Advanced Union-Find

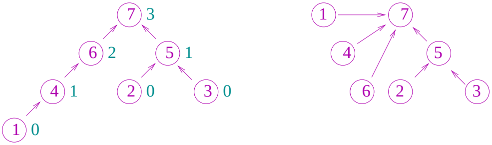
Path Compression

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 $x \rightarrow$ root path.



In array representation:

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

$\longrightarrow$

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

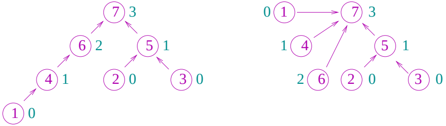
Con: Constant-factor overhead to FIND (from "multitasking").

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 ($\leq n/2^r$ 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) $\boxed{5}$

C) 16

D) 65536

In general: $\log^*(2^{2^{\cdots\ t \text{ times}\ \cdots^{2}}}) = t$

Measuring Progress