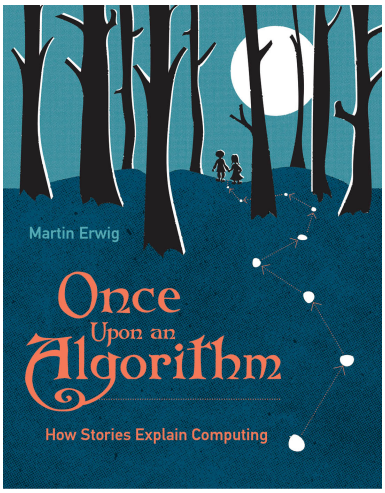

Story Programming Lecture Structure

Version: September 2017

This document contains a lecture plan for teaching an Introduction to Computer Science class based on the book *Once Upon an Algorithm: How Stories Explain Computing*. The material has been selected for a class that has two 50-minute lectures plus a 90-minute lab section per week.

Each week has a main subject and is based on one or more chapters of the book that should be assigned as background reading before the lecture.

Week 1 *Algorithms* [Chapter 1]

- What is an algorithm?
- What is a computer?
- Computation as the result of executing an algorithm by a computer
- Computation transforms representation
- What is a problem, and what does it mean to solve a problem?
- Not every algorithm solves a problem given constraints and context
- What does correctness mean?
- In what ways can an algorithm be incorrect? (wrong results, getting stuck, nontermination)
- How can we asses correctness? (testing, proof)

Week 2 *Algorithms* [Chapter 2]

- Parameters and arguments
- Algorithms solve *classes* of problems
- Runtime complexity measured as growth rate
- Constant vs. linear vs. quadratic runtime
- Trading memory for runtime

Week 3 *Representation* [Chapter 3]

- Signs (signifier vs. signified ≅ syntax vs. semantics)
- Semantics (≅ interpretation) of signs depends on context
- Different representations for the same kind of data (numbers)
- Transformation of signifiers (syntax)

Week 4 *Representation* [Chapter 4]

- Collections
- Data types (set, queue, stack)
- Data structures (list, array, tree)
- Implementing sets with lists and arrays
- Difference between data types and data structures

Week 5 *Searching & Sorting* [Chapters 5 and 6]

- Binary search
- Insertion & selection sort
- Quicksort & mergesort
- Divide-and-conquer
- Linearithmic runtime

Week 6 *Languages & Programming Languages* [Chapters 8 and 9]

- Syntax
- Grammar (nonterminals, terminals, production, sentences, syntax tree)
- Semantics
- Introduction to Haskell / Python

Week 7 *Control Structures* [Chapters 10]

- Sequential composition
- Conditional
- Repeat and while loops

Week 8 *Decomposition & Recursion* [Chapters 12]

- Recursion
- Base case
- Forms of recursion (descriptive vs. unfolded, bounded vs. unbounded, direct vs. indirect)
- Nontermination
- Recursion vs. loops

Week 9 *Types & Abstraction* [Chapters 14 and 15]

- Types vs. values
- Typing rules
- Reasoning (with types) about programs, finding type errors
- Abstraction vs. generalization
- Definition & use of abstractions
- Different forms of abstraction (functional, data, time)

Week 10 *Limits of Computation* [Chapters 7 and 11]

- Intractable problems
- Knapsack problems
- Exponential growths
- P=NP problem
- Undecidable problems
- The halting problem