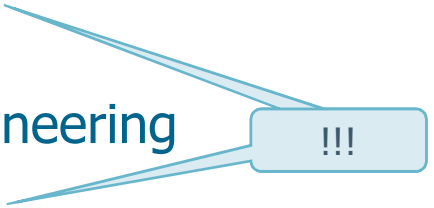
Algorithms and Programming

Course Introduction

Stefano Quer
Department of Control and Computer Engineering
Politecnico di Torino

General Information

- Algorithms and Programming
  - $02OGD_{LM}$ & $01OGD_{LP}$
  - ING-INF/05
- Bachelor-level degree
  - Computer Engineering
    - 2° year, 12 credits, **120** hours
    - All students (from A to Z)
  - Electronic and Communication Engineering
    - 3° year, 10 credits, **100** hours
    - All students (from A to Z)



Teacher and Assistants

Quer Stefano, DAUIN
011-090-7076
stefano.quer@polito.it
http://fmgroup.polito.it/quer/

Gianfranco Politano, DAUIN
011-090-7198
gianfranco.politano@polito.it

Fabrizio Finocchiaro, DAUIN
011-090-7048
fabrizio.finocchiaro@polito.it

Subject Fundamentals

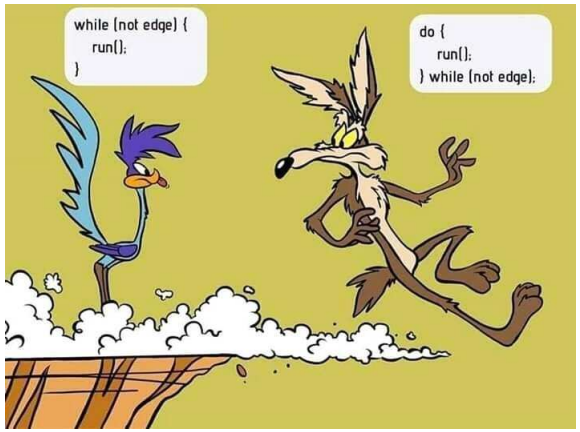
- Acquire **adequate**
  - Knowledge in algorithms, data structures, and their implementation in C
  - Skills to solve complex problems
- Student should gradually evolve from analytic to more design-oriented skills
- The course introduce
  - Theoretical foundations and solutions to "classical" problems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

Learning Outcomes

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

Prerequisites

- Incremental nature of the course with respect to the first year class "**Computer Science**"
- **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



Contents

- Main course items
  - Review of basic language and problem solving
  - Algorithm analysis
  - Sorting algorithms
  - Static and dynamic data structures
  - Modularity and modular implementation
  - Recursion and recursive programs
  - Abstract objects, collections of objects and ADTs
  - Data structures for symbol tables
  - Graph theory

Plus ...
problem solving on all topics

Contents

- 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

Contents

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.

Contents

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.

Contents

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 Modes

- There is no distinction between theory and practice lessons
  - Lectures include practice lessons
  - 5 blocks of 1.5 hours
  - ECE students have to follow 100 hours not 120
    - They will be told when the lecture, practice or laboraotory is optional for them
    - Mainly in the second part of the course (from the Binary Seaerch Trees tropic on)

Delivery Modes

- Lectures and practice are extended with 20 additional hours in laboratory
  - 2 lab teams
    - 2 blocks of 1.5 hours (1 for each lab team)
    - Team A: A – LA, Monday 10.00-11.30, lab 4
    - Team B: LB – Z, Monday 11.30-13.00, lab 4
  - 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)

Texts, Readings, Handouts

- Material
  - Personal student's page (Politecnico portal)
    - Video-recordings
    - Calendar, rules and deadlines
    - Exams bookings and exam results
  - Teacher's personal WEB page
    - **htttp://staff.polito.it/stefano.quer/**
    - **htttp://fmgroup.polito.it/quer/**
    - Material used during all lectures and practice
      - Overheads
      - Laboratory exercises and solutions
      - Examination texts and solutions

Texts, Readings, Handouts

- Material directly used in class
  - Overheads
    - Teacher's personal web page
  - Laboratory specs and solutions
    - Teacher's personal web page
  - Printed material
    - 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

Texts, Readings, Handouts

- Other printed material
  - Theory part
    - T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, "Introduction to Algorithms", McGraw-Hill.
    - R. Sedgewick, "Algorithms in C, Parts 1-4: Fundamentals, Data Structures, Sorting, Searching" and "Algorithms in C, Part 5: Graph Algorithms", Addison-Wesley Professional
  - Programming part
    - P. Prinz and T. Crawford, "C in a Nutshell. The definitive reference", 2nd Edition, O'Reilly

Easier to follow but it adopts pseudo-code (not C)

Harder to follow but it adopts C code

C reference (no problem solving)

Assessment and Grading Criteria

- The exam consists of
  1. A written exam
     - To check the
       - Theory knowledge
       - Problem-solving ability
  2. An off-line section
     - To allow each student to
       - Double-check the written programming part
       - Up-load a final (complete and working) program (based on the written programmng part)
  3. An oral exam
     - To complete the exam and reach a final verdict/mark

Assessment and Grading Criteria

1. The written exam includes

Cell phones have to be turned-off and place on the desk at the beginning of the exam

- A theory part
  - Includes exercises and questions on all theoretical aspects presented during the class
    - Manual application of standard algorithms on data streams and data structures
  - No books, no notes, no transparencies are allowed
  - Maximum time length 60 minutes
  - Evaluation
    - Maximum grade: 12 points
    - Minimum passing threshold: 6 points

The theory part will (often) be different for ECE students

Assessment and Grading Criteria

- A problem solving (programming) part
  - Checks problem solving ability in C
  - At most one book on the C language is allowed
  - Available in two alternative modes
    - A "standard" problem solving task (program) the emphasis being on problem-solving and design skills
    - A "simplified" set of 3 C exercises, with less emphasis on design and more on the ability to use advanced C features
  - Maximum time length 120 minutes

Written part: 180 minutes overall (60+120)

The programming part will (often) be the same for ECE students but possibly solvable with a simplified strategy

Assessment and Grading Criteria

- Each candidate **must** make a copy of the program at the end of the written part
  - All candidates will have a few minutes to take pictures of their paperwork at the end of the exam
- Evaluation
  - Stardard part
    - Maximum grade: 18 points
    - Minimum passing threshold: 9 points
  - Simplified part
    - Maximum grade: 12 points
    - Minimum passing threshold: 6 points

!!!

➢ The final written mark is the sum of the theory part and the programming part

Assessment and Grading Criteria

2. The off-line section
   - Allows to each candidate to decide whether he/she really wants to take the exam or not
   - Each candidate will have 3 working days to
     - Verify the quality and correctness of his/her solution
     - Upload on the **course webpage** a copy of the **working** program (or programs)
       - Follow instruction (**on web side**) to prepare and upload the required material
   - In case the above material is not uploaded on time, the written exam will **not** be ranked otherwise a mark will be uploaded on the course webpage

Assessment and Grading Criteria

- In any case the final report helps the teachers to reach a final verdict but teachers delivers the final mark based on the on-line written version **not** on the off-line final report

Assessment and Grading Criteria

3. Oral examination
   - Students with a written mark
     - < 15 /30 cannot take the oral examination
     - = 15,16,17 /30 must take the oral examination if they want to pass the exam
     - > 18/30
       - May be asked (by the teachers) to sit for the oral exam to consolidate (or check) their evaluation
         - The list of mandatory oral examinations is inserted in the "Avvisi" section of the portal web page when the written results are published
       - May ask to sit for the oral exam to improve their mark
         - The decision can be taken during the oral examination section

Assessment and Grading Criteria

- The oral examination usually consists of two parts with one or more questions/exercises
  - One on all theory topics of the course
  - One on problem solving and programming ability
- The final mark integrates partial results (written part and oral exam)
  - It is not a mere average or sum of those parts

7 students still remain from < 2013-2014 (6+ years)

Results so far ...

- Results "cohort" by "cohort"
  - From 2011-2012 to 2018-2019
  - Evaluation date: 30.09.2019

E = Enrolled
P = Passed

Accademic Year	2013-2014			2014-2015			2015-2016		
	E	P	P[%]	E	P	P[%]	E	P	P[%]
2013-2014	56	12	21						
2014-2015	28	4	7	67	22	33			
2015-2016	23	6	11	37	2	3	62	18	29
2016-2017	16	5	9	31	4	6	36	6	10
2017-2018	9	0	0	22	5	8	26	8	13
2018-2019	8	2	4	14	3	5	16	2	3
Total passed		**29**	**52**		**36**	**54**		**34**	**55**
Dropped-out		**21**	**38**		**20**	**33**		**14**	**23**
Still enrolled		**6**	**11**		**11**	**16**		**14**	**23**

Results so far ...

From 10 (≤2015-2016) to 12 credits (≥ 2016-2017)

Computer Engineeering + Electronic Computer Engineering (≥ 2017-2018)

Longer examination time
Optional oral examination

Accademic Year	2016-2017			2017-2018			2018-2019		
	E	P	P[%]	E	P	P[%]	E	P	P[%]
2013-2014									
2014-2015									
2015-2016									
2016-2017	61	28	46						
2017-2018	27	6	10	87	35	40			
2018-2019	20	2	3	47	13	15	121	53	44
Total passed		**36**	**59**		**48**	**55**	**121**	**53**	**44**
Dropped-out		**7**	**12**		**5**	**6**		**0**	**0**
Still enrolled		**18**	**30**		**34**	**39**		**68**	**56**

Results so far ...

- Results for all academic years
  - From 2011-2012 to 2018-2019
  - Evaluation date: 30.09.2019

Total Number of ...	Total	[%]
... students enrolled	558	100
... exam taken	645	109
... students who never take the exam	**206**	37
... students who dropped-out	**93**	17
... passed (on total = 293/558)	293	53
... passed (on who takes the exam = 293/(558-206))	293	97
Average mark	**22**	

!!!

Not considered 2019-2020 !

Students' comments

- Organization
  - Dedicate less time to the initial review
  - The program of this curse is so extended that ...

> And then ... there is nothing I can really do about that ...

  - It is not good at all to start the first lecture warning the students from the course and how difficult it is and how many students could not even pass the exam!

> Nevertheless ... many students "close the stable door after the horse has bolted"

Students' comments

- Laboratory
  - Going to the laboratory classes isn't so useful …

And then … labs are extremely important: Programming is learn by experience then you need more experience …

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