Algorithmen und Datenstrukturen (IN5004)

Title	Algorithms and Data Structures	
Typ	Vorlesung mit Übungen	
Credits	6	
Lehrform/SWS	3V + 2Ü	
Sprache	Deutsch	
Modulniveau	Bachelor	
Arbeitsaufwand	Präsensstunden	75 Stunden
	Eigenstudium	105 Stunden
	Gesamtaufwand	180 Stunden
Angestrebte Lernergebnisse	Nach erfolgreichem Abschluss des Moduls kennen die Teilnehmer(innen): • Grundlegende Eigenschaften und Entwurfsmethoden für Algorithmen; • Effiziente Algorithmen und Datenstrukturen für grundlegende Probleme; • Wesentliche Komplexitätsklassen für das Laufzeitverhalten und den Speicherplatzbedarf von Algorithmen. Teilnehmer(innen) des Moduls können • das Laufzeitverhalten und den Speicherplatzbedarf für gegebene Algorithmen analysieren; • algorithmische Problemstellungen formal modellieren; • bekannte Datenstrukturen und Algorithmen an modifizierte Problemstellungen adaptieren. Unter Anwendung der erlangten Fähigkeiten und Kenntnisse sind die Teilnehmer in der Lage, • Programme unter Verwendung der erlernten algorithmischen Techniken eigenständig zu konzipieren und in einer Programmiersprache zu implementieren; • Für ein gegebenes algorithmisches Problem die Verwendung verschiedener Lösungsmöglichkeiten nach formalen Kriterien zu beurteilen.	
Intended Learning Outcomes	Participants are familiar with the following topics: • Basic characteristics and development methods for algorithms; • Efficient algorithms and data structures for basic problems; • Important complexity classes for the analysis of	

	runtime and memory complexity. Participants of the module are able to: • analyze the run time and memory requirements for a given algorithm; • formally model and describe algorithmic problem settings; • adapt the introduced data structures and algorithms to modified problem settings. Based on the learned knowledge and abilities the participants obtain the skill to: • develop and implement programs based on the introduced algorithmic techniques in a programming language; • evaluate different solution approaches for a given algorithmic problem based on formal analysis.
Inhalt	Dieses Modul gibt eine Einführung in die Entwicklung effizienter Algorithmen sowie das Zusammenspiel zwischen Algorithmus und Datenstruktur. Grundbegriffe zu Algorithmen und Laufzeitanalyse: • Abgrenzung verschiedener Laufzeitabschätzungen (best-case, worst-case, erwartete Laufzeitkomplexität) • Asymptotische Analyse von oberen und unteren Schranken der Laufzeitkomplexität • Groß O Notation (Definition und Berechnung) • Wichtige Komplexitätsklassen (konstant, logarithmisch, linear, quadratisch, und exponentiell) • Methoden der empirischen Performanz Evaluation • Trade-Off zwischen Zeit- und Speicherverbrauch von Algorithmen • Optionale Themen: klein o, groß Omega und Groß Theta Notation; Rekurrenz-Relationen; Analyse von iterativen und rekursiven Algorithmen; Hauptsatz der Laufzeitfunktionen. Grundlegende Datenstrukturen und Algorithmen: • Elementare Datentypen (Integer, Float, Strings etc.) • Verbunddatentypen, Objekte und Arrays • dynamische Datenstrukturen (einfach und doppelt verkettete Listen, Stapel, Warteschlangen, Bäume) • Implementierung von dynamischen Datenstrukturen • einfache numerische Algorithmen (z.B.

	Bestimmen des Durschnitts, Maximum, oder Minimum einer Liste oder eines Arrays, approximative Berechnung der Quadratwurzel, Bestimmung des größten gemeinsamen Teilers) • sequentielle und binäre Suche in Arrays. Datenstrukturen und Algorithmen zur Schlüsselsuche: • Zusammenhang Suchzeit, Einfügezeit, Löschzeiten und Speicherplatzbedarf • Balancierte Suchbäume (Grundprinzip und Analyse, Beispielalgorithmen z.B. AVL-Bäume, Rot-Schwarzbäume) • Such-Bäume für den Sekundärspeicher (Problemstellung, z.B. B-Bäume) • Grundprinzip des Hashing (Einfache Hashfunktionen, Kollisionsstrategien) • dynamische Hash-Verfahren(z.B. lineares Hashing) • Optionale Themen: weiterführende Algorithmen zur Schlüsselsuche im Hauptspeicher (z.B. optimale binäre Suchbäume, Splay trees, Treaps); weiterführende Indexstrukturen für den Sekundärspeicher (z.B. B*-Baum); weiterführende Hash-Verfahren für den Sekundärspeicher (z.B. lineares Hashing mit partiellen Erweiterungen). Sortierverfahren: • elementare Sortieralgorithmen (Sortieren durch Abzählen, Insertion-Sort, Selection-Sort, Bubble-Sort) • fortgeschrittene Sortierverfahren (Heapsort, Quicksort) • Sortieren auf dem Sekundärspeicher (Merge-Sort) • untere Schranke für vergleichsbasiertes Sortieren • Schlüsselbasiertes Sortieren (Bucketsort). • Optionale Themen: weiterführende Verfahren für das Sortieren großer Schlüsselmengen (bottom-up Heapsort, clever Quicksort); weiterführende Verfahren zur schlüsselbasierten Suche (Radix-Sort); Priotätswarteschlangen (z.B. Fibonacci-Heaps). Graph-Algorithmen: • Grundlegende Eigenschaften von Graphen • Darstellung von Graphen (Adjazenz-Matrizen und -listen) • Graph-Traversierungen (Breitensuche, Tiefensuche)

	• Bestimmung kürzester Pfade (Dijkstras und Floyds Algorithmus) • Minimale Spannbäume (Algorithmen von Prim und Kruskal) • Optionale Themen: Flüsse in Netzwerken (z.B. maximaler Fluss, Max-Flow-Min-Cut Theorem, Maximales bipartites Matching); weitere Graph-Probleme(z.B. Topologische Sortierung, Finden von stark verbundenen Komponenten, Graph Matching). Algorithmische Strategien: • Vollständige Suche • Greedy Algorithmen • Divide-and-conquer • Rekursives Backtracking • Branch-and-bound • Optionale Themen: Reduktion: Transform-and-Conquer. Optionale Themen: • Lineare Programmierung (Dualität, Simplex Algorithmen, Innere Punkt Verfahren) • Pattern Matching und String/Text Algorithmen (z.B. Substring Matching, Reguläre Ausdrücke, längste gemeinsame Teilsequenzen) • String-basierte Datenstrukturen und Algorithmen (z.B. Suffix Arrays, Suffix-Bäume, Tries) • Algorithmen zur Lösung numerischer Probleme (z.B. Primzahlentests, ganzzahlige Faktorisierung) • Geometrische Datenstrukturen und Algorithmen (z.B. Darstellung von Punkten, Liniensegmenten und Polygonen, Eigenschaften und Schnittpunkte, konvexe Hüllen, räumliche Zerlegung, Kollisionserkennung, geometrische Nähe).
Contents	The module gives an introduction to the development of efficient algorithm as well as the interaction between algorithms and data structures. Basic Principles of Algorithms and Runtime Analysis: • Different types of runtime approximations (best-case, worst-case, expected runtime) • Asymptotic analysis of upper and expected complexity bounds • Big O notation (Definition and computation) • Important complexity classes(constant, logarithmic, linear, quadratic, and exponential)

	• Methods for empirical performance evaluation • Time and space trade-off • Optional Topics: Little o, big omega and big theta notation; Recurrence relations; Analysis of iterative and recursive algorithms; some version of a Master Theorem. Basic Data Structures and Algorithms: • Elementary data types (integer, float, strings etc.) • Records, objects and arrays • Dynamic data structures (singly and doubly linked lists, stacks, queues, trees) • Implementations of dynamic data structures • Simple numerical algorithms (e.g. computing the average, maximum, or minimum of a list or an array, approximate computation of the square root, computing the greatest common divisor) • Sequential and binary search in arrays. Data Structures and Algorithms for Key-Searching: • Connection between search, insertion and delete times and memory requirements • Balanced search trees (principles and analysis, example structures e.g. AVL-trees, red-black trees) • Search trees for the secondary storages (basic setting, B-trees) • Principles of hashing (simple hashing functions, basic collision strategies) • Dynamic hashing methods (e.g. linear hashing) • Optional Topics: advanced algorithms for key search in main memory (e.g. optimal binary search trees, splay trees, treaps), advanced index structures for key search in the secondary storage (e.g. B*-trees), advanced hashing methods (e.g. linear hashing with partial extensions). Sorting Methods: • Basic sorting algorithms (counting sort, insertion sort, selection sort, bubble sort) • Advanced sorting algorithms (heapsort, quicksort) • Sorting algorithms for the secondary storage (merge sort) • Lower bounds for sorting based on key comparisons • Key based sorting (bucket sort) • Optional Topics: advanced methods for sorting

	large key sets (button-up heapsort, clever quicksort), advanced key-based sorting methods (radix-sort), priority queues (heaps, Fibonacci heaps). Graph Algorithms: • Basic characteristics of graphs • Graph representations (adjacency matrices and lists) • Graph traversals (breadth first, depth first) • Shortest path computation (Dijkstra's and Floyd's algorithms) • Minimal spanning trees(Prim's and Kruskal's algorithm) • Optional Topics: Network flows (e.g. maximal flow, max-flow–min-cut theorem, maximal bipartite matching), further graph problems(e.g. topological sorting, finding strongly connected components, graph matching). Algorithmic Strategies: • Exhaustive search • Greedy algorithms • Divide-and-conquer • Recursive backtracking • Branch-and-Bound • Optional Topics: Reduction: Transform-and-Conquer. Optional Chapters: • Linear programming (duality, simplex algorithms, interior point methods) • Pattern matching and string/text algorithms (e.g. substring matching, regular expressions, longest common subsequence) • String based data structures and algorithms (e.g. suffix arrays, suffix trees, tries) • Advanced numerical algorithms (e.g. primality tests, integer factorization) • Geometric data structures and algorithms (e.g., points, line segments, polygons, finding convex hull, spatial decomposition, collision detection, geometric search/proximity).
Prüfung	Prüfungsleistung (benotet): Klausur (90 min) Wiederholungsklausur zu Ende des Semesters oder im Folgesemester. Details werden zu Beginn des Moduls bekannt gegeben. In der Klausur weisen die Studierenden nach, dass sie

	die Methoden der Algorithmenanalyse (wie z.B. O-Notation und Master-Theorem) kennen und auf konkrete Problemstellungen anwenden können, einfache Algorithmen und Datenstrukturen für grundlegende Probleme (wie z.B. Listen, Stapel, Bäume, Suchbäume, Hashing, Sortieren, Graph-Algorithmen) wiedergeben und an modifizierte Problemstellungen anpassen können, und die Grundprinzipien des Algorithmenentwurfs (wie z.B. erschöpfende Suche, Divide-and-Conquer, Backtracking) auf einfache Problemstellungen anwenden können. Konkret werden in der Klausur Aufgaben bearbeitet, die eine eigenständige Anwendung der Analysetechniken und Entwurfsmethoden für Algorithmen auf eine algorithmische Problemstellung erfordern.
Examination	Examination requirements: written exam (90-120 min) A makeup exam will be offered at the end of the semester or in the following semester. Details will be announced at the beginning of the module. Within the written exam, students demonstrate that they remember methods for algorithm analyses (e.g., Big-Oh, Master Theorem) and are able to apply them to specific problems, that they are able to reproduce simple algorithms and data structures for fundamental problems (such as lists, stacks, trees, search trees, hashing, sorting, graph algorithms) as well as to adapt them to similar problems, that they are able to apply fundamental techniques for algorithm design (such as exhaustive search, divide and conquer, backtracking) to simple problems. The written exam consists of assignments to solve demanding problems, which require the application of techniques for the analysis and methods for the design of algorithms.
Literatur/Literature	Robert Sedgewick: Algorithmen in Java: Grundlagen, Datenstrukturen, Sortieren, Suchen. Teil 1-4 (Pearson Studium) Thomas Ottmann, Peter Widmayer: Algorithmen und Datenstrukturen (Spektrum Lehrbuch) Thomas H. Cormen et al.: Algorithmen - Eine Einführung (Oldenbourg)
Medienformen	Folienpräsentation, Tafelanschrieb
Media	slide show, blackboard
Lehr- und Lernmethode	Vorlesung, Tutorübung, Aufgaben zum Selbststudium. Das Modul besteht aus einer Vorlesung sowie Übungen

	in kleinen Gruppen. In den Hausaufgaben, die freiwillig abzugeben sind, wird das Verständnis der Konzepte, Analysewerkzeuge und Algorithmen, die in der Vorlesung vorgestellt werden, anhand konkreter Beispiele und Problemstellungen vertieft. In den Hausaufgaben werden selbständig anspruchsvolle Übungsaufgaben bearbeitet, die ähnlich wie die Klausuraufgaben sind und daher zur Vorbereitung darauf dienen. In den Übungen werden mögliche Lösungsansätze der Aufgaben zum Selbststudium diskutiert.
Teaching and Learning Methods	Lecture, tutorial, assignments for individual study. The module consists of a lecture and tutorials in small groups. Within the assignments (submission is optional), concepts, tools for algorithm analysis, and algorithms (presented in the lecture) will be applied to real examples and problems to deepen the understanding. The assignments consist of demanding problems similar to the assignments in the written exam and thus serve as a preparation for the exam. Within the tutorials possible approaches for solutions of the assignments will be discussed.
Turnus	Sommersemester
Modulverantwortlicher	Dr. Matthias Schubert
Dozenten	Professoren der Informatik (LMU)