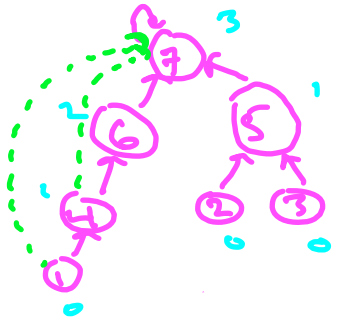
Advanced Union-Find

Path Compression

Algorithms: Design and Analysis, Part II

Path Compression



Idea: why bother traversing a leaf-root path multiple times?

Path Compression: after FIND(x), install shortcuts (i.e., revise parent pointers) to x's root all along the $x \rightsquigarrow$ root path.

In array representation:

1	2	3	4	5	6	7
4	5	5	6	7	7	7

$\rightarrow$

1	2	3	4	5	6	7
7	5	5	7	7	7	7

Con: constant-factor overhead to FIND (from "multitasking")

Pro: speeds up subsequent FINDs. (but by how much?)

On Ranks

Important: maintain all rank fields EXACTLY as without path compression.

- ranks initially all 0
- in UNION, new root = old root with bigger rank
- when merging two nodes of common rank r, reset new root's rank to $(r+1)$

Bad news: now rank[x] is only an upper bound on the maximum number of hops on a path from a leaf to x (which could be much less)

Good news: Rank Lemma still holds ($\leq \frac{n}{2^r}$ objects with rank r)

Also: still always have rank[parent(x)] > rank(x) for all non-roots x

Hopcroft-Ullman Theorem

Theorem: [Hopcroft-Ullman 73] with Union by Rank and path compression, m Union + Find operations take $O(m \log^* n)$ time, where

$\log^* n$ = the number of times you need to apply log to n before the result is ≤ 1.

Quiz on $\log^*$

Question: what is $\log^*(2^{65536})$?

(A) 2

(B) 5

(C) 16

(D) 65536

In general:

$\log^*(2^{2^{\cdot^{\cdot^{2}}}}) = t$ (t twos)

Measuring Progress