**课程地图**

正如这章开头提到的，这份课程地图仅仅是一个仅供参考的课程规划，我作为一个临近毕业的本科生。深感自己没有权利也没有能力向别人宣扬“应该怎么学”。因此如果你觉得以下的课程分类与选择有不合理之处，我全盘接受，并深感抱歉。你可以在下一节定制属于你的课程地图

以下课程类别中除了含有 基础 和 入门 字眼的以外，并无明确的先后次序，大家只要满足某个课程的先修要求，完全可以根据自己的需要和喜好选择想要学习的课程。

**数学基础**

微积分 + 线性代数

作为大一新生，学好微积分线代是和写代码至少同等重要的事情，相信已经有无数的前人经验提到过这一点，但我还是要不厌其烦地再强调一遍：学好微积分线代真的很重要！你也许会吐槽这些东西岂不是考完就忘，那我觉得你是并没有把握住它们本质，对它们的理解还没有达到刻骨铭心的程度。如果觉得老师课上讲的内容晦涩难懂，不妨参考 MIT 的 Calculus Course 和 18.06: Linear Algebra 的课程 notes，至少于我而言，它帮助我深刻理解了微积分和线性代数的许多本质。顺道再安利一个油管数学网红 3Blue1Brown，他的频道有很多用生动形象的动画阐释数学本质内核的视频，兼具深度和广度，质量非常高。

信息论入门

作为计算机系的学生，及早了解一些信息论的基础知识，我觉得是大有裨益的。但大多信息论课程都面向高年级本科生甚至研究生，对新手极不友好。而 MIT 的 6.050J: Information theory and Entropy 这门课正是为大一新生量身定制的，几乎没有先修要求，涵盖了编码、压缩、通信、信息熵等等内容，非常有趣。

**数学进阶**

离散数学

集合论、图论、概率论等等是算法推导与证明的重要工具，也是后续高阶数学课程的基础。但我觉得这类课程的讲授很容易落入理论化与形式化的窠臼，让课堂成为定理结论的堆砌，而无法使学生深刻把握理论的本质，进而造成学了就背，考了就忘的怪圈。如果能在理论教学中穿插算法运用实例，学生在拓展算法知识的同时也能窥见理论的力量和魅力。

概率论

UCB CS70 : discrete Math and probability theory 和 UCB CS126 : Probability theory 是 UC Berkeley 的概率论课程，前者覆盖了离散数学和概率论基础，后者则涉及随机过程以及深入的理论内容。两者都非常注重理论和实践的结合，有丰富的算法实际运用实例，后者还有大量的 Python 编程作业来让学生运用概率论的知识解决实际问题。

数值分析

作为计算机系的学生，培养计算思维是很重要的，实际问题的建模、离散化，计算机的模拟、分析，是一项很重要的能力。而这两年开始风靡的，由 MIT 打造的 Julia 编程语言以其 C 一样的速度和 Python 一样友好的语法在数值计算领域有一统天下之势，MIT 的许多数学课程也开始用 Julia 作为教学工具，把艰深的数学理论用直观清晰的代码展示出来。

ComputationalThinking 是 MIT 开设的一门计算思维入门课，所有课程内容全部开源，可以在课程网站直接访问。这门课利用 Julia 编程语言，在图像处理、社会科学与数据科学、气候学建模三个 topic 下带领学生理解算法、数学建模、数据分析、交互设计、图例展示，让学生体验计算与科学的美妙结合。内容虽然不难，但给我最深刻的感受就是，科学的魅力并不是故弄玄虚的艰深理论，不是诘屈聱牙的术语行话，而是用直观生动的案例，用简练深刻的语言，让每个普通人都能理解。

上完上面的体验课之后，如果意犹未尽的话，不妨试试 MIT 的 18.330 : Introduction to numerical analysis，这门课的编程作业同样会用 Julia 编程语言，不过难度和深度上都上了一个台阶。内容涉及了浮点编码、Root finding、线性系统、微分方程等等方面，整门课的主旨就是让你利用离散化的计算机表示去估计和逼近一个数学上连续的概念。这门课的教授还专门撰写了一本配套的开源教材 Fundamentals of Numerical Computation，里面附有丰富的 Julia 代码实例和严谨的公式推导。

如果你还意犹未尽的话，还有 MIT 的数值分析研究生课程 18.335: Introduction to numerical method 供你参考。

微分方程

如果世间万物的运动发展都能用方程来刻画和描述，这是一件多么酷的事情呀！虽然几乎任何一所学校的 CS 培养方案中都没有微分方程相关的必修课程，但我还是觉得掌握它会赋予你一个新的视角来审视这个世界。

由于微分方程中往往会用到很多复变函数的知识，所以大家可以参考 MIT18.04: Complex variables functions 的课程 notes 来补齐先修知识。

MIT18.03: differential equations 主要覆盖了常微分方程的求解，在此基础之上 MIT18.152: Partial differential equations 则会深入偏微分方程的建模与求解。掌握了微分方程这一有力工具，相信对于你的实际问题的建模能力以及从众多噪声变量中把握本质的直觉都会有很大帮助。

**数学高阶**

作为计算机系的学生，我经常听到数学无用论的论断，对此我不敢苟同但也无权反对，但若凡事都硬要争出个有用和无用的区别来，倒也着实无趣，因此下面这些面向高年级甚至研究生的数学课程，大家按兴趣自取所需。

凸优化

Standford EE364A: Convex Optimization

信息论

MIT6.441: Information Theory

应用统计学

MIT18.650: Statistics for Applications

初等数论

MIT18.781: Theory of Numbers

密码学

Standford CS255: Cryptography

**编程入门**

Languages are tools, you choose the right tool to do the right thing. Since there's no universally perfect tool, there's no universally perfect language.

Shell

MIT-Missing-Semester

Python

CS50P: CS50's Introduction to Programming with Python

Harvard CS50: This is CS50x

UCB CS61A: Structure and Interpretation of Computer Programs

C++

Stanford CS106B/X: Programming Abstractions

Stanford CS106L: Standard C++ Programming

Rust

Stanford CS110L: Safety in Systems Programming

OCaml

Cornell CS3110 textbook: Functional Programming in OCaml

**电子基础**

电路基础

作为计算机系的学生，了解一些基础的电路知识，感受从传感器收集数据到数据分析再到算法预测整条流水线，对于后续知识的学习以及计算思维的培养还是很有帮助的。EE16A&B: Designing Information Devices and Systems I&II 是伯克利 EE 学生的大一入门课，其中 EE16A 注重通过电路从实际环境中收集和分析数据，而 EE16B 则侧重从这些收集到的数据进行分析并做出预测行为。

信号与系统

信号与系统是一门我觉得非常值得一上的课，最初学它只是为了满足我对傅里叶变换的好奇，但学完之后我才不禁感叹，傅立叶变换给我提供了一个全新的视角去看待这个世界，就如同微分方程一样，让你沉浸在用数学去精确描绘和刻画这个世界的优雅与神奇之中。

MIT 6.003: signal and systems 提供了全部的课程录影、书面作业以及答案。也可以去看这门课的远古版本

而 UCB EE120: Signal and Systems 关于傅立叶变换的 notes 写得非常好，并且提供了6 个非常有趣的 Python 编程作业，让你实践中运用信号与系统的理论与算法。

**数据结构与算法**

算法是计算机科学的核心，也是几乎一切专业课程的基础。如何将实际问题通过数学抽象转化为算法问题，并选用合适的数据结构在时间和内存大小的限制下将其解决是算法课的永恒主题。如果你受够了老师的照本宣科，那么我强烈推荐伯克利的 UCB CS61B: Data Structures and Algorithms 和普林斯顿的 Coursera: Algorithms I & II，这两门课的都讲得深入浅出并且会有丰富且有趣的编程实验将理论与知识结合起来。

以上两门课程都是基于 Java 语言，如果你想学习 C/C++ 描述的版本，可以参考斯坦福的数据结构与基础算法课程 Stanford CS106B/X: Programming Abstractions。偏好 Python 的同学可以学习 MIT 的算法入门课 MIT 6.006: Introduction to Algorithms

对一些更高级的算法以及 NP 问题感兴趣的同学可以学习伯克利的算法设计与分析课程 UCB CS170: Efficient Algorithms and Intractable Problems 或者 MIT 的高阶算法 MIT 6.046: Design and Analysis of Algorithms。

**软件工程**

入门课

一份“能跑”的代码，和一份高质量的工业级代码是有本质区别的。因此我非常推荐低年级的同学学习一下 MIT 6.031: Software Construction 这门课，它会以 Java 语言为基础，以丰富细致的阅读材料和精心设计的编程练习传授如何编写不易出 bug、简明易懂、易于维护修改的高质量代码。大到宏观数据结构设计，小到如何写注释，遵循这些前人总结的细节和经验，对于你此后的编程生涯大有裨益。

专业课

当然，如果你想系统性地上一门软件工程的课程，那我推荐的是伯克利的 UCB CS169: software engineering。但需要提醒的是，和大多学校（包括贵校）的软件工程课程不同，这门课不会涉及传统的 design and document 模式，即强调各种类图、流程图及文档设计，而是采用近些年流行起来的小团队快速迭代 Agile Develepment 开发模式以及利用云平台的 Software as a service 服务模式。

**体系结构**

入门课

从小我就一直听说，计算机的世界是由 01 构成的，我不理解但大受震撼。如果你的内心也怀有这份好奇，不妨花一到两个月的时间学习 Coursera: Nand2Tetris 这门无门槛的计算机课程。这门麻雀虽小五脏俱全的课程会从 01 开始让你亲手造出一台计算机，并在上面运行俄罗斯方块小游戏。一门课里涵盖了编译、虚拟机、汇编、体系结构、数字电路、逻辑门等等从上至下、从软至硬的各类知识，非常全面。难度上也是通过精心的设计，略去了众多现代计算机复杂的细节，提取出了最核心本质的东西，力图让每个人都能理解。在低年级，如果就能从宏观上建立对整个计算机体系的鸟瞰图，是大有裨益的。

专业课

当然，如果想深入现代计算机体系结构的复杂细节，还得上一门大学本科难度的课程 UCB CS61C: Great Ideas in Computer Architecture。UC Berkeley 作为 RISC-V 架构的发源地，在体系结构领域算得上首屈一指。其课程非常注重实践，你会在 Project 中手写汇编构造神经网络，从零开始搭建一个 CPU，这些实践都会让你对计算机体系结构有更为深入的理解，而不是仅停留于“取指译码执行访存写回”的单调背诵里。

**系统入门**

计算机系统是一个庞杂而深刻的主题，在深入学习某个细分领域之前，对各个领域有一个宏观概念性的理解，对一些通用性的设计原则有所知晓，会让你在之后的深入学习中不断强化一些最为核心乃至哲学的概念，而不会桎梏于复杂的内部细节和各种 trick。因为在我看来，学习系统最关键的还是想让你领悟到这些最核心的东西，从而能够设计和实现出属于自己的系统。

MIT6.033: System Engineering 是 MIT 的系统入门课，主题涉及了操作系统、网络、分布式和系统安全，除了知识点的传授外，这门课还会讲授一些写作和表达上的技巧，让你学会如何设计并向别人介绍和分析自己的系统。这本书配套的教材 Principles of Computer System Design: An Introduction 也写得非常好，推荐大家阅读。

CMU 15-213: Introduction to Computer System 是 CMU 的系统入门课，内容覆盖了体系结构、操作系统、链接、并行、网络等等，兼具广度和深度，配套的教材 Computer Systems: A Programmer's Perspective 也是质量极高，强烈建议阅读。

**操作系统**

没有什么能比自己写个内核更能加深对操作系统的理解了。

操作系统作为各类纷繁复杂的底层硬件虚拟化出一套规范优雅的抽象，给所有应用软件提供丰富的功能支持。了解操作系统的设计原则和内部原理对于一个不满足于当调包侠的程序员来说是大有裨益的。出于对操作系统的热爱，我上过国内外很多操作系统课程，它们各有侧重和优劣，大家可以根据兴趣各取所需。

MIT 6.S081: Operating System Engineering，MIT 著名 PDOS 实验室出品，11 个 Project 让你在一个实现非常优雅的类Unix操作系统xv6上增加各类功能模块。这门课也让我深刻认识到，做系统不是靠 PPT 念出来的，是得几万行代码一点点累起来的。

UCB CS162: Operating System，伯克利的操作系统课，采用和 Stanford 同样的 Project —— 一个教学用操作系统 Pintos。我作为北京大学2022年和2023年春季学期操作系统实验班的助教，引入并改善了这个 Project，课程资源也会全部开源，具体参见课程网站。

NJU: Operating System Design and Implementation，南京大学的蒋炎岩老师开设的操作系统课程。蒋老师以其独到的系统视角结合丰富的代码示例将众多操作系统的概念讲得深入浅出，此外这门课的全部课程内容都是中文的，非常方便大家学习。

HIT OS: Operating System，哈尔滨工业大学的李治军老师开设的中文操作系统课程。李老师的课程基于 Linux 0.11 源码，十分注重代码实践，并站在学生视角将操作系统的来龙去脉娓娓道来。

**并行与分布式系统**

想必这两年各类 CS 讲座里最常听到的话就是“摩尔定律正在走向终结”，此话不假，当单核能力达到上限时，多核乃至众核架构如日中天。硬件的变化带来的是上层编程逻辑的适应与改变，要想充分利用硬件性能，编写并行程序几乎成了程序员的必备技能。与此同时，深度学习的兴起对计算机算力与存储的要求都达到了前所未有的高度，大规模集群的部署和优化也成为热门技术话题。

**并行计算**

CMU 15-418/Stanford CS149: Parallel Computing

**分布式系统**

MIT 6.824: Distributed System

**系统安全**

不知道你当年选择计算机是不是因为怀着一个中二的黑客梦想，但现实却是成为黑客道阻且长。

理论课程

UCB CS161: Computer Security 是伯克利的系统安全课程，会涵盖栈攻击、密码学、网站安全、网络安全等等内容。

ASU CSE365: Introduction to Cybersecurity 亚利桑那州立大学的 Web 安全课程，主要涉及注入、汇编与密码学的内容。

ASU CSE466: Computer Systems Security 亚利桑那州立大学的系统安全课程，涉及内容全面。门槛较高，需要对 Linux, C 与 Python 充分熟悉。

实践课程

掌握这些理论知识之后，还需要在实践中培养和锻炼这些“黑客素养”。CTF 夺旗赛是一项比较热门的系统安全比赛，赛题中会融会贯通地考察你对计算机各个领域知识的理解和运用。北大今年也成功举办了第 0 届和第 1 届，鼓励大家后期踊跃参与，在实践中提高自己。下面列举一些我平时学习（摸鱼）用到的资源：

CTF-wiki

CTF-101

Hacker-101

**计算机网络**

没有什么能比自己写个 TCP/IP 协议栈更能加深对计算机网络的理解了。

大名鼎鼎的 Stanford CS144: Computer Network，8 个 Project 带你实现整个 TCP/IP 协议栈。

如果你只是想在理论上对计算机网络有所了解，那么推荐计网著名教材《自顶向下方法》的配套学习资源 Computer Networking: A Top-Down Approach。

**数据库系统**

没有什么能比自己写个关系型数据库更能加深对数据库系统的理解了。

CMU 的著名数据库神课 CMU 15-445: Introduction to Database System 会通过 4 个 Project 带你为一个用于教学的关系型数据库 bustub 添加各种功能。实验的评测框架也免费开源了，非常适合大家自学。此外课程实验会用到 C++11 的众多新特性，也是一个锻炼 C++ 代码能力的好机会。

Berkeley 作为著名开源数据库 postgres 的发源地也不遑多让，UCB CS186: Introduction to Database System 会让你用 Java 语言实现一个支持 SQL 并发查询、B+ 树索引和故障恢复的关系型数据库。

**编译原理**

没有什么能比自己写个编译器更能加深对编译器的理解了。

Stanford CS143: Compilers 带你手写编译器。

Web开发

前后端开发很少在计算机的培养方案里被重视，但其实掌握这项技能还是好处多多的，例如搭建自己的个人主页，抑或是给自己的课程项目做一个精彩的展示网页。

两周速成版

MIT web development course

系统学习版

Stanford CS142: Web Applications

**计算机图形学**

Stanford CS148

Games101

Games103

Games202

**数据科学**

其实数据科学和机器学习与深度学习有着很紧密的联系，但可能更侧重于实践。Berkeley 的 UCB Data100: Principles and Techniques of Data Science 通过丰富的编程练习让你在实践中掌握各类数据分析工具和算法，并带领你体验从海量的数据集中提取出想要的结果，并对未来的数据或用户的行为做出相应的预测。但这只是一门基础课，如果想学习工业级别的数据挖掘与分析技术，可以尝试 Stanford 的大数据挖掘课程 CS246: Mining Massive Data Sets。

**人工智能**

近十年人工智能应该算是计算机界最火爆的领域。如果你不满足于整日听各路媒体争相报道人工智能相关的进展，而想真正一探究竟，那么非常推荐学习 Harvard 神课 CS50 系列的人工智能课程 Harvard CS50: Introduction to AI with Python。课程短小精悍，覆盖了传统人工智能领域的几大分支，并配有丰富有趣的 Python 编程练习来巩固你对人工智能算法的理解。美中不足的是这门课因为面向在线自学者的缘故内容较为精简，并且不会涉及特别深入的数学理论，如果想要系统深入地学习还需要一门本科生难度的课程，例如 Berkeley 的 UCB CS188: Introduction to Artificial Intelligence。这门课的 Project 复刻了经典游戏糖豆人，让你运用人工智能算法玩游戏，非常有趣。

**机器学习**

机器学习领域近些年最重要的进展就是发展出了基于神经网络的深度学习分支，但其实很多基于统计学习的算法依然在数据分析领域有着广泛的应用。如果你之前从未接触过机器学习的相关知识，而且不想一开始就陷入艰深晦涩的数学证明，那么不妨先从 Andrew Ng （吴恩达）的 Coursera: Machine Learning 学起。这门课在机器学习领域基本无人不晓，吴恩达以其深厚的理论功底和出色的表达能力把很多艰深的算法讲得深入浅出，并且非常实用。其配套的作业也是质量相当上乘，可以帮助你快速入门。

但上过这门课只能让你从宏观上对机器学习这一领域有一定了解，如果想真正理解那些“神奇”算法背后的数学原理甚至从事相关领域的科研工作，那么还需要一门更“数学”的课程，例如 Stanford CS229: Machine Learning 或者 UCB CS189: Introduction to Machine Learning。

**深度学习**

前几年 AlphaGo 的大热让深度学习进入了大众的视野，不少大学甚至专门成立了相关专业。很多计算机的其他领域也会借助深度学习的技术来做研究，因此基本不管你干啥多少都会接触到一些神经网络、深度学习相关的技术需求。如果想快速入门，同样推荐 Andrew Ng （吴恩达）的 Coursera: Deep Learning，质量无需多言，Coursera 上罕见的满分课程。此外如果你觉得英文课程学习起来有难度，推荐李宏毅老师的 国立台湾大学：机器学习 课程。这门课打着机器学习的名号，却囊括了深度学习领域的几乎所有方向，非常全面，很适合你从宏观上对这个领域有一个大致的了解。而且老师本人也非常幽默，课堂金句频出。

当然因为深度学习领域发展非常迅速，已经拥有了众多研究分支，如果想要进一步深入，可以按需学习下面罗列的代表课程，

计算机视觉

UMich EECS 498-007 / 598-005: Deep Learning for Computer Vision

Stanford CS231n: CNN for Visual Recognition

自然语言处理

Stanford CS224n: Natural Language Processing

图神经网络

Stanford CS224w: Machine Learning with Graphs

强化学习

UCB CS285: Deep Reinforcement Learning

定制属于你的课程地图

授人以鱼不如授人以渔。

以上的课程规划难免带有强烈的个人偏好，不一定适合所有人，更多是起到抛砖引玉的作用。如果你想挑选自己感兴趣的方向和内容加以学习，可以参考我在下面列出来的资源。

MIT OpenCourseWare: 麻省理工学院的课程资源开放共享项目，收录了数以千计的各科课程资源，其中计算机类的课号是 6.xxx。

MIT CS Course List: 麻省理工学院的 CS 课程列表。

UC Berkeley EECS Course Map: UC Berkeley 的 EECS 培养方案，以课程地图的方式将各门课程的类别和先修关系一目了然地呈现，其中绝大多数课程本书中均有收录。

UC Berkeley CS Course List: UC Berkeley 的 CS 课程列表。

Stanford CS Course List: 斯坦福的 CS 课程列表。