★ 21.4 Analysis of union by rank with path compression

As noted in Section 21.3, the running time of the combined union-by-rank and path-compression heuristic is $O(m\,\alpha(n))$ for m disjoint-set operations on n elements. In this section, we shall examine the function α to see just how slowly it grows. Then we prove this running time using the potential method of amortized analysis.

A very quickly growing function and its very slowly growing inverse

For integers $k \geq 0$ and $j \geq 1$, we define the function $A_k(j)$ as

$$A_k(j) = \begin{cases} j+1 & \text{if } k = 0\,, \\ A_{k-1}^{(j+1)}(j) & \text{if } k \geq 1\,, \end{cases}$$

where the expression $A_{k-1}^{(j+1)}(j)$ uses the functional-iteration notation given in Section 3.2. Specifically, $A_{k-1}^{(0)}(j) = j$ and $A_{k-1}^{(i)}(j) = A_{k-1}(A_{k-1}^{(i-1)}(j))$ for $i \geq 1$. We will refer to the parameter k as the ***level*** of the function A.

The function $A_k(j)$ strictly increases with both j and k. To see just how quickly this function grows, we first obtain closed-form expressions for $A_1(j)$ and $A_2(j)$.

Lemma 21.2
For any integer $j \geq 1$, we have $A_1(j) = 2j + 1$.

Proof We first use induction on i to show that $A_0^{(i)}(j) = j + i$. For the base case, we have $A_0^{(0)}(j) = j = j + 0$. For the inductive step, assume that $A_0^{(i-1)}(j) = j + (i - 1)$. Then $A_0^{(i)}(j) = A_0(A_0^{(i-1)}(j)) = (j + (i - 1)) + 1 = j + i$. Finally, we note that $A_1(j) = A_0^{(j+1)}(j) = j + (j + 1) = 2j + 1$. ∎

Lemma 21.3
For any integer $j \geq 1$, we have $A_2(j) = 2^{j+1}(j + 1) - 1$.

Proof We first use induction on i to show that $A_1^{(i)}(j) = 2^i(j + 1) - 1$. For the base case, we have $A_1^{(0)}(j) = j = 2^0(j + 1) - 1$. For the inductive step, assume that $A_1^{(i-1)}(j) = 2^{i-1}(j + 1) - 1$. Then $A_1^{(i)}(j) = A_1(A_1^{(i-1)}(j)) = A_1(2^{i-1}(j + 1) - 1) = 2 \cdot (2^{i-1}(j+1)-1)+1 = 2^i(j+1)-2+1 = 2^i(j+1)-1$. Finally, we note that $A_2(j) = A_1^{(j+1)}(j) = 2^{j+1}(j + 1) - 1$. ∎

Now we can see how quickly $A_k(j)$ grows by simply examining $A_k(1)$ for levels $k = 0, 1, 2, 3, 4$. From the definition of $A_0(k)$ and the above lemmas, we have $A_0(1) = 1 + 1 = 2$, $A_1(1) = 2 \cdot 1 + 1 = 3$, and $A_2(1) = 2^{1+1} \cdot (1 + 1) - 1 = 7$. We also have

$$\begin{aligned} A_3(1) &= A_2^{(2)}(1) \\ &= A_2(A_2(1)) \\ &= A_2(7) \\ &= 2^8 \cdot 8 - 1 \\ &= 2^{11} - 1 \\ &= 2047 \end{aligned}$$

and

$$\begin{aligned} A_4(1) &= A_3^{(2)}(1) \\ &= A_3(A_3(1)) \\ &= A_3(2047) \\ &= A_2^{(2048)}(2047) \end{aligned}$$

$$\begin{aligned} &\gg A_2(2047) \\ &= 2^{2048} \cdot 2048 - 1 \\ &> 2^{2048} \\ &= (2^4)^{512} \\ &= 16^{512} \\ &\gg 10^{80} , \end{aligned}$$

which is the estimated number of atoms in the observable universe.

We define the inverse of the function $A_k(n)$, for integer $n \geq 0$, by

$$\alpha(n) = \min \{k : A_k(1) \geq n\} .$$

In words, $\alpha(n)$ is the lowest level k for which $A_k(1)$ is at least n. From the above values of $A_k(1)$, we see that

$$\alpha(n) = \begin{cases} 0 & \text{for } 0 \leq n \leq 2 , \\ 1 & \text{for } n = 3 , \\ 2 & \text{for } 4 \leq n \leq 7 , \\ 3 & \text{for } 8 \leq n \leq 2047 , \\ 4 & \text{for } 2048 \leq n \leq A_4(1) . \end{cases}$$

It is only for impractically large values of n (greater than $A_4(1)$, a huge number) that $\alpha(n) > 4$, and so $\alpha(n) \leq 4$ for all practical purposes.

Properties of ranks

In the remainder of this section, we prove an $O(m\,\alpha(n))$ bound on the running time of the disjoint-set operations with union by rank and path compression. In order to prove this bound, we first prove some simple properties of ranks.

Lemma 21.4

For all nodes x, we have $rank[x] \leq rank[p[x]]$, with strict inequality if $x \neq p[x]$. The value of $rank[x]$ is initially 0 and increases through time until $x \neq p[x]$; from then on, $rank[x]$ does not change. The value of $rank[p[x]]$ monotonically increases over time.

Proof The proof is a straightforward induction on the number of operations, using the implementations of MAKE-SET, UNION, and FIND-SET that appear in Section 21.3. We leave it as Exercise 21.4-1. ■

Corollary 21.5

As we follow the path from any node toward a root, the node ranks strictly increase. ■

Lemma 21.6
Every node has rank at most $n - 1$.

Proof Each node's rank starts at 0, and it increases only upon LINK operations. Because there are at most $n - 1$ UNION operations, there are also at most $n - 1$ LINK operations. Because each LINK operation either leaves all ranks alone or increases some node's rank by 1, all ranks are at most $n - 1$. ■

Lemma 21.6 provides a weak bound on ranks. In fact, every node has rank at most $\lfloor \lg n \rfloor$ (see Exercise 21.4-2). The looser bound of Lemma 21.6 will suffice for our purposes, however.

Proving the time bound

We shall use the potential method of amortized analysis (see Section 17.3) to prove the $O(m\alpha(n))$ time bound. In performing the amortized analysis, it is convenient to assume that we invoke the LINK operation rather than the UNION operation. That is, since the parameters of the LINK procedure are pointers to two roots, we assume that the appropriate FIND-SET operations are performed separately. The following lemma shows that even if we count the extra FIND-SET operations induced by UNION calls, the asymptotic running time remains unchanged.

Lemma 21.7
Suppose we convert a sequence S' of m' MAKE-SET, UNION, and FIND-SET operations into a sequence S of m MAKE-SET, LINK, and FIND-SET operations by turning each UNION into two FIND-SET operations followed by a LINK. Then, if sequence S runs in $O(m\,\alpha(n))$ time, sequence S' runs in $O(m'\,\alpha(n))$ time.

Proof Since each UNION operation in sequence S' is converted into three operations in S, we have $m' \leq m \leq 3m'$. Since $m = O(m')$, an $O(m\,\alpha(n))$ time bound for the converted sequence S implies an $O(m'\,\alpha(n))$ time bound for the original sequence S'. ■

In the remainder of this section, we shall assume that the initial sequence of m' MAKE-SET, UNION, and FIND-SET operations has been converted to a sequence of m MAKE-SET, LINK, and FIND-SET operations. We now prove an $O(m\,\alpha(n))$ time bound for the converted sequence and appeal to Lemma 21.7 to prove the $O(m'\,\alpha(n))$ running time of the original sequence of m' operations.

Potential function

The potential function we use assigns a potential $\phi_q(x)$ to each node x in the disjoint-set forest after q operations. We sum the node potentials for the potential of the entire forest: $\Phi_q = \sum_x \phi(x)$, where Φ_q denotes the potential of the forest after q operations. The forest is empty prior to the first operation, and we arbitrarily set $\Phi_0 = 0$. No potential Φ_q will ever be negative.

The value of $\phi_q(x)$ depends on whether x is a tree root after the qth operation. If it is, or if $rank[x] = 0$, then $\phi_q(x) = \alpha(n) \cdot rank[x]$.

Now suppose that after the qth operation, x is not a root and that $rank[x] \geq 1$. We need to define two auxiliary functions on x before we can define $\phi_q(x)$. First we define

$$\text{level}(x) = \max\{k : rank[p[x]] \geq A_k(rank[x])\} .$$

That is, level(x) is the greatest level k for which A_k, applied to x's rank, is no greater than x's parent's rank.

We claim that

$$0 \leq \text{level}(x) < \alpha(n) , \tag{21.1}$$

which we see as follows. We have

$$\begin{aligned} rank[p[x]] &\geq rank[x] + 1 && \text{(by Lemma 21.4)} \\ &= A_0(rank[x]) && \text{(by definition of } A_0(j)) , \end{aligned}$$

which implies that level$(x) \geq 0$, and we have

$$\begin{aligned} A_{\alpha(n)}(rank[x]) &\geq A_{\alpha(n)}(1) && \text{(because } A_k(j) \text{ is strictly increasing)} \\ &\geq n && \text{(by the definition of } \alpha(n)) \\ &> rank[p[x]] && \text{(by Lemma 21.6) ,} \end{aligned}$$

which implies that level$(x) < \alpha(n)$. Note that because $rank[p[x]]$ monotonically increases over time, so does level(x).

The second auxiliary function is

$$\text{iter}(x) = \max\left\{i : rank[p[x]] \geq A^{(i)}_{\text{level}(x)}(rank[x])\right\} .$$

That is, iter(x) is the largest number of times we can iteratively apply $A_{\text{level}(x)}$, applied initially to x's rank, before we get a value greater than x's parent's rank.

We claim that

$$1 \leq \text{iter}(x) \leq rank[x] , \tag{21.2}$$

which we see as follows. We have

$$\begin{aligned} rank[p[x]] &\geq A_{\text{level}(x)}(rank[x]) && \text{(by definition of level}(x)) \\ &= A^{(1)}_{\text{level}(x)}(rank[x]) && \text{(by definition of functional iteration) ,} \end{aligned}$$

which implies that $\text{iter}(x) \geq 1$, and we have

$$\begin{aligned} A_{\text{level}(x)}^{(rank[x]+1)}(rank[x]) &= A_{\text{level}(x)+1}(rank[x]) && \text{(by definition of } A_k(j)\text{)} \\ &> rank[p[x]] && \text{(by definition of level}(x)\text{)} \ , \end{aligned}$$

which implies that $\text{iter}(x) \leq rank[x]$. Note that because $rank[p[x]]$ monotonically increases over time, in order for $\text{iter}(x)$ to decrease, $\text{level}(x)$ must increase. As long as $\text{level}(x)$ remains unchanged, $\text{iter}(x)$ must either increase or remain unchanged.

With these auxiliary functions in place, we are ready to define the potential of node x after q operations:

$$\phi_q(x) = \begin{cases} \alpha(n) \cdot rank[x] & \text{if } x \text{ is a root or } rank[x] = 0 \ , \\ (\alpha(n) - \text{level}(x)) \cdot rank[x] - \text{iter}(x) & \text{if } x \text{ is not a root and } rank[x] \geq 1 \ . \end{cases}$$

The next two lemmas give useful properties of node potentials.

Lemma 21.8
For every node x, and for all operation counts q, we have

$$0 \leq \phi_q(x) \leq \alpha(n) \cdot rank[x] \ .$$

Proof If x is a root or $rank[x] = 0$, then $\phi_q(x) = \alpha(n) \cdot rank[x]$ by definition. Now suppose that x is not a root and that $rank[x] \geq 1$. We obtain a lower bound on $\phi_q(x)$ by maximizing $\text{level}(x)$ and $\text{iter}(x)$. By the bound (21.1), $\text{level}(x) \leq \alpha(n) - 1$, and by the bound (21.2), $\text{iter}(x) \leq rank[x]$. Thus,

$$\begin{aligned} \phi_q(x) &\geq (\alpha(n) - (\alpha(n) - 1)) \cdot rank[x] - rank[x] \\ &= rank[x] - rank[x] \\ &= 0 \ . \end{aligned}$$

Similarly, we obtain an upper bound on $\phi_q(x)$ by minimizing $\text{level}(x)$ and $\text{iter}(x)$. By the bound (21.1), $\text{level}(x) \geq 0$, and by the bound (21.2), $\text{iter}(x) \geq 1$. Thus,

$$\begin{aligned} \phi_q(x) &\leq (\alpha(n) - 0) \cdot rank[x] - 1 \\ &= \alpha(n) \cdot rank[x] - 1 \\ &< \alpha(n) \cdot rank[x] \ . \end{aligned}$$

■

Potential changes and amortized costs of operations

We are now ready to examine how the disjoint-set operations affect node potentials. With an understanding of the change in potential due to each operation, we can determine each operation's amortized cost.

Lemma 21.9

Let x be a node that is not a root, and suppose that the qth operation is either a LINK or FIND-SET. Then after the qth operation, $\phi_q(x) \leq \phi_{q-1}(x)$. Moreover, if $rank[x] \geq 1$ and either level(x) or iter(x) changes due to the qth operation, then $\phi_q(x) \leq \phi_{q-1}(x) - 1$. That is, x's potential cannot increase, and if it has positive rank and either level(x) or iter(x) changes, then x's potential drops by at least 1.

Proof Because x is not a root, the qth operation does not change $rank[x]$, and because n does not change after the initial n MAKE-SET operations, $\alpha(n)$ remains unchanged as well. Hence, these components of the formula for x's potential remain the same after the qth operation. If $rank[x] = 0$, then $\phi_q(x) = \phi_{q-1}(x) = 0$. Now assume that $rank[x] \geq 1$.

Recall that level(x) monotonically increases over time. If the qth operation leaves level(x) unchanged, then iter(x) either increases or remains unchanged. If both level(x) and iter(x) are unchanged, then $\phi_q(x) = \phi_{q-1}(x)$. If level$(x)$ is unchanged and iter(x) increases, then it increases by at least 1, and so $\phi_q(x) \leq \phi_{q-1}(x) - 1$.

Finally, if the qth operation increases level(x), it increases by at least 1, so that the value of the term $(\alpha(n) - \text{level}(x)) \cdot rank[x]$ drops by at least $rank[x]$. Because level(x) increased, the value of iter(x) might drop, but according to the bound (21.2), the drop is by at most $rank[x] - 1$. Thus, the increase in potential due to the change in iter(x) is less than the decrease in potential due to the change in level(x), and we conclude that $\phi_q(x) \leq \phi_{q-1}(x) - 1$. ■

Our final three lemmas show that the amortized cost of each MAKE-SET, LINK, and FIND-SET operation is $O(\alpha(n))$. Recall from equation (17.2) that the amortized cost of each operation is its actual cost plus the increase in potential due to the operation.

Lemma 21.10

The amortized cost of each MAKE-SET operation is $O(1)$.

Proof Suppose that the qth operation is MAKE-SET(x). This operation creates node x with rank 0, so that $\phi_q(x) = 0$. No other ranks or potentials change, and so $\Phi_q = \Phi_{q-1}$. Noting that the actual cost of the MAKE-SET operation is $O(1)$ completes the proof. ■

Lemma 21.11

The amortized cost of each LINK operation is $O(\alpha(n))$.

Proof Suppose that the qth operation is LINK(x, y). The actual cost of the LINK operation is $O(1)$. Without loss of generality, suppose that the LINK makes y the parent of x.

To determine the change in potential due to the LINK, we note that the only nodes whose potentials may change are x, y, and the children of y just prior to the operation. We shall show that the only node whose potential can increase due to the LINK is y, and that its increase is at most $\alpha(n)$:

- By Lemma 21.9, any node that is y's child just before the LINK cannot have its potential increase due to the LINK.
- From the definition of $\phi_q(x)$, we see that, since x was a root just before the qth operation, $\phi_{q-1}(x) = \alpha(n) \cdot rank[x]$. If $rank[x] = 0$, then $\phi_q(x) = \phi_{q-1}(x) = 0$. Otherwise,

  $$\begin{aligned} \phi_q(x) &= (\alpha(n) - \text{level}(x)) \cdot rank[x] - \text{iter}(x) \\ &< \alpha(n) \cdot rank[x] \qquad \text{(by inequalities (21.1) and (21.2))} . \end{aligned}$$

  Because this last quantity is $\phi_{q-1}(x)$, we see that x's potential decreases.
- Because y is a root prior to the LINK, $\phi_{q-1}(y) = \alpha(n) \cdot rank[y]$. The LINK operation leaves y as a root, and it either leaves y's rank alone or it increases y's rank by 1. Therefore, either $\phi_q(y) = \phi_{q-1}(y)$ or $\phi_q(y) = \phi_{q-1}(y) + \alpha(n)$.

The increase in potential due to the LINK operation, therefore, is at most $\alpha(n)$. The amortized cost of the LINK operation is $O(1) + \alpha(n) = O(\alpha(n))$. ∎

Lemma 21.12
The amortized cost of each FIND-SET operation is $O(\alpha(n))$.

Proof Suppose that the qth operation is a FIND-SET and that the find path contains s nodes. The actual cost of the FIND-SET operation is $O(s)$. We shall show that no node's potential increases due to the FIND-SET and that at least $\max(0, s - (\alpha(n) + 2))$ nodes on the find path have their potential decrease by at least 1.

To see that no node's potential increases, we first appeal to Lemma 21.9 for all nodes other than the root. If x is the root, then its potential is $\alpha(n) \cdot rank[x]$, which does not change.

Now we show that at least $\max(0, s - (\alpha(n) + 2))$ nodes have their potential decrease by at least 1. Let x be a node on the find path such that $rank[x] > 0$ and x is followed somewhere on the find path by another node y that is not a root, where $\text{level}(y) = \text{level}(x)$ just before the FIND-SET operation. (Node y need not *immediately* follow x on the find path.) All but at most $\alpha(n) + 2$ nodes on the find path satisfy these constraints on x. Those that do not satisfy them are the first node

on the find path (if it has rank 0), the last node on the path (i.e., the root), and the last node w on the path for which $\text{level}(w) = k$, for each $k = 0, 1, 2, \ldots, \alpha(n) - 1$.

Let us fix such a node x, and we shall show that x's potential decreases by at least 1. Let $k = \text{level}(x) = \text{level}(y)$. Just prior to the path compression caused by the FIND-SET, we have

$$\begin{aligned}
rank[p[x]] &\geq A_k^{(\text{iter}(x))}(rank[x]) && \text{(by definition of iter}(x)\text{)} ,\\
rank[p[y]] &\geq A_k(rank[y]) && \text{(by definition of level}(y)\text{)} ,\\
rank[y] &\geq rank[p[x]] && \text{(by Corollary 21.5 and because}\\
& && \quad y \text{ follows } x \text{ on the find path)} .
\end{aligned}$$

Putting these inequalities together and letting i be the value of $\text{iter}(x)$ before path compression, we have

$$\begin{aligned}
rank[p[y]] &\geq A_k(rank[y])\\
&\geq A_k(rank[p[x]]) \qquad \text{(because } A_k(j) \text{ is strictly increasing)}\\
&\geq A_k(A_k^{(\text{iter}(x))}(rank[x]))\\
&= A_k^{(i+1)}(rank[x])) .
\end{aligned}$$

Because path compression will make x and y have the same parent, we know that after path compression, $rank[p[x]] = rank[p[y]]$ and that the path compression does not decrease $rank[p[y]]$. Since $rank[x]$ does not change, after path compression we have that $rank[p[x]] \geq A_k^{(i+1)}(rank[x]))$. Thus, path compression will cause either $\text{iter}(x)$ to increase (to at least $i + 1$) or $\text{level}(x)$ to increase (which occurs if $\text{iter}(x)$ increases to at least $rank[x] + 1$). In either case, by Lemma 21.9, we have $\phi_q(x) \leq \phi_{q-1}(x) - 1$. Hence, x's potential decreases by at least 1.

The amortized cost of the FIND-SET operation is the actual cost plus the change in potential. The actual cost is $O(s)$, and we have shown that the total potential decreases by at least $\max(0, s - (\alpha(n) + 2))$. The amortized cost, therefore, is at most $O(s) - (s - (\alpha(n) + 2)) = O(s) - s + O(\alpha(n)) = O(\alpha(n))$, since we can scale up the units of potential to dominate the constant hidden in $O(s)$. ■

Putting the preceding lemmas together yields the following theorem.

Theorem 21.13

A sequence of m MAKE-SET, UNION, and FIND-SET operations, n of which are MAKE-SET operations, can be performed on a disjoint-set forest with union by rank and path compression in worst-case time $O(m\,\alpha(n))$.

Proof Immediate from Lemmas 21.7, 21.10, 21.11, and 21.12. ■