

This and other notes are available from <http://www.cs.berkeley.edu/~yozo/cs170.fa05/>

Union-Find Data Structure

- Rank by union. Tries to make tree depths shallow. If a node has rank k then the size of the tree rooted at this node is at most 2^k . (Why?)
- Path compression. Tries to shorten paths for free (in asymptotic sense) when we are visiting nodes.

Fun with $\log^* n$

- Compute $\log^* 2^{123}$. ($2^{123} \approx 10^{37}$)
- Give the smallest and largest number n such that $\log^* n = 4$.

Complexity Analysis of Union-Find Algorithm with Path-Compression

Assume there are n nodes (n make-set operations). Now separate the nodes into groups C_0, C_1, \dots , so that node with (final) rank r belongs to group $C_{\log^* r}$. So, for example, a node with rank 27 belongs to C_4 since $\log^* 27 = 4$. Key observations:

1. Path compression does not affect the rank of any node. (Why?)
2. For each rank r , there are at most $\frac{n}{2^r}$ nodes with rank r . (Why? *Hint*: Think about just using rank-by-union, then apply the above observation.)
3. There are at most $\log^* n$ (non-empty) groups. (Why?)
4. Each group consists of nodes of ranks $k+1, k+2, \dots, 2^k$ for some k . For example C_4 has nodes of rank 17, 18, $\dots, 256 = 2^{17-1}$.
5. The group above contains at most $\frac{n}{2^k}$ elements. This is because

$$\sum_{r=k+1}^{2^k} \frac{n}{2^r} \leq \sum_{r=k+1}^{\infty} \frac{n}{2^r} = \frac{n}{2^k} \left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \dots \right) = \frac{n}{2^k}$$

6. Give each item in the above group 2^k tokens. Then each group will have at most n tokens. Since there are $\log^* n$ groups, the total number of tokens given out is $n \log^* n$.
7. Rank of parent of x is strictly larger than the rank of x . (Why?)
8. Path compression can only make the rank of parent of x larger. (Why?)

For each link (u, v) that FIND traverses, there are two cases:

1. u and v are in the same group C_ℓ , consisting of elements of rank $k+1, k+2, \dots, 2^k$. In this case we let u pay one of its tokens. Since there are at most 2^k elements in C_ℓ , there can only be at most 2^k such v 's (for each u), so the pre-paid token suffices. (Why?)

2. u and v are in different group. This cost is assessed to the FIND operation. There are at most $\log^* n$ of these per find operation. (Why?)

Hence the total cost is $O(n \log^* n + m \log^* n) = O((n + m) \log^* n)$.

Questions

1. Let N_k be the largest number such that $\log^* N_k = k$. Show that

$$N_k = 2^{2^{\dots^2}}$$

where there are k twos on the right hand side. Compute N_1, N_2, N_3, N_4 , and N_5 . How many digits does N_5 have?

2. Is $\log \log^* n = O(\log^* \log n)$?
3. Suppose you have the vertices 1, 2, 3, 4, and 5. Suppose UNION(1, 3), UNION(2, 4), UNION(2, 3), FIND(3) is executed in this order. What is the result of this operation, using the algorithm presented in class with path-compression?
4. Why are we only concerned with FIND operations to find the complexity of m UNION and FINDS?
5. Answer all ‘Why?’ that appear in the notes.
6. Reorganize the above complexity analysis without using the idea of “tokens”.
7. More amortized analysis. Consider an implementation of growable list, where it is implemented as a simple array. Whenever the array becomes too short, it just allocates a new one of double the size and copies over all the elements. Assume no deletes. Give the time complexity of n inserts.