YSC2229: Introductory Data Structures and Algorithms

Wrapping Up

Why take this class?

- You will learn:

Why take this class?

- You ~~will~~ have learned:
  - To *understand* and *evaluate* classic algorithms
  - How to design algorithms that are *fast*
  - How to choose *the right* data structures for your problems
  - How to exhaustively *test* your code
  - A little bit about *compilers* and *memory management*
  - More functional and imperative programming in OCaml
  - How to be a better programmer (not just in OCaml, but any language)

Now you know about…

- Correctness and Loop Invariants
- Time Complexity and Order Notation
- Reasoning about Recursive Algorithms
- Searching Algorithms
- InsertSort, MergeSort, QuickSort
- Best-case Sorting Complexity
- Sorting in Linear Time: BucketSort, RadixSort
- Binary Heaps, HeapSort, Priority Queues
- Abstract Data Types: Stacks, Queues
- Dynamic Memory Allocation and Reclamation
- Hash-Tables
- Equivalence Checking and Union-Find
- Bloom Filters and False Positives
- Substring Search Algorithms
- Constraint Solving and Backtracking
- Optimisation and Dynamic Programming
- Input/Output and Binary Encodings
- Data Compression and Huffman Encoding
- Representing Sets via Binary Search Trees
- Graphs, Graph Traversals, Topological Sort
- Shortest Paths, Spanning Trees
- Computational Geometry: Segments, Intersections
- Operations with Segments, Polygons, Convex Hulls

Now you know about…

- Correctness and Loop Invariants
- Time Complexity and Order Notation
- Reasoning about Recursive Algorithms
- Searching Algorithms
- InsertSort, MergeSort, QuickSort
- Best-case Sorting Complexity
- Sorting in Linear Time: BucketSort, RadixSort
- Binary Heaps, HeapSort, Priority Queues
- Abstract Data Types: Stacks, Queues
- Dynamic Memory Allocation and Reclamation
- Hash-Tables
- Equivalence Checking and Union-Find
- Bloom Filters and False Positives
- Substring Search Algorithms
- Constraint Solving and Backtracking
- Optimisation and Dynamic Programming
- Input/Output and Binary Encodings
- Data Compression and Huffman Encoding
- Representing Sets via Binary Search Trees
- Graphs, Graph Traversals, Topological Sort
- Shortest Paths, Spanning Trees
- Computational Geometry: Segments, Intersections
- Operations with Segments, Polygons, Convex Hulls

Before IDS&A

After IDS&A

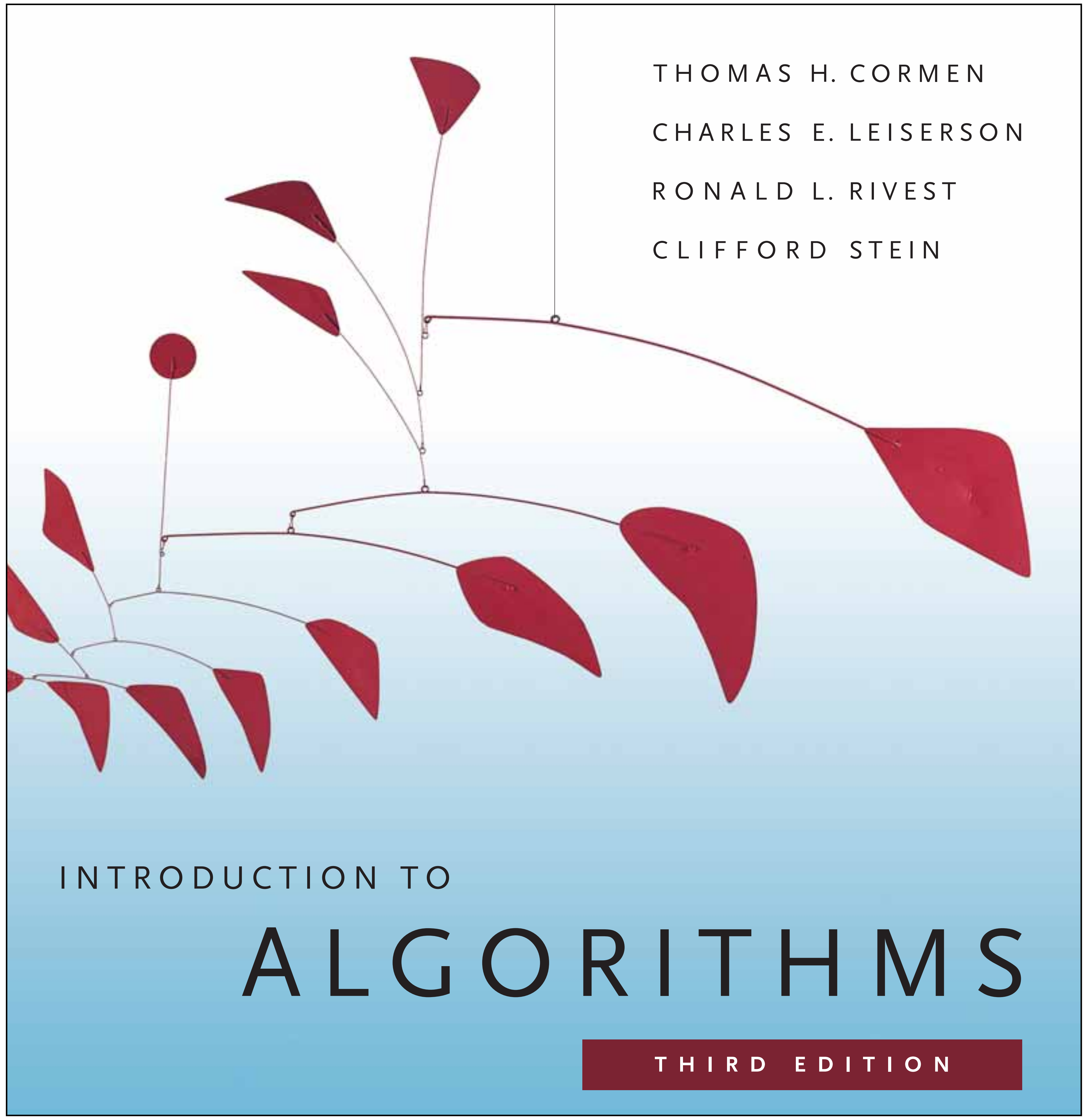
pls no more oh-caml

I use functors and Bloom filters
to write tail-recursive SAT-solvers
and compress binary files
before breakfast

More DS&A in Yale-NUS MCS

- YSC3203 Advanced Algorithms and Data Structures
  - *probabilistic algorithms, formal proofs of complexity*
- YSC3236 Functional Programming and Proving
  - *formal reasoning about correctness of algorithms*
- YSC3232 Software Engineering
  - *team work on large software projects*
- YSC4231: Parallel, Concurrent and Distributed Programming
  - *algorithms for multiprocessor computers and distributed systems*
- YSC4230: Programming Language Design and Implementation
  - *more on compilers, memory management, program optimisations*

Where to Go From Here



THOMAS H. CORMEN

CHARLES E. LEISERSON

RONALD L. RIVEST

CLIFFORD STEIN

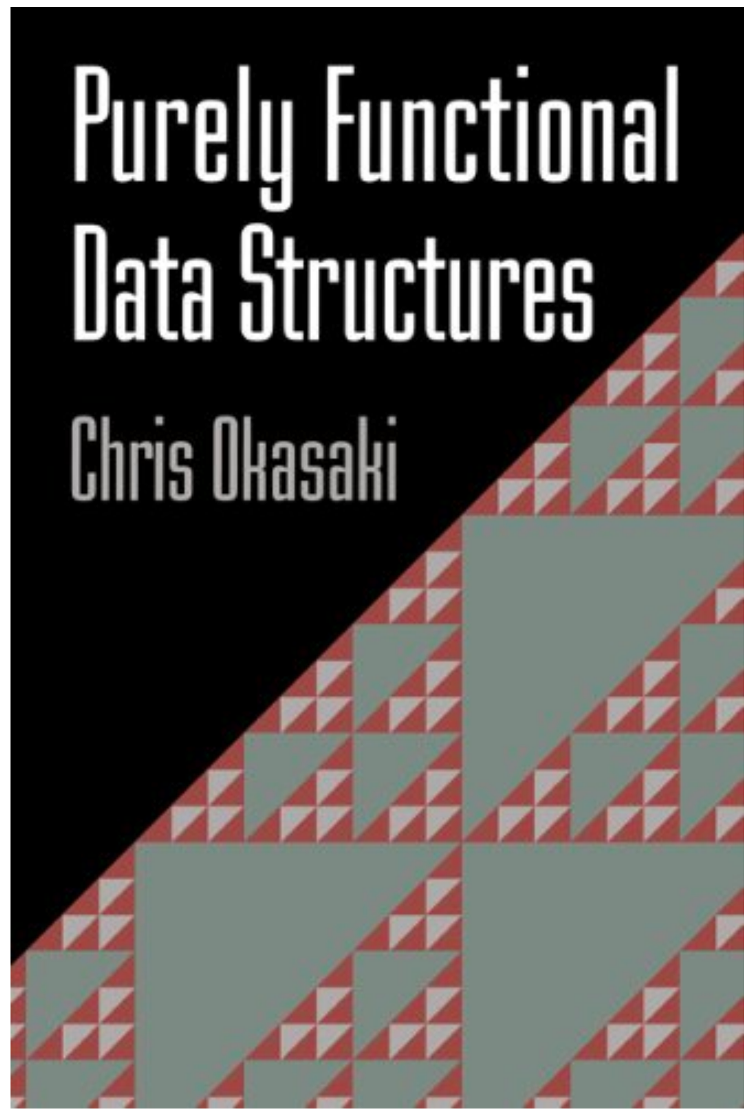
INTRODUCTION TO

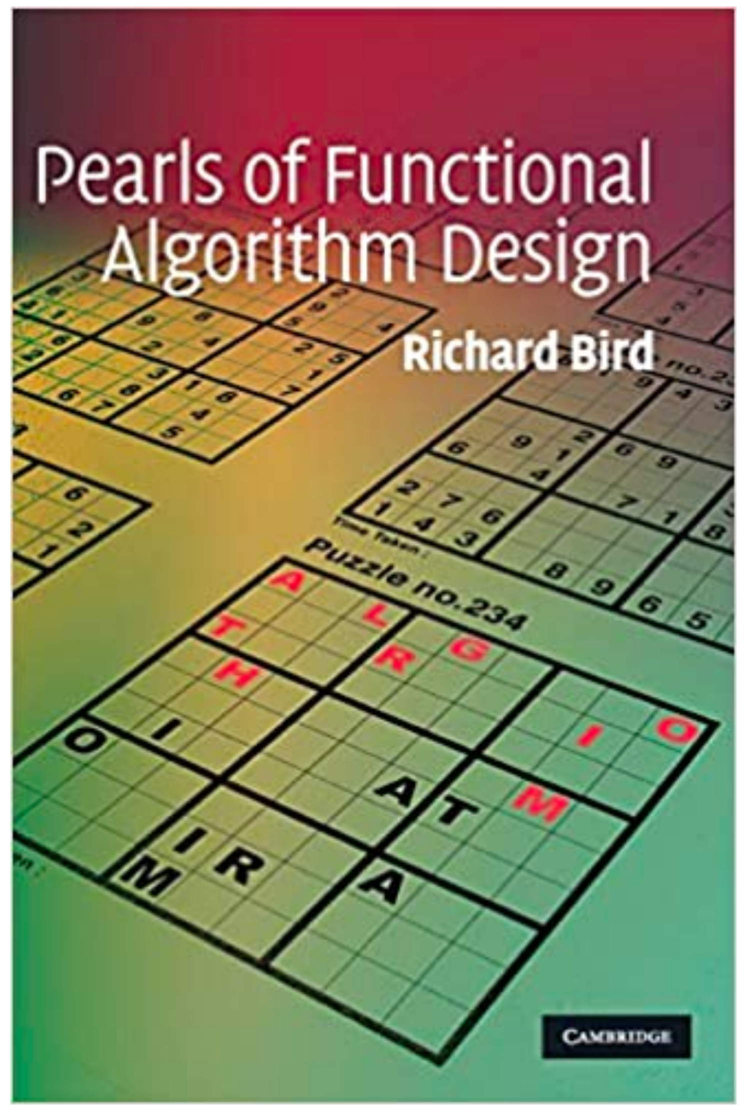
ALGORITHMS

THIRD EDITION

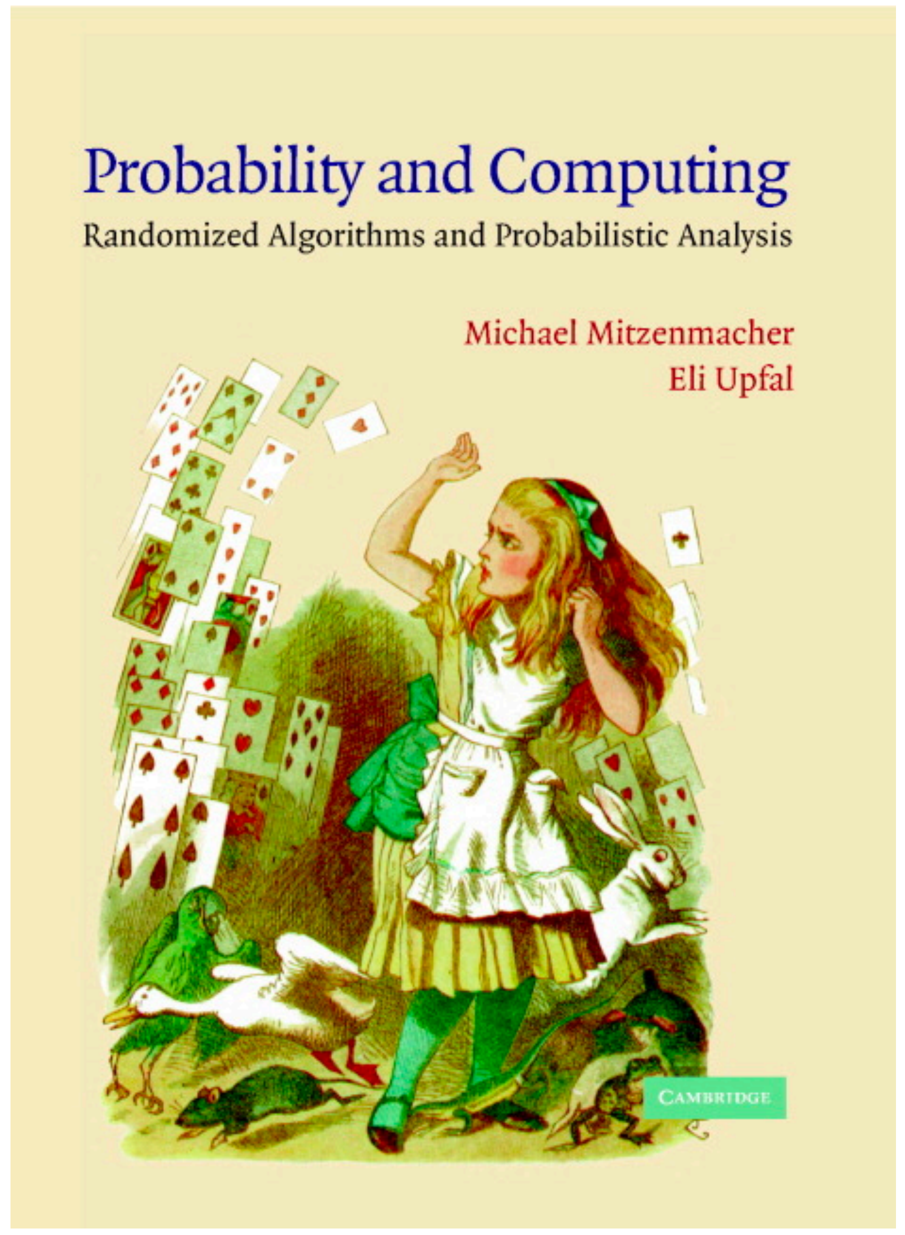
Where to Go From Here

- Any area that uses *computing* deals with *algorithms*
- Algorithms and data structures are *everywhere!*
- Here are some further reading suggestions…

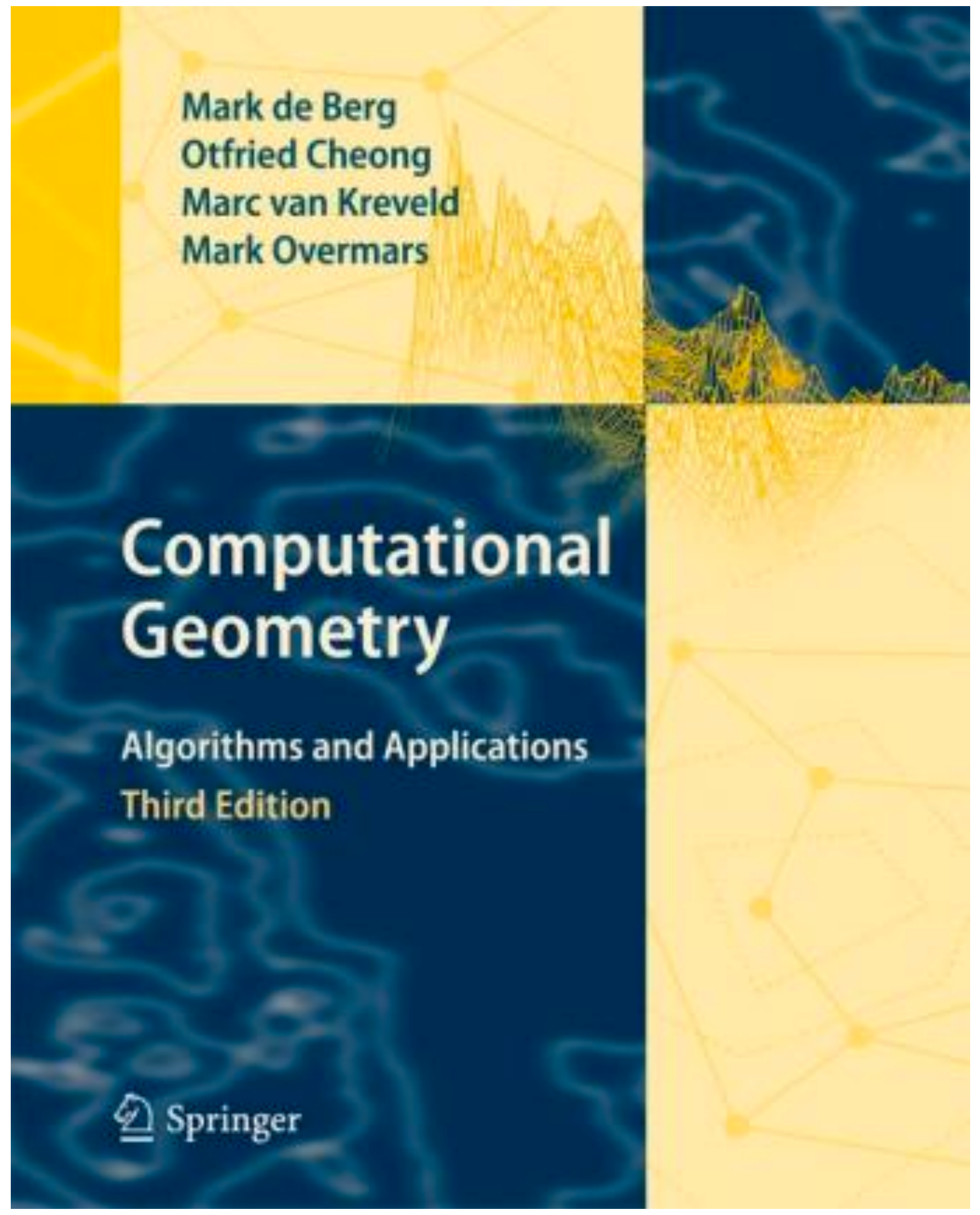




… if you're into functional programming


Probability and Computing
Randomized Algorithms and Probabilistic Analysis
Michael Mitzenmacher
Eli Upfal
CAMBRIDGE

… if you're into probabilities and big data



… if you're into elegant math

Donald Knuth
Author of KMP, TeX, "The Art of Computer Programming"
1974 Turing Award Winner

The *best programs* are written
so that computing machines can *perform them quickly*
and so that human beings can *understand them clearly*.

The End

Thanks!

P. S.

Please, don't forget to submit the module evaluation
by **23 April 2021!**