Algorithms and Data Structures

Introduction to AADS

Stefano Quer
Department of Control and Computer Engineering
Politecnico di Torino

General Information

- Algorithms and Programming
  - From 2011-2012 to 2020-2021
  - Disciplinary Scientific Sectors
    - ING-INF/05, Information Processing Systems
  - Bachelor-level degree
    - Computer Engineering ($02OGD_{LM}$)
      - 2° year, **12** credits, **120** hours
    - Electronic and Computer Engineeering ($01OGD_{LP}$)
      - 3° year, **10** credits, **100** hours until 2019-2020
      - 3° year, **12** credits, **120** hours in 2020-2021

Number of students enrolled
A.Y. 2018-2019 – 236
A.Y. 2019-2020 – 257
A.Y. 2020-2021 – 290

General Information

- Algorithms and Data Structures (01URNLM)
  - From 2021-2022
  - Disciplinary Scientific Sectors
    - ING-INF/05, Information Processing Systems
  - Bachelor-level degree
    - Computer Engineering
      - 2° year, 8 credits, **80** hours
      - All students (from A to Z)

Number of students enrolled
A.Y. 2021-2022 – 175

Instructors

Politecnico di Torino
1859
Department of Control and Computer Engineering

DAUIN

Consiglio Nazionale delle Ricerche

Quer Stefano
011-090-7076
stefano.quer@polito.it

Cheminod Manuel
manuel.cheminod@ieiit.cnr.it

Learning Outcomes

- Acquire **adequate**
  - Knowledge in algorithms, data structures, and their implementation in C
  - Skills to solve complex problems
- Student should gradually evolve from analytic to more design-oriented skills
- The course introduce
  - Theoretical foundations and solutions to "classical" problems
  - Advanced aspects in C language and problem-solving

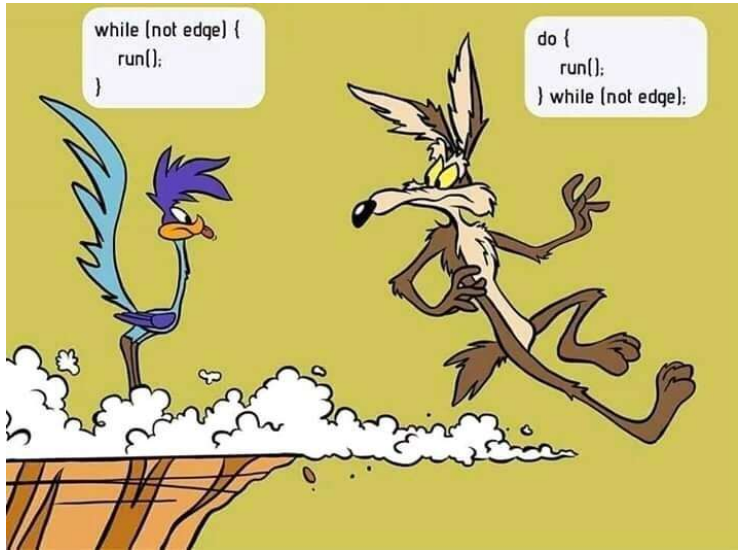
Learning Content

- Main topics
  - Dynamic memory allocation and use of pointers
  - Complexity analysis
  - Recursion
    - Recursive programming, recursive sorting, combinatorics
  - Dynamic programming
  - Greedy problem-solving paradigms
  - Advance C and modular programming
  - Complex data structures and Abstract Data Types
    - Linked lists, queues, stacks, trees, binary search trees, hash tables, heaps, graphs

Plus ... problem solving on all topics

Prerequisites

- Incremental nature of the course with respect to the first year class "**Programming Techniques**"
- **Strict** prerequisites in terms of programming skills and programming language knowledge
  - Elementary computer systems architecture
  - Numeric systems, numbers and types
  - Syntax of C, basic data types and basic constructs
  - Basic programming skills in C for elementary problem solving



Curricula impact

- Course impact for your curricula?

Google

Preparing for Google Technical Internship Interviews

This guide is intended to help you prepare for Software Engineering internship and Engineering Practicum internship interviews at Google. If you have any additional questions, please don't hesitate to get in touch with your recruiter.

Recruitment Process: Engineering Practicum Internships

Recruitment Process: Software Engineering Internships

Interview Tips

Technical Preparation

Extra Prep Resources

Curricula impact

Google | Preparing for your Interview

Technical Preparation

Coding: Google Engineers primarily code in C++, Java, or Python. We ask that you use one of these languages during your interview. For phone interviews, you will be asked to write code real time in Google Docs. You may be asked to:

- Construct / traverse data structures
- Implement system routines
- Distill large data sets to single values
- Transform one data set to another

Algorithms: You will be expected to know the complexity of an algorithm and how you can improve/change it. You can find examples that will help you prepare on TopCoder. Some examples of algorithmic challenges you may be asked about include:

- Big-O analysis: understanding this is particularly important
- Sorting and hashing
- Handling obscenely large amounts of data

Sorting: We recommend that you know the details of at least one n*log(n) sorting algorithm, preferably two (say, quicksort and merge sort). Merge sort can be highly useful in situations where quicksort is impractical, so take a look at it. What common sorting functions are there? On what kind of input data are they efficient, when are they not? What does efficiency mean in these cases in terms of runtime and space used? E.g. in exceptional cases insertion-sort or radix-sort are much better than the generic QuickSort / MergeSort / HeapSort answers.

Reminder:
Engineering Practicum interviews will focus on coding, algorithms and data structures and content will be level appropriate. See slide 2.

Curricula impact

Google | Preparing for your Interview

Technical Preparation

Data structures: Study up on as many other structures and algorithms as possible. We recommend you know about the most famous classes of NP-complete problems, such as traveling salesman and the knapsack problem. Be able to recognize them when an interviewer asks you in disguise. Find out what NP-complete means. You will also need to know about Trees, basic tree construction, traversal and manipulation algorithms, hash tables, stacks, arrays, linked lists, priority queues.

Hashtables and Maps: Hashtables are arguably the single most important data structure known to mankind. You should be able to implement one using only arrays in your favorite language, in about the space of one interview. You'll want to know the O() characteristics of the standard library implementation for Hashtables and Maps in the language you choose to write in.

Trees: We recommend you know about basic tree construction, traversal and manipulation algorithms. You should be familiar with binary trees, n-ary trees, and trie-trees at the very least. You should be familiar with at least one flavor of balanced binary tree, whether it's a red/black tree, a splay tree or an AVL tree. You'll want to know how it's implemented. You should know about tree traversal algorithms: BFS and DFS, and know the difference between inorder, postorder and preorder.

Min/Max Heaps: Heaps are incredibly useful. Understand their application and O() characteristics. We probably won't ask you to implement one during an interview, but you should know when it makes sense to use one.

Reminder:
Engineering Practicum interviews will focus on coding, algorithms and data structures and content will be level appropriate. See slide 2.

Curricula impact

Google | Preparing for your Interview

Technical Preparation

Graphs: To consider a problem as a graph is often a very good abstraction to apply, since well known graph algorithms for distance, search, connectivity, cycle-detection etc. will then yield a solution to the original problem. There are 3 basic ways to represent a graph in memory (objects and pointers, matrix, and adjacency list); familiarize yourself with each representation and its pros/cons. You should know the basic graph traversal algorithms, breadth-first search and depth-first search. Know their computational complexity, their tradeoffs and how to implement them in real code.

Recursion: Many coding problems involve thinking recursively and potentially coding a recursive solution. Prepare for recursion, which can sometimes be tricky if not approached properly. Practice some problems that can be solved iteratively, but a more elegant solution is recursion.

Operating systems: You should understand processes, threads, concurrency issues, locks, mutexes, semaphores, monitors and how they all work. Understand deadlock, livelock and how to avoid them. Know what resources a process needs and a thread needs. Understand how context switching works, how it's initiated by the operating system and underlying hardware. Know a little about scheduling. The world is rapidly moving towards multi-core, so know the fundamentals of "modern" concurrency constructs.

Mathematics: Some interviewers ask basic discrete math questions. This is more prevalent at Google than at other companies because counting problems, probability problems and other Discrete Math 101 situations surrounds us. Spend some time before the interview refreshing your memory on (or teaching yourself) the essentials of elementary probability theory and combinatorics. You should be familiar with n-choose-k problems and their ilk - the more the better.

Still want more info?

Tech interviews @ Google

Distributed systems & parallel programming

Scalable Web Architecture & Distributed systems

How search works

Delivery Mode

- There is no distinction between theory and practice lessons
  - Lectures include practice lessons
  - 3 blocks of 1.5 hours
    - Monday, 13.00-14.30, classroom 1
    - Recovery slot
      - Tuesday, 16.00-17.30, classroom 4P
    - Thursday, 11.30-13.00, classroom 8
    - Thursday, 13.00-14.30, classroom 8

Done during weeks 1, 2, and 3

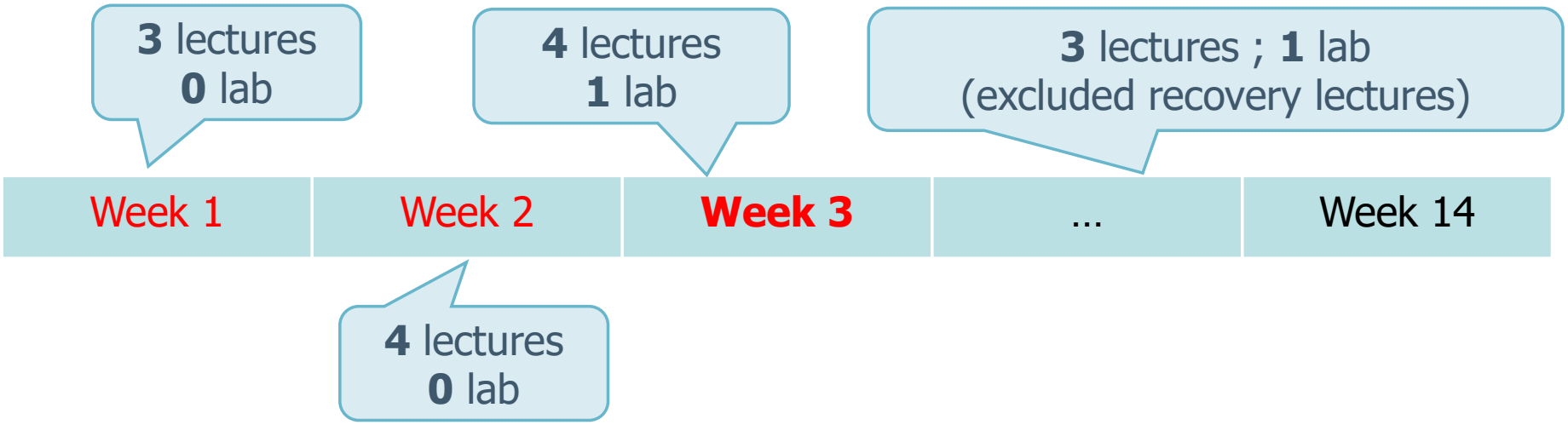
Delivery Mode

- Lectures and practice are extended with 15 additional hours in laboratory
  - 2 lab teams
    - 2 blocks of 1.5 hours (1 for each lab team)
    - Team A: A – LA, Wednesday 08.30-10.00, laib 3
    - Team B: LB – Z, Wednesday 10.00-11.30, laib 3
  - Problem solving in C language
    - From specs to code through editing, compilation, debugging, and execution of programs
    - Windows operating system
    - Codebock (or similar) API (Application Programming Interface)

Not done during weeks 1 and 2

Delivery Mode

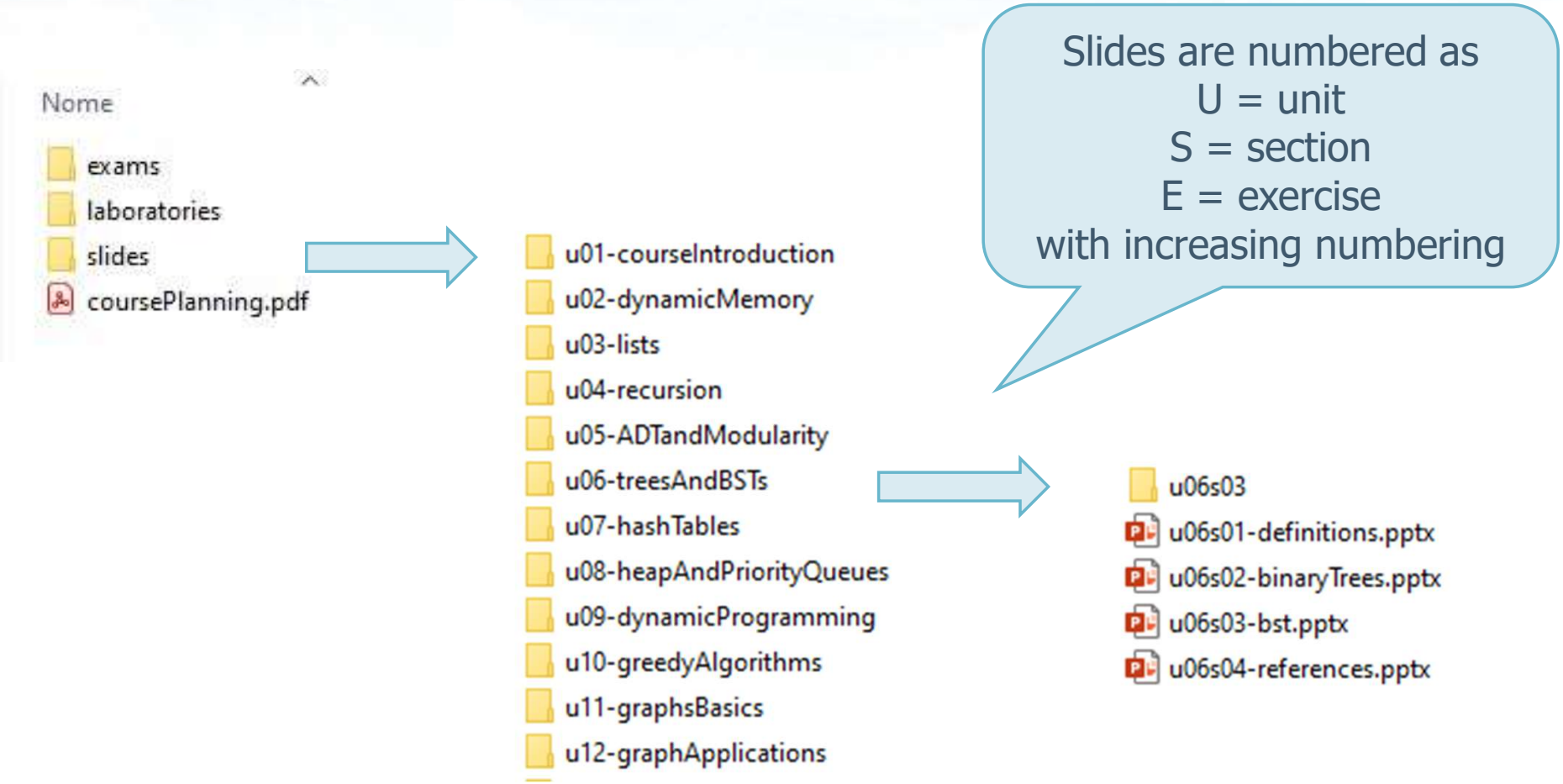
- To summarize
  - All lectures, practice lessons, and laboratories are done in-classroom
  - All lectures and practices will be video-recorded
  - The complete schedule of the course is avaiable on Dropbox



Texts, Readings, Handouts

- General information (teaching portal)
  - Calendar, rules and deadlines
  - Exams bookings and exam results
- Material used (teaching portal, dropbox)
  - Overheads
  - Laboratory
    - 1 specification every week
    - 1 solution every week (previous week)
  - Examination texts
    - All "compatible" versions are made available
    - Solution (almost all theory solved)

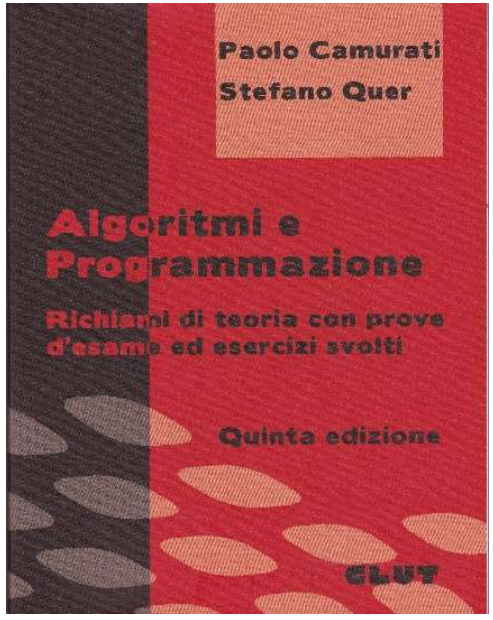
Texts, Readings, Handouts



Texts, Readings, Handouts

- Printed material
  - Theory part
    - P. Camurati, S. Quer, "Advanced Programming and Problem-Solving Strategies in C. Part II: Algorithms and Data Structures", Second Edition, **CLUT**, Sept. 2018

In **Italian**
Includes some theory and many solutions of theory exercises

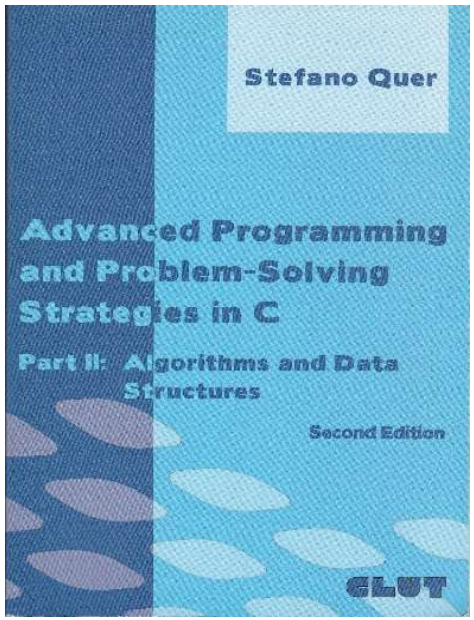


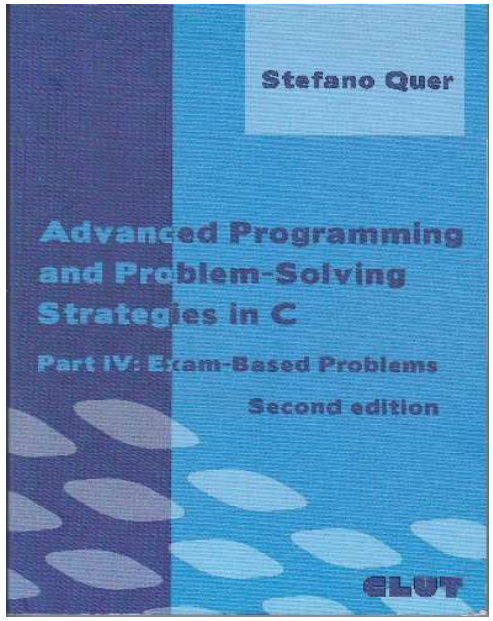
Texts, Readings, Handouts

- Printed material
  - Programming part
    - S. Quer, "Advanced Programming and Problem-Solving Strategies in C. Part II: Algorithms and Data Structures", Second Edition, **CLUT**, Sept. 2018
    - S. Quer, "Advanced Programming and Problem-Solving Strategies in C. Part IV: **Exam-Based Problems**", Second Edition, **CLUT**, Sept. 2018

Theory and problem solving

Problem solving

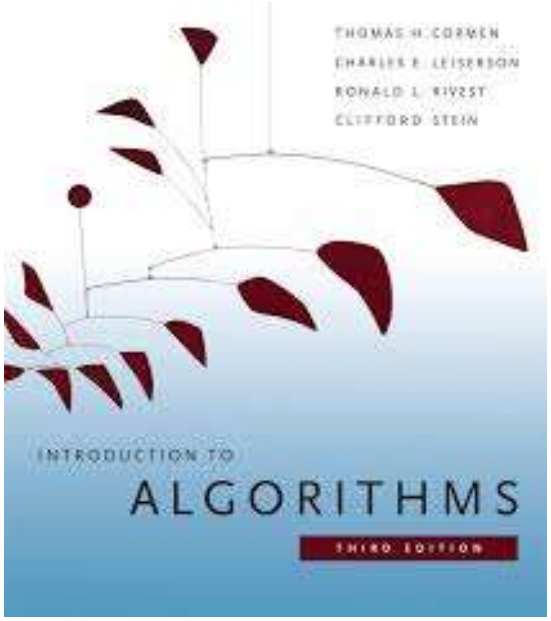




Texts, Readings, Handouts

- Other printed material
  - Theory part
    - T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, "Introduction to Algorithms", McGraw-Hill

Easy to follow
It adopts pseudo-code (not C)

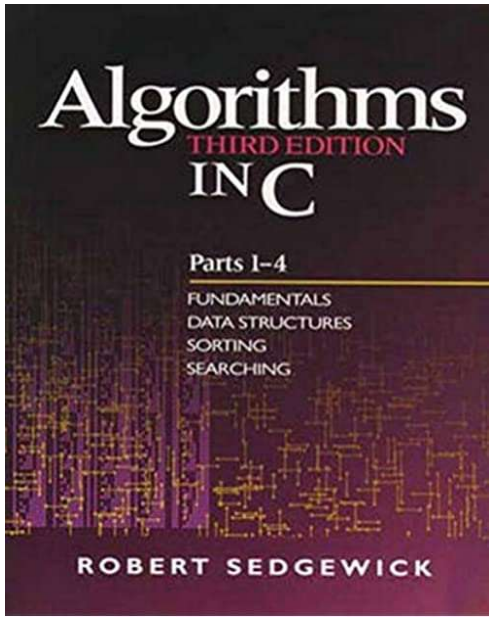


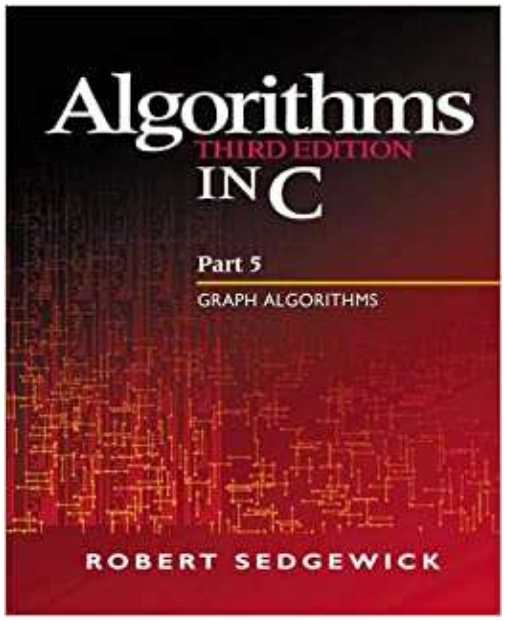
Texts, Readings, Handouts

- Other printed material
  - Theory and programming part
    - R. Sedgewick, "Algorithms in C, Parts 1-4: Fundamentals, Data Structures, Sorting, Searching" and "Algorithms in C, Part 5: Graph Algorithms", Addison-Wesley Professional

Hard to follow
they adopt C code

All others books by
R. Sedgewick are
not suggested

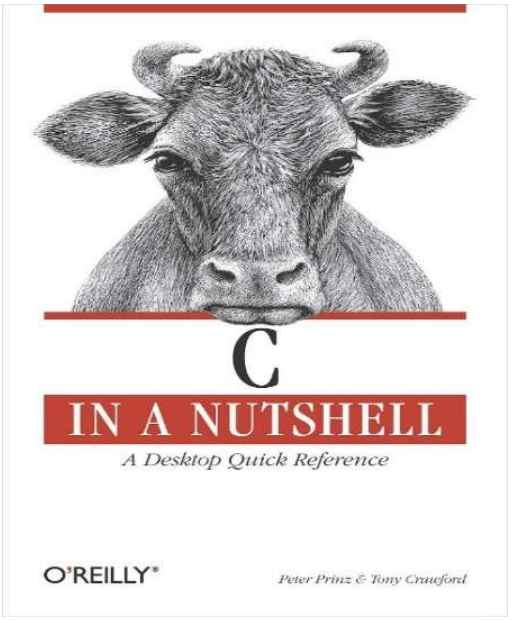




Texts, Readings, Handouts

- Other printed material
  - Programming part
    - P. Prinz and T. Crawford, "C in a Nutshell. The definitive reference", 2nd Edition, O'Reilly

C reference
(no problem solving)



Assessment and Grading Criteria

- The exam consists of
  - A **unique** written exam
  - It can be taken
    - In-classroom
      - In a laboratory, using the Exam platform, with the assistance of one of the instructors
    - Virtually
      - From a remote location, using the Exam platform, integrated with the Respondus and LockDown Browser tools
      - In this case no virtual-classroom will be open

Assessment and Grading Criteria

- Please
  - Enroll for the written exam

Warning: The Exam platform **does not allow** to non-enrolled students to take the exam

Warning: The instructors **cannot** help much

  - Verify your hardware devices (e.g., the keyboard) and software tools far in advance
- No books, no notes, no transparencies are allowed
  - Respect the etical behaviour

Assessment and Grading Criteria

- Maximum time length
  - 3 – 3.5 hours
  - Plus 30 minutes "connection/disconnection" times
- Maximum grade
  - 36 points
  - Rounding-up is applied in all cases
    - Marks ≥ x.5 are rounded-up to x+1
      - 25.49→25, 25.50→26
    - Marks ≥ 17.25 (and ≤18) are rounded-up to 18
      - 17.24→Fail, 17.25→18
    - Marks ≥ 32 are transformed into 30 with honour

Extra points with round-up
No extra-discussion, no oral, etc.

Assessment and Grading Criteria

- The written test include
  - Theory questions and programming puzzles
    - Theory questions include exercises and questions on all theoretical aspects presented during the class
    - Programming puzzles include problem solving, i.e., the manual application of standard algorithms on data streams and data structures

Assessment and Grading Criteria

- Questions and puzzles can be
  - Closed
    - Automatically corrected, by the system
    - On/off evaluation, i.e., a partially wrong response is a wrong response
    - Pay attention to the indicated format, **format errors count as all other errors**
  - Open
    - Manually corrected, by the instructors

Assessment and Grading Criteria

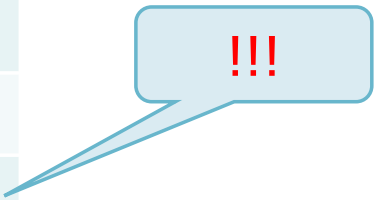
- The "standard" composition of a written test
  - All closed questions have an **on/off** evaluation
  - In all closed questions "**format errors**" count as **errors**

	Number of Exercises	Marks	Total
Closed theory questions	9	1	9
Open theory questions	1	4	4
Closed programming questions	3	2	6
Open programming puzzles	1 1 1	3 5 9	17
Total	**16**		**36**

Results so far

- Results for AADS for 1 academic years
  - 2021-2022
  - Evaluation date: 20.09.2022

Total Number of ...	Total	[%]
... students enrolled	175	100
... exam taken	101	57
... students who take the exam (at **least** once)	**71**	41
... students who **never** take the exam	**104**	59
... passed (on total = /175)	50	29
... passed (on who takes the exam = /71)		71
Average mark	**24.9**	



Results so far

Computer sciences

Programming techniques

Algorithms and Data Structures

Results so far

Computer sciences

Bottleneck

Programming techniques

Funnel

Algorithms and Data Structures

Results so far

Computer sciences

Programming techniques

Algorithms and Data Structures

Programming puns

- Old programmers never die
  - They just don't C as well
- Measuring programming progress by lines of code is like measuring aircraft building progress by weight [Bill Gates]
- Why did the functions stop calling each other?
  - Because they had constant arguments
- How does a computer scientist order three beers?
  - He holds up two fingers [Sriram Gopalan]

Programming puns

- Programming is like sex because...
  - One mistake and you have to support it for the rest of your life [Michael Sinz]
  - You can do it for money or for fun
  - Public schools don't do a very good job teaching kids about it
  - Some people are just naturally good at it ... but some people will never realize how bad they are, and you're wasting your time trying to tell them
  - One little thing going wrong can ruin everything
  - Beginners do a lot of clumsy fumbling about
  - You'll miss it if it's been a while since you've done it

To summarize

- During the course we will face both theory and practice problems

To summarize

- During the course we will face both theory and practice problems
  - Theory is when you know everything but nothing works

To summarize

- During the course we will face both theory and practice problems
  - Theory is when you know everything but nothing works
  - Practice is when everything works but no one knows why

To summarize

- During the course we will face both theory and practice problems
  - Theory is when you know everything but nothing works
  - Practice is when everything works but no one knows why
  - **In this class, theory and practice will be combined: Nothing will work and no one will know why**

(possibly) Albert Einstein, 1879-1955