

Chapter 3, Case Studies:

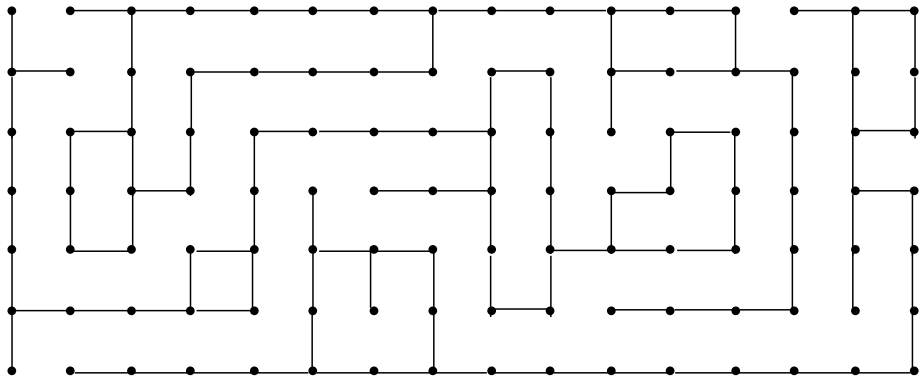
- ▶ Linear-time sorting algorithms (Monday and Wednesday)
- ▶ Disjoint sets and array forests (**Today**)
- ▶ Priority queues and heaps (next week Monday)
- ▶ N -sets and bit vectors (next week Wednesday)

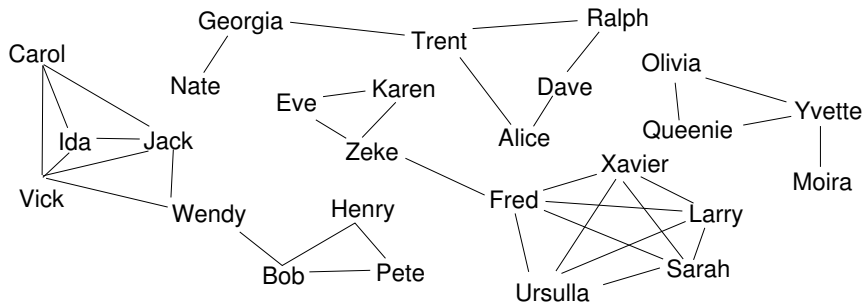
Today:

- ▶ Problem statement
- ▶ Disjoint set ADT details
- ▶ The array forest abstraction and data structure
- ▶ Find and union strategies, with optimizations

Problem statement:

Suppose we have a collection of items connected by an unknown equivalence relation. Efficiently find the equivalence classes in this collection as information about the relation is discovered.





$$a = c$$

$$e = a + b$$

$$d = b$$

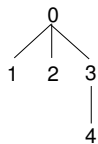
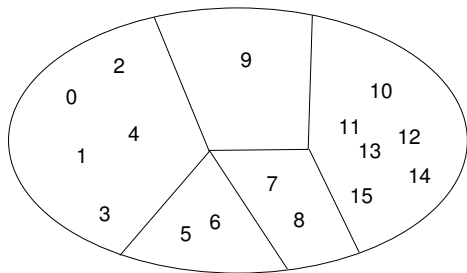
$$g = 1$$

$$f = d + c$$

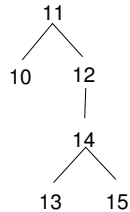
$$h = e * g$$

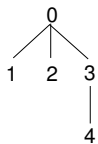
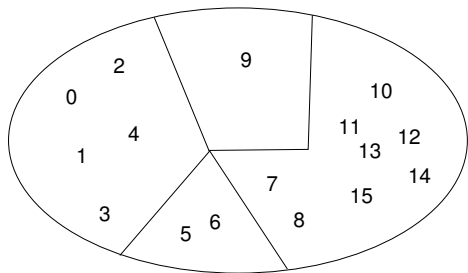
The *disjoint set* ADT:

- ▶ Main operations: union two sets, find a set for a given element, and test if two elements are in the same set.
- ▶ The universe is closed.
- ▶ We assume all elements can be indexed, $[0, N)$.
- ▶ A set in the partition is identified by a leader.

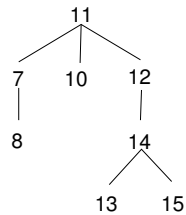


9





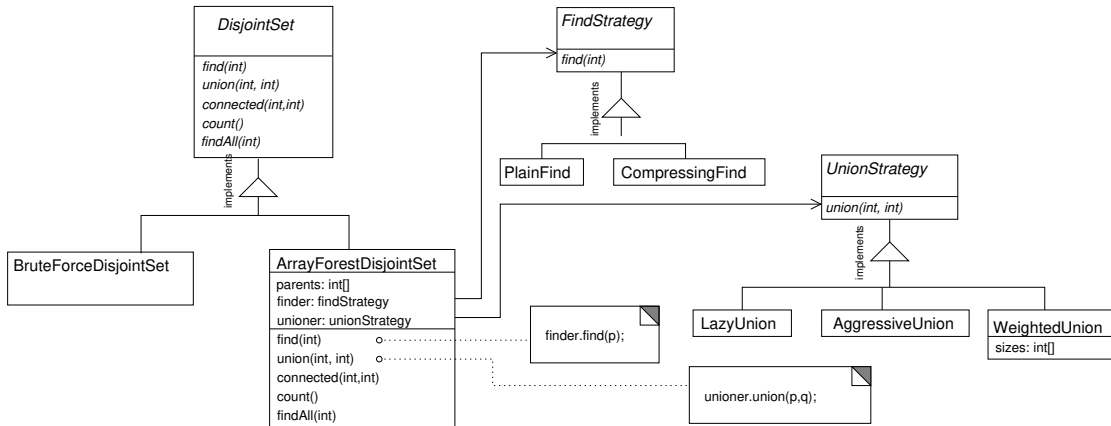
9



Invariant (Class ArrayForestDisjointSet)

For all $i \in [0, n)$,

- (a) $leader(i) = leader(parents(i))$, that is, $id(i)$ points to another element in the same set as i .
- (b) $leader(i) = parents[leader(i)]$, that is, leaders all point to themselves.
- (c) Following a finite number links implied by $parents$ will converge, that is, there is no circularity in the tree.



Union strategy	LazyUnion	AggressiveUnion	WeightedUnion	LazyUnion	WeightedUnion
Find strategy	PlainFind	PlainFind	PlainFind	CompressingFind	CompressingFind
Find heavy:	1.30E7 (5.68E6)	3.34E7 (8.40E3)	7.40E5 (1.80E4)	9.26E5 (2.38E4)	6.68E5 (9.34E3)
Even mix:	9.89E7 (1.22E7)	4.41E7 (9.93E3)	1.20E6 (1.97E4)	1.56E6 (2.12E4)	9.80E5 (9.96E3)
Union heavy:	1.62E8 (1.26E7)	4.39E7 (9.99E3)	1.40E6 (2.01E4)	1.71E6 (1.59E4)	1.04E6 (1.00E4)

Coming up: (all end-of-day)

*Do **linear sorting** project (suggested by Mon, Feb 6)*

*Due **Today:***

Finish reading Section 3.2 (disjoint sets and array forests)

Take disjoint-sets quiz

*Due **Tues, Feb 7:***

Read Section 3.3 (heaps and priority queues)

Take heap/pq quiz

*Due **Thurs, Feb 9:***

Read Section 3.4

Do Exercises 3.(26 & 27).

Take N-sets quiz