

For Monday

- Read Becker, chapter 4, sections 1-3
- Recommended practice problems:
 - chapter 3, problem 3
 - chapter 4, problems 1-4

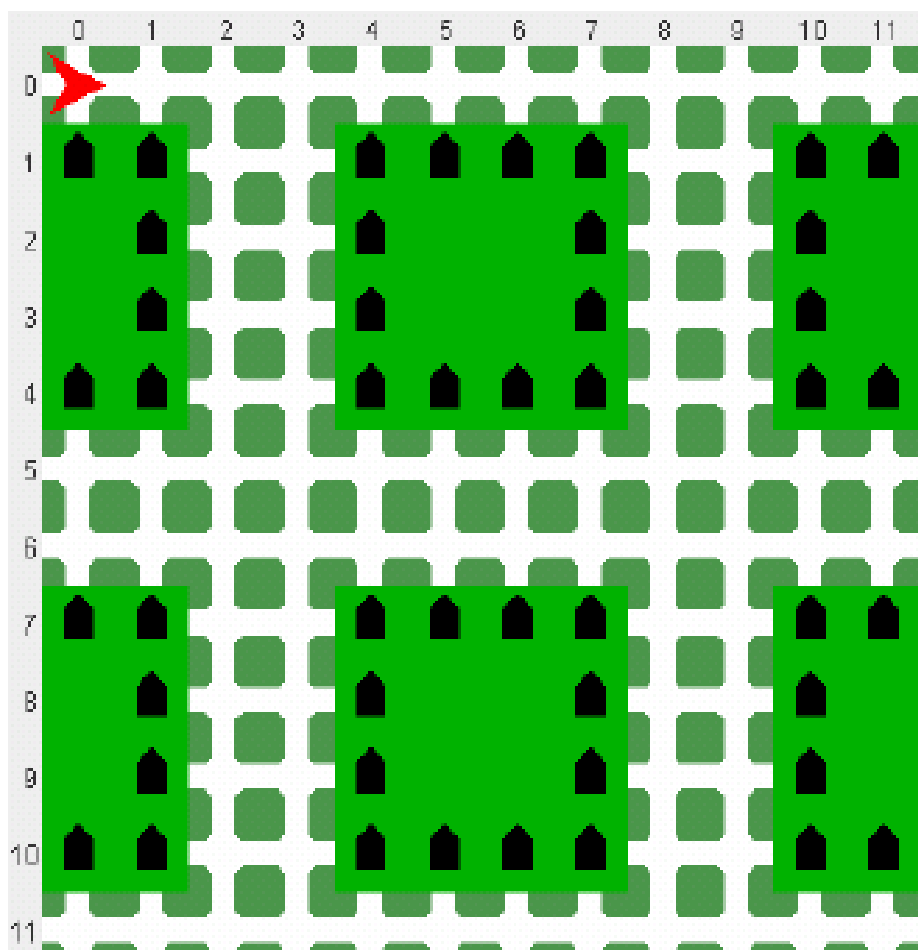
Program 2

Questions before the quiz?

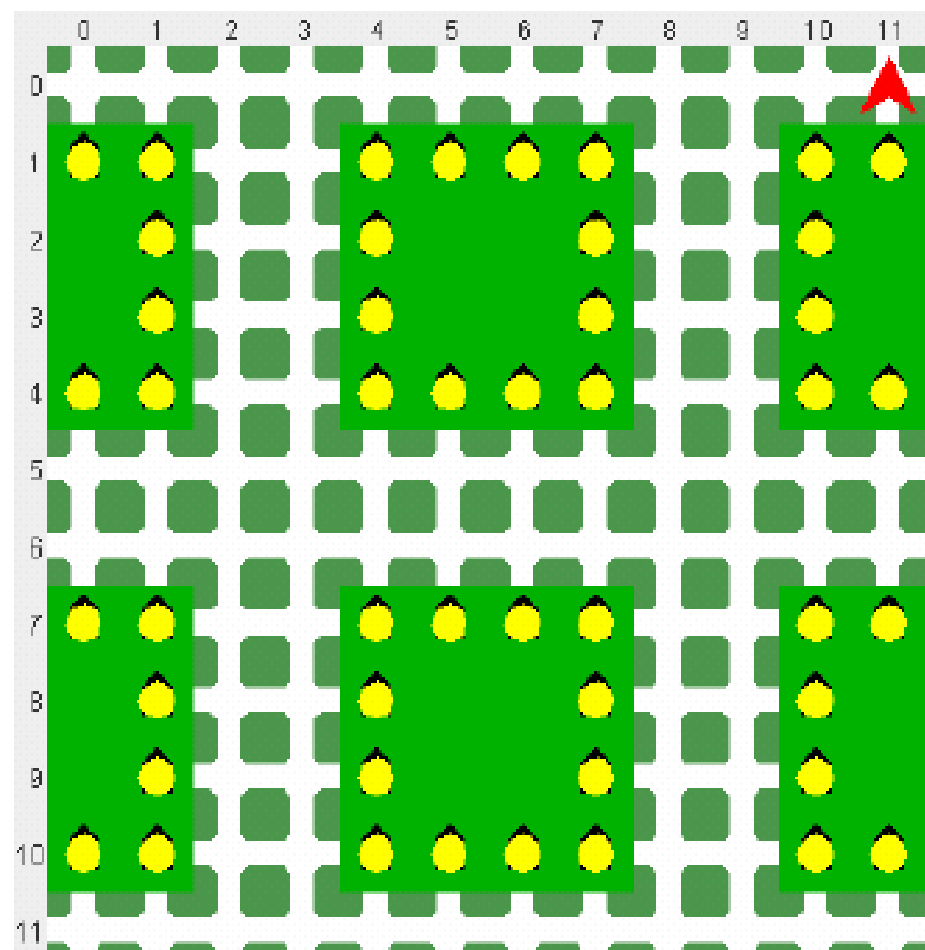
Quiz

Delivering Flyers

- You need a robot to deliver flyers to houses on a delivery route. The route is shown on the next slide. The robot must visit all of the houses on the route and should stay off the green areas as much as possible.



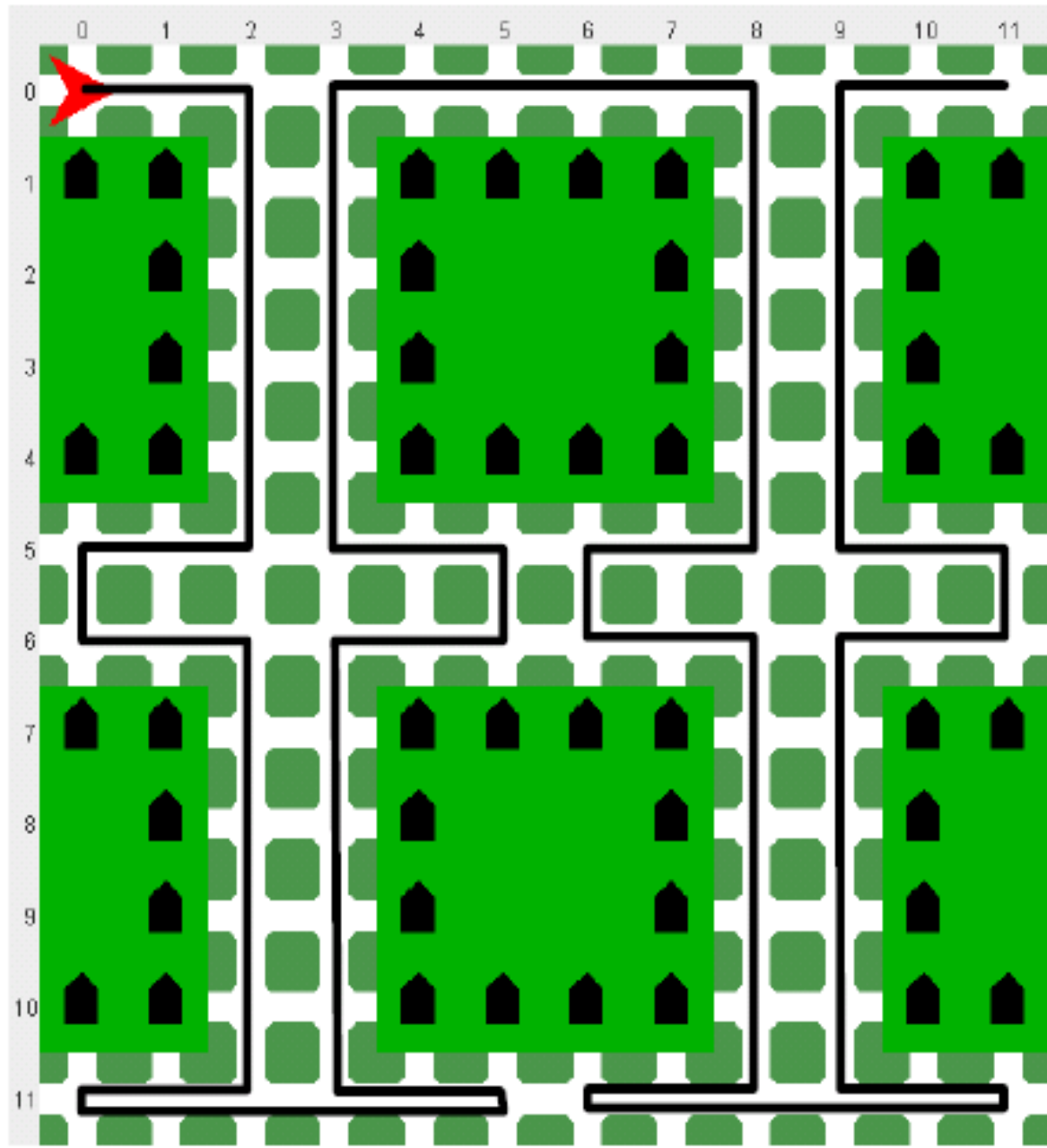
Initial Situation



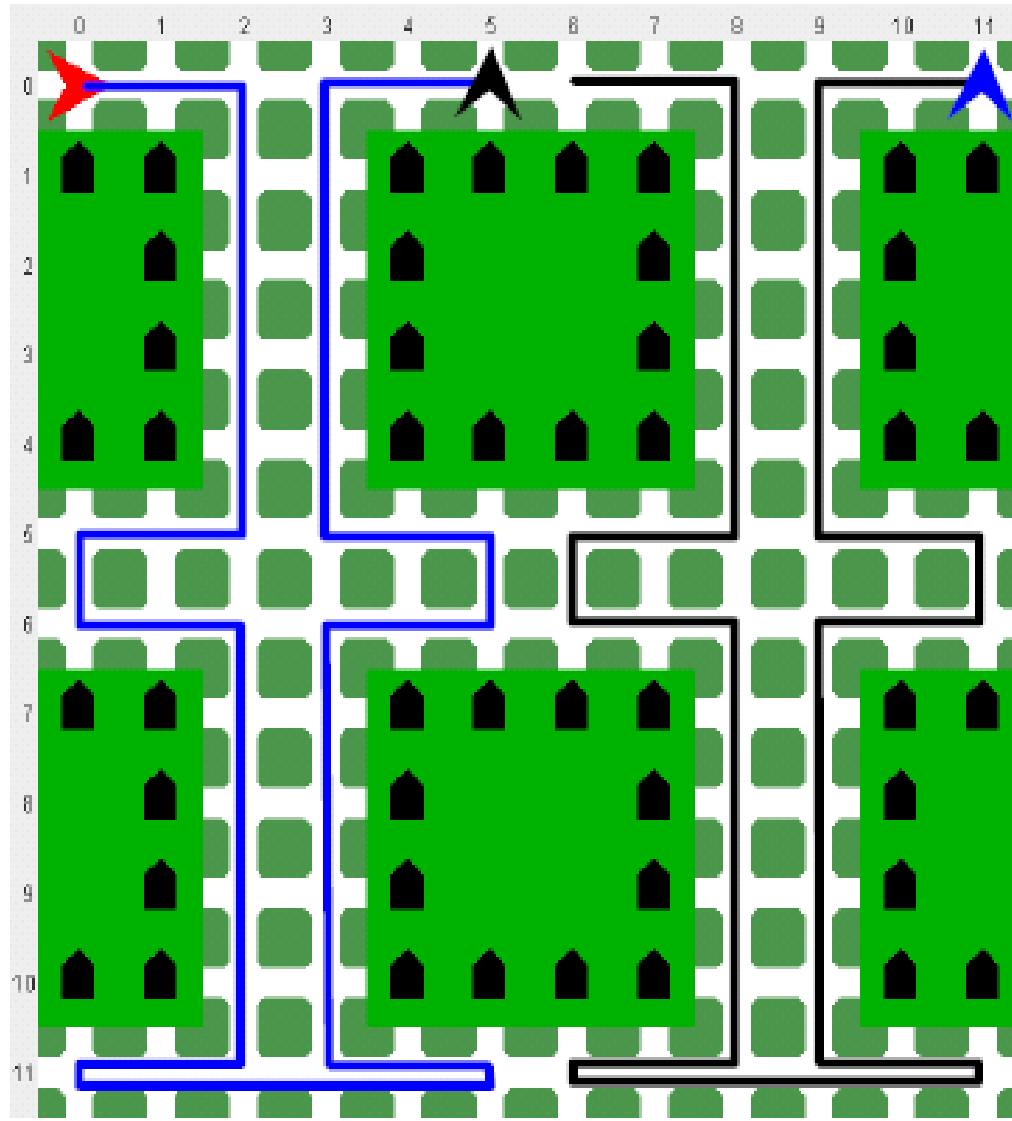
Final Situation

What route?

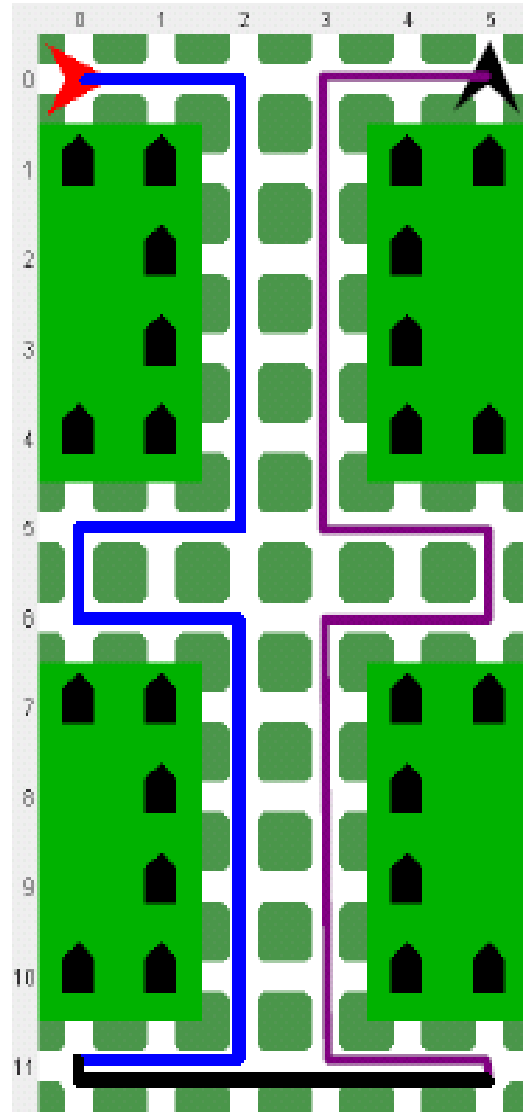
- This shows a possible route (not counting the actual delivery to the houses)



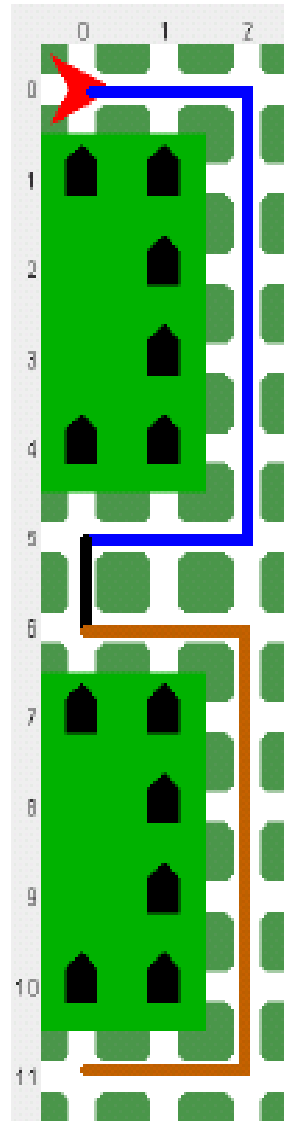
Breaking up the Work



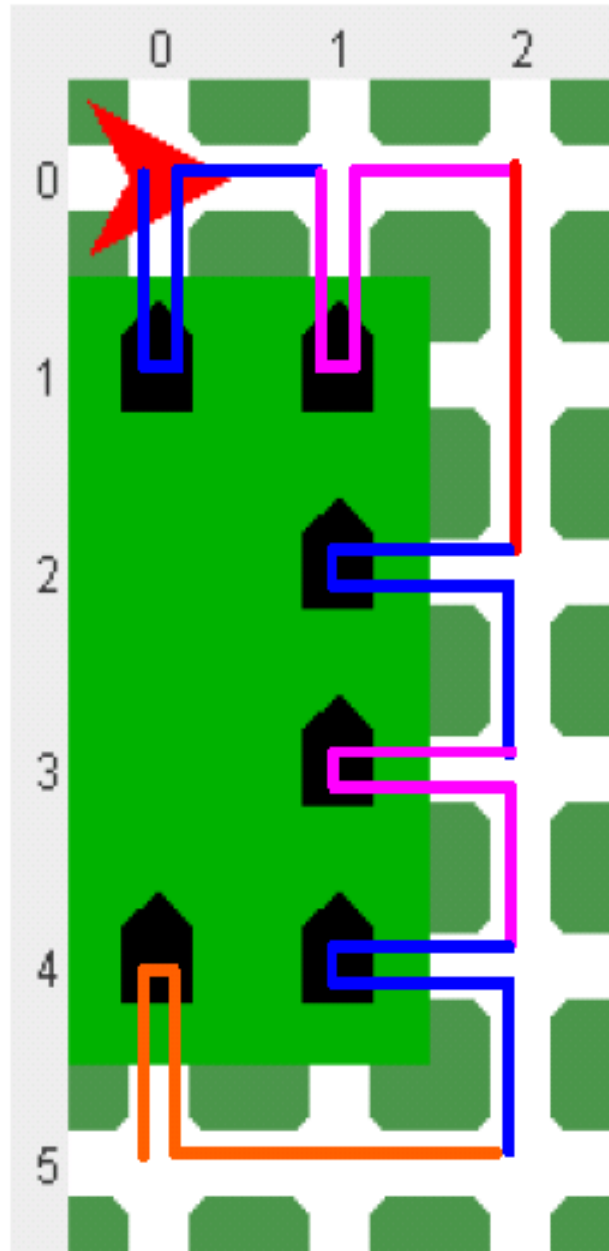
Breaking up DeliverOneAvenue

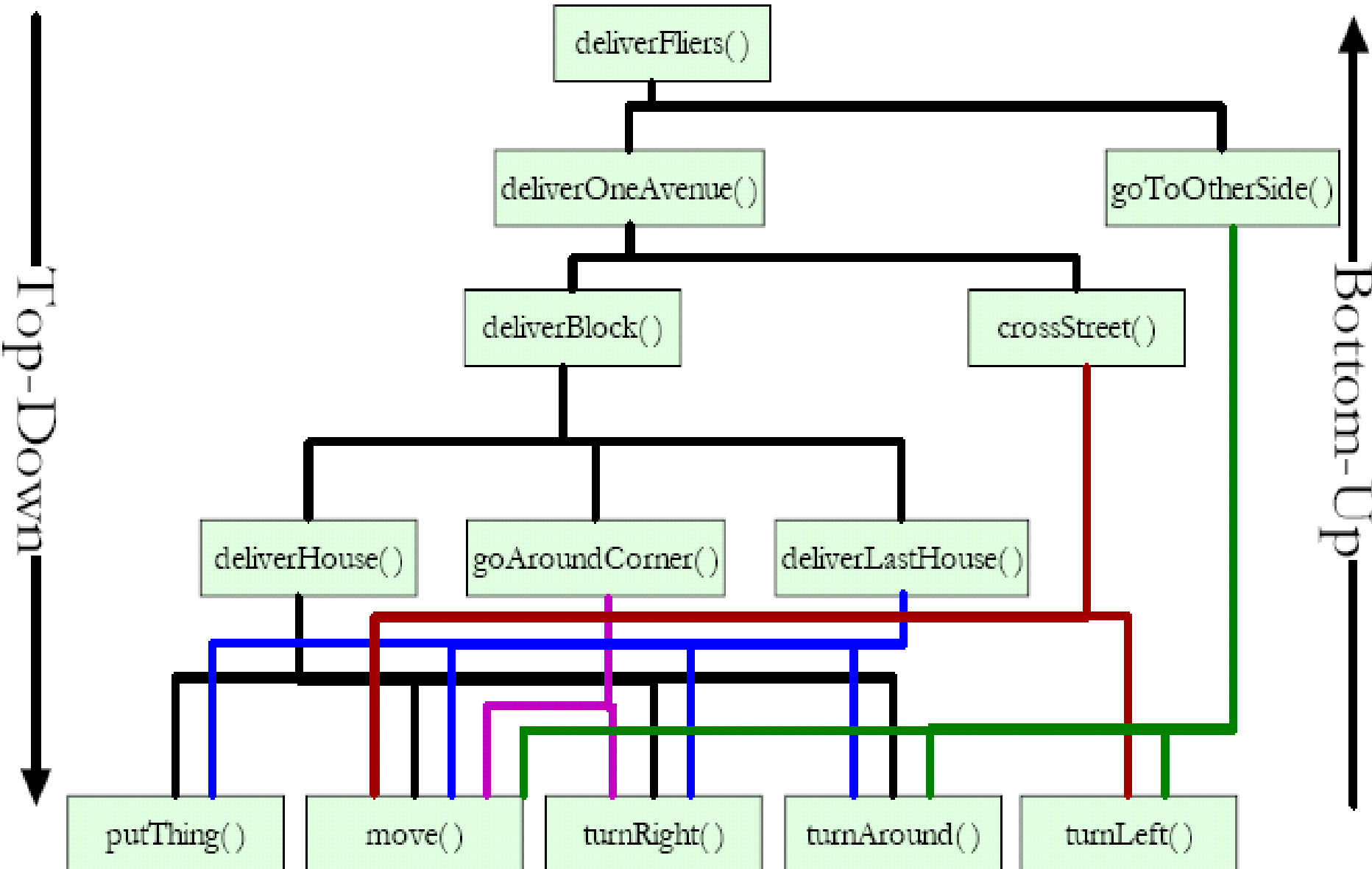


Breaking up DeliverOneSide



DeliverBlock





Advantages of Stepwise Refinement

- Tends to make programs
 - Easier to understand
 - Free of errors
 - Easier to test and debug
 - Easier to modify
- Why?

The Reasons

- People can't keep all the details in their heads at once.
- Helps to impose structure on the problem
- The descriptive names allow us to ignore details

Pseudocode Algorithms

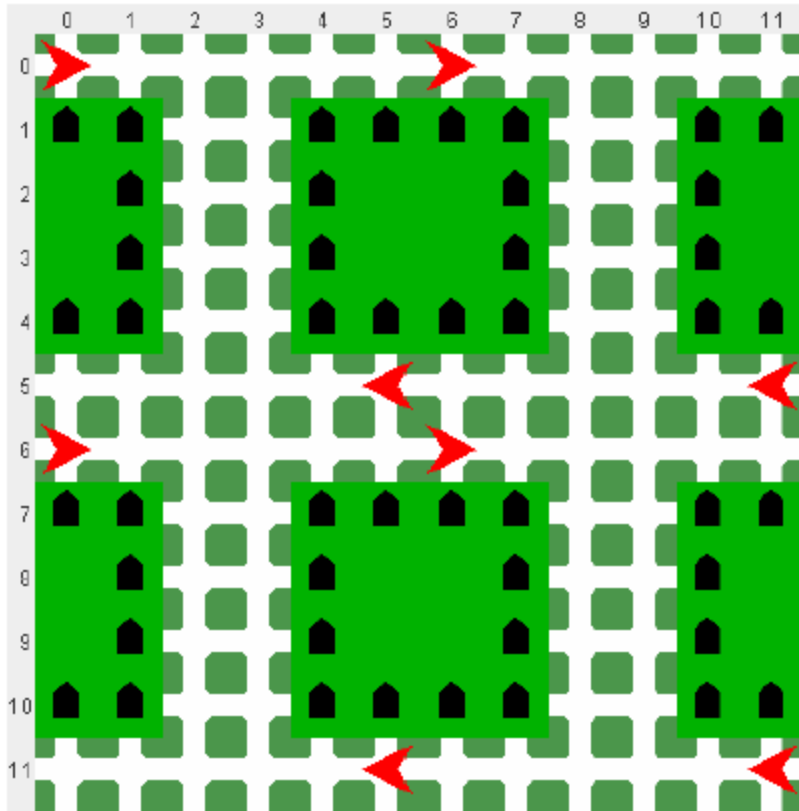
- Let us focus on the algorithm, not the details of Java, first
- Combines natural language with structure
- Very important as we move on to writing programs that make decisions next week.
- Example:
 - deliver fliers to each house up to the corner*
 - turn the corner*
 - deliver fliers to each house up to the corner*
 - turn the corner*

Advantages of Pseudocode

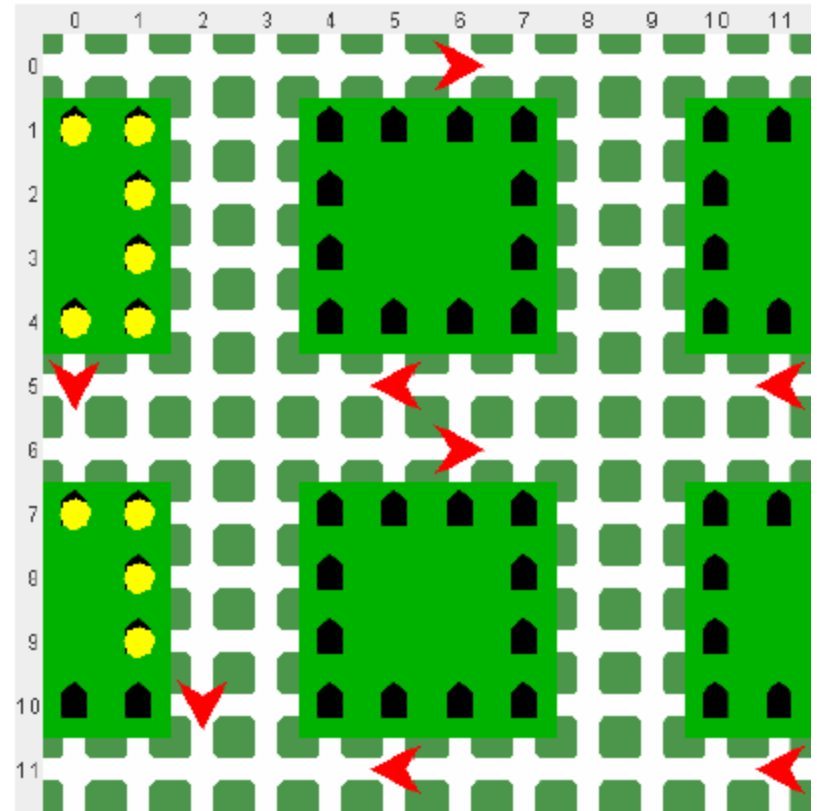
- Pseudocode helps us think more abstractly, allowing us to ignore many irrelevant details.
- Pseudocode allows us to trace our programs very early in development.
- Pseudocode can provide a common language on a development team, even with non-technical users.
- Algorithms expressed in pseudocode can be implemented in a variety of programming languages.

Using Multiple Robots

- How could we do the flyer problem efficiently with multiple robots?
- Would we have to change the DeliveryBot code?



Initial Situation



During Execution

```
import becker.robots.*;

public class DeliverFlyers
{

    public static void main(String[ ] args)
    {
        Route route = new Route();
        DeliveryBot db1 = new DeliveryBot(route, 0, 0, Direction.EAST, 6);
        DeliveryBot db2 = new DeliveryBot(route, 6, 0, Direction.EAST, 6);
        DeliveryBot db3 = new DeliveryBot(route, 5, 5, Direction.WEST, 6);
        DeliveryBot db4 = new DeliveryBot(route, 11, 5, Direction.WEST, 6);
        DeliveryBot db5 = new DeliveryBot(route, 0, 6, Direction.EAST, 6);
        DeliveryBot db6 = new DeliveryBot(route, 6, 6, Direction.EAST, 6);
        DeliveryBot db7 = new DeliveryBot(route, 5, 11, Direction.WEST, 6);
        DeliveryBot db8 = new DeliveryBot(route, 11, 11, Direction.WEST, 6);

        db1.deliverBlock();
        db2.deliverBlock();
        db3.deliverBlock();
        db4.deliverBlock();
        db5.deliverBlock();
        db6.deliverBlock();
        db7.deliverBlock();
        db8.deliverBlock();
    }
}
```

Factoring out Differences

- What if I want a CollectionBot?
- What do I need to change?
- How do I avoid duplicating code?

Helper Methods

- What is a helper method?
- Who should call a helper method?
- How do we enforce that?

Access Modifiers

- **Public**
 - Access from anywhere
 - Use for the services a robot provides to others
- **Protected**
 - Access from methods in the class or its subclasses
 - Use for methods that may be overridden but don't need to be public
- **Private**
 - Access from only methods in the class
 - Use for anything that doesn't need to be public or protected