

## 1.引言

现实与比尔盖茨的梦想之一的距离越来越近了,即计算机在家家户户的普及率越来越高。在这个科技普及速度相当快的年代,我们多数学生或多或少的有一定的计算机操作基础。如今我进入了计算机专业,这与榜样的作用是分不开的,当然更重要的是兴趣。不过这计算机专业现实情况与外人的认识还是有很大区别的,面对庞大而繁重的数学,渺小的我,还好有老师的指引和同学的鼓励,我当然会坚持下去。现在,网络的速度可以达到几何的速度,而且它的速度还会进一步的增快。看看我们过去十年的数字生活方式和数字工作方式,这意味着这些工具已成为主流。计算机是当今社会发展不可或缺的重要元素,它自问世以来一直走在科技前沿,几乎个领域都离不开计算机,计算机无时无刻不在推动者社会发展。作为计算机专业的学生我们充满了信心与斗志!“计算科学导论”这门课为我们敲开了专业之门,我从中受益匪浅。

## 2.对于计算科学导论的认识与体会

通过学习“计算机科学与技术导论”课程,我对计算机发展史又有了新的认识。例如,20世纪30年代是计算模型取得突破进展的时期,哥德尔、丘奇、图灵、波斯特等人分别有了建树,为计算科学技术奠定了基础。1966年美国还设立了计算科学大奖图灵奖,以纪念这位杰出的科学巨匠。图灵和冯诺伊曼贡献了存储式通用电子计算机,人类使用自动计算装置代替人的人工计算

机和手工劳动的梦想成为现实。在此基础上，才吸引了大批人才开展对计算机的研究，这为后来的比尔盖茨成为传奇人物在一定程度上奠定了基础。

学习这门课程之前，一直以为除了发明者，只有比尔盖茨才是对计算机产业贡献最大的人。后来才知道，原来图灵、冯诺伊曼和乔布斯等人在计算机发展史上也是有不可磨灭的重要地位的。在计算机科学导论课上，我们还了解了大名鼎鼎、耳熟能详但以前没有接触过的二进制，怀着崇高的敬意，我发现二进制还是比较有趣的。计算机的世界永远都那么吸引人。这些基础中的基础，我们必然要熟练掌握的。

21 世纪，进入了知识爆炸的年代。计算机领域也一样，技术更新迭代的速度如同火箭的速度一样。新的技术框架和组件层出不穷。但是，万变不离其宗，掌握扎实的基础，面对技术浪涌，也能应对自如！

## **2.1 基础的重要性**

从世界范围内来看，知名院校计算机学院都会开设上述基础课程，例如：国内的清华、北大，美国的 MIT、伯克利、斯坦福、CMU。这些课程开设的目的就是夯实理论基础，才能构建宏伟建筑。罗马不是一天建成的，但是没有基础，罗马不可能建成的！

快速学习能力的提升依赖于扎实的计算机基础。这是一个需要终身学习的年代，不仅要关注学习本身，还得关注学习效率的

提升。基础夯实，意味着面对技术潮涌，也能应对自如。金融领域的技术，技术难点之一就是实现数据一致性，数据一致性依赖于事务，最基础的一致性模型就是分布式事务，而分布式事务的基础是本地事务，《数据库原理》就提供了本地事务的基本原理。如果理解了本地事务原理，学习分布式事务理论就事半功倍。在《操作系统》中，有一个最基本的模型：生产者-消费者，而消息队列的实现本质上也是生产者-消费者模型。生产者-消费者模型实现了空间和时间的解耦，应用场景广泛，例如：异步解耦、削峰、数据一致性、数据通道。如果掌握了扎实的数学基础，要转型算法工程师、从事数据挖掘、机器学习等行业，面对眼花缭乱的数学公式，也可从容不迫！其实，各行各业都一样，根基稳固了，学习能力自然就提升了！

计算机基础是否扎实决定了职业生涯能走多远！

## 2.2 思维的重要性

程序员的圈子总是有一种焦虑感，程序员是年轻人的天下、35岁达到职业瓶颈期。如果花5年甚至10年重复得干CRUD工作，每天忙碌于解决线上问题、修复Bug，35岁能不到达职业瓶颈？为什么计算机基础扎实了，有助于职业生涯走得更远？这得从计算机思维说起。基础不仅仅带来的是理论知识，更重要的是带来了思维方式的提升。其中最重要的两个思维是：自动化思维和抽象思维。

计算机是按照指令顺序自动执行，自动化思维的启示是解放

双手、关注核心的项目，能自动化的事情就应该自动化。计算机领域层有出不穷的自动化工具，充分利用自动化工具，解放双手，就解放了焦虑，做更有意义的事情，才能有更大的成长空间。

抽象思维，解决复杂问题的利器！为什么有些人是行业专家，指点江山？为什么有人是架构师，高屋建瓴？为什么有些人只是辛勤的搬砖族？本质的区别在于抽象思维。行业专家制定行业标准，例如网络原理的七层模型，构建该模型首先得具备抽象能力，分层的边界是什么？每一层具体的职责是什么？各层之间如何交互。架构师解决系统架构问题，就得摸清系统的边界？系统的职责是什么？系统之间如何通信？应用哪些技术组件？这些都需要抽象思维，站得足够高，才能望得足够远。而计算机基础可以训练抽象思维，是否具备抽象思维决定了你是陷入忙碌的死循环还是做更有意义的事情，也同样决定了你是搬砖族还是高屋建瓴者！

### 3.进一步的思考

在课程学习的过程中，我们还进行了一些深入的了解，我与搭档一同对“拜占庭将军问题”进行了研究，参考国内文献了解到，拜占庭位于如今的土耳其的伊斯坦布尔，是古代东罗马帝国的首都。拜占庭罗马帝国国土辽阔，为了达到防御目的，每块封底都驻扎着一支由将军统领的军队，每个军队都分隔很远，将军与将军之间只能靠信差传递消息。在战争的时候，拜占庭军队内所有将军必须达成一致的共识，决定是否有赢的机会才去攻打敌

人的阵营。但是，在军队内有可能存有叛徒和敌军的间谍，左右将军们的决定影响将军们达成一致的共识。在已知有将军是叛徒的情况下，其余忠诚的将军如何达成一致协议的问题，这就是拜占庭将军问题。应该明确的是，拜占庭将军问题中并不去考虑通信兵是否会被截获或无法传递信息等问题，即消息传递的信道绝无问题。我还参考了国外期刊，了解到 Byzantine failures, a reliable computer system must be able to cope with the failure of one or more of its components. A failed component may exhibit a type of behavior that is often overlooked- -namely, sending conflicting information to different parts of the system. The problem of coping with this type of failure is expressed abstractly as the Byzantine Generals Problem. We devote the major part of the paper to a discussion of this abstract problem and conclude by indicating how our solutions can be used in implementing a reliable computer system. 该问题的本质则是区块链技术，而为了解决这一问题，出现了拜占庭容错，但这并不是某一个具体算法，而是能够抵抗拜占庭将军问题导致的一系列失利的系统特点。这意味着即使某些节点出现缺点或恶意行为，拜占庭容错系统也能够继续运转。本质上来说，拜占庭容错方案就是少数服从多数。拜占庭将军问题的原始论文给出了一些解决思路，但其更注重理论上的可行性。算法效率不高，算法复杂度为指数级，且文中明确指出时间成本及消息传递数量很大。因此

不具备太大的实用价值。

所以在 1999 年, 麻省理工学院的 Miguel Castro 和 Barbara Liskov 提出了实用拜占庭容错, 称为 PBFT, 对于 PBFT 算法, 因为 PBFT 算法的除了需要支持容错故障节点 之外, 还需要支持容错作恶节点。假设集群节点数为  $n$ , 有问题 的节点为  $f$ 。有问题的节点中, 可以既是故障节点, 也可以是作 恶节点, 或者只是故障节点或者只是作恶节点。那么会产生以下 两种情况: 第一种情况,  $f$  个有问题节点既是故障节点, 又是作恶节点, 那么根据少数服从多数的原则, 集群里正常节点只需要比  $f$  个节 点再多一个节点, 即  $f+1$  个节点, 正确节点的数量就会比故障节 点数量多, 那么集群就能达成共识。也就是说这种情况支持的最 大容错节点数量是  $(n-1)/2$ 。第二种情况, 故障节点和作恶节点都是不同的节点。那么就 会有  $f$  个故障节点和  $f$  个作恶节点, 当发现节点是故障节点后, 会被集群排除在外, 剩下  $f$  个作恶节点, 那么根据少数服从多数 的原则, 集群里正常节点只需要比  $f$  个节点再多一个节点, 即  $f+1$  个节点, 正确节点的数量就会比作恶节点数量多, 那么集群 就能达成共识。所以, 所有类型的节点数量加起来就是  $f+1$  个正 确节点,  $f$  个故障节点和  $f$  个作恶节点, 即  $3f+1=n$ ,  $f=(n-1)/3$ 。举个例子, 我是一个愚昧的国王, 没有自己的判断力, 我不 知道应该对敌国进攻还是投降好。我有一些大臣, 我希望听从他 们的意见做出决定, 但是他们现在都离我很远, 我只能通过飞鸽 传书

的方式告知他们目前的问题，得到他们的选择。然而，可能出现大臣叛变，故意提出相反的观点（作恶节点），也可能出现鸽子在传输过程中飞错了，我没有得到该大臣的选择（网络堵塞）。PBFT 可以保证如果我有  $3f+1$  的大臣的话，即使其中有  $f$  个大臣叛变或者没有响应，我依然可以得出共识的正确结果。这里一个核心问题就是，为什么有  $f$  个节点未响应或出错时，为了保证系统的正常，需要总共有  $3f+1$  个节点进行投票。同样用国王的例子，假设除了我（国王）一共有  $n$  个大臣，我知道其中有  $f$  个大臣是叛徒或者未响应，所以我一定要能从  $n-f$  个大臣的回应中进行判断。然而由于是飞鸽传书，所以当我陆续收到  $n-f$  个传来的消息后，我并不知道之后是否还会有新的消息传来。因为如果  $f$  个有问题的大臣都是未响应，那么我将不会收到新的消息，如果其中有大臣是叛徒，我之后还会收到消息，但作为国王的现在不知道是哪种情况，却需要立刻作出进攻还是投降的判断。最坏的情况下，剩下的  $f$  个大臣都是好人，只是鸽子飞得慢我还没收到消息，也就是说我收到消息的  $n-f$  的大臣中有  $f$  个大臣都是叛徒，即  $f$  个叛徒和  $n-2f$  个好人。由于多数者胜，所以只有当  $n-2f > f$  的情况下，作为国王的我会做出正确的决定，即  $n > 3f$ ， $n$  最小需要取  $3f+1$ 。pbft 算法的基本流程主要有以下四步：

- （1）客户端发送请求给主节点；
- （2）主节点广播请求给其它节点，节点执行 pbft 算法的三阶段共识流程。
- （3）节点处理完三阶段流程后，返回消息给客

户端。 (4) 客户端收到来自  $f+1$  个节点的相同消息后, 代表共识已经正确完成。为什么收到  $f+1$  个节点的相同消息后就代表共识已经正确完成? 从上面的介绍里可知, 无论是最好的情况还是最坏的情况, 如果客户端收到  $f+1$  个节点的相同消息, 那么就代表有足够多的 正确节点已全部达成共识并处理完毕了。算法的核心三个阶段分别是 pre-prepare 阶段 (预准备阶段), prepare 阶段 (准备阶段), commit 阶段 (提交阶段)。图中的 C 代表客户端, 0, 1, 2, 3 代表节点的编号, 打叉的 3 代表可能是 故障节点或者是问题节点, 这里表现的行为就是对其它节点的请求无响应。0 是主节点。 Pre-prepare 阶段: 节点收到 pre-prepare 消息后, 会有两种选择, 一种是接受, 一种是不接受。什么时候才不接受主节点发来的 pre-prepare 消息呢? 一种典型的情况就是如果一个节点接受到 了一条 pre-pre 消息, 消息里的  $v$  和  $n$  在之前收到里的消息是曾经出现过的, 但是  $d$  和  $m$  却和之前的消息不一致, 或者请求编号 不在高低水位之间 (高低水位的概念在下面会进行解释), 这时候就会拒绝请求。拒绝的逻辑就是主节点不会发送两条具有相同的  $v$  和  $n$ , 但  $d$  和  $m$  却不同的消息。 Prepare 阶段: 节点同意请求后会向其它节点发送 prepare 消息。这里要注意一点, 同一时刻不是只有一个节点在进行这个过程, 可能有  $n$  个节点也在进行这个过程。因此节点是有可能收到 其它节点发送的 prepare 消息的。在一定时间范围内, 一个节点



如果收到  $2f$  个不同节点的 prepare 消息, 就代表 prepare 阶段已经完成。Commit 阶段: 于是进入 commit 阶段。向其它节点广播 commit 消息, 同理, 这个过程可能是有  $n$  个节点也在进行的。因此可能会收到其它节点发过来的 commit 消息, 当收到  $2f+1$  条 commit 消息后 (包括该节点本身), 代表大多数节点已经进入 commit 阶段, 这一阶段已经达成共识, 于是节点就会执行请求, 写入数据。

之后我上网络进行查询, 了解到对 PBFT 算法也已经有了改进, 该改进提出了适用于联盟链环境的共识算法, IPBFT。针对 PBFT 算法中通信复杂度高的问题, 在 IPBFT 中, 引入了协调节点, 将原来两节点之间互换消息变为了节点先将消息发给协调节点进行验证, 协调节点验证通过后再发给其他节点, 从而减少了通信量; 针对原来算法扩展性差的问题, 新算法引入了节点加入、退出、恢复协议, 使得节点可以自由地出入网络; 针对原来算法中错误节点反复作恶的情况, 新算法还增加了错误节点清除协议和信誉积分机制, 以及时剔除错误节点, 提高系统稳定性。因此, 相对于 PBFT, 改进的算法有以下优势: (a) 扩展性好, 通过对节点加入或退出进行共识验证, 使得节点可以动态地参与到网络中。 (b) 通信复杂度低, 在每轮共识中, 随机选择一个协调节点参与到共识中, 通过主节点与协调节点的配合, 可将通信复杂度大大降低。 (c) 系统稳定性高, 新算法引入了信誉积分机制和错误节点清除协议, 适时的清除了错误节点, 避免了错误节点二次作恶。最后, 本文设

计了简易的区块链系统,并对 IPBFT 算法的性能和功能进行了测试。实验结果也同样表明,IPBFT 算法在通信复杂度,延迟等方面都优于 PBFT 算法,也更适合区块链环境。

在随后的时间里,中本聪提出“比特币”,并在比特币白皮书中,通过“比特币协议”给出了终极解决方案 (1) 引入“工作量证明”机制 (PoW), 只有第一个完成规定计算工作的国家才能传播信息, 从而保证一段时间内只有一个提案。工作量证明 (Proof-of-Work, PoW) 是一种对应服务与资源滥用、或是阻断服务攻击的经济对策。一般是要求用户进行一些耗时适当的复杂运算, 并且答案能被服务方快速验算, 以此耗用的时间、设备与能源做为担保成本, 以确保服务与资源是被真正的需求所使用。PoW 算法来解决拜占庭问题的思路有两条: 一、限制一段时间内网络中出现提案的数量。也就是说每个节点在提出提案之前需要运行哈希现金 (HashCash) 的算法, 即需要在一个交易块中找到一个随机数, 计算机只能通过穷举法来找到这个随机数。同时中本聪设计了时间戳的机制, 为每个节点的出块时间加上印章来保证公平性。二、弱化对于一致性确认的需求。在比特币网络中各个节点都确定验证已知的最长链进行延伸。由于 POW 机制弱化了网络对于一致性 (Consistency) 的需求, 所以同一时刻访问比特币的不同节点, 得到的内容并不一致, 甚至网络会出现分叉的情况。但是整个网络的可用性 (Availability) 和分区容忍性 (Partition) 都得到了提升。工作量证明, 简单的理

解就是一份证明，现实中的毕业证、驾驶证都属于工作量证明，它用以检验结果的方式证明你过去所做过了多少工作，在拜占庭的系统里，加入工作量证明，其实就是简单粗暴地引入了一个条件：大家都别忙着发起消息，都来做道题，看谁最聪明，谁就有资格第一个发起消息。如果不同的将军先后解出了题，各自先后向这个网络发布消息，于是各个节点都会收到来自不同节点发起的进攻或者不进攻的消息，那怎么办的？只有时间最早的发起者才是有效的。中本聪巧妙的设计了一个时间戳的东西，为每个将军在解好题的时间（出块时间）盖上时间印章。将军们那又凭为什么要一起做工作量证明呢？中本聪也完全可以设置一个奖励机制，比特币的奖励机制是每打包一个块，目前是奖励 25 个比特币，当然，拜占庭将军问题的奖励机制可以是瓜分拜占庭获得的利益。（2）引入非对称加密算法，为信息传递提供签名技术支持，以保证消息传递的私密性，且不可抵赖、不可篡改。非对称加密算法的加密和解密使用不同的两个密钥。这两个密钥就是我们经常听到的“公开密钥”(公钥)和“私有密钥”(私钥)，公钥和私钥一般成对出现，如果消息使用公钥加密，那么需要该公钥对应的私钥才能解密；同样，如果消息使用私钥加密，那么需要该私钥对应的公钥才能解密，非对称加密的作用是保护消息内容，并且让消息接收方确定发送方的身份，比如，将军 A 想给将军 B 发送消息，为防止消息泄露，将军 A 只需要使用 B 的公钥对信息加密，而 B 的公钥是公开的，B 只需要用只有他自己有的私钥解密

即可。将军 B 想要在信件上声明自己的身份，他可以自己写一段“签名文本”，并用私钥签名，并广播出去，所有人可以根据 B 的公钥来验证该签名，确定的 B 的身份。由此，一个不可信的分布式网络变成了一个可信的网络，所有的参与者可以在某件事上达成一致。这就是拜占庭将军问题的两个具体解决方法。

#### 4.总结

目前我们正在学习或将要学习的各门课程，对我们今后的专业学习都有重要意义，我们要注重全面均衡发展。自计算机诞生以来，大量的人才前赴后继的投入计算机研究领域，举世瞩目的成果不断被推出，计算机的发展史告诉我们，在学好专业知识的基础上，我们要注重培养创新思维，努力做创新型人才。老师在课堂上通过对计算科学学科的定义、基本问题、发展主线、主流方向、学科方法论、历史渊源、学科特点、发展变化、知识组织结构与分类体系、学科发展潮流与未来发展方向等学科发展历程和学科范型理论知识的介绍，使我们对计算科学学科有了一个正确、初步的认识和了解。虽然我们目前对许多知识不能深入理解或根本不能理解，但也不影响我们对本学科整体上形成初步的认知。学习计算机专业要掌握坚实的基础知识，这是实践与创新的前提。二进制、布尔代数等等，以后的专业课程中我们将会用到，所以一开始就要有足够的重视，以后更要认真学习。

“计算科学导论”这门课是我们专业学习的敲门砖，通过一段时间的学习，我简单初步认识了计算机科学与技术，对以后的学习有

启发作用，受益匪浅。在今后的学习中我会明确目标，努力走好这条路。

路漫漫其修远兮，吾将上下而求索！

## 5. 参考文献

[1]赵致琢，《计算科学导论（第三版）》，科学出版社，2008.

[2] jackben, 简书; [WWW.hn1c.com](http://WWW.hn1c.com).


[3]孙耀景. 基于实用拜占庭容错算法的区块链共识算法研究[D]. 湘潭大学,2020.


[4]Leslie Lamport,Robert Shostak,Marshall Pease, The Byzantine Generals Problem, Volume 4, Issue 3. 1982. PP 382-401.

[5][1]高丽芬,胡全贵.区块链共识机制之拜占庭算法[J].数字通信世界,2019(01):43+49.

## 附录

Github 账号 : Amaury093 ; 网址 :  
<https://github.com/Amaury093>;

 Search or jump to... Pull requests Issues Marketplace Explore



浮云里

Amaury093

Edit profile

Joined 13 days ago

OverviewRepositories1ProjectsPackages

Popular repositories

LYBB

3 contributions in the last year

JanFebMarAprMayJunJulAugSepOctNovDec

Mon

Wed

Fri

Learn how we count contributions.

LessMore

Contribution activity

2021


January 20212020


Amaury093 has no activity yet for this period.

Show more activity

Seeing something unexpected? Take a look at the GitHub profile guide.

哔哩哔哩





0

粉丝

127

关注

0

获赞

编辑资料

FYLXYQ

大会员

勋章 >

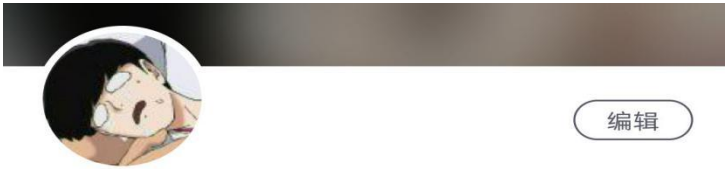
LV4

8160/10800

这个人很神秘，什么都没有写

详情

CSDN



编辑

浮云里 ID

码龄89天

此人很懒，什么都没有写



博客排名 -- 访问 0 粉丝 0

学习强国

观察者

奕霖

付奕霖

段位:积分不足，暂无段位

收藏

订阅

历史

2

积分

我的



浮云里 观察员



0  
关注

0  
粉丝

0  
文章

博客园账户：浮云里；网址：<https://home.cnblogs.com/>；

[园子](#) [关注](#) [粉丝](#) [随便看看](#) [消息](#)

博客园

欢迎你，浮云里

 浮云里 · [我的博客](#) · [设置](#) · [退出](#)

闪存

博客

小组

新闻

博问

收藏

招聘

文库

问题反馈

写博

发言

投递

提问

添加

发布



浮云里

[编辑个人资料](#)

你在做什么？你在想什么？

☒ 公开

☐ 私有

[加标签](#)

发布

我的关注: 0  
我的粉丝: 0

[» 申请博客](#)

园友搜索

搜索

[我关注的人\(0\)](#)  
[我的粉丝\(0\)](#)

- 最新动态

我的

全站

刷新

全部
- 

[-小马](#) 发表博客: [装饰者模式](#) 2021-01-07 17:03

今天来介绍装饰者模式（Decorator Design Pattern）。假设我们需要给一家火锅店设计一套结账系统，也就是统计顾客消费的总价格。怎样才能设计出一个好的系统呢？1，结账系统需求分析 既然要设计一个结账系统，当然需要知道火锅店都有哪些食品及食品的价格，假如我们从火锅店老板那里得到以下
- 

[colazcy](#) 发表博客: [题解 P3321 【\[SDOI2015\]序列统计】](#) 2021-01-07 17:03

题目链接 Solution [SDOI2015]序列统计 题目大意：给定一个集合  $\setminus(S)$ ，求生成一个长为  $\setminus(n)$ ，每个元素属于  $\setminus(S)$ ，所有元素积  $\setminus(mod\setminus m=x)$  的数列的方案数。 $\setminus(m)$  为质数。原根，NTT 分析：首先我们想办法把乘法变为加法，这样我们就可以用卷积来计
- 

[小菜qi](#) 发表博客: [Python运算符和数据类型](#) 2021-01-07 17:03
- 

[初衷的味道](#) 发表博客: [个人总结——磨刀不误砍柴工](#) 2021-01-07 17:03

回望自己在软件工程实践中走过的路：1.做了哪些作业： 个人作业： 第一次作业 第一次个人

小木虫账户：浮云里；网址：  
<http://muchong.com/bbs/space.php?uid=24919720;>



小木虫论坛-学术科研互动平台 » 我的主页

个人首页	主题	草稿箱	订阅	提醒	听众	收藏	淘贴	相册	私密空间	钱包	金币荣誉
------	----	-----	----	----	----	----	----	----	------	----	------



浮云里

/bbs/space.php?uid=24919720

个人设置面板

金币: 0

组别: 新虫 注册: 2021-01-05 23:05:39 虫号:24919720 听众:0 红花:0 VIP:0 帖子:0

撰写主题

浮云里 基本资料					
注册时间	2021-01-05 23:05:39	最后活跃	2021-01-07 17:15:27	最后发表	x
身份与荣誉					
虫号	24919720	用户组 (金币)	新虫	应助	0
贵宾	0	金币	0	散金	0
沙发	0	帖子	0	管辖	
在线时间		在线状态	在线	专业	
性别	0	来自		生日	0000-00-00