Analysis with Improved Link(x,y)

First of all, the height of a tree is at most the rank of its root. Second, a tree of rank j contains at least 2^j nodes. Proof by induction: both true for $j = 0$.

Consider first a Link(x,y) that does not modify ranks (say, rank of x is higher). Then the height of the subtree of y is at most $rank(y) < rank(x)$, and the longest path from x to a node in the subtree of y is at most $1 + rank(y) \leq rank(x)$.

And when the rank goes from $j - 1$ to j, the height of the tree increases by at most 1. Also we union two disjoint trees with at least 2^{j-1} nodes each, for a total of at least 2^j nodes.

As the number of nodes in the tree is at most n, we get $2^j \leq n$, or $j \leq \log_2 n$. This makes Find(x) run in $O(\log n)$ time (worst-case).

Path compression doubles the time of Find(x), but saves over the long term.

Worst-case, Find(x) is still $\Theta(\log n)$ each but a sequence of m operations with n elements will take at most $(m+n)\alpha(m,n)$ running time, where $\alpha(m,n)$ grows very very slowly. Analysis to follow.