**专业探索-计算机**

作者：LR，清华大学交叉信息学院本科毕业，博士在读

首先声明一下，任何一行都会有各种各样的具体情况，作为一个个体很难绝对完善的覆盖各个方面，笔者亦不能免俗。这篇文章的某些观点仅代表个人意见，具体情况还希望多加调查以确保不要陷入个人偏见。

信息技术是当今信息社会非常基础的存在和生产力，也正因此，在很长的一段时间里计算机学科都显得很吃香，也成为了诸多学生选择专业的首选。

但和很多其他专业一样，计算机这个学科有着自己的特点、适应人群和并不简单轻松的职业发展路径。因此，本文将试图从专业学习、未来出路和生涯规划三个方面来介绍一下计算机专业的全貌。

# 关于学科本身

计算机是一门非常年轻的学科，其历史至今还不足百年。但是作为第三次工业革命的主要技术，计算机依然是这个时代最核心的学科之一。

简单来说，最狭义的计算机学科，研究的是如何通过现代的各种电子元器件（如最早的电子管、晶体管、集成电路）的组合，使得它可以按照人类预先设定的流程工作，从而完成各种计算问题。这是抽象到最抽象的计算机学科，也是其原本最根本的出发点和整个学科的基础核心。

事实上，计算机这个学科是一个非常特殊的存在，因为它是如此的包罗万象，以至于从各个不同的侧面和不同的层级看去，计算机科学可以呈现出各类学科的特点。例如，在最底层的实现部分，计算机学科是一门妥妥的工程学科，如何搭建电路以完成基本逻辑运算、如何提高效率、如何大规模集成，这些和设计精密的机械仪器和汽车发动机别无二致；然而到了抽象层面，算法、数据结构这些组织数据和解决问题的方法论，却是满满的理论研究的味道，到处充满了组合数学；在中间的各种应用层级，计算机学科又像是材料、建筑或是生物，在有限的抽象层级和有限的工具范围内进行组合，在某一特定应用领域中发现一些算不上是基础突破、但又不完全纯粹是技巧的技术和知识（例如优化计算机网络结构等）。

现如今，因为信息系统已经深度融合进入了各个学科行业，所以计算机学科也变得格外重要，其下属的具体方面也愈加丰富多元。有硬件层面的不断精益求精、试图突破摩尔定律的极限，有软件应用层面的工程代码的设计、网络系统设计，有交叉学科的人机交互、图形学，有试图破解人类智慧的人工智能，有在理论上探究更加算法与计算本质的理论计算机科学和量子计算等等。

# 学科的知识结构

## 本科课程体系

计算机学科的培养方案比较庞杂，不同学校也有各自的特点，不能一概而论，但大体上以下部分是几乎所有的培养方案都会包含的：

* 数理基础课：大一基本上都是上这些，一部分课程可能要到大二。其主要内容除去理工科基本必须的各项现代数学（微积分、线性代数、概率论）等之外，还有程序员必备的核心本领——代码能力，以及一些比较特色的、只有在计算机这个世界里显得格外重要的一些分支（如离散数学等）。部分学校会要求修大学物理
  + 数学课程：高等数学/微积分、线性代数/高等代数、概率论、复变函数、离散数学、数值分析
  + 代码课程：程序设计基础、面向对象程序设计、算法设计
  + 物理课程：大学物理
* 核心专业课：课程分布在大二、大三两年，据我所知，无论是哪个大学，计算机核心专业课都是十分之多的，因为计算机本身包含的内容确实太丰富了。不然怎么都调侃计算机系是熬夜脱发院系呢（摊手）
  + 代码进阶：汇编，Java，图像化编程，软件工程，数据结构
  + 硬件课程：电路原理，模拟电路，数字电路
  + 计算机系统：编译原理，计算机组成原理，操作系统
  + 计算机应用：网络原理，人工智能原理，信号原理，计算机图形学，数据库
* 可选方向的专业选修课：这些选修课的内容为计算机应用的最前沿，覆盖计算机各类子学科，一般会要求学生选修若干学分。很多课程会有一线的科研内容，以下是一些例子
  + 密码学，网络安全，数据挖掘，人工智能，神经网络，图像处理，软件开发，嵌入式系统，模式识别，高性能计算，多媒体，数字系统，博弈论，复杂度分析……

其中数理基础课和核心专业课程都属于必修内容，而后续的选修课程则与各类细分的专业方向相关。

## 细分方向

计算机有很多细分方向，不同方向之间也并非泾渭分明，大致归类如下

* **人工智能：**可能是最近最火的方向了，致力于用计算机系统和算法来重现、模拟甚至超越人类的智能，从而解决各种实际问题。其具体领域包括机器学习、模式识别、计算机视觉、自然语言处理、强化学习等等。
* **高性能计算：**致力于充分发挥计算机强大的计算能力、追求其极限，最广为人知的例子就是超级计算机了，如何设计并使得超大的计算集群得以运转并解决规模庞大的任务、尽力将每一个计算过程的效率发挥到极致、降低能耗、设计全新的计算架构（包括硬件和系统组织层面）是这个学科的核心问题。
* **计算机系统：**在执行最基本运算的硬件和实现各种纷繁复杂应用的软件之间，计算机系统是一个至关重要的平台和桥梁，你很难想象像先人那样用命令行操作你的电脑，用0和1编程。好的系统需要快速、安全、稳定，这其中涉及到大量的底层问题，而这也正是计算机系统的研究内容。
* **网络技术：**我们现在已经很习惯于网络了，但是互联网的到来是一件非凡的事情，其遗留的问题和未来的发展一直在继续。网络结构如何优化？远隔重洋的计算机如何安全的完成互相的身份确认和信息传输？黑客是怎么一回事、又该如何防范？这些都属于网络技术的范畴。
* **多媒体：**多媒体技术包括图像和视频的处理，人机交互过程中可能出现的问题等等。因为其跟新闻传媒、社会科学以及人体工程有很多交集，所以是计算机学科中“人文味”相对比较浓的一个领域。
* **理论计算机：**一切现在存在的计算机本质上都是图灵机，或者更确切的说，是基于冯诺依曼架构体系设计而成的计算机。这一理论本质带来一系列有趣的问题，例如计算能力的边界在哪里？面对一类问题，如何设计更好的算法，或者证明不存在更好的算法？对于一些本质十分困难的算法问题，如何设计好的近似或者随机算法？量子计算机真的在理论上可以完成传统计算机几乎不能完成的任务吗？为什么我们认为我们现在使用的密码系统是安全的？这些都属于理论计算机的范畴。

## 交叉学科

计算机学科与其他学科的交叉非常之多。例如与经济学交叉的算法经济学和算法博弈论，研究各种定价和拍卖的问题；与社会科学结合的数据（AI）伦理；与数学结合的计算数学和理论计算机；与生物结合的计算生物学和生物统计，用计算机的力量帮助破解基因密码，又比如alphafold可以用于破译蛋白质结构……

可以说，只要一门科学里涉及到数学、并可以将某些核心问题转化成数学问题，计算机学科就可以用计算发挥其威力。

# 计算机专业怎么学？

前面两个部分对计算机专业的知识结构与关注的主要方向做了基础的介绍，但很多同学关注的除了“计算机专业学什么”之外，更多是“怎么学”以及“难不难学”两个问题，这里会针对这两个问题，简单给出一个回答。

在展开介绍之前，有必要先明确一个十分根本的认识问题：**计算机学科到底在干啥？**很多人怀揣着对于信息时代黑科技的向往或是对高薪资的追求，就非常热切的选择了计算机专业，这种本质上基于浪漫的想象和功利的眼红的动机，在本科的学习中很容易出现落差或者不适应，即使是在笔者所在的清华大学，计算机类专业学习过程中出现不适应、跟不上的学生也比比皆是。

计算机学科最原始的初衷是用机械代替人力计算，后来机械装置变成电路，即使用复杂的电路完成预先规划的计算行为。因此，从本质上说，计算机学科是一个研究“如何设计并使用电子器件收集、存储、处理信息进行计算以达成某种目的”的学科，而如何控制此种计算或处理流程的预设逻辑，即为大家所熟知的“程序”或“代码”。

以此脉络展开，就不难理解世界上各个大学惊人一致的计算机专业的培养方案了：除去是一个理工科学生都必须要学的微积分、线性代数、概率论之外，一个计算机专业的学生首先需要理解基本的电路的工作原理、以及如何使用电路进行逻辑运算，所以就会有一系列的硬件课，包括电路原理、数字电路、计算机组成结构；

其次，计算机专业的学生一定要十分熟悉并且喜欢一种“安排”感，即从一个策划者的角度、像设计多米诺骨牌或哥德堡装置一样，环环相扣、如何一步步完成某一个目标的流程，这些本质上即“代码思维”，是一个在我国基础教育中相对较为缺失的一环，大学有关的课程是程序设计、算法设计、数据结构等；

接下来，是如何沟通抽象的逻辑流程设计（程序）到具体实践的电路操作，就有一系列的系统课：编译原理、汇编、操作系统、计算机组成原理、网络原理等。

以上是计算机这个学科的绝对核心，也是学起来最为硬核的部分，还有一些剩余的技术领域如人工智能、网络安全、高性能计算、图形学、软件工程等等，是由这些核心所延展出的具体应用。

所以，其实诸如重装系统、恢复硬盘、黑客技术等等，本质上其实是计算机具体应用层面（如计算机系统硬件、网络安全等）非常具体细化的专业知识技术，学习计算机主干学科内容的同学不会修电脑，实在是再正常不过的事情了。

从上面不难看出，想要学好计算机这个学科，其实在思维上要特别适应并且擅长“程序化流程思考”的模式，很多人常问学好计算机是不是需要很好的数学，答案当然是肯定的，不过和实分析复分析代数拓扑泛函分析这些数学系的“数学”不同，计算机专业学生需要特别拥有的数学思维能力，是偏组合的、偏概率的、面向某一个具体问题的流程的能力，习惯于从算法程序的角度思考、分析问题。

线性代数、离散数学、概率论、组合数学是计算机学生的数学基础核心，而面向对象的体系化程序设计，学会分析和操作一个极高复杂性的系统（包括装环境、de一些谜之bug），并忍受该过程中的繁难、没有线索、牵一发而动全身的复杂性，是成为一个好的计算机专业的学生必备的基本素质，面对全新的知识和事物能够迅速把握关键、迁移自己已有的知识快速学习，也是计算机专业的学生必须具备的能力。

在这个角度上，我认为大学之前从来没有写过代码的同学，最好还是相对慎重的选择计算机专业；如果选择了，则程序设计、面向对象、数据结构、算法设计、软件工程这之类的代码基础课一定要把基础打的极其扎实，否则一定会苦不堪言，将来在具体岗位上的可替代性也会很大。

总之，在计算机的专业课程方面，我的建议是：**一定要动手做**。对计算机学科内容充分的掌握=熟练理论知识+自己动手写（改）过一遍代码+de过一遍bug+最终跑通。

**这个过程一定要独立、千万不要耍滑，可以寻求指点，但一定不要去找大佬直接复制他们的代码或者结果，一定要自己结结实实吃下所有shit并最终趟过去、完全理解自己写了什么**，只需要经过几次这样的艰苦流程之后，基本就会在代码能力和专业素养上有质的提升，渐渐上道，很多在大学之前完全没有编程经验的同学度过了这一难关之后，和那些一直参加编程竞赛的大佬的专业水平差距小到基本可以忽略不计。

一个小技巧和建议是，国外的很多大学（如四大：MIT, CMU, Berkeley，Stanford）有质量很高的网课和材料，国内的许多大学课程抄的邯郸学步、深入深出、讲的云里雾里，如果有时间可以去把英文原版的课程啃透，对于整体提升自己的理解“内功”有奇效。

# 专业前景

计算机行业的出路大体上其实就两条路，一条是学术，一条是就业。前者的主要去向是大学高校，角色是大学老师，而后者则是进入企业（俗称大厂）成为一名码农。但这一点也不绝对，也有一些不是教职、但在工业界从事科研的人，像各种大厂包括微软的研究院，其工作核心更接近于学术的创新、而非工业界的开发和“出活儿”。

以下，我将简要介绍这两大出路的工作内容、选拔侧重、优劣分析。

## 学术道路

学术这条路主要的归宿是成为一名大学老师，少数是在某些有技术研发的公司中成为研究员。

想要走上这条路基本上都得读博士，并在博士期间积累论文产出，想要获得比较好的教职坑位往往还需要出国读博士后（进入工业界科研相对而言不太需要）。

大学教授的主要任务就是科研，其次是承接一些重要项目以及教学。在刚毕业入职时基本都是“助理教授”/“青年教师”，需要在最长六年的时间内在科研、国家项目、博士生培养、教学、社会服务多个方面达到学校的考核要求，成为“副教授”的终身教职，俗称tenure或“上岸”。这个过程是比较辛苦和艰辛的，在某些大学里更是招募一大群助理教授内卷争夺上岸席位。

学术对于工作的创新性要求很高，你永远需要追逐在某一个领域的技术和科学发展的前沿，提出全新的方法、做出前所未有的工作。换言之，学术研究的过程里永远在探索未知，没有人告诉你该怎么做，也没有任何人可以保证现在采取的方法路径是否能够有成果。这是学术迷人却又痛苦的地方。

所以，如果想要走学术这条道路，必须有强烈的intellectual interest，对于某类问题的研究很感兴趣；百折不挠，不会轻易被失败的探索打倒；能够忍受学术天然的judge属性，成天被审稿人指指点点、质疑、不理解；善于系统性规划时间，并能够对于某一个特定问题深入且不懈的挖掘探索。

这条路的优势在于，如果你对于自己研究的内容拥有浓厚的兴趣，那么相较于打工而言，研究全新的技术、拓展知识边界时工作的意义感会更强，而且学术界的一个重要好处是每个人都可以完全获得自己工作的credit（可以理解为著作权，这一点在公司中则非常不同）。

此外，教职相对于工业界大厂而言更为稳定、好的大学往往还能解决住房和子女教育问题（上岸后）。

劣势在于基本收入相对而言较为清贫，相比业界的收入要少很多，想在大城市维持生活基本都要靠项目等副业（而这件事高度取决于学校资源的好坏）；其次，科研教学社工等压力非常非常大，而且学术岗的坑位相对而言比较少、竞争激烈，且存在不能上岸的风险。

## 大厂打工

走上学术之路的毕竟还是少数，绝大多数人可能还是想要进入大厂（如腾讯、阿里、字节、微软等）成为程序员。

这里需要首先说的是，“码农”或“程序员”的称呼是一种太过概括的统称，其内部具体的细分工作性质仍然有很多不同，例如**“开发岗”**要做的事情就是开发大型的软件系统、服务维护、成套的解决方案；**“研发岗”**负责研发新的技术路径；**“安全岗”**负责的就是筛查网络和系统漏洞，应对潜在威胁；**“算法岗”**负责优化整个公司产品的服务逻辑、推荐系统、匹配调度等；**“前端”**负责设计交互页面、优化用户体验；**“后端”**则涉及到应用程序的实际逻辑，需要处理并发、数据库维护等等问题。

在大厂工业界打工，其实说白了要的就是一个“业务熟练”。每一个具体的岗位会有不同的知识和技术的侧重，其实日常涉及到的东西也不算太多，但是作为一名搬砖的码农一定要做到足够熟练。

如果能够稍微多一些更深层的理解（例如作为算法岗的人，除了会python调包调参之外，懂得很多传统机器学习的算法以及整个“机器学习”的底层逻辑），那么就会相对有竞争力的多，其在岗位上的不可替代性也就更强。如果学习能力很强，及时丰富自己的知识和技能树，那么也会更吃香一些。

因此，大厂在招人时，往往也会格外关注基本技术的熟练程度和贯穿这些技术之上的理解深刻性。

在目前互联网整体旺盛的生态下，大厂的薪资还是可观的，一个能力基本过关的985计算机专业的学生，本科应届生拿到一万到两万的月薪不是难事，硕士和博士毕业生也常见能够拿到几十万的年薪。随着在公司工作年数的增加，资历更深的员工如果可以上升到主管的位置，年薪基本上也有接近翻番的提升空间。

当然，缺点也大量存在，一方面是工作压力同样很大，且目前的生态中相当多的公司存在比较严重的加班现象；二是当前形势下，无论是大到某一个产业的公司、还是小到每个人具体所在的职位，市场和人事都存在不稳定性，没有学术的编制那么“铁饭碗”。

有关“程序员35岁被榨干后失业”这一说法，笔者自己作为一名博士生，其实也没有什么发言权。多方打听之后大家的共识观点是，近期大厂的裁员潮主要来自互联网此前过度狂热、垄断内卷的降温，但从其客观体量上来说，其实并非每一个程序员到35岁都要面临“卸磨杀驴”的处境（事实上，任何一个大厂里都有相当数量的35岁以上的员工）。对个体而言，掌握真正的核心技术、提升自己整体的专业能力、永远学习新知识新技术，就是能做到的最好的事情了。

## 对学弟学妹的职业规划建议

对于一名高中生或本科生而言，如果想要走上计算机专业的道路，应该准备些什么？又该在每个阶段做些什么？

首先，对于没有学过编程、中学也没有参加过编程竞赛的同学而言，我认为在本科初期以稳定持续的频率（例如每天几道题）刷一些编程OJ（online judge）以及LeetCode的题目是有益处的，至少可以迅速提升基本的算法思维素养和动手能力，但不宜迷信也不必沉迷，记住其目的是为了补足此前并不习惯的思维模式就行，每道题踏踏实实自己想过、做过、实在不会再看题解，在精不在多。

其次，一定一定要拥有**良好（超强）的自学能力**，因为到了大学绝大部分的课程最终还将是以自学为主，到了工作岗位上，赶鸭子上架、需要现学现用的时候也比比皆是。善于从互联网中搜索好的博客、教程、课程等，迅速学会新知识也是必须的本领。

以下，根据想要选择路径的分化，各有不同的选择去向。

如果想要从事学术，最好在有一定的基础的情况下（大二以上）找到一个本校（或者某些公司研究所）靠谱的实验室课题组，再找到一个靠谱的学长/学姐，开始学习上手科研。在进行具体项目的科研过程中，学习补足基本知识，学会如何阅读论文，掌握基本的技能和实验流程，熟悉科研思考问题的方法。

如果想要和工业界接轨，那么建议到公司（大厂）实习。不同的大厂实习要求的标准不同，基本笔试面试都会问一些基本的数学知识（线性代数、概率论），手写某些数据结构、算法流程或现场给出一些算法题的思路，再有就是跟具体想要实习的岗位对口的一些专业知识，需要事先准备扎实。如果能够找到学长学姐内推自然是最好，不然的话就关注一下群里的实习信息或者自己投递简历。

再往后的路，就只有自己扎扎实实的走了，变数很多、每个人也不尽相同。信息技术这个领域瞬息万变，永远没有一个简单和永恒的最佳规划，保持开放、向前辈沟通、不断学习才是最核心的。

# 专业整体的文化氛围

计算机学科难度高、压力大，能选择进来的学生大多也很聪明很拼命，基本上在各个院校，计算机系都是工科院系内卷的巅峰。进入计算机系（或相关学科）需要有比较好的数理基础和代码思维，更要有强大的抗压能力和心态，能够应对竞争激烈、强手如云、生活学习压力较大的日常状态。

不过因为大多是男生，环境也比较纯粹，大家在一起也很开心，很多院系都会需要计算机系的同学作为核心生产力和帮手，对外的身份还是很吃香的。