YAŞAR UNIVERSITY
FACULTY OF ENGINEERING
DEPARTMENT OF SOFTWARE ENGINEERING

CENG 502 ALGORITHM ANALYSIS AND COMPLEXITY THEORY
Spring semester 2013

ECTS: 8 (lectures 3, labs/tutorials 0)

INSTRUCTOR: Assoc. Prof. Dr. Kostadin Kratchanov

OFFICE: U107

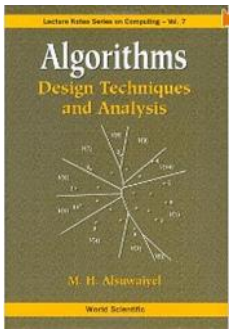
PHONE: 411-5289

EMAIL: kostadin.kratchanov@yasar.edu.tr

OFFICE HOURS: Tue, Thu 13:30 – 14:30

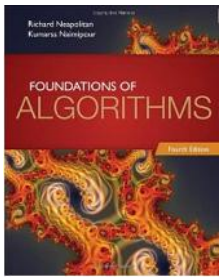
LECTURES: Tue 9:55 – 12:30 Room: network lab

TEXTBOOK:

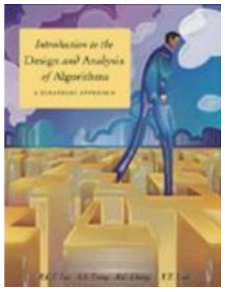


- Algorithms: Design Techniques and Analysis by M. H. Alsuwaiyel. World Scientific, 2010

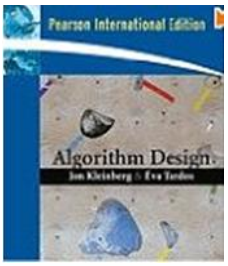
REFERENCE BOOKS:



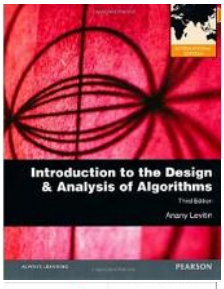
- Foundations of Algorithms, 4th ed. by R. Neapolitan and K. Naimipour. Jones and Bartlett, 2011



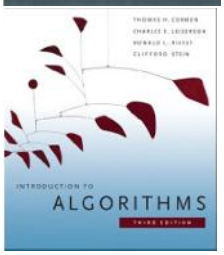
- Introduction to the Design and Analysis of Algorithms: A Strategic Approach by R. C. T. Lee, S.S. Tseng, R.C. Chang and Y. T. Tsai. McGraw-Hill, 2006



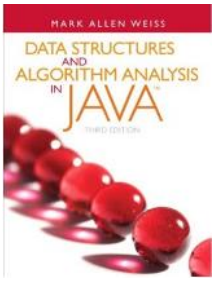
- Algorithm Design by J. Kleinberg and E. Tardos. Pearson (Addison-Wesley), 2005



- Introduction to the Design and Analysis of Algorithms, 3rd ed. by A.V. Levitin. Pearson, 2012

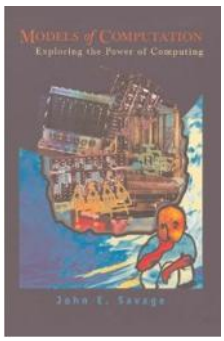


- Introduction to Algorithms, 3rd ed. by T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. The MIT Press, 2009

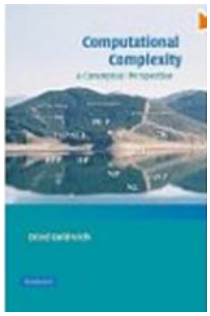


- Data Structures and Algorithm Analysis in Java, 3rd ed. by Mark Allen Weiss. Addison-Wesley, 2011

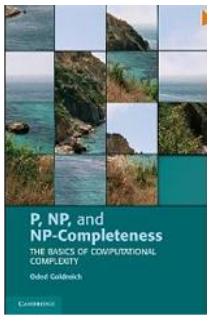
- Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties by G. Ausiello, P. Crescenzi, G. Gambosi and V. Kann. Springer, 2003



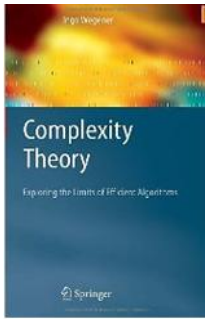
- Models of Computation. Exploring the Power of Computing. John Savage. Addison-Wesley, 1998
  New version (currently 2008) is freely available at http://www.cs.brown.edu/~jes/book/home.html



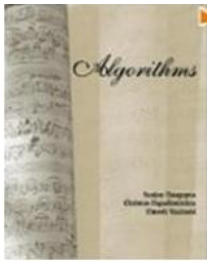
- Computational Complexity: A Conceptual Perspective by O. Goldreich. Cambridge University Press, 2008



- P, NP, and NP-Completeness: The Basics of Computational Complexity by O. Goldreich. Cambridge University Press, 2010

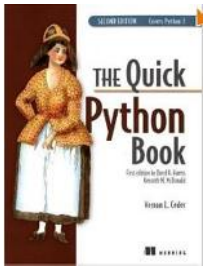


- Complexity Theory: Exploring the Limits of Efficient Algorithms by Ingo Wegener. Springer, 2005

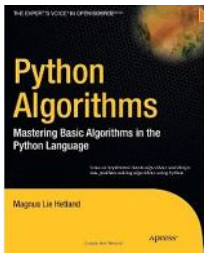


- Algorithms by S. Dasgupta, C.H. Papadimitriou and U. Vazirani. McGraw-Hill, 2006.
  A free electronic copy is available at:
  http://www.cs.berkeley.edu/~vazirani/algorithms/all.pdf

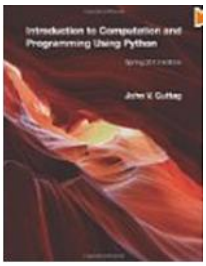
We will be using the Python programming language for representing (and running) the algorithms studied. The next book is an excellent (and really quick) introduction to Python for readers with background in other procedural programming languages. The books that follow introduce a number of important for us algorithms using Python.



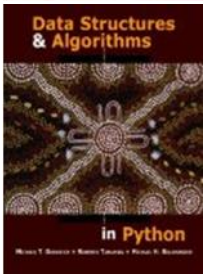
- The Quick Python Book, 2nd ed. by V. L. Ceder. Manning, 2010



- Python Algorithms: Mastering Basic Algorithms in the Python Language by M.L. Hetland. Apress, 2010



- Introduction to Computation and Programming Using Python by J.V. Guttag. MIT Press, 2013



- Data Structures and Algorithms in Python by M.T. Goodrich, R. Tamassia and M.H. Goldwasser. John Wiley, expected to be available on 28 Mar 2013

GENERAL OVERVIEW (COURSE OBJECTIVES)

Algorithms play the central role in both the science and practice of computing. **Analysis and Design of Algorithms** is one of the most fundamental courses in Computer Science and part of any Computing curriculum. Algorithm Analysis means determining the quantity of resources needed for an algorithm as a function of the size of the instances considered. This is of immense practical importance when considering the practical usability of a given algorithm.

In other words, the analysis of an algorithm concerns determining its efficiency. Efficiency of an algorithm and its complexity are reciprocal concepts. Therefore, analyzing an algorithm and studying its complexity are in principle synonymic. However, although with the same subject in principle, algorithm analysis and complexity theory have in practice different nuances and foci, with complexity theory having a much more theoretical flavor.
Complexity Theory is one of the two major branches of the Theory of Computation – Computability Theory and Complexity Theory. The Theory of Computation is the area of computer science and mathematics that deals with whether and how efficiently problems can be solved on a model of computation, using an algorithm. Historically the first, fundamental research question in computer science is: *"Are there well-defined problems that cannot be automatically (by a computer, regardless of the computational powers of contemporary computers or futuristic ones) solved?"* Efforts to answer this question led to the founding of Computer Science as an independent science. Today, this field of computer science is known as Computability Theory. After developing methods for classifying problems according to their algorithmic solvability, one asks the following question: *"How difficult are concrete algorithmic problems?"* Today, this branch of computer science is known as Complexity Theory.

This course has five major objectives.

The first objective is, building upon students' previous exposure in the undergraduate curriculum, to substantially strengthen students' understanding of the fundamental concepts of algorithm analysis and apprehension of the importance of algorithm efficiency. Students are equipped with knowledge and skills that allow them to effectively perform analysis of new algorithms. This part of the course is of higher level and more intensive and rigorous than a typical undergraduate course on Algorithm analysis.

This Algorithm analysis part then develops naturally into a study of the basics of the Complexity theory with its fundamental theoretical results and consequences. In particular, the practical importance of NP-complete (computationally hard) problems will be shown, and a number of such algorithms will be discussed.

The third major goal of the course is to build an understanding of the approaches used to practically address NP-complete problems. A number of strategies will be introduced such as approximate algorithms, randomized algorithms, heuristic algorithms, etc.

This will be preceded by an overview of Algorithm Design techniques and a concise summary of more traditional design approaches such as divide-and-conquer, decrease-and-conquer, greedy algorithms, branch-and-bound, etc. Thus, the fourth major goal of the course is a review of design methods for developing precise algorithmic solutions.

In the process of studying algorithm analysis, design and complexity issues a number of known algorithms will be revised, classified and analyzed. Also, many new algorithms of interest will be considered. This is a fifth, substantial "by-product" of the course.

Note that this course is <u>not</u> a course on Data structures and algorithms to process this data structures.

Finally, students will get exposed to the new for them Python programming language which has been gaining wide popularity recently.

LEARNING OUTCOMES

After completing the course the students will be able to:

- Demonstrate an understanding of the notion of algorithm efficiency and appreciate its importance.
- Compare the order of growth of functions, algorithms, and problems.
- Analyze time efficiency of algorithms and programs in the worst, best, and average cases, as well as their space efficiency.
- Solve recurrence relations.
- Discuss various problem solving techniques and some of their applications
- Utilize these approaches for solving new problems.
- Recognize the concepts of undecidable and hard problems, NP-complete problems, and similar theoretical notions.
- Understand the reasons for using approximate and randomized algorithms and display knowledge of such algorithms of practical importance.
- Have a general understanding of heuristics and metaheuristics and be aware of certain important examples and techniques.
- Understand major issues and approaches in sorting and search algorithms and their analysis, as well as certain graph and other algorithms.
- Understand algorithms specified in the Python programming language, and write Python programs using its basic features.

COURSE FORMAT

There are three hours of scheduled instruction per week. Some of the new material will be formally presented in lectures. However, in many cases, students will be required to prepare a specified topic for the next lecture; during that lecture there will be a discussion and additions to the topic, and numerous practical problems will be solved.

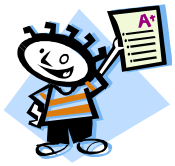
Short quizzes will be given on the lecture topics. Some of the scheduled lecture time may be used for tutorials whose goal is to reinforce the lectures by providing an opportunity to study examples and practice the application of concepts as well as to gain hands-on experience in solving problems. Students may undertake to prepare and give short presentations on topics that are not covered in the lectures.

An important part of the course is the set of 2 or 3 course assignments solved by the students at home. These assignments may include theoretical problems, applying studied approaches for solving given problems by hand, or writing and experimenting with programs.

To be successful, you will need to devote significant time outside of class to studying, practicing skills, and solving assignment problems..

ASSESSMENT

Attendance, participation and quizzes	15 %
Assignments	25 %
Midterm	20 %
Final Examination	40 %



EXAMINATIONS

The exact date and time of the midterm and final exams will be determined later in the semester.

The examinations will focus on understanding and applying the concepts taught in class. The general types of exam questions will be announced in advance.

ASSIGNMENTS

Problem solving and analysis techniques and skills can only be acquired through practice and through the study of increasingly difficult problems.

All assignments are to be completed individually. See the section below which discusses academic honesty.

Assignments can be very time consuming. Start early and budget sufficient time for completion.

Students will typically have around two weeks to complete each assignment, depending on its complexity. Submission deadlines will be enforced strictly. An early/late submission policy will be applied.

Students are responsible for maintaining backups of their work, and for maintaining an unaltered copy of each submission until it has been graded and returned.

ATTENDANCE, PARTICIPATION AND QUIZZES

The mark for Attendance, Participation, and Quizzes is made up of the specified three components.

Lecture attendance is compulsory. Student participation in class and completed

homeworks will be used to determine the second component of the mark. Students are strongly encouraged, individually or teaming up with another student, to prepare and give a short presentation on a topic not covered in the lectures. Outstanding presentations may earn bonus points.

Quizzes will be normally announced in advance. Quizzes will usually be closed–book (some might be open-book) and will typically require about 10-20 minutes to complete. A student who misses a quiz for any reason will be assigned a score of zero for that quiz.

ELECTRONIC DISTRIBUTION OF FILES AND INFORMATION TO STUDENTS

Numerous files will be made available to you electronically during the semester: documents, lectures, examples, homeworks, etc. Conversely, you might be asked to submit completed assignments electronically.

Our main tool for communication will be the course web site available from http://lectures.yasar.edu.tr. Students can access the course web site using the following "enrollment key": **CENG5022013EK.**

Note that urgent announcements will also be posted at this site.

Students are responsible to regularly check the announcement, documents and other uploaded materials. Failing to do so cannot be accepted as an excuse.

STUDENT FEEDBACK

Feedback from students is extremely important to me.

Please speak up and share with me any concerns or questions that you have. Feel free to raise an issue or ask a question in class, after class, during my office hours, or at any time you meet me.

You are also very welcome to use the email or telephone for communication.

ACADEMIC MISCONDUCT

Cheating and Plagiarism will not be tolerated at any stage during your studies at Yaşar University. These are a serious violation of academic ethical standards and are unfair to other students. Moreover, plagiarism and cheating contradict the main purpose of course work, which is to assist students in understanding and learning the course material.

It is expected that all work handed in by a student will be original work that has been done by the individual. If it is not, then this act of intellectual dishonesty will be dealt with severely.

Normally, the sanction for any student accused of cheating will be zero on the assignment. In the case of one student giving part of his/her assignment to another student, both students are considered to be cheating. Sanctions for further incidents of cheating by the same student may ultimately result in expulsion from the University.

While students are expected to work reasonably independently, we do not expect you to work in isolation. Often you learn best when working with others on an assignment. So what degree of collaboration is expected and, indeed, encouraged, and what is deemed to be cheating?

In general, we encourage things like bouncing ideas off one another, discussing which of

two alternate solutions might be better (and why), and getting another's ideas on how to resolve a difficulty that you have already spent time on. However, you should not be working so closely together that someone else's solution becomes incorporated into your product. These general guidelines apply to any type of assignment.

TOPIC OUTLINE

The **tentative** course topics are listed below. Not all topics will be covered in the same degree of detail and the sequence may differ somewhat from the list. The main purpose of the list below is to show the logical structure of the material rather than any temporal relationships. The inclusion of certain topics depends on the time available.

A. Introduction

- Mathematical review.
- Problems, algorithms, instances and programs.

B. Algorithm Analysis

- Analysis framework.
- Empirical and theoretical analysis.
- Asymptotic notations and basic rules. The limit rule.
- Techniques and examples of algorithm analysis.
- Recursively defined sequences and solving recurrence relations.
- Review of sorting algorithms and their analysis.
- More examples of algorithm analysis – the traveling salesperson problem, knapsack problem, assignment problem.

C. Traditional Algorithm Design Strategies

- Introduction to algorithm design.
- Brute force.
- Divide-and Conquer.
- Decrease-and-Conquer.
- Revision of exponentiation – brute force, divide-and-conquer, decrease-by-constant, decrease-by-a-constant-factor.
- Greedy algorithms.
- Dynamic programming.

D. Complexity of Algorithms

- Introduction to the theory of algorithm complexity.
- Introduction to NP-completeness.
- Important NP-complete problems.
- Other complexity classes. Space and time hierarchy theorems.

E. Coping with Hardness

- Backtracking.
- Branch-and-Bound.
- Approximation algorithms.
- Heuristic algorithms.
- Local search.

- Randomized algorithms.

A detailed current List of Topics will be maintained on the course online site.

THE PYTHON PROGRAMMING LANGUAGE

Python is a modern, high-level language, with many features that make it not only an excellent programming language for teaching but also one used in an increasing number of practical applications by hundreds of thousands of programmers. It is also gaining an increasing popularity as the first programming language to be taught at universities worldwide. Among the major features that Python exhibits, are:

- It is easy to use. Its syntax and semantics are very simple and consistent. Beginners can absorb enough Python syntax to write useful code quickly. It isn't unusual for coding an application in Python to take one-fifth of the time it would if coded in C or Java, and to take as little as one-fifth of the number of lines of the equivalent C program.
- It operates at a much higher level of abstraction and is very expressive and easy to read and understand. Basically, Python is more or less executable pseudocode – another feature that is extremely useful for our course!
- Python is free. It is very easy to install and use. A standard installation automatically includes all the libraries typically used.
- It is used interactively. The user simply communicates with the computer without the boring and time-consuming overhead of compiling, linking, etc. We simply write the algorithm and play with it – there are no declarations, classes, constructors, heavy IDE's and all other components that take unnecessary almost all of our time and effort if we write programs in Java or C#.
- You can write object-oriented programming but you don't have to – we will simply code the algorithms we study and will not need all the unnecessary and unimportant for our purposes overheads related to OO programming and compilation.
- Python is truly cross-platform. It runs on Windows, Mac, Linux, and so on. There are even versions of Python compatible with .NET and the Java virtual machine.

Python will be used in two ways in our course:

- Many of the algorithms that we study will be given in Python (in addition to some others that are presented in pseudocode or Java). That means that you not only have a clear and easy to understand specification of the algorithm (as in pseudocode) but you can also directly run and explore that algorithm. Moreover, we can also use freely available resources that trace and visualize the execution of the Python program which provides invaluable help in teaching and learning algorithms.
- There are tasks in your homeworks and assignments that require programming and experimenting with the programs you write. You will be required to write such programs in Python. First, this should be much easier for you. Secondly, I won't have to mark your programming skills (in Java or C# for example) that you obtained in other courses.

To install Python to your computer, browse to http://python.org/download/ and download the installer for the latest Windows version (if your computer uses Windows). Currently, this is Python 3.3.0 Windows X86-64 MSI Installer (64-bit) or Python 3.3.0 Windows x86

MSI Installer (32-bit). We will use Python 3, not Python 2. Note that the two versions are not compatible. You have two built-in options for obtaining interactive access to the Python interpreter: the basic command-line mode and the more advanced IDLE which is a simple integrated development environment for Python. Install and use IDLE.

A great book on Python for persons like you, who have programmed in another programming language already, is *The Quick Python Book* which I have included in the list of references above. Of course, there are numerous other useful books and other resources on Python. I strongly encourage you to obtain and use the *Quick Book*. Basically, chapters 1 to 5, 7, and 8 is what you definitely need. If necessary, you may find deeper explanations in the other chapters.

Another useful resource is the Python official website http://python.org/, and in particular the Python documentation http://docs.python.org/py3k/.

For tracing and visualizing Python programs we will use the *Online Python Tutor* at http://www.pythontutor.com/visualize.html (see also http://www.pythontutor.com/).