Disjoint Sets

Chapter 8 Overview

- **Equivalence Relations**
- **Dynamic Equivalence Classes**
- **Basic Union-Find Structure and Algorithms**
- **Smarter Union Algorithms**
- **Path Compression**
- **Complexity (not the proof)**
- **Applications (Efficient Connected Components)**

Representing Partitions

Any set S can be partitioned into a set of equivalence classes defined by some relation R.

Example:

S = { 1, 2, 3, 4, 5, 6, 7, 8}

R = {(x,y) | x mod 3 = y mod 3}

The equivalence relation R partitions S into 3 sets S1, S2, and S3 whose union is S.

$$\left.\begin{array}{l} S1 = \{1, 4, 7\} \\ S2 = \{2, 5, 8\} \\ S3 = \{3, 6\} \end{array}\right\} \text{Partition of S into 3 sets}$$

The Data Structure

We need a data structure to represent partitions.

The structure must allow us to:

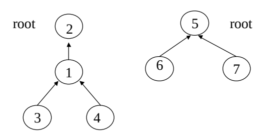
- find the "name" of an equivalence class (find)
- merge two equivalence classes (union)
- determine if two elements x and y are in the same equivalence class

The structure is sometimes called the union-find data structure.

Implementation

The union-find structure is just a set of trees, one for each equivalence class, with a simple array implementation strategy.

Example: S1 = {1, 2, 3, 4}, S2 = {5, 6, 7}



	1	2	3	4	5	6	7
S	2	0	1	1	0	5	5

S[i]=0 if i is a root. else it is the parent node of node i.

The root is the name of the class.

Smarter Unions

Instead of always making the second tree a subtree of the first, be smarter about it.

Union by Size: Keep track of the set sizes and make the smaller tree a subset of the larger.

Union by Height: Keep track of the tree heights and make the shorter tree a subset of the deeper.

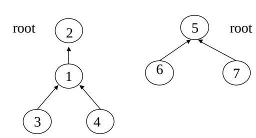
Implementation: Instead of adding a size (or height) field, replace each initial zero with –1, meaning trees of size 1. As the tree grows

$$S(i) = \begin{cases} \text{parent(i) if node i has a parent} \\ \text{negated size of i's subtree, otherwise} \end{cases}$$

Smart Implementation

Same trees, slightly different array.

Example: S1 = {1, 2, 3, 4}, S2 = {5, 6, 7}



	1	2	3	4	5	6	7
S	2	-4	1	1	-3	5	5

S[i]= –size if i is a root, else it is the parent node of node i.

2	-2	1	1	-1	5	5

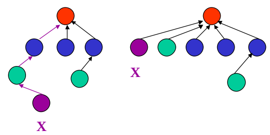
Use height, instead.

DS.S.7

Path Compression

Path compression is another smart technique that works for union by size.

During a Find(X) operation, change the parent of every node on the path from X to the root to the root itself.



How does this affect complexity of Find?

DS.S.8

Union by Rank

Path compression is not compatible with union by height, because it can change the heights of trees.

Union by rank is like union by height, but it does not recompute heights of trees, so it leaves us with only an estimate of height.

The estimated heights are called ranks.

DS.S.9

Complexity

- For N nodes, any sequence of $M = \Omega(N)$ Union/Find operations takes a total of $O(M \log^* N)$ running time. (6 page proof)
- Worst case for Find is $O(N)$. Why?
- Union takes constant time, BUT, usually you do 2 Finds before calling it.

$\log^* N$ is the number of times the logarithm of N needs to be applied till N becomes ≤ 1.

Result 1. is true for union by rank or union by size.

DS.S.10

Application: Efficient Connected Components

The efficient connected components makes two sequential passes down the image and uses the union-find structure to keep track of different labels that all belong to the same component.

-Look at 2 rows of the binary image at a time

	7	7	7	7		10		5	5					previous
1	1	1		1	1	1			1	1		1	1	current

-If a 1-pixel has no already-labeled neighbors in the previous row or to its left in the current row, assign it a new label.

-If it has such neighbors, and they all have the same label, assign it that label.

DS.S.11

-If it has 2 neighbors with different labels, use the smaller one and equivalence those labels.

	7	7	7	7		10		5	5				
11	**7**	7		7	7	**7**			5	5		12	12

$7 \equiv 11$ $\quad$ $7 \equiv 10$

Use the union-find structure to keep track of equivalence classes.

1	2	3	4	5	6	7	8	9	10	11	12
-1	–1	-1	-1	–1	-1	-3	-1	-1	7	7	-1

Example is using 4-neighbor labeling.
What would happen in 8-neighbor labeling?