Opening Discussion

- Do you have any questions about the quiz?
- Minute essay comments
  - Two paths of the same length.
  - No path to exit.
  - Not retracing steps.
  - Nothing can go wrong with me writing the code. (Wishful thinking on the part of one student.)
Mazes

- Finishing shortest path.
- Adding breadcrumbs.
- Slow in the worst case because this does all possible paths.
We can also use recursion to write some better sorts.

All of our old sorts could have been written with recursion, but only as a substitute for iteration.

With recursion we can do sorts that work by repeatedly breaking the set down then work recursively on the pieces.

Do they do the work on the way down the stack or back up?

Work fairly well on lists.
Merge Sort

- Simple description
  - Break the collection in two and make a recursive call on the two halves.
  - Merge together the sorted results with an O(n) merge.
- Can't be done in place, but that is advantageous for lists which are immutable.
- O(n log n) all the time.
Quick Sort

- **Description**
  - Pick a pivot and move everything less than the pivot below and everything greater above.
  - Recurse on the two sides of the pivot.

- Can be done in place, but Scala collection methods allow very simple form that isn't in place. We'll wrote both.

- Speed depends on pivot selection. $O(n \log n)$ on average with random data, but can be as bad as $O(n^2)$ with bad pivots.
Object-Orientation

- We have been dealing with objects all semester, but we haven't really faced object-orientation head on.
- The OO paradigm is characterized by encapsulation, the grouping of data and functions together into objects.
- The data is called members and the functions are called methods.
- The idea is that an object knows some things and how to do some things.
Scala is a class-based OO language. In the code we write classes which act as the blueprints of objects.

- These start just like the case classes we saw before, but the word case isn't required.
- Put the body of the class in curly braces after the declaration and arguments.
Differences from Case Classes

- Members are private by default so you can only see them in the class.
- Have to be made with `new`.
- Code in the body of the class is executed immediately.
- Functions defined in the body are methods of the objects.
- Data defined in the class are members of the objects.
- You can make things private.
Making Objects

- The class is only a blueprint. To get an object we have to instantiate an instance form the class.
  - `new ClassName(arguments)`
  - This expression can be assigned to values or passed into functions. The type is the name of the class.
- Once you have an object you can access members and methods using the dot notation.
Operators as Methods

- You can use symbols for method names and use them with operator syntax.
- This lets you do things like a+b when a and b are of a type you created.
Minute Essay

- Do you have any final requests before the last day of class?