,\underbrace{\log^*(n)})$$

Iterated log function.

↳ It turns out that

$\alpha(n)$ grows excruciatingly more slowly than $\log^*(n)$ ...

↳ ... and, in fact, $O(m\,\alpha(n))$ is the optimal bound for 'disjoint-set data structure, amortised over 'm' find/union operations.

END OF ANALYSIS