General remarks

- The recommended bibliography of the courses usually contains four or five books but I include only the ones that I consulted during the course.
- "Chapters x to y" means that chapter y is included.
- Computer Science courses usually require one or more projects in which the student applies techniques and implements the algorithms studied in the course.
- All the Computer Science courses offer course notes.
- The outlines of Computer Science courses are much more succinct, since this is how they are presented in the official study plans.
- I had to manually translate the outlines, so in this sense they are not official.

Department of Computer Science – required courses

Probability and Statistics [Computer Science]

No projects.

Objectives

Introduce the basic concepts in probability theory and statistics

Contents

Descriptive statistics. Probability spaces. Discrete random variables. Continuous random variables. Generation of random number. Elemental statistics. Applications.

Bibliography

Subsumed by the bibliography of Probability and Statistics [Mathematics].

Numerical Methods

Two group projects.

Objectives

Give the basic tools for the treatment of numeric problems. Study the most important numerical methods and their computational implementations.

Contents

Rounding error. Numerical representations. Solving non linear systems. Interpolation and polynomial approximation. Least squares. Numerical differentiation and integration. Ordinary differential equations. Solving linear systems: direct methods and iterative methods. Eigenvalues and eigenvectors.

Bibliography

- Watkins. Fundamentals of matrix computations.
  Complete.
- Course notes.

Algorithms and Data Structures I

Three group projects.

Objectives

Present tools to solve problems about sequences and lists. Formally prove the correctness of programs. Make small projects to apply the learned tools.

Contents

Specification and implementation of programs. Correctness of programs. Types. Abstract types. Treatment of sequences and lists. Sequential files.

Bibliography

- Course notes.

Algorithms and Data Structures II

Three group projects.

Objectives

Introduce abstract recursive data types. Present specification tools. Present techniques of analysis and design of algorithms. Proof the correctness of the constructed programs. Implement the learned techniques in projects. Analyze time and space complexity of various algorithms.

Contents

Recursion. Examples of abstract types, lists, queues, dictionaries, trees, graphs,. . . . Methodologies of formal specification. Formal specifications and implementation of these types using various data structures. Basics on time and space complexity.

Bibliography

- Cormen, Leiserson, Rivest and Stein. Introduction to Algorithms. 3 ed.
  Chapters 1 to 5 of Part I. Parts II and III.
- Course notes.

Algorithms and Data Structures III

Three group projects.

Objectives

Introduce the notion of graph. Implement using graphs various types of abstraction. Present the notion of exponential time algorithms, approximate solutions and various heuristics. Apply the learned concepts in various projects.

Contents

Graphs. Basic topics in graph theory. Solving problems using graphs. Basic topics in complexity theory. Recurrence. Heuristic solutions.

Bibliography

- Cormen, Leiserson, Rivest and Stein. Introduction to Algorithms. 3 ed.
  Parts IV and VI.
- Course notes.

Paradigms of Programming Languages

Three group projects.

Objectives

Evaluate the concepts of programming language in terms of its contribution to the developing of software.

Contents

The imperative paradigm. The functional paradigm. The logic paradigm. The equational paradigm. Elements for the evaluation of languages.

Bibliography

- Pierce. Types and Programming Languages. Part II. And chapter 15 of Part III.
- Lloyd. Foundations of Logic Programming. Chapters 1 and 2.
- Course notes.

Computer Architecture I

Two group projects.

Objectives

Understand the components in a computational system, their interaction and operation, abstracted by Von Neumann.

Contents

Representation of information. Components in a classical computational system. Languages. Microprogramming.

Bibliography

- Stallings. Computer Organization and Architecture.
  Part Two.
- Course notes.

Computer Architecture II

One individual project and two group projects.

Objectives

Develop projects to experiment with various computational systems.

Contents

Programming using microinstructions. Program interactions between various components of the system. Study a particular implementation of assembly language. Program a microkernell using INTEL assembly and C.

Bibliography

- Course notes.
- Tom Shamley. The Unabridged Pentium 4. IA32 Processor Genealogy, MindShare, INC. Addison-Wesley.
  Used as a reference book.

Operating Systems

Three group projects.

Objectives

Introduce the main functionalities of an operating system. Present the interrelation between operating system and the computer architecture.

Contents

Operating Systems. Components and functions. Administration of the various components of the system. Concurrent and distributed processes.

Bibliography

- Tanenbaum. Modern Operating Systems.
  Chapters 1 to 6 and 9.
- Course notes.

Communication Theory

Three group projects.

Objectives

Give the knowledge needed to understand the principles of transmission and manipulation of information. Study the design of information networks.

Contents

Information theory. Coding of information. Transmission mediums. Bandwidth and capacity. Fourier analysis. Modems. Multiplexers. Queue Theory. Protocols.

Bibliography

- Peterson. Computer Networks: A System Approach.
  Complete.
- Course notes.

Language Theory

One group project.

Objectives

Present the notion of formal languages, syntax and semantics of languages, with compilers in mind.

Contents

Formal languages. Syntax and semantic of programming languages. Construction of some simple compilers.

Bibliography

- Hopcroft, Ullman. Introduction to Automata Theory, Languages, and Computation.
  Chapters 1 to 7.
- Aho, Sethi, Ullman. Compilers: Principles, Techniques, and Tools.
  Used as a reference book for one of the projects.
- Course notes.

Software Engineering I

Two group projects.

Objectives

Present and practice traditional techniques of Software Engineering and design, to be able to cope with complex systems.

Contents

Planning of software projects. Requirements analysis. Specification. Design. Studying software quality: correctness, trustworthiness. Strategies of software verification. Maintenance. Metrics. Cost computation. CASE tools.

Bibliography

- Axel van Lamsweerde. Requirements Engineering: From System Goals to UML Models to Software Specifications. Wiley 2009.
  Chapters 1, 7, 8. Chapters 10 to 17.
- Course notes.

Software Engineering II

Two group projects.

Objectives

Present and practice traditional techniques of software engineering and design, to be able to cope with complex systems.

Contents

Techniques of object oriented software engineering. Fast prototyping. Exercise the planning and management of software projects.

Bibliography

- Gamma, Helm, Johnson, Vlissides. Design Patterns - Elements of Reusable Objet-Oriented Software.
  Chapter 1. Chapters 3 to 6.
- Course notes.

Databases

Three group projects.

Objectives

Extend the concept of data structure to the requirements involved in the solving of complex problems.

Contents

Functionalities of Database systems. Models of data. Query languages. Design of Databases. Physical data structures. Query optimization. Concurrence and recovering. Implementations.

Bibliography

- Ullman. Principles of Database and Knowledge Base Systems.

- Course notes.

Department of Computer Science – elective courses

Modal Logic

No projects.

- Formulas. Induction principle. Recursive definitions. Subformulas. Substitution.
- Classical, regular and normal logics. Axiomatizable and recursively axiomatizable logics. Deductibility. Consistence. The K logic. Tautologic consequence.
- Frames. Models. The logic of a class of frames. Completeness. Correctness. Subframes and submodels generated by indexes. Strong and weak consequence.
- Properties of relations. Correspondence between modal formulas and properties of relations. Correspondence between modal formulas and first and second order formulas.
- Sets of consistent formulas and L-consistent formulas. Canonical models. Completeness of modal logics. Existence theorems of models and frames.
- The property of finite models and the property of finite frames. Decidability from the property of finite frames.

Bibliography

- Ramón Jansana. Una Introduccion a la Lógica Modal.
  Chapters 1 to 7.

Positional Games

No projects.

- Positional games, examples. Different sets of rules. Tic-Tac-Toe, generalizations. Hex.
- Strong games, pairing draw. Maker-Breaker games. Biased games, threshold bias. General tools.
- Standard games on graphs – Connectivity game, Degree game, Hamilton Cycle game. Games with non-spanning winning sets, Clique game, Planarity game, Colorability game.
- Avoider-Enforcer games, two variants of rules, threshold biases. Standard games on graphs.
- Winning fast, implications in strong games.
- Games on sparse graphs, variations. Games on random boards.
- Neighborhood conjecture and related problems.

Bibliography

- Handouts that were parts of the book: Hefetz, Krivelevich, Stojakovic, Szabo. Positional Games.
  Parts of chapters 1 to 4.

Interactive Theorem Proving: theory and practice

One group project.

- Brief presentation of Proof Theory and Coq.
- Propositional Calculus based on natural deduction. Rules, examples and basic theorems. Proving propositional theorems in Coq.
- Lambda Calculus. Curry-Howard correspondence. Inspection of proof terms.
- First Order Calculus based on natural deduction. Rules, examples and basic theorems. Proving first order theorems in Coq.
- Inductive Types and structural induction.
- Fixed points and its relation to structural induction.
- Lambda Calculus with fixed points. Non termination and inconsistency.
- Proof automation: proof search, Ltac, typed automation.

Bibliography

- Y. Bertot and P. Castéran: Interactive Theorem Proving and Program Development, Springer, 1998.
  Chapters 1 to 4 and 6 to 8.
- Benjamin C. Pierce. Software Foundations.
  In the lab we made exercises from this book. Chapters: Basics, Induction, Lists, MoreCoq.
- Course handouts.

Operations Research

One individual project.

Objectives

Lots problems in Combinatorial Optimization can be modeled as lineal programming problems. Many of these problems are difficult to solve exactly. The purpose of this course is learn to formulate models and study the methods for solving a big variety of optimization problems that can be modeled as integer and linear programming problems.

Contents

- Linear Programming.
- Convex sets and functions. Polyhedra and cones. Convex hull. Basic lemmas and theorems.
- Simplex Method. Convergence. Complexity. Dual problem. Dual theorems.
- Examples of Integer Programming problems. Good and bad formulations.
- Characterizations of the convex hull of an Integer Programming problem. Classical inequalities.
- Algorithms for the resolution of Integer Programming problems. Branch and Bound. Branch and Cut. Usage and experimentation with CPLEX. Small project in CPLEX.

Bibliography

- Chvatal. Linear Programming.
  Part I.
- Nemhauser. Integer and Combinatorial Optimization.
  Used as a reference book.