1. Introduction to algorithms (6 days)
   1.1 Terminology and definitions [secs 1.1–1.2]
   1.2 Unsolvable and Intractable problems; NP complete problems
   1.3 Designing and analyzing algorithms; fast exponentiation, insertion sort [secs 2.1–2.2, fast exponentiation handout, straight insertion sort handout]
   1.4 Classifying functions by their rates of growth [sec 3.1, rate of growth of functions handout]
   1.5 Common functions [sec 3.2, factorials handout, rate of growth of polynomials, exponentials, and logarithms handout]

2. Recursion, recurrence equations and inequalities (6 days)
   2.1 Recursive vs iterative definitions, recursive algorithms [sec 2.3]
   2.2 Analysis of recursive algorithms, recurrences [chap 4 introduction]
   2.3 Solving recurrences: The substitution method [sec 4.1]
   2.4 Solving recurrences: The recursion tree method (brief discussion) [sec 4.2]
   2.5 Solving recurrences: The Master Theorem and an extension [secs 4.3, sec 4.4 to page 80]

3. Introduction to sorting algorithms (4 days)
   3.1 The sorting problem, types of sorting algorithms, stability [pp 123–126]
   3.2 Heapsort and priority queues [chap 6]
   3.3 Lower bounds for sorting [sec 8.1]

4. Divide-and-conquer algorithms (9 days)
   4.1 The divide-and-conquer method [sec 2.3.1, first page]
   4.2 Multiplication of polynomials, multiplication of integers
   4.3 Quicksort [chap 7]
   4.4 Selecting the \( k \)th smallest in linear expected time [secs 9.1–9.2]
   4.5 Multiplication of polynomials and the FFT [secs 30.1–30.2]

5. Dynamic Programming (5 days)
   5.1 Dynamic programming vs divide-and-conquer [chap 15 introduction]
   5.2 Matrix chain multiplication [sec 15.2]
   5.3 More about dynamic programming [sec 15.3]
   5.4 Longest common subsequence [sec 15.4]
   5.5 All-pairs shortest paths in graphs and digraphs [sec 25.2]

6. Greedy algorithms (4 days)
   6.1 The greedy method, simple examples [sec 16.1–16.2]
   6.2 Huffman codes (brief discussion, time permitting) [sec 16.3]
   6.3 Minimal spanning trees: Prim’s algorithm and Kruskal’s algorithm [chap 23]

7. Graphs and digraphs, searching in graphs (6 days)
   7.1 Definitions, adjacency matrix and adjacency list representations [sec 22.1]
   7.2 Breadth-first search in graphs and digraphs (brief discussion) [sec 22.2]
   7.3 Depth-first search in graphs and digraphs; classification of edges [sec 22.3]
   7.4 Directed acyclic graphs; topological sort [sec 22.4]
   7.5 Strongly connected components of a digraph [sec 22.5]

8. String-matching algorithms (3 days)
   8.1 The string-matching problem, straightforward solutions [chap 32 intro, sec 32.1]
   8.2 String-matching with finite automata [sec 32.3]
   8.3 The Knuth-Morris-Pratt algorithm (brief discussion, time permitting) [sec 32.4]