Summary: Algorithmics

Chapter 1: Introduction into formal algorithmics

Comparison of fundamental sorting algorithms (function and run time).
Landau symbols (big O notation): Definition, hierarchies, comparison of different classes, relevance for the complexity of algorithms and problems, upper and lower limits.
Independent construction of easy recursive run time formulae (e.g. mergesort, quicksort) and proof of their (nonrecursive) asymptotic run time behaviour via mathematical induction.
~~Master-Theorem for run time estimates in recursive formulae.~~
RAM: fundamental comprehension, relevance for complexity.
Proof of the lower bound for sorting via comparisons.

Chapter 2: Advanced searching and sorting

Order statistics: Randomised und deterministic algorithm, recursive run time formula for run time, knowledge and proof (only outline) of complexity class.
Searching in sorted arrays:
- binary search, interpolation search, quadratic binary search:
  - functionality, knowledge of complexity class, proofs for this (no interpolation search).

Sorting in finite or limited domains:
- Countingsort, radixsort, bucketsort:
  - functionality, knowledge of complexity class, ~~proofs for this~~.

Summary: Algorithmics

Chapter 3: Solutions for the dictionary problem

Definition of the dictionary problem, run time goals.
Hashing: Basic idea, comparison to tree methods (also storage).
(2,3)-trees: exact functionality, generalisation to (a,b)-trees.
~~AVL trees~~, red-black trees: Transformation into (2,4)-trees.
~~Trie trees, B trees~~.
For all methods: Basic ideas for the proof of run time estimates (no details)

Bellman's algorithm: Functionality, abstract and at an example (code is given, but not explanations).
Proofs of correctness (only idea) and run time estimate (exact!)
~~Improvement by Knuth~~.

Summary: Algorithmics

Chapter 4: Graph algorithms

Data structures heap und union-find with run times and applications in the implementation of graph algorithms.
Kruskal‘s algorithm: Exact functionality, run time estimate ~~and proof of correctness.~~
Dijkstra‘s algorithm: Exact functionality, run time estimate ~~and proof of correctness~~.
All Pair Shortest Path problem: Algorithmus of Floyd-Warshall (including proofs of correctness and run time)
Relation between path computations and matrix multiplication. Fast computation of matrix powers. Matrix multiplication and applications according to last assignment.

Maximum flows:
Terminology: Augmenting paths, residual graph, level graph.
Theorem of Ford-Fulkerson (min cut = max flow): proposition, only trivial proof directions.
Algorithm of Edmonds-Karp: Exact functionality, run time estimate, proof: outline only, no details.
Algorithm of Dinic: Exact functionality, run time estimate, proof: outline only, no details.
Illustration of both flow algorithms at an example.

Different variants of the definition of the matching problem für k (kDM), complexity classes.
Bipartite matching: Relation to the max flow problem (Transformation), ~~run time estimates, proofs~~.
General matching: Algorithm of Edmonds, illustration at an example (blossom detection), ~~proof of correctness~~.

Summary: Algorithmics

Chapter 5: String matching

Trivial method: Run time.
Algorithm of Knuth-Morris-Pratt: exact functionality (including prefix function),
illustration at an example,
run time estimate: crucial idea (consumer argument),
~~proof of correctness (invariants)~~.

Chapter 6: Fundamentals of algorithmic geometry

Basic problems Closest Reference Point, Closest Pair, Minimum Spanning Tree, Convex Hull:
Trivial algorithms with run time estimates.
Voronoi diagrams: Definition and data structure, application to basic problems with run time estimates, relation to Delaunay triangulation.
Sweep method: Basic idea, characteristic properties.
Explanation of terminology for the following examples:
Closest Pair für d=1,2.
Computation of a Voronoi diagram.
Exact comprehension of the invariants of plane sweep for Voronoi diagrams, details of the operations for maintaining SSS and EPS, ~~details with mathematic formulae~~, run time estimates, no correctness proof, for run time estimates only crude arguments (transformation to the dictionary problem).