CH08-320201
Algorithms and Data Structures

Prof. Dr.-Ing. Lars Linsen

Jacobs University
Spring 2016
0. Introduction
0.1 Syllabus and Organization
Course website

http://www.faculty.jacobs-university.de/llinsen/teaching/320201.htm
(accessible through CampusNet)
Content

• This course introduces a basic set of data structures and algorithms that form the basis of almost all computer programs. The data structures and algorithms are analyzed in respect to their computational complexity with techniques such as worst case and amortized analysis.

• Topics: Fundamental data structures (lists, stacks, trees, hash tables), fundamental algorithms (sorting, searching, graph traversal).
Prerequisites
Lectures

• **Times:**
  - Tuesday 11:15am–12:30am,
  - Thursday 9:45am–11:00am.

• **Location:** Research III Lecture Hall
Instructor

- Lars Linsen
- Office: Res I, 128.
- Phone: 3196
- E-Mail: l.linsen [@jacobs-university.de]
- Office hours: by appointment
Tutorials

- Times: tbd
- Location: tbd
Assignments

• The homework assignments include theoretical and practical problems that tackle topics from the lectures.
• The homework assignments are handed out on a regular basis.
• Solutions that are handed in late lead to reduced credit (-15% per day).
• Exceptions are only made with an official excuse.
• With an official excuse of up to 4 days, the deadline for the respective homework is extended by the same amount of days.
• With an official excuse of more than 4 days, the respective homework will not count.
• Handing in via JGrader.
Exams

- There will be one midterm and a final exam.
- There are no quizzes planned.
Grading

- **Homework** assignments contribute 35% to the overall grade.
- The **midterm** exam contributes 25%.
- The **final** exam contributes 40%.
## Dates - Lectures (1)

<table>
<thead>
<tr>
<th>Week 1</th>
<th>02.02.</th>
<th>Lecture 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>04.02.</td>
<td>Lecture 2</td>
</tr>
<tr>
<td>Week 2</td>
<td>09.02.</td>
<td>Lecture 3</td>
</tr>
<tr>
<td></td>
<td>11.02.</td>
<td>Lecture 4</td>
</tr>
<tr>
<td>Week 3</td>
<td>16.02.</td>
<td>Lecture 5</td>
</tr>
<tr>
<td></td>
<td>18.02.</td>
<td>Lecture 6</td>
</tr>
<tr>
<td>Week 4</td>
<td>23.02.</td>
<td>Lecture 7</td>
</tr>
<tr>
<td></td>
<td>25.02.</td>
<td>Lecture 8</td>
</tr>
<tr>
<td>Week 5</td>
<td>01.03.</td>
<td>No lecture - off campus</td>
</tr>
<tr>
<td></td>
<td>03.03.</td>
<td>Lecture 9</td>
</tr>
<tr>
<td>Week 6</td>
<td>08.03.</td>
<td>Lecture 10</td>
</tr>
<tr>
<td></td>
<td>10.03.</td>
<td>Lecture 11</td>
</tr>
<tr>
<td>Week 7</td>
<td>15.03.</td>
<td>Lecture 12</td>
</tr>
<tr>
<td></td>
<td>17.03.</td>
<td>Lecture 13</td>
</tr>
</tbody>
</table>

Jacobs University

Visualization and Computer Graphics Lab
## Dates - Lectures (2)

<table>
<thead>
<tr>
<th>Week 8</th>
<th>29.03.</th>
<th>Midterm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>31.03.</td>
<td>Lecture 14</td>
</tr>
<tr>
<td>Week 9</td>
<td>05.04.</td>
<td>Lecture 15</td>
</tr>
<tr>
<td></td>
<td>07.04.</td>
<td>Lecture 16</td>
</tr>
<tr>
<td>Week 10</td>
<td>12.04.</td>
<td>Lecture 17</td>
</tr>
<tr>
<td></td>
<td>14.04.</td>
<td>Lecture 18</td>
</tr>
<tr>
<td>Week 11</td>
<td>19.04.</td>
<td>Lecture 19</td>
</tr>
<tr>
<td></td>
<td>21.04.</td>
<td>Lecture 20</td>
</tr>
<tr>
<td>Week 12</td>
<td>26.04.</td>
<td>Lecture 21</td>
</tr>
<tr>
<td></td>
<td>28.04.</td>
<td>Lecture 22</td>
</tr>
<tr>
<td>Week 13</td>
<td>03.05.</td>
<td>Lecture 23</td>
</tr>
<tr>
<td></td>
<td>05.05.</td>
<td>Holiday</td>
</tr>
<tr>
<td>Week 14</td>
<td>10.05.</td>
<td>Lecture 24</td>
</tr>
<tr>
<td></td>
<td>12.05.</td>
<td>Lecture 25</td>
</tr>
</tbody>
</table>
Dates - Exams

- **Midterm:** March 29, 2016
- **Final:** tbd (finals' week)
## Dates - Assignments

<table>
<thead>
<tr>
<th>Homework</th>
<th>Handed out</th>
<th>Due</th>
</tr>
</thead>
<tbody>
<tr>
<td>Homework 1</td>
<td>05.02.</td>
<td>12.02. at 7pm</td>
</tr>
<tr>
<td>Homework 2</td>
<td>12.02.</td>
<td>19.02. at 7pm</td>
</tr>
<tr>
<td>Homework 3</td>
<td>19.02.</td>
<td>26.02. at 7pm</td>
</tr>
<tr>
<td>Homework 4</td>
<td>26.02.</td>
<td>04.03. at 7pm</td>
</tr>
<tr>
<td>Homework 5</td>
<td>04.03.</td>
<td>11.03. at 7pm</td>
</tr>
<tr>
<td>Homework 6</td>
<td>11.03.</td>
<td>18.03. at 7pm</td>
</tr>
<tr>
<td>Homework 7</td>
<td>18.03.</td>
<td>01.04. at 7pm</td>
</tr>
<tr>
<td>Homework 8</td>
<td>01.04.</td>
<td>08.04. at 7pm</td>
</tr>
<tr>
<td>Homework 9</td>
<td>08.04.</td>
<td>15.04. at 7pm</td>
</tr>
<tr>
<td>Homework 10</td>
<td>15.04.</td>
<td>22.04. at 7pm</td>
</tr>
<tr>
<td>Homework 11</td>
<td>22.04.</td>
<td>29.04. at 7pm</td>
</tr>
<tr>
<td>Homework 12</td>
<td>29.04.</td>
<td>06.05. at 7pm</td>
</tr>
<tr>
<td>Homework 13</td>
<td>06.05.</td>
<td>13.05. at 7pm</td>
</tr>
</tbody>
</table>
Literature

0.2 Goals
Goals

• The objective of the course is to learn about
  - fundamental algorithms for solving problems efficiently,
  - basic algorithmic concepts,
  - the analysis of algorithms, and
  - fundamental data structures for efficiently storing, accessing, and modifying data.
0.3 Content
Content

1. Foundations
2. Sorting & Searching
3. Fundamental Data Structures
4. Design Concepts
5. Graph Algorithms
6. Computational Geometry
1. Foundations
1.1 Definitions
Definition: Algorithm

- An algorithm is a sequence of computational steps which transforms a set of values (input) to another set of values (desired output).
- It is a tool for solving a well-defined computational problem.
- Step-wise procedure that can be implemented in a computer program.
- Consists of a finite list of well-defined instructions (Turing machine).
- 'Algorithm' stems from 'Algoritmi', the Latin form of al-Khwārizmī, a Persian mathematician, astronomer and geographer.
Example: Sorting Problem

• Input:
  sequence $\langle a_1, a_2, \ldots, a_n \rangle$ of numbers.

• Output:
  permutation $\langle a'_1, a'_2, \ldots, a'_n \rangle$
  such that $a'_1 \leq a'_2 \leq \cdots \leq a'_n$

• Example (instance of sorting problem):
  Input: 8 2 4 9 3 6
  Output: 2 3 4 6 8 9
Example: Searching

Algorithm - Wikipedia, the free encyclopedia
In mathematics and computer science, an algorithm is a step-by-step procedure for calculations. Algorithms are used for calculation, data processing, and ... 
al-Khwārizmī - List of algorithms - Euclidean algorithm - Automated reasoning

List of algorithms - Wikipedia, the free encyclopedia
The following is a list of algorithms along with one-line descriptions for each.

Bilder zu algorithms

Weitere Bilder zu algorithms
Example: Road map

Graph algorithm
Example: DNA sequences

String matching
Analysis of algorithms

- The theoretical study of computer-program performance and resource usage.
- Other design goals?
  - correctness
  - functionality
  - robustness
  - reliability
  - user-friendliness
  - programmer time
  - simplicity
  - modularity
  - maintainability
  - extensibility
Performance of Algorithms

• Analysis helps us to understand scalability.
• Performance often draws the line between what is feasible and what is impossible.
• Algorithmic mathematics provides a language for talking about program behavior.
• “Performance is the currency of computing.”
• The lessons of program performance generalize to other computing resources.
Definition: Data Structure

• A data structure is a way to store and organize data in order to facilitate access and modification.

• There is typically no best data structure, but each data structure has its strengths and weaknesses.

• Which data structure to use, depends on the problem that is to be solved.

• Sometimes there is a trade-off between storage (in a data structure) and speed (in accessing a data structure or of an algorithm).
1.2 First example: Insertion Sort
Sorting problem

- First algorithm: Insertion Sort
**Insertion Sort**

**INSERTION-SORT** \((A, n)\)

\[
\textbf{for } j = 2 \textbf{ to } n \\
key = A[j] \\
// Insert } A[j] \text{ into the sorted sequence } A[1 \ldots j - 1]. \\
i = j - 1 \\
\textbf{while } i > 0 \textbf{ and } A[i] > key \\
i = i - 1 \\
A[i + 1] = key
\]
Example

- Sort $A = < 5, 2, 4, 6, 1, 3 >$

\[ \begin{array}{cccccc}
(a) & 5 & 2 & 4 & 6 & 1 & 3 \\
(b) & 2 & 5 & 4 & 6 & 1 & 3 \\
(c) & 2 & 4 & 5 & 6 & 1 & 3 \\
(d) & 2 & 4 & 5 & 6 & 1 & 3 \\
(e) & 1 & 2 & 4 & 5 & 6 & 3 \\
(f) & 1 & 2 & 3 & 4 & 5 & 6 \\
\end{array} \]
Correctness

**INSERTION-SORT**(*A, n*)

```
for j = 2 to n
    key = A[j]
    // Insert A[j] into the sorted sequence A[1 .. j - 1].
    i = j - 1
    while i > 0 and A[i] > key
        A[i + 1] = A[i]
        i = i - 1
    A[i + 1] = key
```

- **Loop invariant:**
  At the start of each iteration of the for loop, the subarray *A[1..j-1]* consists of elements originally in *A[1..j-1]*, but in sorted order.