Guide for Technical Development

Having a solid foundation in Computer Science is important in being a successful Software Engineer. This guide is a suggested path for Undergraduate students to develop their technical skills academically and non-academically through self paced hands-on learning. The guide is broken down by year to help build your skills as you progress in your studies. You may use this guide to determine courses to take. However, please make sure you are taking courses required for your major or faculty in order to graduate. The online resources provided in this guide are not meant to replace courses available at your University. However, they may help supplement your learnings or provide an introduction to the topic.

Using this guide

- Please use this guide at your discretion
- This is a suggested timeline and not meant to be a strict timeline you need to follow
- There may be other things you want to learn or do outside of this guide - go for it!
- Checking off all items in this guide does not guarantee a job at Google
- This guide will evolve or change - check back for updates

Join our Google+ Community to get additional tips, resources, and other students interested in development.

Guide Updated July 2013

☐ **Take at least one CS course**
Notes: Introduction to Computer Science Course that provides instructions on coding
Online Resources: Udacity - intro to CS course, Coursera - Computer Science 101

☐ **Become familiar with and able to code in at least one object oriented programming language: C++, Java, or Python**
Online Resources: Coursera - Learn to Program: The Fundamentals, MIT Intro to Programming in Java, Google's Python Class, Coursera - Introduction to Python

☐ **Learn other Programming Languages**
Notes: Add to your repertoire - Java Script, HTML, Ruby, or PHP.
Online Resources: w3school.com - HTML Tutorial

☐ **Work on at least one major CS project outside of the classroom.**
Notes: Create and maintain a website, build your own server, or build a robot.

☐ **Work on a small piece of a large system (codebase), read and understand existing code, track down documentation, and debug things.**
Notes: Github is a great way to read other people's code or contribute to a project.
Online Resources: Github, Kiln

☐ **Develop logical reasoning to derive conclusion**
Online Resources: Coursera - Introduction to Logic

- [ ] **Strong understanding of Algorithms and Data Structures**

Notes: Learn about fundamental data types (stack, queues, and bags), sorting algorithms (quicksort, mergesort, heapsort), and data structures (binary search trees, red-black trees, hash tables) Online Resources: MIT Introduction to Algorithms, Coursera Introduction to Algorithms Part 1 & Part 2, List of Algorithms, List of Data Structures, Book: The Algorithm Design Manual

- [ ] **Become Proficient in C++, Java, or Python**

Online Resources: Udacity's Design of Computer Programs, Coursera - Learn to Program: Crafting Quality Code, Coursera - Programming Languages, Brown University - Introduction to Programming Languages

- [ ] **Develop your UI / Front-end design knowledge and skills**

Notes: Learn HTML, CSS, and JavaScript.

- [ ] **Work on several CS projects outside the classroom**

Notes: Get involved in CS clubs or organizations that have group projects. Check out projects available in the Open Source community
Online Resources: Apache List of Projects, Google Summer of Code, Google Developer Group

- [ ] **Work on project with other programmers.**

Notes: This will help you improve your ability to work well in a team and enable you to learn from others.

- [ ] **Build a solid foundation in discrete math**

Online Resources: Coursera - Linear and Discrete Optimization, Coursera - Probabilistic Graphical Models, Coursera - Game Theory

☐ **Strong proficiency C++ or Java**
Online Resources: Udacity - Design of Computer Programs

☐ **Strong knowledge of algorithms / data structures: Big O, sorting hashtables, trees, graphs**
Notes: Practice your algorithmic knowledge through coding competitions like CodeJam or ACM's International Collegiate Programming Contest.
Online Resources: CodeJam, ACM ICPC

☐ **Learn other Programming Languages**
Notes: Learn C, Perl, Shell. Lisp, or Scheme.

☐ **Develop a strong knowledge of operating systems**
Online Resources: UC Berkeley Computer Science 162

☐ **Test Your Code**
Notes: Learn how to catch bugs, create tests, and break your software
Online Resources: Udacity - Software Testing Methods, Udacity - Software Debugging

☐ **Internship experience in software engineering**
Notes: Make sure you apply for internships well in advance of the period internships take place. In the US, internships take place during the summer, May-September, and applications are usually open several months in advance. You don't have to wait until your
Online Resources: google.com/jobs

☐ **Learn how to build compilers**
Online Resources: Coursera - Compilers

☐ **Learn cryptography**
Online Resources: Coursera - Cryptography

☐ **Learn Parallel Programming**
Online Resources: Coursera - Heterogeneous Parallel Programming

***Thinking about going to graduate school?**
You can still have an internship before you go.