Part I of the course: Optimization Problems

we want to find valid solution minimizing cost (or maximizing profit)

⇒ always start by investigating structure of optimal solution

**backtracking**: just try all solutions (pruning when possible)
- extra exercises: backtracking algorithms for NP-hard problems
  Vertex Cover, 3-SAT, Hamiltonian Cycle, …

**greedy algorithms**: make choice that seems best, recursively solve remaining subproblem
- proof is crucial; make sure you know proof structure by heart
- extra exercises: greedy algorithms explained in book

**dynamic programming**: give recursive formula, fill in table
- make sure you know steps to design dynamic-programming algorithm
- extra exercises: dynamic-programming algorithms explained in book
Part II of the course: optimization problems on graphs

single-source shortest paths
Find shortest path from source vertex to all other vertices

all-pairs shortest paths
Find shortest paths between all pairs of vertices

maximum flow
Find maximum flow from source vertex to target vertex

maximum bipartite matching
Find maximum number of disjoint pairs of vertices with edge between them
single-source / all-pairs shortest paths

- you should know and understand the various algorithms
  SSSP: Bellman-Ford, Dijkstra, algorithm on DAGs
  APSP: relation to matrix multiplication, Johnson’s reweighting
- extra exercises: give algorithms, give proofs of important lemmas

maximum flow + maximum bipartite matching

- definition of flow network, flow
- Ford-Fulkerson method + Edmonds-Karp implementation
  (concept of augmenting flow, residual path, etcetera)
- relation between flows and cuts (e.g. max flow = min cut)
- integer capacities: there is always a max flow that is integral
  ➔ basis for many applications, e.g. for max bipartite matching)
Part III of the course: Miscellaneous topics

NP-completeness
- definitions and concepts: P, NP, reductions
- list of NP-complete problems
  you should be able to reproduce NP-completeness proofs …
  (they also teach you tricks for other NP-completeness proofs)

approximation algorithms
- definitions and concepts: approximation factor, (F)PTAS
- how to prove approximation ratio: compare to lower (upper) bound

linear programming
- definitions and concepts: LP, standard form integer LP, 0/1-LP
- how to model a problem as an LP
The exam

- most exercises are of the same type as the homework exercises
- about 1/3 of exercises is (almost) identical to homework exercises
- exam may also have some questions where the answer can be found literally on the slides or in the book (reproduce algorithm or proof, definitions)

How to prepare

1. study the slides (only look into the book when there are things in the slides that you do not understand)
2. solve all the homework exercises again!
3. reproduce algorithms / proofs from the book
What we did not do in the course …

… lots, in particular …

- algorithms for big data
- geometric algorithms
- algorithms for geographic data
Algorithms for Big Data

Data, data, everywhere. Economist 2010
I/O- and caching behavior

analysis of running time: (asymptotically) determine number of elementary operations as a function of input size

basic assumption: elementary operations take roughly same amount of time

is this justified?

Not when data is stored on disk:
- disk operations can be factor 100,000 (or more) slower than operations on data in main memory

- how can we design I/O-efficient algorithms?
- how should we analyze such algorithms?
- can / should we also worry about caching?
streaming algorithms

data stream model:
- data is read sequentially in one (or few) passes
- size of the working memory is much smaller than size of data

example: traffic going through high speed routers (*data can not be revisited*)

- how can we design algorithms for data streams?
  (e.g. maintain approximate median of stream of numbers)
- how should we analyze such algorithms?

➔ take *Advanced Algorithms*  
 (*big data + more approximation algorithms*)
Geometric algorithms
computer-aided design and manufacturing (CAD/CAM)

- motion planning
- virtual walkthroughs

3D model of power plant
(12,748,510 triangles)
3D copier

3D scanner

3D printer
many areas where geometry plays important role:

- geographic information systems
- computer-aided design and manufacturing
- robotics and motion planning
- computer graphics and virtual reality
- databases
- structural computational biology
- and more …
Geometric data structures

- store geometric data in 2-, 3- or higher dimensional space such that certain queries can be answered efficiently

- applications: GIS, graphics and virtual reality, …

Examples:

- point location
- range searching
- nearest-neighbor searching

want to know more about algorithms and data structures for spatial data?

⇒ take Geometric Algorithms
Algorithms for Geographic Data
geographic information systems (GIS)

30 cities: $4^{30} = 10^{18}$ possibilities
automated cartography

examples:

- map labeling
- scaling and simplification
- schematization
- maps for various types of data, e.g., flows
trajectory data

data of
- animals, humans, vehicles, …

Examples of analysis tasks:
- How similar are two trajectories?
- How to find clusters?
- How to represent a cluster?
- How to simplify a trajectory?
- …

➔ take *Algorithms for Geographic Data*
well, except for the exam ...

comments / suggestions for improvements are welcome