Chapter 3, Case Studies:

- Linear-time sorting algorithms (last week Wednesday and Friday)
- Disjoint sets and array forests (**Today**)
- Priority queues and heaps (Wednesday and Friday)
- N-sets and bit vectors (Thursday lab)
- (Begin Graph unit in lab next week)

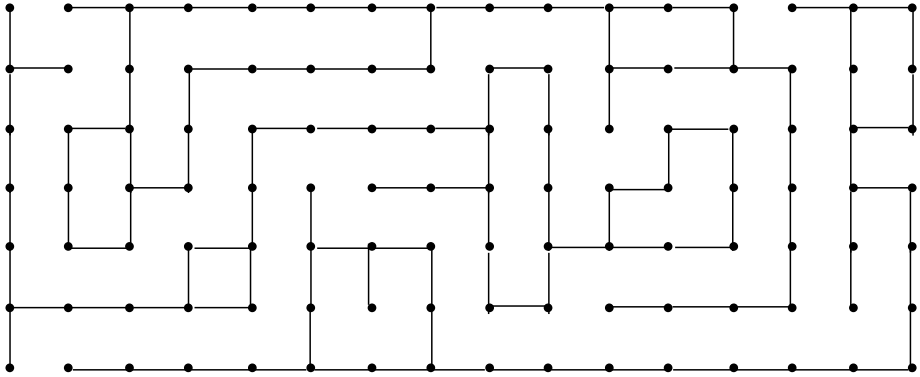
Today:

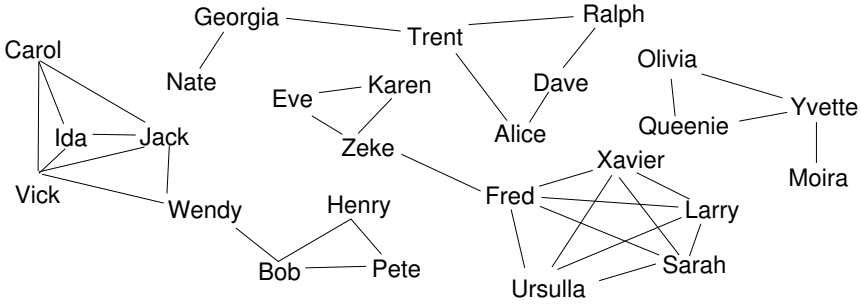
- Quiz solutions
- Problem statement
- Disjoint set ADT details
- The array forest abstraction and data structure
- Find and union strategies, with optimizations

```
static Node arrayToList1(int[] array) {
    Node toReturn = new Node(array[0], null);
    for (int i = 1; i < array.length; i++) {
        Node current = toReturn;
        while (current.next() != null)
            current = current.next();
        current.setNext(new Node(array[i], null));
    }
    return toReturn;
}

Node arrayToList2(int[] array) {
    Node toReturn = null;
    for (int i = array.length - 1; i >= 0; i--)
        toReturn = new Node(array[i], toReturn);
    return toReturn;
}

static int[] listToArray(Node head) {
    int size = 0;
    for (Node current = head; current != null; current = current.next())
        size++;
    int[] toReturn = new int[size];
    int i = 0;
    for (Node current = head; current != null; current = current.next())
        toReturn[i++] = current.datum();
    return toReturn;
}
```

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.

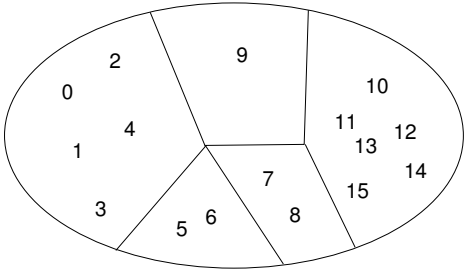


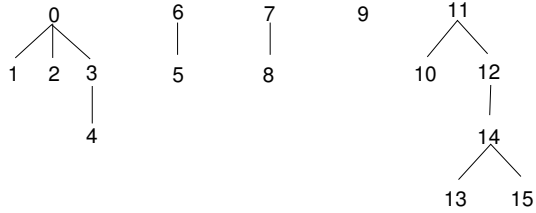

Georgia
Ralph
Trent
Carol
Nate
Dave
Olivia
Eve
Karen
Yvette
Queenie
Ida
Jack
Alice
Zeke
Xavier
Moira
Vick
Fred
Henry
Larry
Wendy
Bob
Pete
Sarah
Ursulla

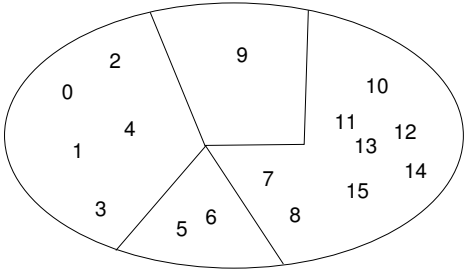
```
a = c
e = a + b
d = b
g = 1
f = d + c
h = e * g
```

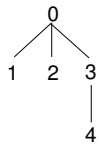
The *disjoint set* ADT:

- 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.

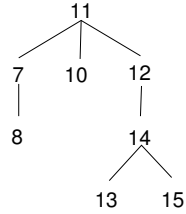

2
9
0
10
4
11
12
13
1
14
7
15
3
6
8
5


0
1
2
3
4
6
5
7
8
9
11
10
12
14
13
15





9



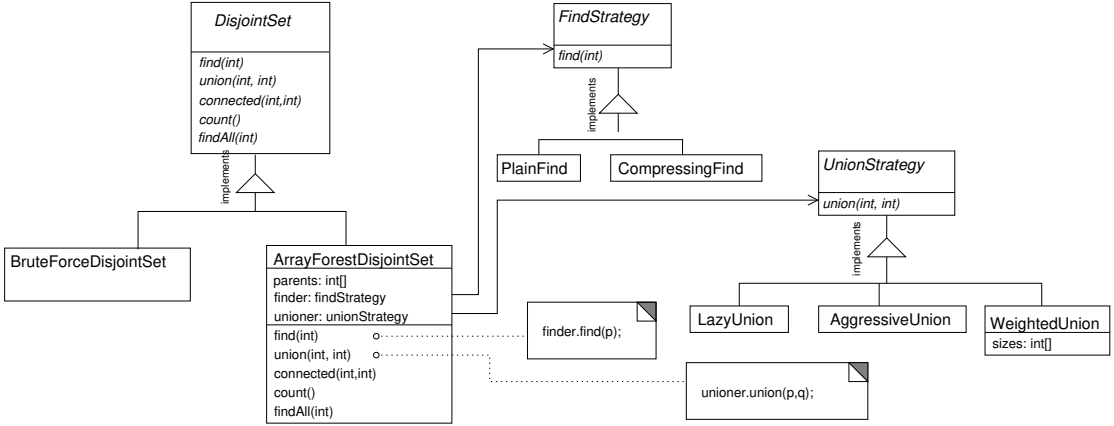
Invariant (Class `ArrayForestDisjointSet`)

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

(a) $\mathit{leader}(i) = \mathit{leader}(\texttt{parents}[i])$, *that is,* $\texttt{parents}[i]$ *points to another element in the same set as* i.

(b) $\mathit{leader}(i) = \texttt{parents}[\mathit{leader}(i)]$, *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.*


DisjointSet
find(int)
union(int, int)
connected(int,int)
count()
findAll(int)
implements
BruteForceDisjointSet
ArrayForestDisjointSet
parents: int[]
finder: findStrategy
unioner: unionStrategy
find(int)
union(int, int)
connected(int,int)
count()
findAll(int)
FindStrategy
find(int)
implements
PlainFind
CompressingFind
finder.find(p);
UnionStrategy
union(int, int)
implements
LazyUnion
AggressiveUnion
WeightedUnion
sizes: int[]
unioner.union(p,q);

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

Coming up: (all end-of-day)

Do **linear sorting** *project (Wed, Sept 23)*

Due **Today**:
Finish reading Section 3.2 (disjoint sets and array forests)
Do Ex 2.(12 & 16) and 3.8 Take disjoint-sets quiz

Due **Thurs, Sept 26**:
Read Section 3.4
Do Exercises 3.(26 & 27).
Take N-sets quiz

Due **Fri, Sept 27**:
Read Section 3.3 (heaps and priority queues)
(no exercises)
Take heap/pq quiz