Case Study 1:

Computation Complexity

From the Introduction:

The need to be able to **measure the complexity of a problem, algorithm or structure**, and to obtain bounds and quantitive relations for complexity arises in more and more sciences: besides computer science, the traditional branches of mathematics, statistical physics, biology, medicine, social sciences and engineering are also confronted more and more frequently with this problem. In the approach taken by computer science, complexity is measured by the quantity of computational resources (time, storage, program, communication). These notes deal with the foundations of this theory.

Computation theory can basically be divided into three parts of different character. **First**, the exact notions of algorithm, time, storage capacity, etc. must be introduced. For this, different mathematical machine models must be defined, and the time and storage needs of the computations performed on these need to be clarified (this is generally measured as a function of the size of input). By limiting the available resources, the range of solvable problems gets narrower; this is how we arrive at different complexity classes. The most fundamental complexity classes provide important classification even for the problems arising in classical areas of mathematics; this classification reflects well the practical and theoretical difficulty of problems. The relation of different machine models to each other also belongs to this first part of computation theory.

Second, one must determine the resource need of the most important algorithms in various areas of mathematics, and give efficient algorithms to prove that certain important problems belong to certain complexity classes. In these notes, we do not strive for completeness in the investigation of concrete algorithms and problems; this is the task of the corresponding fields of mathematics (combinatorics, operations research, numerical analysis, number theory).

Third, one must find methods to prove "negative results", *i.e.* for the proof that some problems are actually unsolvable under certain resource restrictions. Often, these questions can be formulated by asking whether some introduced complexity classes are different or empty. This problem area includes the question whether a problem is algorithmically solvable at all; this question can today be considered classical, and there are many important results related to it. The majority of algorithmic problems occurring in practice is, however, such that algorithmic solvability itself is not in question, the question is only what resources must be used for the solution. Such investigations, addressed to lower bounds, are very difficult and are still in their infancy. In these notes, we can only give a taste of this sort of result.

It is, finally, worth remarking that if a problem turns out to have only "difficult" solutions, this is not necessarily a negative result. More and more areas (random number generation, communication protocols, secret communication, data protection) need problems and structures that are guaranteed to be complex. These are important areas for the application of complexity theory; from among them, we will deal with **cryptography**, the theory of secret communication.

CASE STUDY 2:

Complexity Theory: A Modern Approach

Computational complexity theory has developed rapidly in the past three decades. The list of surprising and fundamental results proved since 1990 alone could fill a book: these include new probabilistic definitions of classical complexity classes (**IP = PSPACE** and the **PCP** Theorems) and their implications for the field of approximation algorithms; Shor's algorithm to factor integers using a quantum computer; an understanding of why current approaches to the famous **P** versus **NP** will not be successful; a theory of derandomization and pseudorandomness based upon computational hardness; and beautiful constructions of pseudorandom objects such as extractors and expanders.

This book aims to describe such recent achievements of complexity theory in the context of the classical results. It is intended to be a text and as well as a reference for self-study. This means it must simultaneously cater to many audiences, and it is carefully designed with that goal. The book will explain the context in which a certain notion is useful, and why things are defined in a certain way. Examples and solved exercises accompany key definitions.

The book has three parts and an appendix. **Part I** covers basic complexity classes; it provides a broad introduction to the field and covers basically the same ground as Papadimitriou's text from the early 1990s -- but more quickly. **Part II** covers lowerbounds for concrete computational models; it concerns lowerbounds on resources required to solve algorithmic tasks on concrete models such as circuits, decision trees, etc. **Part III** covers advanced topics; this constitutes the latter half of the book and is largely devoted to developments since the late 1980s. It includes average case complexity, derandomization and pseudorandomness, the **PCP** theorem and hardness of approximation, proof complexity and quantum computing. Finally, the **Appendix** outlines mathematical ideas that may be useful for following certain chapters, especially in parts II and III.

Intended Audience:

This book assumes essentially no computational background (though a slight exposure to computing may help) and very little mathematical background apart from the ability to understand proofs and some elementary probability on finite sample spaces. A typical undergraduate course on "Discrete Math" taught in many math and CS departments should suffice (together with the Appendix).