Long-term Goal:

Ability to design and analyze algorithms

1. COMS10007: Algorithms (1st year)
2. COMS21103: Data Structures and Algorithms (2nd year)
3. COMS31900: Advanced Algorithms (3rd year)
4. ... (4th year)

Projects:

- Final projects
- Summer internships (after the second year)
- PhD theses
Exam

- June, 6th at 9:30 am
- 2 hours

The Exam will test both your **skills** and **knowledge**

**Two Key Ingredients:**

1. **Tools/skills:** $O$-notation, recurrences, loop invariants (induction), mathematics (e.g., bounding sums), . . .

2. **Knowledge:** algorithms, algorithmic design principles (divide-and-conquer, dynamic programming, . . .)
Key Skills:
- $O$-notation ($\Omega, \Theta$), formal proofs (e.g. is $f \in O(g)$?), racetrack principle, . . .
- Recurrences, substitution method, recursion tree method, . . .
- Runtime analysis

Key Knowledge:
- Precise definitions, principles (e.g., dynamic programming)
- Algorithms (e.g. sorting, binary search, Fibonacci, . . .)

Material: Everything, except excluded material
<table>
<thead>
<tr>
<th>Week</th>
<th>Day</th>
<th>Date</th>
<th>Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Mon</td>
<td>28-Jan</td>
<td>Lecture 1: Introduction, peak finding</td>
</tr>
<tr>
<td></td>
<td>Tue</td>
<td>29-Jan</td>
<td>Lecture 2: O-notation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Remarks: Throughout this course, ( \log(n) ) denotes the binary logarithm, i.e., ( \log(n) = \log_2(n) ). In the slides used in this course, ( \log(n) ) was wrongly computed with ( 1/n ), while it should of course be ( 1/(n \ln(2)) ), with ( \ln(n) ) being the logarithm to base ( e ) in the slides.</td>
</tr>
<tr>
<td>14</td>
<td>Mon</td>
<td>04-Feb</td>
<td>Lecture 3: Theta, Omega, RAM Model</td>
</tr>
<tr>
<td></td>
<td>Tue</td>
<td>05-Feb</td>
<td>Lecture 4: Linear and binary search, proofs by induction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Exercise class 1</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Remarks: Exercises on O-notation, Omega, and Theta can all be solved by finding constants ( c,n_0 ) such that the definition of Theta, Omega is fulfilled. Recall that any constants that fulfill the respective definition are fine, i.e., you do not need to find the best ones. Solutions will be provided soon.</td>
</tr>
<tr>
<td>15</td>
<td>Mon</td>
<td>11-Feb</td>
<td>Lecture 5: Loop Invariants, Insertion sort</td>
</tr>
<tr>
<td></td>
<td>Tue</td>
<td>12-Feb</td>
<td>Lecture 6: Sorting problem, first part of merge sort algorithm</td>
</tr>
<tr>
<td>16</td>
<td>Mon</td>
<td>18-Feb</td>
<td>Lecture 7: Second part of merge sort, maximum subarray problem (slides of lectures 6 and 7 are combined)</td>
</tr>
<tr>
<td></td>
<td>Tue</td>
<td>19-Feb</td>
<td>Lecture 8: Trees, first part of heap-sort</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Exercise class 2</strong></td>
</tr>
<tr>
<td>17</td>
<td>Mon</td>
<td>25-Feb</td>
<td>Lecture 9: Second part of heap-sort (slides of lectures 8 and 9 are combined)</td>
</tr>
<tr>
<td></td>
<td>Tue</td>
<td>26-Feb</td>
<td>Lecture 10: Quicksort</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td><strong>Reading Week</strong></td>
</tr>
<tr>
<td>19</td>
<td>Mon</td>
<td>11-Mar</td>
<td>Lecture 11: Runtime of Quicksort</td>
</tr>
<tr>
<td></td>
<td>Tue</td>
<td>12-Mar</td>
<td>Lecture 12: Sorting LB, Countingsort, Radixsort</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>In-class Test</strong></td>
</tr>
<tr>
<td>20</td>
<td>Mon</td>
<td>18-Mar</td>
<td>Lecture 13: Recurrences (substitution method, recursion-tree method)</td>
</tr>
<tr>
<td></td>
<td>Tue</td>
<td>19-Mar</td>
<td>Lecture 14: Recurrences continued (slides of lectures 13 and 14 are combined)</td>
</tr>
<tr>
<td>21</td>
<td>Mon</td>
<td>25-Mar</td>
<td>Lecture 15: Fibonacci Numbers</td>
</tr>
<tr>
<td></td>
<td>Tue</td>
<td>26-Mar</td>
<td>Lecture 16: Dynamic Programming - Pole Cutting</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Exercise class 3</strong></td>
</tr>
<tr>
<td>22</td>
<td>Mon</td>
<td>01-Apr</td>
<td>Lecture 17: Dynamic Programming - Matrix Chain Parenthesization</td>
</tr>
<tr>
<td></td>
<td>Tue</td>
<td>02-Apr</td>
<td>Lecture 18: Elements of Dynamic Programming 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Easter Break</strong></td>
</tr>
<tr>
<td>23</td>
<td>Mon</td>
<td>29-Apr</td>
<td>Lecture 19: Elements of Dynamic Programming 2</td>
</tr>
<tr>
<td></td>
<td>Tue</td>
<td>30-Apr</td>
<td>Lecture 20: Peak Finding</td>
</tr>
</tbody>
</table>
Material

O-notation
↓
Loop Invariants
↓
Sorting
↓
Recurrences (Divide-and-Conquer)
↓
Dynamic Programming
No Questions on: (only holds for exam, not for repeat exam!)

- RAM Model
- 2D peak finding
- Maximum Subarray
- Exercise sheet 4
- No loop invariant...!

(However still helpful to know!)
How to study for the Exam?

Overview
- Get a complete picture of the material
- Story behind it

Algorithms
- Understand algorithms, explain in words
- Example runs
- Understand their runtimes
- No need to give code
Answer the questions: Knowledge

- What is ... (any key word/concept/algorithm)?
  

- How does ... work? (e.g. any algorithm, design principle, concepts, . . .)

- What is the definition of...?
How to study for the Exam?

**Exercise sheets:**
- Problem solving
- Exam questions similar (slightly easier, fewer technicalities)

**Concepts:**
- Know what to do (O-notation, recurrences, etc.)
- Algorithm versus concept

**Mock Exam:**
- Will be put online later this week
- Attention: Tests only a subset of what is relevant
**Things I was very pleased with:**

- Attendance
- Office hours, drop-ins
- In-class test

**Discussion:**

- More/longer exercise classes
- More exercises of different difficulties
- Practical relevance, toy problems