Sergio E. Garcia Tapia
Algorithms by Sedgewick and Wayne (4th edition) [SW11]
October 6th, 2024

1.5: Case Study: Union-find

Exercise 1. Show the contents of the `id[]` array and the number of times the array is accessed for each input pair when you use quick-find for the sequence

```
9-0 3-4 5-8 7-2 2-1 5-7 0-3 4-2
```

Solution.

		id[]										
p	q	0	1	2	3	4	5	6	7	8	9	Array Accesses
9	0	0	1	2	3	4	5	6	7	8	9	
		0	1	2	3	4	5	6	7	8	0	15
3	4	0	1	2	3	4	5	6	7	8	0	
		0	1	2	4	4	5	6	7	8	0	15
5	8	0	1	2	4	4	5	6	7	8	0	
		0	1	2	4	4	8	6	7	8	0	15
7	2	0	1	2	4	4	8	6	7	8	0	
		0	1	2	4	4	8	6	2	8	0	15
2	1	0	1	2	4	4	8	6	2	8	0	
		0	1	1	4	4	8	6	1	8	0	16
5	7	0	1	1	4	4	8	6	1	8	0	
		0	1	1	4	4	1	6	1	1	0	16
0	3	0	1	1	4	4	1	6	1	1	0	
		4	1	1	4	4	1	6	1	1	4	16
4	2	4	1	1	4	4	1	6	1	1	4	
		1	1	1	1	1	1	6	1	1	1	18

Each input sequence incurs the cost of a `connected()` operation which is two calls to `find()`, and hence 2 arrays accesses. Then in calling `union()`, we have two more array accesses since we call `find()` twice again. Then, we have at least 10 array accesses as we iterate through the `id` array. Finally, we have an extra array access for each identifier matching `pID`, the identifier of the component of the first site given.

Exercise 2. Do Exercise 1.5.1, but use quick-union (page 224). In addition, draw the forest of trees represented by the `id[]` array after each input pair is processed.

Solution.

		id[]										
p	q	0	1	2	3	4	5	6	7	8	9	Array Accesses
9	0	0	1	2	3	4	5	6	7	8	9	
		0	1	2	3	4	5	6	7	8	0	3
3	4	0	1	2	3	4	5	6	7	8	0	
		0	1	2	4	4	5	6	7	8	0	3
5	8	0	1	2	4	4	5	6	7	8	0	
		0	1	2	4	4	8	6	7	8	0	3
7	2	0	1	2	4	4	8	6	7	8	0	
		0	1	2	4	4	8	6	2	8	0	3
2	1	0	1	2	4	4	8	6	2	8	0	
		0	1	1	4	4	8	6	2	8	0	3
5	7	0	1	1	4	4	8	6	2	8	0	
		0	1	1	4	4	8	6	2	1	0	9
0	3	0	1	1	4	4	8	6	2	1	0	
		4	1	1	4	4	8	6	2	1	0	5
4	2	4	1	1	4	4	8	6	2	1	9	
		4	1	1	4	1	8	6	2	1	0	5

See Figure 1 for the forest of trees representation of `id[]`.

Exercise 3. Do Exercise 1.5.1, but use weighted quick-union (page 228).

Solution.

		id[]										
p	q	0	1	2	3	4	5	6	7	8	9	Array Accesses
9	0	0	1	2	3	4	5	6	7	8	9	
		9	1	2	3	4	5	6	7	8	9	
3	4	9	1	2	3	4	5	6	7	8	9	
		9	1	2	3	3	5	6	7	8	9	
5	8	9	1	2	3	3	5	6	7	8	9	
		9	1	2	3	3	5	6	7	5	9	
7	2	9	1	2	3	3	5	6	7	5	9	
		9	1	7	3	3	5	6	7	5	9	
2	1	9	1	7	3	3	5	6	7	5	9	
		9	7	7	3	3	5	6	7	5	9	
5	7	9	7	7	3	3	5	6	7	5	9	
		9	7	7	3	3	7	6	7	5	9	
0	3	9	7	7	3	3	7	6	7	5	9	
		9	7	7	9	3	7	6	7	5	9	
4	2	9	7	7	9	3	7	6	7	5	9	
		9	7	7	9	3	7	6	7	5	7	

See Figure 2.

Exercise 7. Develop classes `QuickUnionUF` and `QuickFindUF` that implement quick-union and quick-find, respectively.

Solution. See the `com.segarciat.algs.ch1.sec5.ex07` package.

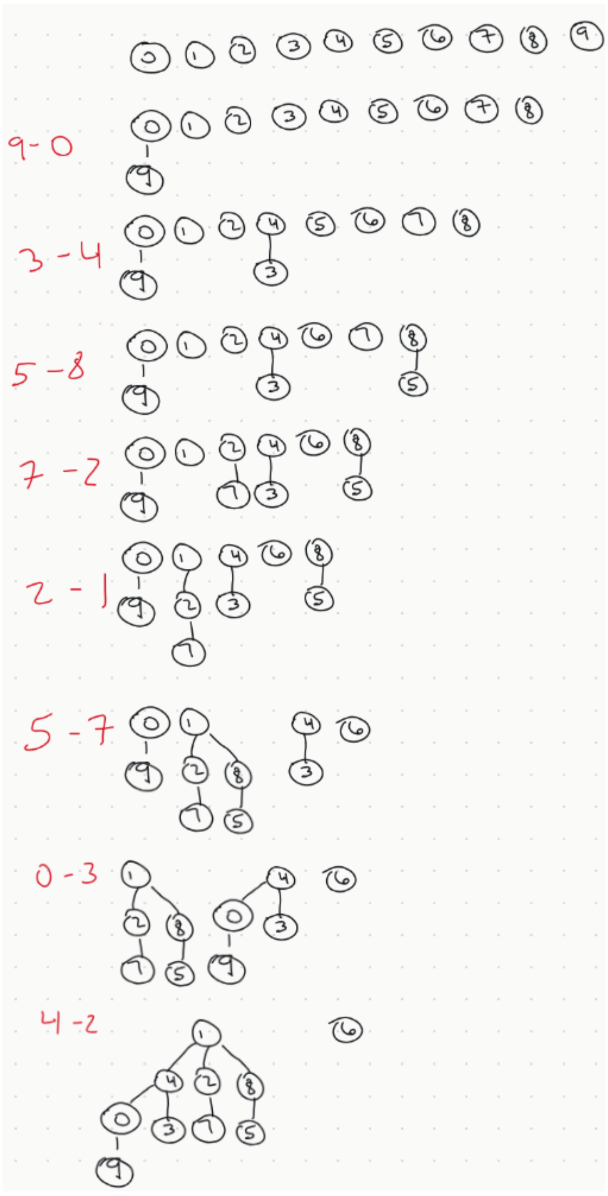

Figure 1: Forest of trees representation of `id[]` for quick-union in Exercise 2.

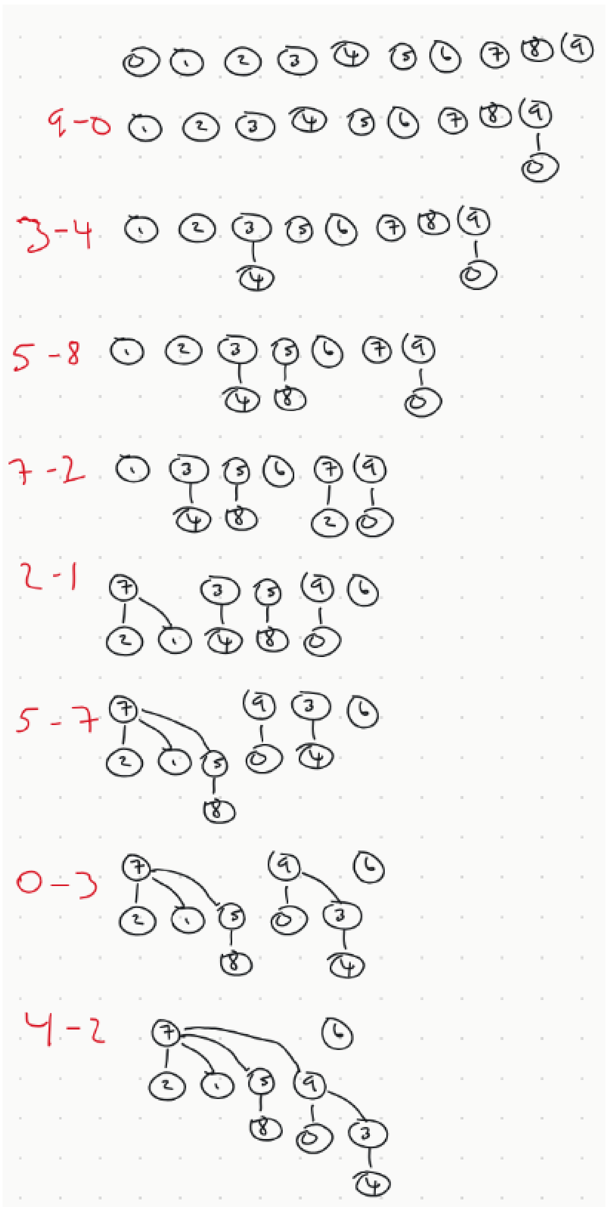

Figure 2: Forest of trees representation of `id[]` for quick-union in Exercise 2.

References

[SW11] Robert Sedgewick and Kevin Wayne. *Algorithms*. 4th ed. Addison-Wesley, 2011. ISBN: 9780321573513.