

1 UNION-FIND With Path Compression

We will study the amortized complexity of the UNION-FIND data structure with path compression. Our treatment is based on *A Potential-Based Amortized Analysis of the Union-Find Data Structure* by Harfst and Reingold, from SIGACT NEWS, Volume 31, Issue 3, September 2000.

The data structure does the following.

1. Holds disjoint sets which are represented by trees.
2. Every node is an element. The root of a tree is the name of that set.
3. Will assign to every node a rank. The rank will measure, in some sense, how many nodes are in the tree with that node as root. It will not be a precise measure of this, and in fact it may not even be close. But this definition works well with the proof

NOTE: The CLR or CLRS book have nice pictures and examples of this data structure.

2 INSERT

To INSERT x make x a 1-node tree. Assign Rank 0.

3 UNION

UNION x and y . Note that x and y are roots of trees and have ranks.

1. If $\text{rank}(x) < \text{rank}(y)$ then make x a child of y . Do nothing else. NOTE: you DO NOT increase the rank of y .
2. If $\text{rank}(x) > \text{rank}(y)$ then make y a child of x . Do nothing else. NOTE: you DO NOT increase the rank of x .
3. If $\text{rank}(x) = \text{rank}(y)$ then (arbitrarily) make y a child of x and (NOTE!) $\text{rank}(x) = \text{rank}(x) + 1$.

It may seem odd that we only increase the rank when there is a tie; however, this makes sure that the rank never gets bigger than $\lg n$. The following lemma is left to the reader.

Lemma 3.1 *Start with the empty data structure. If m operations are performed but at most n are INSERTS then, for all x , $\text{rank}(x) \leq \lg x$.*

Note 3.2 I do not see that we ever use Lemma 3.1. Extra Credit to anyone who can show that we need this.

4 FIND

Path compression: When you do a FIND on a node, the node and all its parents you find on the way to the root become children of the root (a sacrifice for the greater good).

(SEE CLRS for nice examples).

5 A Lemma on Ranks

Ranks are in strictly decreasing as you descend though the tree.

Lemma 5.1 *If x_1, \dots, x_L is a path going up the tree then*

$$\text{rank}(x_1) < \dots < \text{rank}(x_L).$$

Proof: Induction on the number of UNIONS performed.

Base Case: If no unions are performed then the only branches are of length 1 and the lemma is true vacuously.

Even though this is not needed, I will do the case where 1 union is performed. The first union is between two nodes of rank 0. Now the top one has rank 1 and the node below it has rank 0, so the lemma holds.

Induction Step: Assume true for m applications of UNION. Assume that you are just about to do the $(m + 1)$ st. You are going to UNION sets with roots x and y .

There are three cases.

1. $\text{rank}(x) < \text{rank}(y)$. Then x becomes a child of y and the ranks remain what they were. Since the induction hypothesis applies to the branch below x , and $\text{rank}(x) < \text{rank}(y)$, the lemma holds.
2. $\text{rank}(x) > \text{rank}(y)$. Similar to the above.
3. $\text{rank}(x) = \text{rank}(y)$. Then x becomes a child of y and the $\text{rank}(y) = \text{rank}(y) + 1$. Since the induction hypothesis applies to the branch below x , and given the increase in $\text{rank}(y)$, the lemma holds.

■

The following lemma looks helpful but we do not use it. The proof is left to the reader.

Lemma 5.2 *Start with the empty data structure. If m operations are performed but at most n are INSERTS then, for all x , the depth of x is $\leq \lg n$.*

6 Motivation Behind Potential Function

When we do a Path Compression this may take a lot of work. We want to somehow argue that the tree is so much better off afterwards that some potential function goes down enough to offset the work.

What nice thing happens to the data structure after a path compression? Several nodes are now children of the root when they were not before. By Lemma 5.1 these nodes now have a LARGER difference of rank with their parent than they had before.

We first want to define a potential function ϕ just for nodes. We will later sum them to get one for the entire structure. Given a node x we will want

$$\text{rank}(\text{parent}(x)) - \text{rank}(x) \text{ large} \Rightarrow \phi(x) \text{ small}$$

$$\text{rank}(\text{parent}(x)) - \text{rank}(x) \text{ small} \Rightarrow \phi(x) \text{ large}$$

Hence we need a function $F(a, b)$ that we intend to use for $F(\text{rank}(x), \text{rank}(\text{parent}(x)))$ such that

$$b - a \text{ large} \Rightarrow F(a, b) \text{ small}$$

$b - a$ small $\Rightarrow F(a, b)$ large

We also want $F(a, b) \geq 0$.

SO, our final goal is to come up with a function F with these properties and then define the potential Φ as follows:

$$\Phi = \sum_x F(\text{rank}(x), \text{rank}(\text{parent}(x)))$$

SO, what might F look like?

We cannot use $a - b$ since it is negative. What about $2a - b$? $3a - b$? $4a - b$? If a and b are ‘close enough’ then we can use $3a - b$.

We will define:

$$F(a, b) = 3a - b \text{ if } \lfloor \lg a \rfloor = \lfloor \lg b \rfloor$$

Lemma 6.1 *If $\lfloor \lg a \rfloor = \lfloor \lg b \rfloor$, $a < b$ then $2a - b \geq 1$.*

Proof: Teach CMSC 250 and assign it as homework. ■

Note 6.2 For the definition of F we only need that $3a - b \geq 1$. But for other reasons later we will need $2a - b \geq 1$.

What if $\lfloor \lg a \rfloor \neq \lfloor \lg b \rfloor$? What if they are still, given that fact, ‘somewhat close together’? We cut to the chase:

$$F(a, b) = a - \lfloor \lg b \rfloor \text{ if } \log^* a = \log^* b$$

Lemma 6.3 *If $a < b$, $\lg a < \lg b$, but $\log^* a = \log^* b$ then $a - \lfloor \lg b \rfloor \geq 1$*

Proof: Since $\log^* a = \log^* b$ there exists an i such that

$$\text{tow}(i) \leq a < b < \text{tow}(i + 1).$$

Take the \lg of that equation to obtain

$$\text{tow}(i - 1) \leq \lfloor \lg b \rfloor < \text{tow}(i) \leq a.$$

(The reader may question if we really get $\lfloor \lg b \rfloor < \text{tow}(i)$. This is true since $\text{tow}(i)$ is a power of 2 so the floor function does not effect its \lg at all.)

Putting it together we obtain $\lfloor \lg b \rfloor < \text{tow}(i) \leq a$. ■

We now define ϕ formally. We will not use the function F , we just used it above to demonstrate what we want our ϕ to satisfy.

Def 6.4 Definition of $\phi(x)$

(1) If x is root then

$$\phi(x) = 3\text{rank}(x).$$

(2) If $\lfloor \lg(\text{rank}(x)) \rfloor = \lfloor \lg(\text{rank}(\text{parent}(x))) \rfloor$ then

$$\phi(x) = 3\text{rank}(x) - \text{rank}(\text{parent}(x)).$$

(3) If $\lfloor \lg(\text{rank}(x)) \rfloor \neq \lfloor \lg(\text{rank}(\text{parent}(x))) \rfloor$ and $\log^*(\text{rank}(x)) = \log^*(\text{rank}(\text{parent}(x)))$ then

$$\phi(x) = \text{rank}(x) - \lfloor \lg(\text{rank}(\text{parent}(x))) \rfloor.$$

(4) if $\log^*(\text{rank}(x)) \neq \log^*(\text{rank}(\text{parent}(x)))$ then

$$\phi(x) = 0.$$

Def 6.5 Define $\Phi = \alpha \sum_x \phi(x)$ where we determine α later. This is our potential function.

Note 6.6 In the paper we are working out of they use, in case (3) above,

$$\phi(x) = \text{rank}(x) + \lfloor \lg(\text{rank}(x)) \rfloor - \lfloor \lg(\text{rank}(\text{parent}(x))) \rfloor.$$

The proof in these notes seems to work fine with the simpler function I have chosen. Extra credit if you can show me where my proof breaks down by not using that function.

7 INSERT Has Amortized Cost $O(1)$

INSERT is $O(1)$ actual cost. Adding a root of rank 0 causes

$$\Delta\Phi = 0.$$

Therefore the amortized cost is $O(1)$.

8 FIND Has Amortized Cost $O(\log^* n)$

Theorem 8.1 *Assume that the Data Structure has $\leq n$ elements in it. Using the potential function defined above, the amortized cost of FIND is $O(\log^* n)$.*

Proof:

Assume there are n elements in the data structure. Assume that a FIND is called and the element is L away from the root. Let the branch from the element to the root be

$$x_1, \dots, x_L$$

where x_1 is the element FIND was called on, x_L is the root. We denote x_L by root. We assume $L \geq 3$ (if $L \leq 2$ then it is easy to show that actual cost is $O(1)$ and $\Delta\Phi \leq 0$ so amortized cost is $O(1)$). Since $L \geq 3$ we have that $\text{rank}(\text{root}) \geq 2$.

When we are done ALL of x_1, \dots, x_{L-1} are direct children of the root. For each i , $0 \leq i \leq L-2$, $\phi(x_i)$ may change, perhaps alot. We will show that for all but $\log^* n + O(1)$ of the elements x on the branch,

$$\text{NEW } \phi(x) < \text{OLD } \phi(x).$$

Since $\phi(x)$ is integer values this means that NEW $\phi(x)$ is at least ONE less than OLD $\phi(x)$. Hence

$$\Delta\Phi = \alpha(L - (\log^* L + O(1))).$$

The actual work is $O(L)$. So the amortized cost is

$$O(L) - \alpha(L - \log^* L + O(1)) \leq O(L) - \alpha L + \alpha \log^* L + O(\alpha) \leq O(L) - \alpha L + \alpha \log^* n + O(\alpha).$$

One can pick α large enough so that this is $O(\log^* n)$.

There are several cases. Let $x = x_i$ and assume that $i \leq L-3$, so $\text{parent}(x)$ exists and is not the root.

Case I: $\lfloor \lg(\text{rank}(x)) \rfloor = \lfloor \lg(\text{rank}(\text{parent}(x))) \rfloor$. By the definition of ϕ

$$\text{OLD } \phi(x) = 3\text{rank}(x) - \text{rank}(\text{parent}(x)).$$

AFTER FIND: What is $\phi(x)$? The new parent of x is root. So the NEW $\phi(x)$ depends on how $\text{rank}(x)$ and $\text{rank}(\text{root})$ compare. There are cases.

1. $\lfloor \lg(\text{rank}(x)) \rfloor = \lfloor \lg(\text{rank}(\text{root})) \rfloor$: Then

$$\text{NEW } \phi(x) = 3\text{rank}(x) - \text{rank}(\text{root}).$$

By Lemma 5.1 $\text{rank}(\text{parent}(x)) < \text{rank}(\text{root})$. Hence NEW $\phi(x) <$
 OLD $\phi(x)$

2. $\lfloor \lg(\text{rank}(x)) \rfloor \neq \lfloor \lg \text{rank}(\text{root}) \rfloor$ but $\log^*(\text{rank}(x)) = \log^*(\text{rank}(\text{parent}(x)))$:
 Then

$$\text{NEW } \phi(x) = \text{rank}(x) - \lfloor \lg \text{rank}(\text{root}) \rfloor.$$

We want to show that

$$\text{NEW } \phi(x) < \text{OLD } \phi(x).$$

Hence we need the following to be positive:

$$\begin{aligned} & (\text{rank}(x) - \text{rank}(\text{parent}(x))) - (\text{rank}(x) - \lfloor \lg(\text{rank}(\text{root})) \rfloor) \\ &= 2\text{rank}(x) - \text{rank}(\text{parent}(x)) + \lfloor \lg \text{rank}(\text{root}) \rfloor. \end{aligned}$$

Since $\lfloor \lg(\text{rank}(x)) \rfloor = \lfloor \lg(\text{rank}(\text{parent}(x))) \rfloor$, by Lemma 6.1,

$$2\text{rank}(x) - \text{rank}(\text{parent}(x)) \geq 1.$$

Hence we have

$$2\text{rank}(x) - \text{rank}(\text{parent}(x)) + \lfloor \lg(\text{rank}(\text{root})) \rfloor \geq \lfloor \lg(\text{rank}(\text{root})) \rfloor.$$

Since $\text{rank}(\text{root}) \geq 2$, $\lfloor \lg \text{rank}(\text{root}) \rfloor \geq 1$. Hence the quantity is positive.

3. $\log^*(\text{rank}(x)) \neq \log^*(\text{rank}(\text{root}))$. In this case NEW $\phi(x) = 0$. Since
 OLD $\phi(x) \geq 1$ we have that NEW $\phi(x) <$ OLD $\phi(x)$.

Case II: $\lfloor \lg(\text{rank}(x)) \rfloor \neq \lfloor \lg(\text{rank}(\text{parent}(x))) \rfloor$ but $\log^*(\text{rank}(x)) = \log^*(\text{rank}(\text{parent}(x)))$.

By the definition of ϕ

$$\text{OLD } \phi(x) = \text{rank}(x) - \lfloor \lg \text{rank}(\text{parent}(x)) \rfloor.$$

Since $\text{rank}(\text{parent}(x)) < \text{rank}(\text{root})$ and $\lfloor \lg(\text{rank}(x)) \rfloor \neq \lfloor \lg(\text{rank}(\text{parent}(x))) \rfloor$,
 we must have $\lfloor \lg(\text{rank}(x)) \rfloor \neq \lfloor \lg(\text{rank}(\text{root})) \rfloor$. Hence this time we have only
 two cases.

1. $\log^*(\text{rank}(x)) = \log^*(\text{rank}(\text{root}))$:

NEW $\phi(x)$ is $\text{rank}(x) - \lfloor \lg(\text{rank}(\text{root})) \rfloor$.

We need for this to be LESS THAN the OLD ϕ . Hence we need

$$\begin{aligned} \text{rank}(x) - \lfloor \lg(\text{rank}(\text{root})) \rfloor &< \text{rank}(x) - \lfloor \lg(\text{rank}(\text{parent}(x))) \rfloor \\ - \lfloor \lg(\text{rank}(\text{root})) \rfloor &< - \lfloor \lg(\text{rank}(\text{parent}(x))) \rfloor \\ \lfloor \lg(\text{rank}(\text{parent}(x))) \rfloor &< \lfloor \lg(\text{rank}(\text{root})) \rfloor \end{aligned}$$

This might not always hold! We show that the number of times it is false is $O(1)$. So when is it false? Combining its falsity with the Case II condition we have

$$\lfloor \lg \text{rank}(x) \rfloor < \lfloor \lg \text{rank}(\text{parent}(x)) \rfloor = \lfloor \text{rank}(\text{root}) \rfloor$$

Hence $\text{parent}(x)$ is the element of the lowest rank such that

$$\lfloor \lg(\text{rank}(\text{parent}(x))) \rfloor = \lfloor \text{rank}(\text{root}) \rfloor .$$

There is only one such element. Hence this only happens once for all x in this case. For all the other x in this case we have

NEW $\phi(x) < \text{OLD } \phi(x)$.

2. $\log^*(\text{rank}(x)) \neq \log^*(\text{rank}(\text{root}))$. In this case NEW $\phi(x) = 0$. Since OLD $\phi(x) \geq 1$ we have that NEW $\phi(x) < \text{OLD } \phi(x)$.

Case 3: $\log^*(\text{rank}(x)) \neq \log^*(\text{rank}(\text{parent}(x)))$: For how many x can this occur? Imagine the ranks of the elements listed out:

$$\text{rank}(x_1) < \text{rank}(x_2) < \dots < \text{rank}(x_L)$$

Now imagine putting them into blocks as follows:

All i that have $\log^*(\text{rank}(x_i)) = 1$ go to the first block. Hence the first block has elements whose rank is in the set

$$\{1, 2, 3\}.$$

All i that have $\log^*(\text{rank}(x_i)) = 2$ go to the second block. Hence the first block has elements whose rank is in the set

$$\{4, 5, 6, \dots, 15\}.$$

All i that have $\log^*(\text{rank}(x_i)) = 3$ go to the second block. Hence the first block has elements whose rank is in the set

$$\{16, 17, 18, \dots, 2^{16} - 1\}.$$

All i that have $\log^*(\text{rank}(x_i)) = 3$ go to the third block.
etc.

Since $L \leq n$ there will be $\log^* L \leq \log^* n$ blocks.

Case 3 only happens when x has the largest rank in its block. This only happens $O(\log^* n)$ times. ■

Note 8.2 In Case 3 we used that the depth $L \leq n$ hence the rank is $\leq n$. In reality both the depth and the rank are $\leq \lg n$. We do not seem to need that in this proof. Extra credit for someone who can show where I need it in the proof.

9 UNION has Amortized Cost $O(1)$

Theorem 9.1 *Assume that the Data Structure has $\leq n$ elements in it. Using the potential function defined above, the amortized cost of UNION is $O(1)$.*

Proof: If a union is performed then the actual cost is $O(1)$. We need to prove that the potential increases by at most a constant.

Let x and y be the names of the sets. There are two cases.

Case I: $\text{rank}(x) < \text{rank}(y)$. In this case no ranks change. The node x goes from being a root from being a child of a y , so its ϕ value will change. No other nodes ϕ value changes.

OLD $\phi(x) = 3\text{rank}(x)$.

NEW $\phi(x)$ depends on how $\text{rank}(y)$ (the new parent of x) and $\text{rank}(x)$ compare.

1. $\lfloor \lg(\text{rank}(x)) \rfloor = \lfloor \lg(\text{rank}(y)) \rfloor$. Then

$$\text{NEW } \phi(x) = 3\text{rank}(x) - \text{rank}(y) < 3\text{rank}(x) = \text{OLD } \phi(x).$$

Hence $\Delta\Phi \leq 0$.

2. $\lfloor \lg(\text{rank}(x)) \rfloor < \lfloor \lg(\text{rank}(y)) \rfloor$ but $\log^* \text{rank}(x) = \log^* \text{rank}(y)$. Then
 NEW $\phi(x) = \text{rank}(x) - \lfloor \lg(\text{rank}(y)) \rfloor$.
 NEW $\phi(x)$ - OLD $\phi(x)$ is
 Hence we look at
 $(\text{rank}(x) - \lfloor \lg(\text{rank}(y)) \rfloor) - (3\text{rank}(x)) = -2\text{rank}(x) - \lfloor \lg(\text{rank}(y)) \rfloor \leq 0$.
3. $\log^* \text{rank}(x) \neq \log^* \text{rank}(y)$.
 NEW $\phi(x) = 0$ so the change in potential is ≤ 0 .

■

Note 9.2 The paper claims that amortized UNION takes $O(\log^* n)$. I get $O(1)$. Extra Credit who can tell me if I'm right or their right, and whats going on.