Algorithmic Thinking and Structured Programming (in Greenfoot)

Teachers:
Renske Smetsers-Weeda
Sjaak Smetsers
Ana Tanase

Today’s Lesson plan (11)

- Retrospective
  - Previous lesson
- Theory: tasks
- Theory NP-C: Travelling Salesman problem
- Explanation: Dodo’s race

Retrospective

- **Nested if-then-else => else-if**
- **class constant**
- declare and use List variables (with primitive or object types)
- **object types (vs. primitive types)**
- assigning values to object types (vs. primitive types)
- **Null**
- List methods (i.e. getting and deleting elements)
- for-each-loop
- use Java Library Documentation to look for and use existing Java (List) methods;

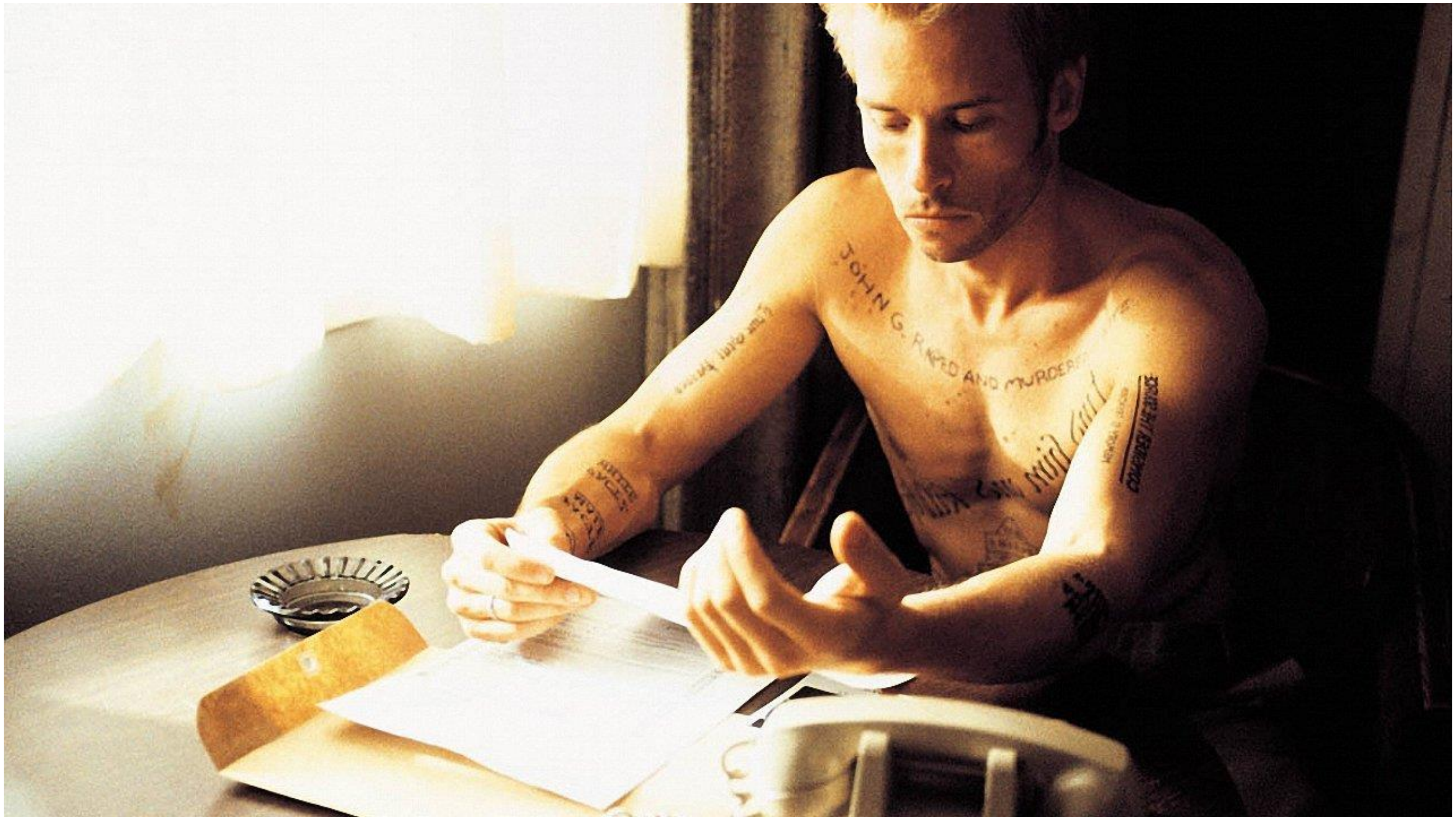
Topics for assignment (rest of 7)

- Splitting complex algorithm into subalgorithms (Dodo's race)
- Implementing sequence of subalgorithms in Greenfoot
- Swapping elements in a list

- Note: some of the subalgorithms you may have already implemented)

- NPC: Travelling salesman problem
- Dodo's race

Remembering things



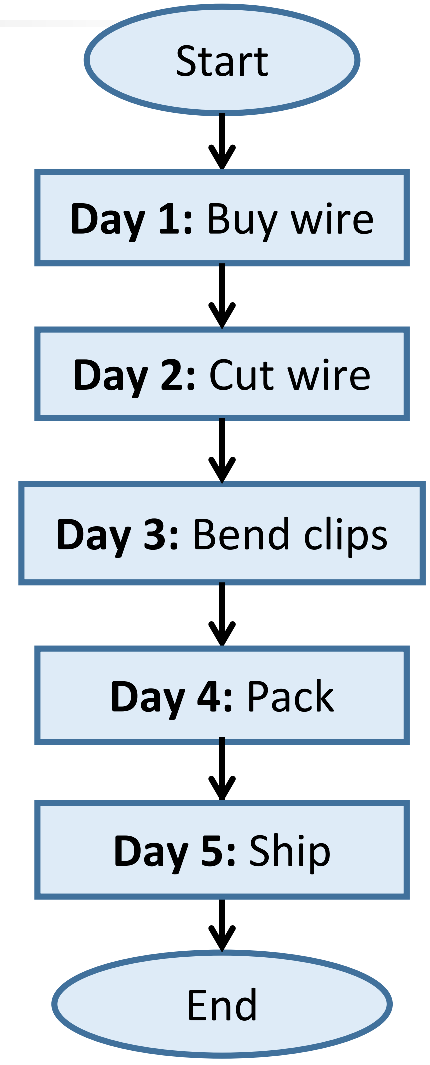
Splitting complex into subalgorithms

Manufacturing paper clips

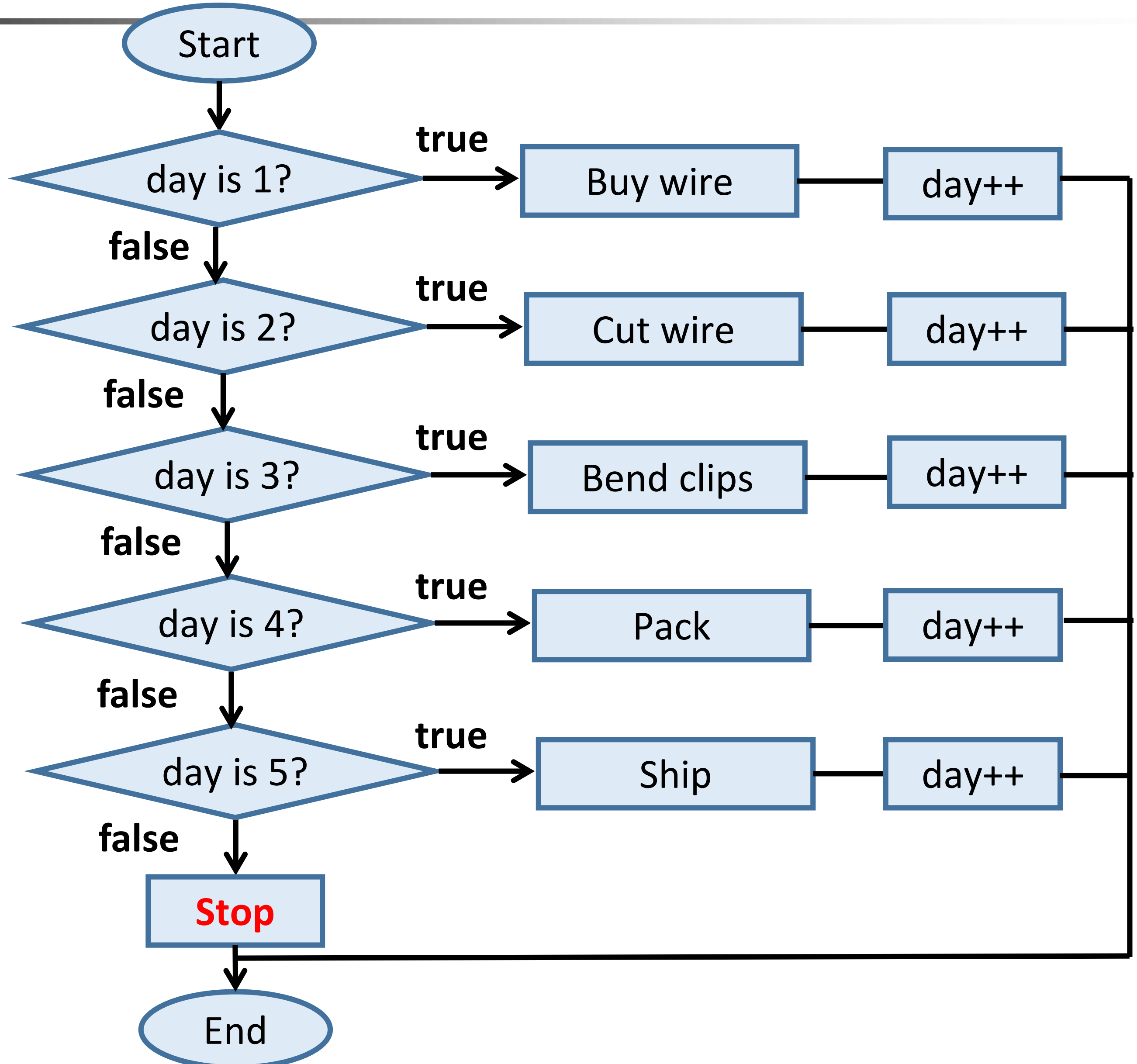
- Day 1: Buy steel wire
- Day 2: Cut wire into pieces
- Day 3: Bend the clips
- Day 4: Pack them
- Day 5: Ship the packages

What happens if you've forgotten what you did the day before?

Keep track of the day number (or the current task)



Manufacturing paper clips (2)



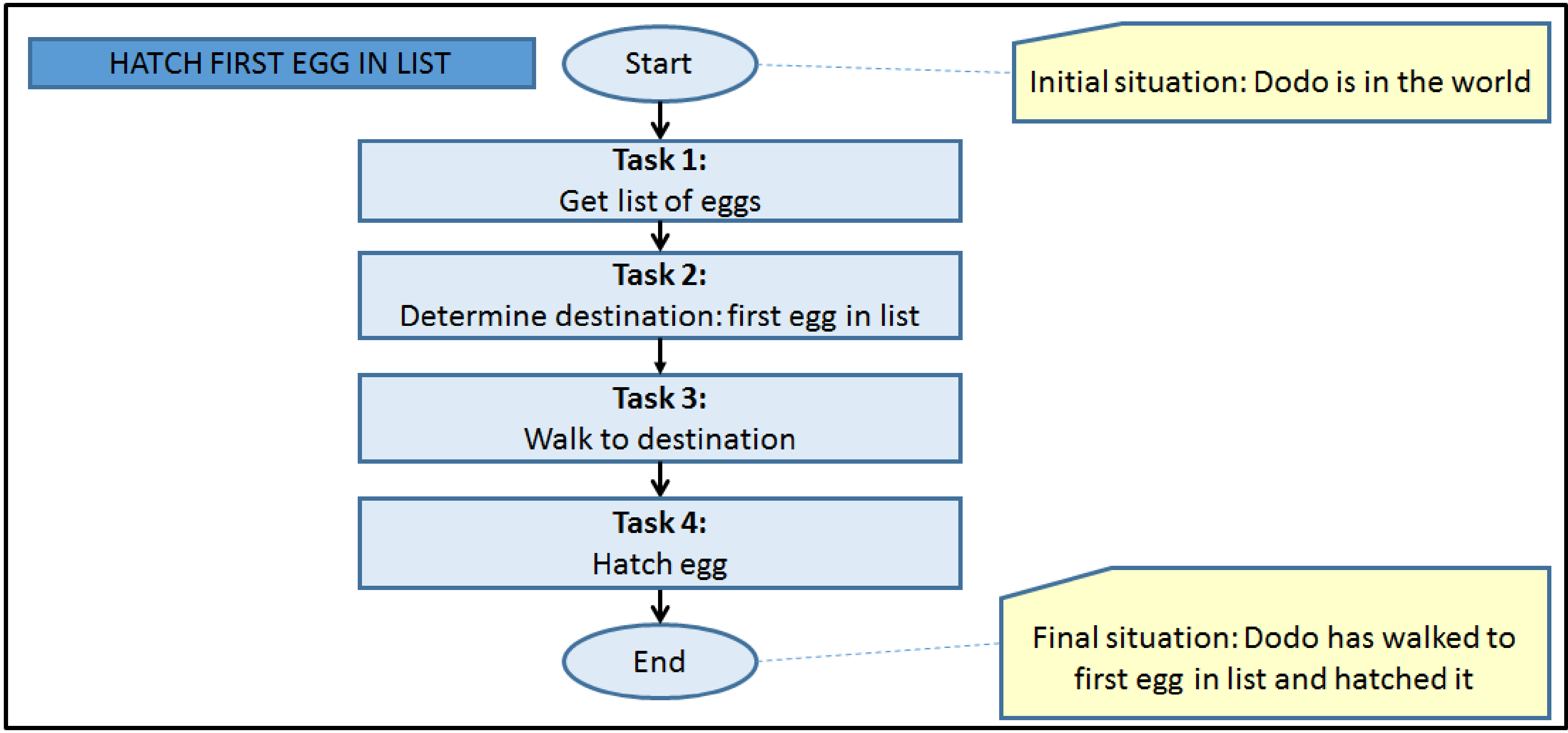
Manufacturing paper clips (3)

```
private int myCurrentTask;
```

Instance variable

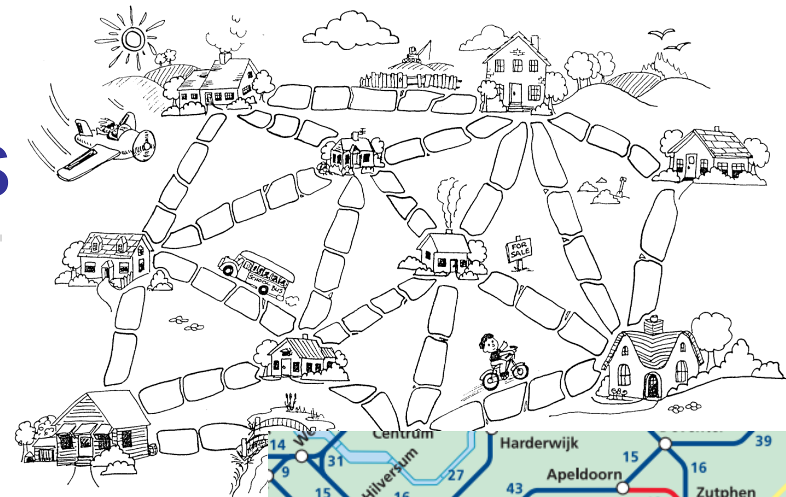
```
public void makePaperClips() {
    if ( myCurrentTask == 1 ) {
        buyWire();
    } else if ( myCurrentTask == 2 ) {
        cutWire();
    } else if ( myCurrentTask == 3 ) {
        bendClips();
    } else if ( myCurrentTask == 4 ) {
        packClips();
    } else if ( myCurrentTask == 5 ) {
        shipPackages();
    }
}
```

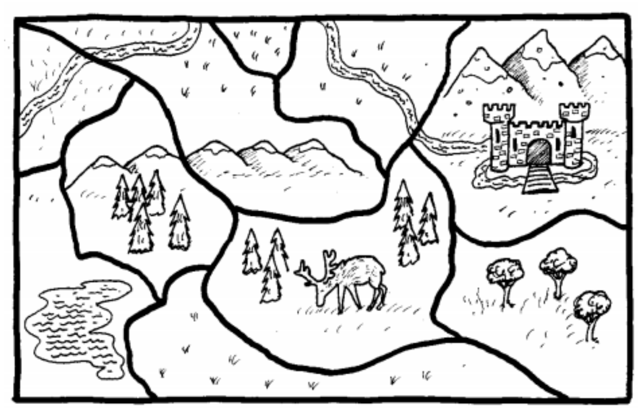
Greenfoot: subalgorithms in sequence

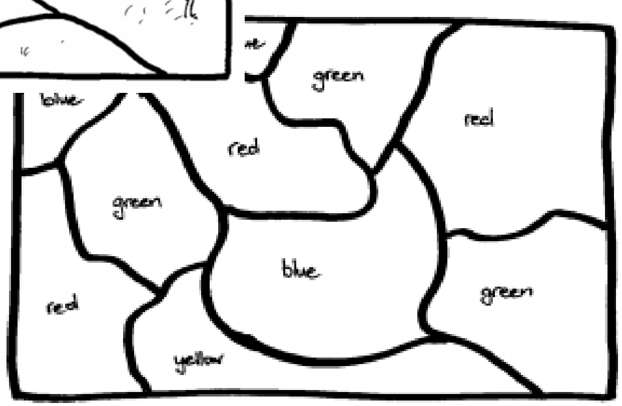


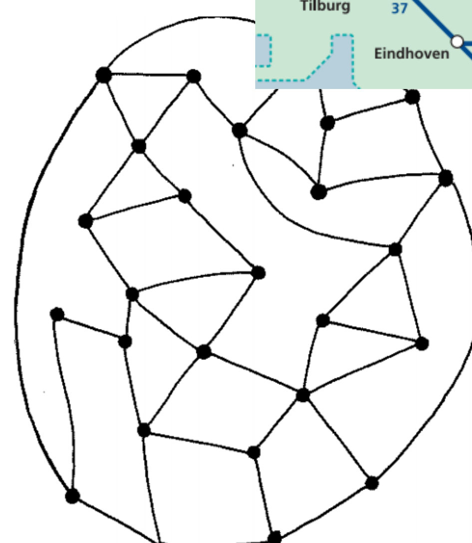
NP-Complete problems

- In search of efficient solutions

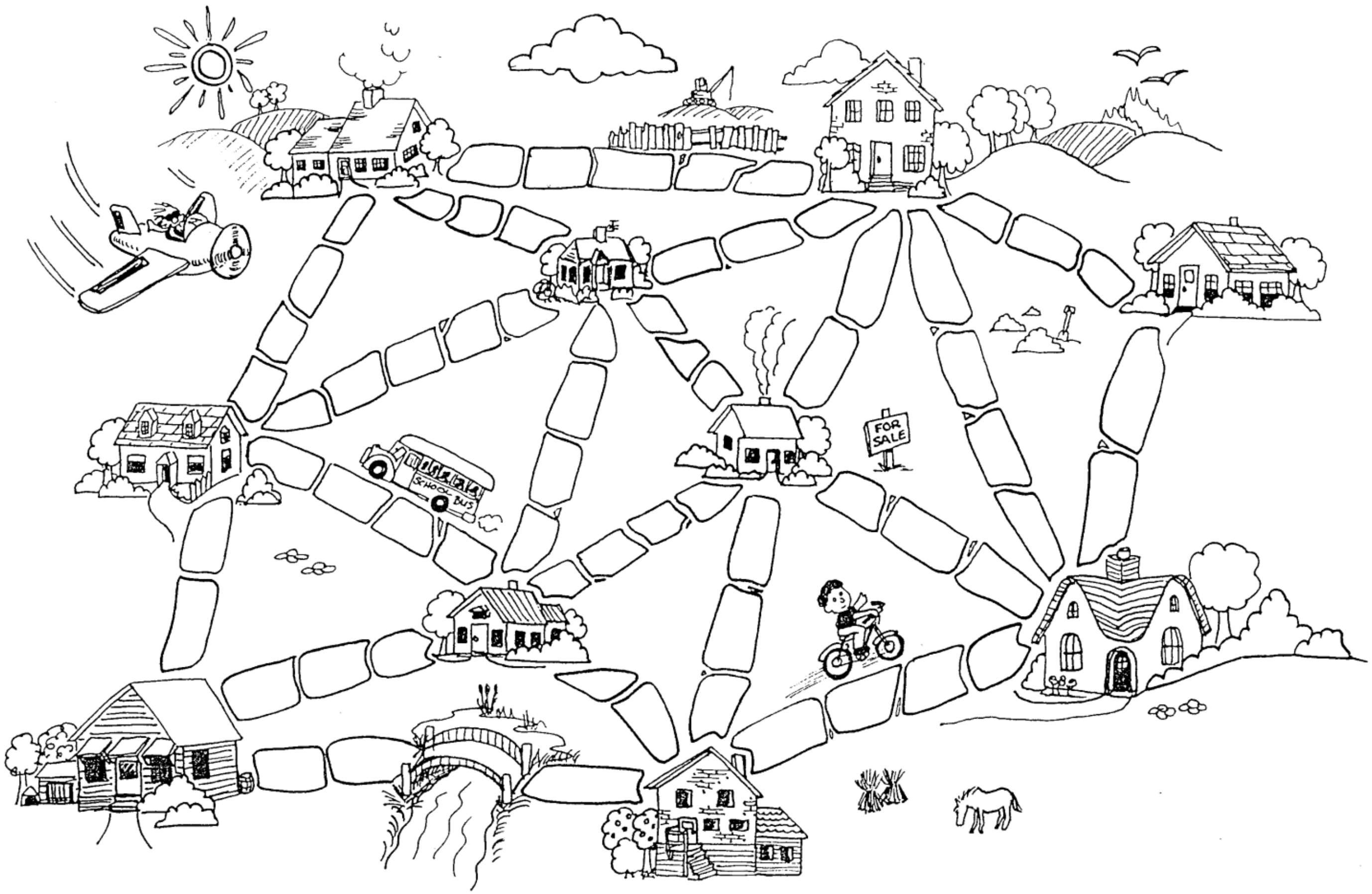








Muddy City



The Muddy City challenge

Problem:

- City has no roads
- Rain? Muddy boots!
- Not too much money: also want to build swimmingpool

Solution:

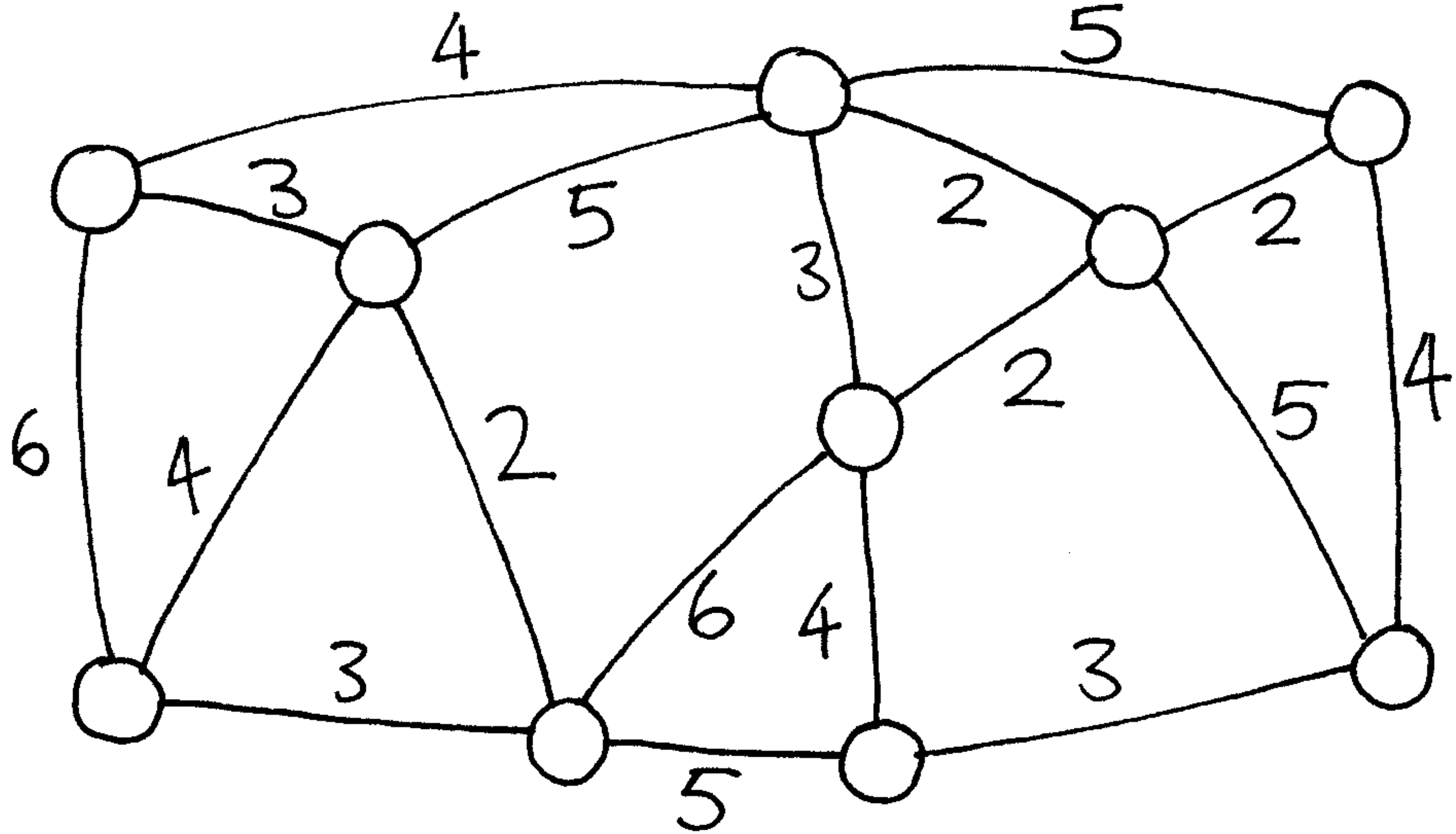
- Pave some streets
- Just enough for everyone to get around
- Cheap as possible (road length is price)

Your Muddy City: solutions

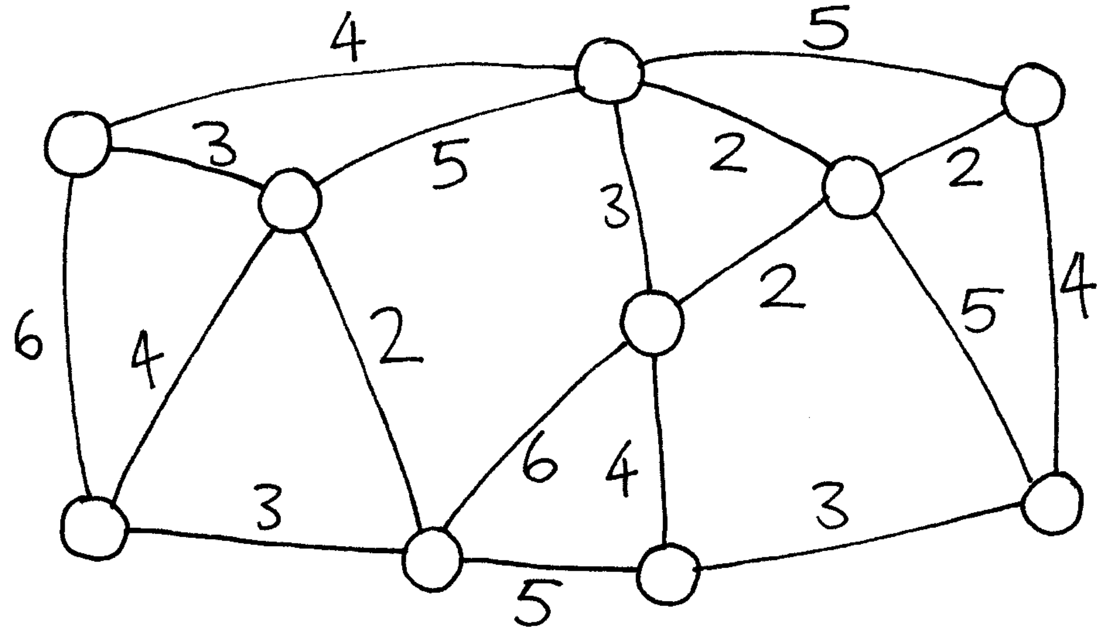
What did you come up with?

- How much will it cost?
- What was your strategy?

Muddy City Graph



Muddy city



Model: make a map (abstraction)

Strategy 1:

- Start with full map
- Remove expensive streets

Strategy 2:

- Try all possibilities and decide which is best

Strategy 3:

- Draw houses and add in streets, cheapest first

Muddy city

Strategy 1: Removing expensive streets

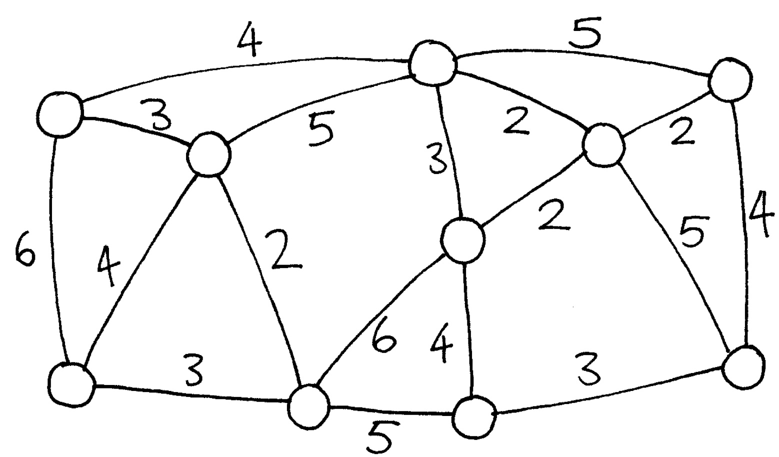
- Takes some effort

Strategy 2: Try all possibilities and decide which is best

- Brute force
- Lots of computation

Strategy 3: Draw in cheapest links

- Kruskal
- Efficient algorithm



Greedy Algorithm

1. Draw the houses (as nodes)
2. Sort the values (street lengths) into a list
3. Create a graph (in CS terms) by:
   a) For each element in the list
   b) Select cheapest link
   c) If no closed circuit will be made, draw link
   d) Remove value from list (whether drawn or not)

   Kruskal (1956)

Optimal solution **can be** found**:**

n houses => (n-1) streets

Efficient: in polynomial time in O(log n)

Muddy city: what it’s about

- **Minimal spanning tree** problem
  - Connecting all nodes
  - Minimal totaal length
- Efficient algorithms do exist

- Networks:
  - Power networks
  - Gas Pipelines
  - Computer networks
  - Telephone networks

NS price map:


Harderwijk
Apeldoorn
Zutphen
Hilversum
Amersfoort
Ede-Wageningen
Arnhem
Utrecht Centraal
Rhenen
Doetinchem
Tiel
Nijmegen
Geldermalsen
Oss
's-Hertogenbosch
Venray
Tilburg
Helmond
Eindhoven
Venlo
14
31
9
15
16
27
43
15
39
16
18
18
31
54
21
34
41
45
29
17
43
26
30
44
19
12
41
49
24
22
19
39
22
21
32
23
37
13
39

More complex: shortest route

People interested in shortest (not cheapest) route when travelling:

- by car
- airplane

Searching for the **best route** seams like a similar problem (to cheapest route), however:

- Convenience!
- Shorter more important than cheaper

Travelling Salesman problem

- Finding the shortest route passing by all houses

- No efficient method to find optimal solution is **EXISTS!**

- **NP-Complete**
- Cannot be calculated in **polynomial** time.

Brute-force algorithm:

- Always finds best solution
- Algorithm is easy to describe (and thus implement)
- Long time to solve: not efficient

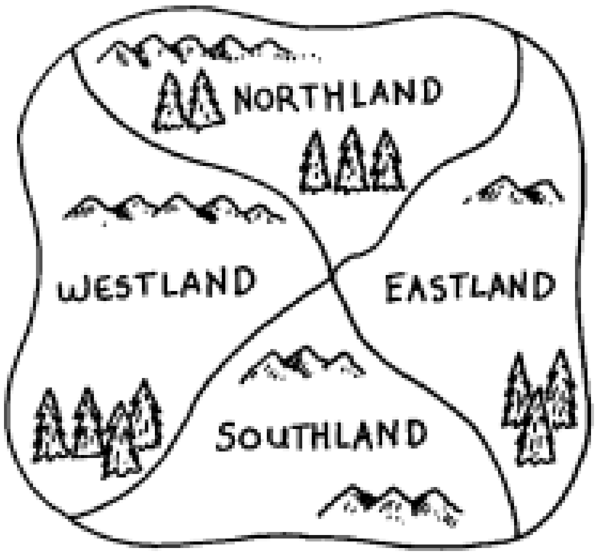
- 5 houses: (5-1)! /2 = 12 paths 10 sec
- 12 houses: (12-1)! /2=19958400 paths >31yr

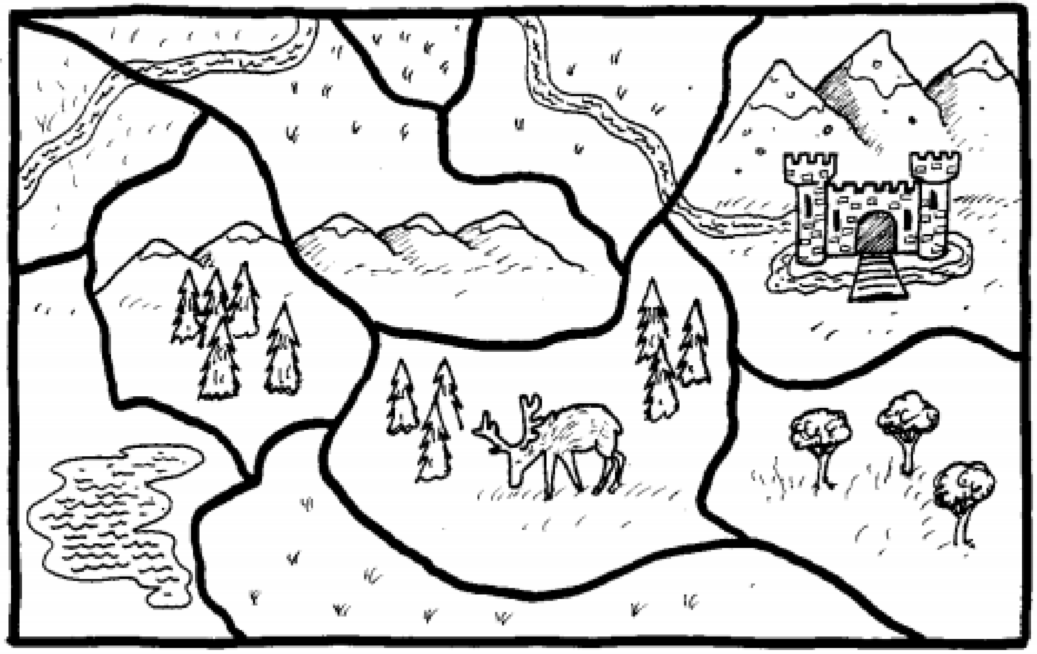
- Imagine scheduling for the whole country!
- Or re-scheduling after a snow-storm or power failure…

Graph coloring

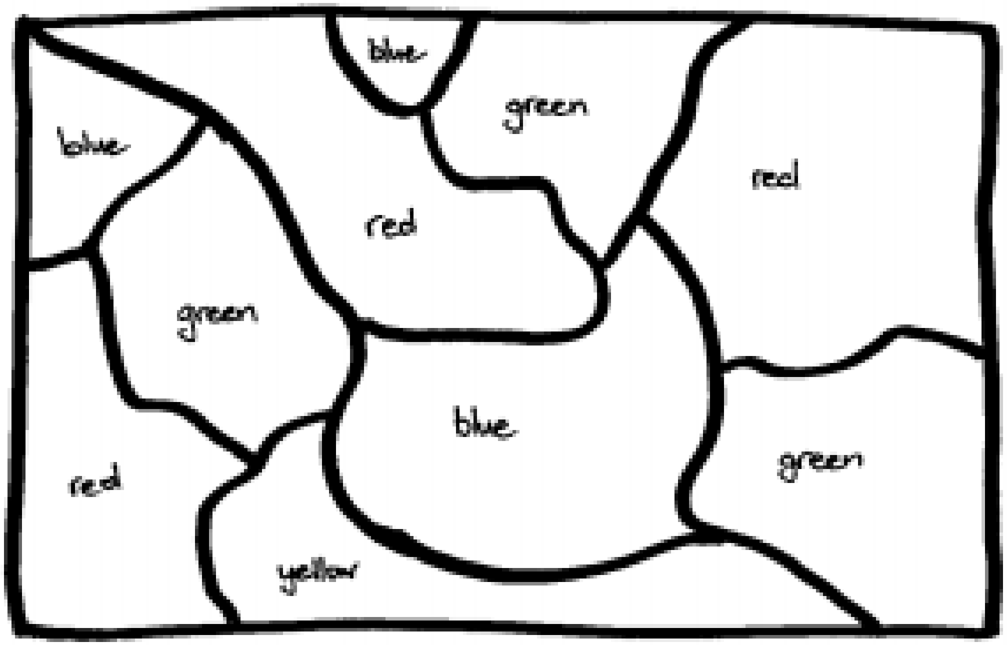
- No adjacent countries same color
- Minimum #colors?

How many colors do you need?





Solution to graph coloring



Graph coloring

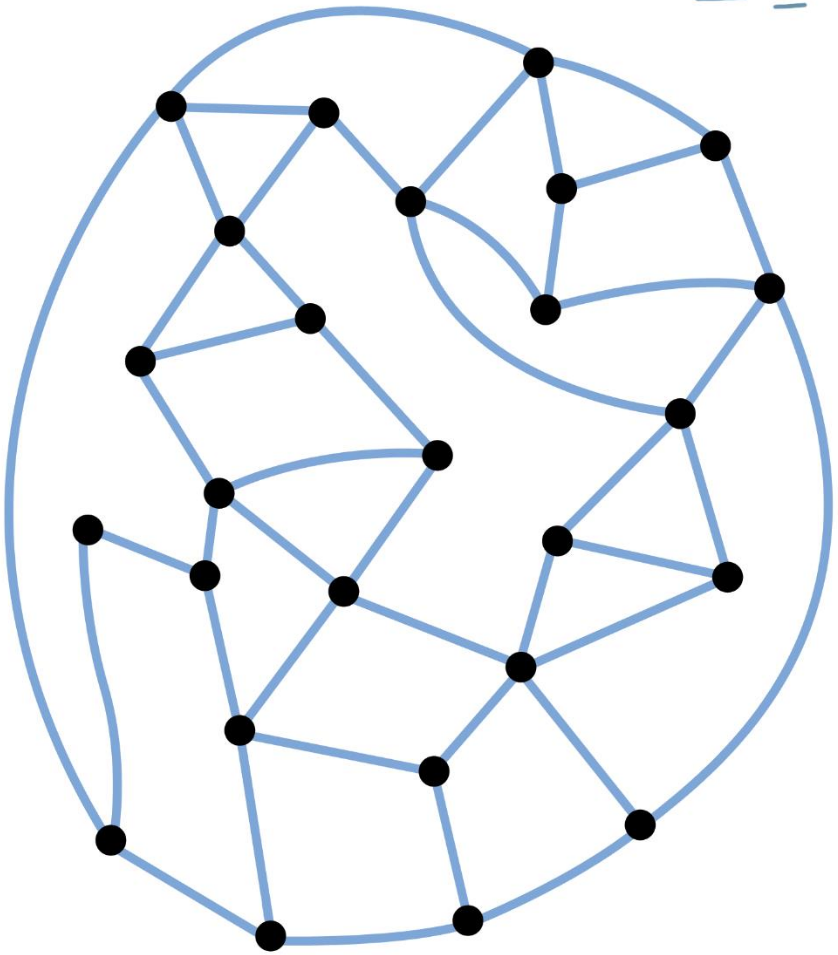
- **Any** map can be colored with **4** colors
- Theorem took 120 years to prove!
- But can it be done better?

- No efficient algorithm known for minimum #colors!

Dominating Sets

Goal: Max distance from house to mailbox is 1.5 blocks

- Where to place a mailbox?
- What is the minimum number of mailboxes you will need?



Minimum Dominating Sets

- Strategies for placing?
- What is minimum # of mailboxes needed?

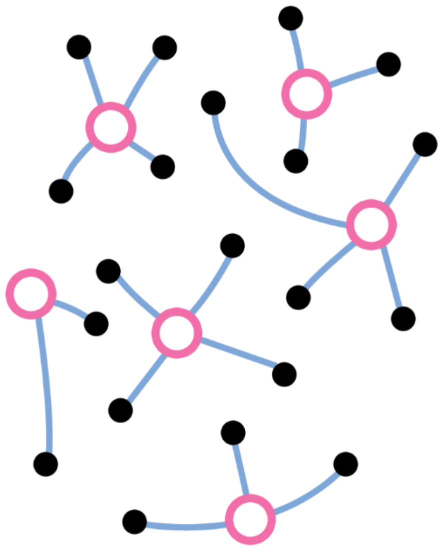
Possible strategy?

- Try **all** possibilities? Brute force
- Number of possibilities: $2^{\#intersections}$
- Big city? You'll be here for a while!
- Efficiency: exponential

- No efficient algorithm **known** to determine the minimum set!

Minimal solution: 6

- If you know the solution
- Creating the map is easy

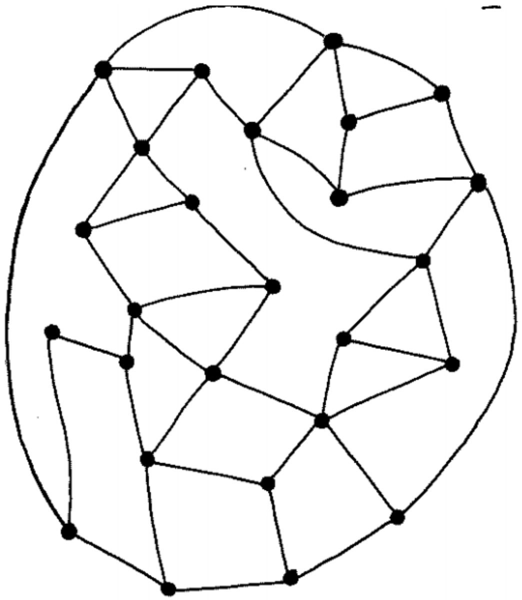


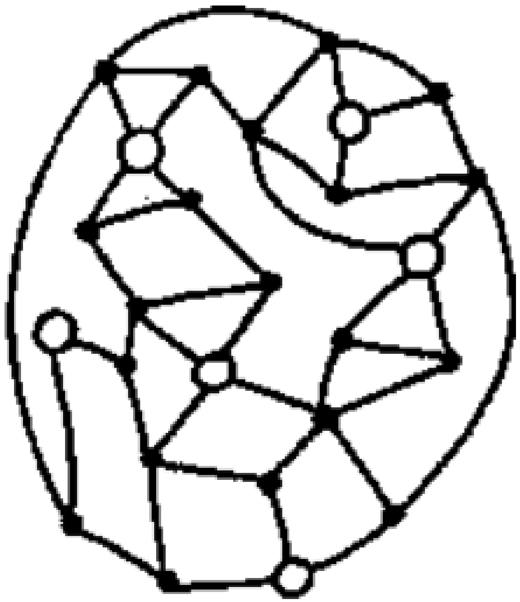
- Next step: connect all the dots

Minimum Dominating Sets

- Brute force: try all possibilities
- Possibilities: $2^{\#intersections}$

- Lots of intersections? Lots of work!!





Similar real-life situations

Stationing:

- Ambulances (max 15’)
- Placing stores, ice-cream trucks
- Army base-stations, watch towers, marine ships

Comparable optimization problems

- Travelling Salesman problem
- Minimum dominating sets
- Graph coloring
- Scheduling processes

    Best timetable for 30 teachers & 800 students?

    Takes years to solve!

    By then you'll be out of school

- In all cases:
- Optimal solution hard to find
- However, easy to check if a solution is optimal

Similar problems

- NP-Complete problems:
  - Graph-coloring
  - Minimum Dominating sets
  - Travelling Salesman problem

- All these problems have been proved to be similar
- You solve one, then you solve them all!

- Are you looking for eternal fame? (Turing Award)

Efficiency

- Trying to find better solution than brute force

Dodo's race (goal)

Who can make Dodo the smartest?

- Competition in class on March 18th
  - everyone's program will be run!

Highest score in

max 40 moves WINS!

 1 point

 5 points

Dodo’s race (rules)

Ground rules:

- Maximum steps: 40
- 15 blue eggs: each worth 1 point
- 1 Golden Egg: worth 5 points
- Mimi only moves using move()
- Max 1 move() per act()
- Competition will be held in a new world
- Highest score wins

Presentation: March 18th

- Presentation:
  - Present (describe) your algorithm to the class (2 minutes)
  - Test your algorithm against classmates

- Who will make the smartest Dodo?
- Think about efficiency (vs brute force)!

Computational thinking

- **Working in a structured manner:**
  - Breaking problems down into subproblems
  - Design, solve and test solutions to subproblems
  - Combining these (sub)solutions to solve problem
- **Analyzing** the quality of a solution
- **Reflecting** on the solution chosen and proces
- **Generalizing** and re-use of existing solutions

- Maps are an **abstraction** of reality

Wrapping up

Homework for Wednesday 8:30 March 9th:

- Assignment 7:
  - **Finish assignment 7 up to and including 4.3.7**
  - ZIP code and 'IN' and **email** to **Renske.weeda@gmail.com**
- If you want extra credit: do all of 4.4 and next week extra credit assignment (you may choose what you want to do)
- **Next week's plans:**
  - **in class- no theory (work on 4.4 – Dodo's race)**