<https://www.1point3acres.com/bbs/thread-678970-1-1.html>

天下刷题方法千千万万，每个人的资质、特点、喜好都不同，所以大可选择适合自己的方法，不必刻意拘泥于某一种形式，只要对你有效果，就是好方法。

神有神的天赋，霸有霸的套路，对于学弱和咸鱼，也可以有帮助自己成长翻身的方法。

本篇仅是一名转专业学弱总结的算法题刷题方法，也许看起来不那么聪明，但对于没有基础的小白来说，可能也是一种启发和参考。当然如果你是大佬，那自然可以略过。

# 第一章    一些“方法论”

在展开具体的干货前，先唠叨一些看起来没用的“废话”。

## “废话”1：几条需要着重强调的学习要点

系统性的分类学习

先学会看懂答案

注意代码风格的养成

反复训练

详细的笔记

提升讲题能力

任务分配管理

### 系统性的分类学习：

对于计算机基础不扎实的同学，如果按照高频顺序刷三百题，通过每道题反向获取知识，这样的方法不是不可以，但效率可能不高，还会造成知识零散化，不能融会贯通。另外，松散地做300道题，对于我等学弱而言，从执行的层面而言也略显艰难。

面试算法体主要就十几类题型，套路性比较强，经过前人的总结，已经形成了比较完善的体系（300~400题左右）。即便出现很多新题（1600+），大多也能用现有知识体系去化解。

所以不如站在前人的肩膀上，通过中国学生更熟悉的“分类知识学习+核心题目+反复训练+进阶题目+复习升华+融会贯通”的系统性方法进行学习，不但基础更为牢固，而且遇到新题也有现场化解的可能。

在后面的章节里，胖头龙可以分享自己总结的，面试中常见的算法类别，大致介绍每一类需要自学掌握的要点，并附加上题目列表。

（当然大部分培训机构都有，我只是作为小白分享一点自己的心得。）

### 学会看懂答案：

作为一条咸鱼，在刷透300题之前，不要试图靠自己平庸的大脑跟任何题目死磕。毕竟很多算法问题的最优解，放在几十或上百年前也都是人类知识边缘的数学问题，当年大师们也花了不少功夫才研究出来发论文的，如果被我等弱鸡几十分钟就凭空搞定，大师们不要面子的吗？

所以，对于初学者，一道题十分钟没有想法就不要开脑洞了，而应该直接去寻找标准答案，以最快的速度爬到前人的肩膀上。

首先可以看LeetCode的标准答案，大部分解释的很清楚。你可以先看懂最优解，然后用自己的代码风格写一遍（答案上的代码风格参差不齐，不要被带偏），并记入笔记。如果觉得还是不清楚，那么可以直接在Google上搜题目，CSDN，博客里会出来各种大神的中文附图讲解，更方便理解。但如果一道题，你研究了一个小时还没完全领悟答案的思路（对于新手这很正常，不慌），可以先把这题暂时标注在你的记录上，过几天再回来看，不用浪费时间死磕。

注意，看答案的目的不是为了让这题混个pass，而是要理解答案中的思路，并逐渐积累形成自己的思维体系。就好比课本里的例题答案，是帮助你更好地去理解基本理论知识的运用方法，课后习题的答案，是为了方便你在做题中自我校正提升。如果你看答案只是稀里糊涂抄上去混个pass，那就没什么意义了。虽然我们是咸鱼，但并不一定要做渣，而是要做一条对自己负责任的咸鱼。

所以，不要为抄答案而羞愧，有成长的抄答案，是一种高效学习的方式。

### 注意代码风格的养成：

这是很多新手会忽视，也不容易掌握的点。

大多数时候我们仅仅满足于LC测试通过。但是在面试中，代码风格会直接暴露出你的实际项目经验。虽然面试更关注的是算法本身，但是如果你能写的一手正确而又整洁的代码，那绝对是加分项。我听过周围很多朋友抱怨新人同事写的便便一般的代码，可见做一个代码不整洁的人，会多么令人一言难尽。

但大部分人的整洁代码，是被大公司里严格的CR训练出来的。对于初学者怎么办呢？

首先，我们在学习变成语言的时候，就可以同时读一读这一语言的官方代码规范，虽然不可能每条都记住，但看几遍多多少少有印象，自己写到相关内容的时候，自然会想起来一些。

其次，很多IDE都自带Lint功能，平时字节写project的时候，按照lint要求把warning都修了，基本也就差不多了。自己总结的标准答案，也可以复制粘贴进去看看有没有什么问题。

最后，参考网上答案的时候要小心，很多人的答案看起来很短很酷，但是在现实中却是很糟的代码风格，或者是隐形增加了时空复杂度，所以需要有选择的汲取。一旦形成自己的代码模板，就坚持下去。

### 反复训练：

很多同学会说几百题刷了一两遍，但似乎依旧没有什么印象。

但是，作为一条咸鱼，你刷了一两遍就把那么多题吃透了，这个，题它不要面子的吗？它们也是花了七八年才增长到上千道的，被你两刷就搞定了颜面何在？

想想当年复习托福，GRE，那些单词你背两遍就都记住了吗？况且算法题比单词复杂多了吧。

我们不是神，并且同学们大多都是二十多岁的中老年人，不应该太苛求自己的记忆力，刷一两遍题肯定是要忘的，这一点不要惊慌。

我等学弱虽然脑子慢一些，记忆力差一些，但是多来几轮还是能补上的。而且这里的反复训练中的很多轮是在回顾思路，不是反复重写，所以一道题几分钟就能过一遍，并不是那么费时间。

总之，如果一道题你刷了两三遍以下还没记住，那说明你是正常人，不要灰心，不要怀疑，再来几遍就好。

### 详细的笔记：

很多同学每道题写完就算了，不总结也不记录。

刚才说到需要反复训练，但刷一遍题也挺费脑细胞和卡路里的，难道不应该留下点什么吗？你的大脑也许确实带不走一丝云彩，但可以试图留下点来过的痕迹。

特别是找到并形成一道题的标准答案后，在笔记上写下主要思路和要点，对于日后复习有神奇的作用。

另外，一个好的笔记体系，能够帮你构建一个完整的知识体系，养成这个习惯，会有意想不到的效果。

其次，很多人同学觉得刷题像考托福GRE申请学校一样，只是一次性的投入。但实际上，目前互联网程序员平均在一家公司的在职时间还不到3年（包括你自己跳槽以及公司逼你跳槽），这意味着在你职业生涯的前十年，大概需要找三四次工作。在北美的就业体系内，很多人最终也不会走向管理岗位（一言难尽。。。），所以即便你有五到十年工作经验，只要你面试的岗位还是一个Individual Contributor，在面试当中很大可能还是逃不过算法题。

不过面试算法题的大致类型和范围相对比较固定，比如之前七八年虽然LC题目从100增长到1600+，但是主要知识点还是那么多，只是题目变难了，要求变高了。因此你现在总结的笔记，在接下来十年内，都是有用的，而且会被用很多次。

所以一份详细的笔记，不仅对你这次找工作有益，对你未来很多年都有用。

### 提升讲题能力：

很多同学面试题在LC上刷得很熟练很顺利，但是在面试白板/文档上却写不完讲不利索。

不幸的是，面试中的表达是一个很重要的环节，直接影响面试官对你的判断。特别是疫情以后基本都是线上面试，比现场面试更难用手比划，所以相比于仅仅刷题，一定要着重提高对自己讲题能力的训练。虽然你不可能总是找别人帮你mock interview，但在自己平时在做题之后，对着小黄鸭自言自语，复述解题思路是非常重要的。

除了说，学会在共享文档里画图举例子，走case，对于把问题讲清楚有着极其美妙的效果。

能不能把题讲明白和能不能把题做出来同等重要。

### 任务分配管理：

作为一个学弱，看到两三百道题的任务量，大约是颤抖并退缩的。特别是体会过那种刷过一两遍，脑海中却依然没有半点痕迹的感觉的人。

但是我们可以把这些学习任务，细化出来，制定成相对具体的学习计划，分配到每周甚至每天。

每完成一个任务点之后，都应该及时进行标注并可以对自己进行小小的鼓励和奖励。

这也是自我管理的技能，很多人在职场上工作过一些年可能都没有意识到，虽然总觉得老板安排的任务似乎都完成了，但其实在老板眼中你可能是个做事没有条理没有计划让领导不完全放心的员工。

其实如果你养成了合理安排工作计划，并按计划执行方案的习惯，这在实际工作中将是一个加分项。不管是老板还是同事，都希望自己的组员是个有条理的人。因为在公司里，良好的执行力是实现各种炫酷idea的必要条件，而合理的任务计划和进度管理又是团队执行力的基础。

另外，能够管理好自己也是未来能够管理好别人的重要前提。

在后面的章节里，作者会分享一些自己具体的刷题计划日志，供参考。

## “废话”2：基本的学习过程（是否需要培训课程）？

在全民转行敲代码风行了这么多年的今天，网上各类求职辅导课程自然是。。。和南湾的乌鸦一样多。

对于有一定学习能力的人来说，根据网上免费的信息，自己刷题是完全没有问题的。

当然对于零基础的人，选一些合适的课程，站在前任的肩膀上，会节省很多时间，避免一些坑（当然前提是你选的课程它自己不是个坑）。

所以这个问题没有最优答案，需要根据自己的情况权衡选择。

胖头龙因为转专业，不小心上过不少主流、非主流的课程（因为自己太弱，给机构送钱送到钱包哭了），掉过坑也有过收获，所以多多少少有一些自己的心得体会。

当然我也不会对具体机构和课程给出评价，免生是非。

不过从方法论出发，站在求学者的角度，探讨一下什么是一门好课，还是可以的。

对于我个人而言，选择短期培训课程，是希望对于某些需要训练才能提升的问题（算法、系统设计、面向对象设计、基础知识、技术面试套路、行为类问题、项目实战等），能获取整套系统性解决方案，并且这个方案是能够让我在课下时间重复练习以达到持续提升的目的。

### 针对某一类具体问题，一个“好”的，系统性的解决方案应该包括以下几个要点：

【要点-1】： 准备适当的学习材料（武功秘籍）

【要点-2】： 给出原理性解决问题的方法论 （内功心法）

【要点-3】： 给出针对各个具体问题的系统性解决方案/模板 （剑法招式）

【要点-4】： 系统性地指出难点和要点，以及需要避免的易错点和误区 （命门要害）

【要点-5】： 通过具体实例展示如何通过套路解决问题，启发思路，训练思维方式 （演示指导）

【要点-6】： 能够独立反复训练并验证，逐渐提升自己对套路的掌握，对内在原理的领悟，得到相应技能的持续进步（勤学苦练）

其中【要点1~5】是我们希望从课程中获得的，【要点-6】除了要求课程的具有可重复训练，主要靠自己努力。

其实仔细想想，会发现这些要点不仅仅是我们对课程的诉求，也可以是我们自学刷题的整体思路。因此即便我们完全自己学习，也可以按照这些内容进行准备。

以解决算法题面试这个目标为例，稍微展开一下这些要点：

### （1） 准备适当的学习材料（武功秘籍）

如何从海量的信息中提取恰到好处的内容，也是一门技术。

有些课程大而全，会提供很多详细的内容，当然从学习的角度而言，学了并没有坏处。只是面试基本考不到，如果你时间紧，那么就是浪费时间。

有些机构比较懒，连基本材料都懒得给你准备，丢给你一个网站，或者让你自己google。当然他们会说这是培养你自主学习的能力，但是你买不买账，全看自己判断。

不过好在对于算法题而言，目前的学习资源还是比较明确的。

<1> 所选刷题语言的入门材料（大部分人都是 Java 或 Python），不管选哪个语言，常用数据结构的增删查改操作要熟练，基本逻辑、类，要会写，代码风格、规范要掌握。

<2> 十几种常见算法的基本原理和思路。

<3> 题库，一般是 Leetcode 为主，可以偶尔辅助 LintCode。题目范围主要是总表前两三百道高频，按公司分类的高频，和网上的各种面经。

<4> 解题答案，课程应该提供标准答案，自学的话自己总结。

<5> 面试技巧方面的内容和经验。这个对新人来说并不容易获取，多看看帖子，找过来人mock一下效果会比较好。

### （2） 给出原理性解决问题的方法论 （内功心法）

解决算法题的核心思维方式，破解问题的思考步骤，优化取舍的基本判断依据。

一般来说自己领悟到这些需要大量时间的训练，另外大部分机构的课程在这方面做得也一般。他们更多地是讲解每题的具体解法（毕竟这样比较容易），能做到醍醐灌顶，融会贯通的，毕竟只是少数。

### （3）给出针对各个具体问题的系统性解决方案/模板 （剑法招式）

针对题型分类准备相应的模板和一般思维流程，对一些更具体的子问题做一些准备和探讨。

这一点，通过自己钻研也是可以总结出不错的模板的，当然大部分课程也都能做到。

### （4）系统性地指出难点和要点，以及需要避免的易错点和误区 （命门要害）

一些题目的易错点和需要小心处理的点，面试中容易犯错的误区，以及与面试官的沟通要点。

除了熟练做题之外，清楚地讲题也是重点。

### （5）通过具体实例展示如何通过套路解决问题，启发思路，训练思维方式 （演示指导）

老师演示指导面对一道题，如何提问明确面试官需求，如何破题，该往哪个方向思考，然后如何套用模板把问题解决。

教会一个人思考是很难的，所以很多课程更多的局限于知识的传授，而不是启发思考。

另外少部分课程“觉得”他们做到了启发思考，但其实仅限于老师不断的说“我在教你思考”，却没有实际效果。

所以市面上真正能做到“启发思考，传授思维方式”的好课，目前屈指可数。

### （6）能够独立反复训练并验证，逐渐提升自己对套路的掌握，对内在原理的领悟，得到相应技能的持续进步（勤学苦练）

这个比较直观，以上面总结出的方法，练吧。

当然具体训练的时候，也需要分主次，分轻重。

这里刷几百题是有质量区别的，不是说均匀的每题过一两遍，而是按照优先级高低分主次。

### 题目优先级分类

一般来说，在我的个人笔记里，把题目分为四个优先级：

<1> **必背**，大约20~30道，都是各类型题目的典型模板题，基本需要刷十几遍，做到迷迷糊糊半昏迷状态也能熟练默写的肌肉记忆状态。

<2> **核心**，大约100~150道，主要是各种高频题和经典题，基本在5~8遍以上，需要做到最优解，medium难度10分钟以内，hard难度15分钟以内，无错一遍过，同时要能解释清楚思路。另外有多重解法的也要掌握，知道不同解法间的优缺点和trade off原因。

<3> **重点**，大约200~300道，核心题之外的高频题，基本在5遍左右。这样遇到原题或者类似题的时候，基本思路、逻辑不会错。能不能临场完全bug free要看基本功和运气。

<4> **普通**，上述题之外所有你刷过的题。基本上做过一两遍，有个思路，临场遇见了不会慌。

总体而言，在分类学习入门之后，300~400的题量（也就是优先级重点及以上）差不多是可以找工作的（当然有大佬告诉你100道也能找到那也是真的，不过那可是大佬）。

### 刷题力度分类

如何在短时间内刷/过这么多遍也是有技巧的，并不是每次都要手写一遍，并不是每道题都要死缠烂打不会做不罢休，适当的迂回转进，提高整体效率才是目的。

下面是个人对这些基本操作（刷题力度）的定义，比如刚开始刷题的时候初刷、精刷比较多，中期过题、复刷比较多，最后阶段过题和模拟比较多。

具体的分配示例，在后面计划日程的章节有所分享。

#### 初刷：

新手期遇到新题，如果短时间内（五分钟）没有思路没关系，不要死缠烂打，早点看答案。

找到最优的答案并尽力去理解，用自己的面试语言和规范的代码风格进行“抄写”，不完全懂也没关系，标注一下，以后再看。

每题时长控制在半小时之内。

#### 精刷：

针对初刷过还未充分理解的题。

尽量全面理解答案的要义，并且至少在能自己“默写”出来，并调试通过。

将解题思路和形成的标准答案加入到笔记当中。

如果实在不能完全领悟，需要在笔记中用显著符号标注该题，写下困惑的地方，过一段时间再试试看。

每题时长控制在一小时之内。

#### 过题：

按照笔记的标签（比如按分类，或者按优先级）大量复习题目。

针对做过的题进行复习，如果对这一题还有映象，则在大脑中过一遍解题思路及实现中的细节要点。

如不记得，则进行主动思考寻求解法，之后对比之前的答案和笔记，看是否正确。

如果不正确则需要强化思考并记忆，不需要写题。

每题时长控制在5-10分钟内。

#### 复刷：

针对做过的题进行复习。好

自言自语clarify题意，讲请楚自己的思路，写题，检查，跑case，争取bug free一遍过。

只有参考笔记和标准答案复盘自己的错误和不足。

每题时长控制在20分钟内。

#### 模拟：

抽取一道新题，看能不能做到复刷中的效果目标。

把题目记入笔记。

每题时长控制在30分钟内。

回到话题本身，如果你决定准选择一门课，但你在上完课前，并不知道这门课/机构靠不靠谱，怎么尽量避免掉坑呢？

（1）研究一下课程表，至少知道大概要说什么内容，然后看这些内容是否满足我们说的这些要点，另外从课程内容和价格也可以了解性价比，不同课程间可以比一比。

（2）上试听课，和主讲老师聊一聊（和老师聊，不是客服，客服只负责营销和教务），感觉一下靠不靠谱。不过大部分机构的试听课都一半的篇幅是营销，另外一半大概能透露一些这个老师的讲课风格，所以可以稍微感受一下。

（3）最可靠的方式是找到你周围上过这些课的同学、朋友，聊一下看看他们有没有通过这些课找到工作，或者细聊一下课程内容和上课方式，看能不能达到你的要求。

（4）网上、论坛上明显有差评和争议的课程需要谨慎。另外，满屏彩虹屁的营销文章基本不用看，很多文章作者都是国内写手，连班加罗尔在美国还是印度都弄不清。

（5）不过核心目的还是学习知识，真上当了除了尽快解决纠纷，还需要尽量调整心态，专注于找工作。

最重要的，不管报不报班，还是得靠自身努力，题还是得自己刷；可以花钱买来武功秘籍提升效率，但是最终的汗水和收获，都属于你自己。

## “废话”3：需要刷到什么程度？

在讨论详细的学习计划前，先定义一下学习目标是什么，或者说期望的学习效果。

对于算法题，作者拍了一下肚子，装X地分成了四层境界（仅代表个人意见及恶趣味）：

### 第一层境界：昨夜西风凋碧树，独上高楼，望尽天涯路。 （手中有题）

（一只学弱从坐标0点出发，100小时左右可以达到的状态）

总题量 <100 。

能认出旧题，识别出考点并且记得大致的思路，

在不限时间，不限提交次数的情况下，一般也能勉强完成代码。

对新题基本上无能为力。

### 第二层境界：谁家玉笛暗飞声，散入春风满洛城。（心中有题）

（200 ~ 300小时）

总题量 200+。

对算法模板、数据结构形成初步的认识，偶尔也能产生醍醐灌顶、幡然感悟的快乐。

对于旧题有较深的映象，有完整的解题思路，能在不看答案的情况下实现代码，基本上能在两三次提交内通过。

对于新题，大约能判断出题目的类型和解题的方向，但是缺乏对细节的实现能力，有时写的出，有时写不出。

### 第三层境界：衣带渐宽终不悔，为伊消得人憔悴。（人题合一 ）

（500 ~ 1000小时）

以下内容需要积分高于 50 您已经可以浏览

总题量 400~500。

熟练掌握十几种类型，300~400的核心/重点题，对每一类题的套路了然于心。

看到原题或类似的题基本能在规定时间内完成高质量，最优的代码。可能有一些小问题（typo，API记错），导致不能一遍过，但没有思路和逻辑上的问题，基本上不会提交两次以上。

能够清晰讲解解题思路，并让别人听懂。

对于没做过的题目，能迅速地进行分类；或者对于有多重解法的题目，能分析不同类解法的要点和优劣，理解取舍的原因。

基本上能根据自己想出的思路正确地实现最优解的代码，但可能会有一些细节考虑地不周到，造成提交超过两次。

在讲解这些没做过的题目的时候，可能不太流畅，但最终也能说得通。

### 第四层境界：两岸“猿”声啼不住，轻舟已过万重山。（天下无题）

（几千小时？我也不知道要久能到这个境界，显然我自己没练到。。。不过不影响，乾坤大挪移的作者也没修炼满，照样编第七层）

实现了对面试中所有常见类别的算法题套路与解法轻车熟路，完成了理论体系的融会贯通。

深刻理解题目的内涵，清楚的理解当一些题目条件发生变化时将对题目的解法与复杂度产生什么样的影响，能够大致猜到follow up的方向。

不管题目有没有做过，看到以后都很快形成清晰的思路，找到面试环境下的最优解，同时对一些需要注意的常见代码细节了然于胸。

能清晰流畅地表达讲解自己的思路。

在较短时间内流畅且无错地完成符合规范的代码，一气呵成。

整体而言，为了能较好的通过大部分面试，一般需要修炼到第三层境界。

当然如果经过努力最终到达第四层境界，至少在算法这个分项上，你大概已经咸鱼翻身成为大佬，具备手握包括顶级大厂在内的N个offer，在HR之间比价聊包的资本了。

如果你说你刷了一遍题就到达第四层境界，很好，大神你很优秀，目送您出门好走。

当然具体的时间和题量都因人而异，毕竟面试这件事并不是一个标准化事件，很难一概而论。

所以，我只是分享自己一只学弱的经历。

------------------------------- 第一章 的分割线 ----------------------------------------

好的，终于差不多聊完了第一章的三段“废话”。

接下来的章节计划先分享一些笔记体系和刷题日程安排，然后再按算法分类进行一些分享。

（未完待续）

补充内容 (2020-10-26 06:41):

更新番外篇: 转行刷题，是否一定要报班？有疑问的小伙伴可以看作者的另一个帖子，防坑指南：

https://www.1point3acres.com/bbs/thread-680829-1-1.html

补充内容 (2020-10-26 06:44):

更新至第二章：刷题笔记体系，作者用回复功能续更在这个主题里。为了方便大家使用“只看作者”功能，我可能就不单独回复大家的提问了，请见谅~

补充内容 (2020-11-7 11:35):

更新至第三章：刷题方法

补充内容 (2020-11-16 08:00):

更新至第四章：程序基础

补充内容 (2020-11-16 08:23):

更新至第五章：二分法

补充内容 (2020-12-6 07:28):

更新至第六章：多指针 （更新的时候，系统显示需要审核~可能需要一两天）

补充内容 (2020-12-31 09:44):

更新至第七章：宽度优先搜索 （最近比较忙，更新略慢，请见谅）

补充内容 (2021-1-18 06:47):

更新至第八章：二叉树与递归（有同学问优先级是怎么判断的？根据题目代表性和高频程度选择）

补充内容 (2021-1-31 09:24):

更新至第九章：深度优先搜索 （等待审核中；关于重复题目的问题：因为有些题目有多种解法，所以会出现在不同的章节里）

补充内容 (2021-2-21 11:47):

更新到第十章：数据结构 （后面再来一章DP应该就可以完结了）

补充内容 (2021-3-14 07:28):

更新到第十一章：动态规划 （算法篇已完结，以后如果有需要添加的内容会根据情况进行增补）

# 第二章 刷题笔记体系

胖头龙推荐使用完整的笔记系统来记录我们已经刷过的题目内容。

有些同学会觉得题目直接刷不就行了？还花时间整这些没用的？

对于一些几个小时的任务，直接莽一波确实没太大问题；但是刷题是一个几百上千小时的任务，多花一些时间在笔记和整理上，长远看来会有事半功倍的效果。

## 做笔记的好处

### （1）方便查找、记录：

很多时候我们做完、想明白一道题以后其实很快就会遗忘（领悟一时爽，事后忘光光）。

其实在思考过程中，我们会有一些很好的结论和过程，值得及时记下来。

虽然可以通过查看leetcode的submission来查看过往提交的代码，但是这样效率比较低，而且debug次数多的话，会把自己弄混。

所以记录一个属于自己的“golden” 答案是有必要的。

### （2）方便复习过题

整理过的笔记对于复习过题非常有价值，只需要点开相应的标签（分类或优先级），可以在几个小时内回顾几十道题的思路、要点和解法，对于加深记忆有奇效。

### （3）方便下次找工作

在“前言”部分也提到过，作为一个程序员，从毕业到工作十年左右，你可能需要找三四次工作，而且基本都逃不过算法面试。

因此，这次准备的笔记，以后都可以使用。

### （4）培养良好工作习惯

作者在几家公司工作过几年，身边很多正例、反例都说明一个有条理的员工是非常受欢迎的，无论是对老板还是同事。

不思考、不总结、上来就莽代码，想到哪做到哪的工作风格会让别人十分头痛，对你自己也没有好处。

Organized是一个工程师的基本素养，而且这和move fast并不冲突。

在本文的刷题笔记里，作者把每道题写成一页笔记，并且打上不同的标签。这样我们可以根据不同标签来快速查找分类题目。

我自己用的Evernote（不是广告，纯粹因为当年找了一个名气还可以而且免费的，你可以用你喜欢的任何App），比较喜欢里面的多标签功能。

一开始我十分痛苦它为什么不使用hierarchy 的 文件夹形式，后来发现扁平的 notes 外加 hierarchy 的 tag 反而更高效，因为这样同一篇note 可以出现在不同的 Tag Tree 里，非常方便。

## 标签分类

按题目类型分：

Binary Search, BFS, DFS...等算法分类，同时在按数据类型分，就像leetcode的标签一样，每道题因为有多重属性所以可以打上多个标签。

按重要性分：

在“前言”里面也说了，作者把题目分为四个优先级，所以刷题力度也不同：

<1> **必背**，大约20~30道，都是各类型题目的典型模板题，基本需要刷十几遍，做到迷迷糊糊半昏迷状态也能熟练默写的肌肉记忆状态。

<2> **核心**，大约100~150道，主要是各种高频题和经典题，基本在5~8遍以上，需要做到最优解，medium难度10分钟以内，hard难度15分钟以内，无错一遍过，同时要能解释清楚思路。另外有多重解法的也要掌握，知道不同解法间的优缺点和trade off原因。

<3> **重点**，大约200~300道，核心题之外的高频题，基本在5遍左右。这样遇到原题或者类似题的时候，基本思路、逻辑不会错。能不能临场完全bug free要看基本功和运气。

<4> **普通**，上述题之外所有你刷过的题。基本上做过一两遍，有个思路，临场遇见了不会慌。

当然你也可以创建更多的分类，比如难度、公司什么的，看你自己的需要。

进行过题复习的时候，直接点开某一类标签，然后几分钟过一题，非常有效率。

一些标签示例（之前截的图，有些中英文混用了，请忽略这个细节）：







## 笔记内容

每一题记录为一页笔记。

首先会抄下题号、难度、链接和截图题面。这样的好处是复习的时候不用再点开leetcode，省时间。

其次作者一般会添加这几个部分内容：

### 讨论澄清：

这个其实也是你需要和面试官clarify的内容。

因为实际面试不会像leetcode把一道题明明白白的摆在你面前，而是先给你一个模糊的描述，需要你问问题、讨论逐步细化明确需求。

所以在平时做题的时候，多想一想这些问题，对面试非常有帮助。

一个小tip，leetcode题目里面的notes，constraints 很多时候就是我们面试中需要先澄清的问题。

仔细想一想为什么要加这些条件，如果不加的话题目解法会出现什么样的变化，带着这样的思路去看题，也就知道面试的时候需要clarify什么了。

### 解法思路：

如果有多重解法，分开写。

对于每一种解法，描述一下核心思路和主要解法步骤。

其实这也是面试中，你向面试官简要描述你的解法的过程，对着屏幕自言自语练一练。

### 注意要点：

代码实现中需要注意的细节，这主要来自于你写这道题是遇到的具体问题，和 debug中领悟的要点，往往也是易错点或者比较tricky的地方。

### 复杂度：

常见的复杂度分析，每次做题想一想，分析一遍，自然就熟悉了。

同时也是不同算法之间trade off重要准则。

### 代码：

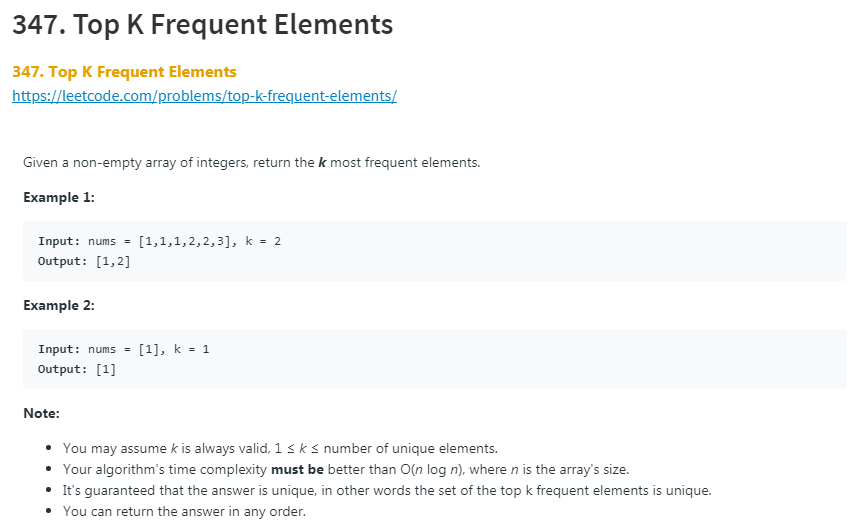
贴上你自己准备的"golden"答案，一方面方便复习，同时也促进了代码风格的形成。

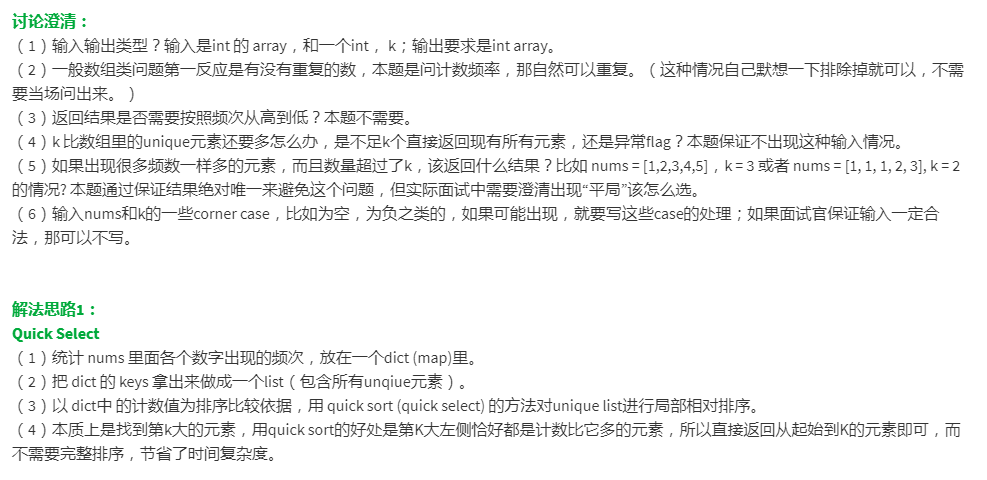
下面是一个笔记的例子，因为优先级相对较高，所以写的比较细。

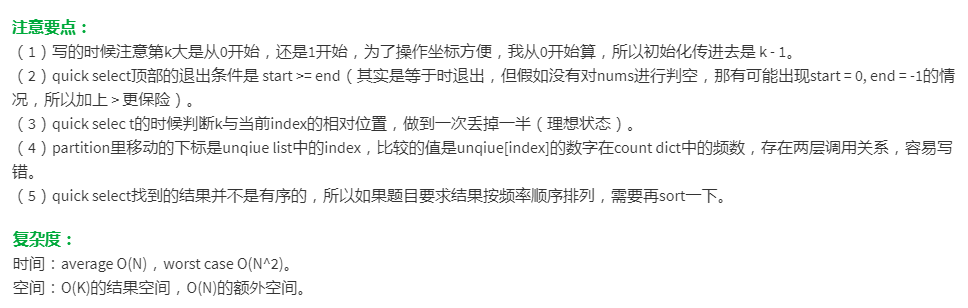
其实不用每题都写这么复杂，抓准每题的重点即可。

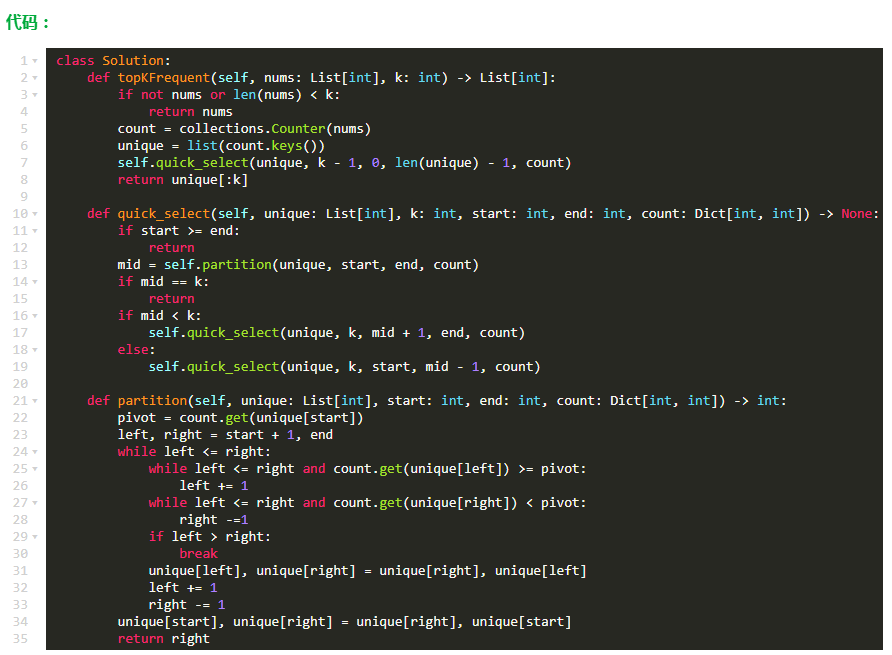
也不需要一次性写这么多，很多内容实在反复刷反复想的过程中逐步添加的。

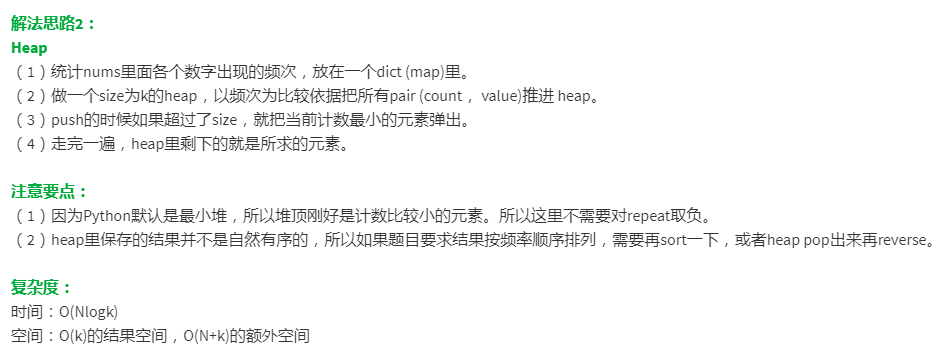
这个例子里面有三个标签：Pointers-array，Heap，优先级2\_核心。













## 刷题记录表

除了分类笔记，我们还需要一张总体的刷题表来记录自己当前的状态。

它可以帮助我们一目了然的看出哪些题我们做的还不够，下一步需要加强哪些方向。

这张表包含：

**要求**：必背，核心，重点，普通，分别用不同颜色区分，方便观察。

**题目链接**：指向leetcode的链接，并用颜色区分的难度。

**分类**：分类标签。

**当前刷题状态：**

包括最近刷题、过题的一些信息。特别是最近一次的写题时长和重交次数，这个决定刷题状态的颜色条：

**深红**：连答案都没看懂。

**浅红**：最终提交通过，但是时间太长或者重交次数过多（简单/中等超过10min，hard超过15min, 重交达到2次且有逻辑漏洞）。

**黄色**：第一次能在规定时间内完成，顶多有一两个typo造成的错误。

**绿色**：连续两次按时无错通过。

备注：一些其他信息，灵活使用，比如某题是其他题的follow up之类的。

截图一个之前的栗子：

一开始可能很空旷而且都是红色，后来渐渐变绿就越来越稳了。



以上就是胖头龙记笔记时的一些要点。

之前有同学建议胖头龙可以把刷题记录表和计划表（后面章节会提到）做成一个小App，方便大家使用，但确实业余时间有限，最近看来是没戏了。

其实具体形式不是最关键的，重要的是学习、工作的时候，形成一套自己的信息总结体系，做一个有条理的工程师。

# 第三章 刷题方法

## 基本思路

作为小白刚入门的时候，第一反应以为所谓刷500题就是把 LeetCode 上面前500题都做一遍；再后来有些经验了意识到刷一遍是不够的，基本做了就忘记了，所以需要刷5遍，也就是500 x 5的由来。

这样也对，也不对。

对于大部分资质平均水平的同学而言，刷题确实需要反复训练领悟，但是反复训练不代表死板的暴力重复。当然，暴力重复确实会有作用，但是有系统、有方法的学习，一般来说会更有效率。

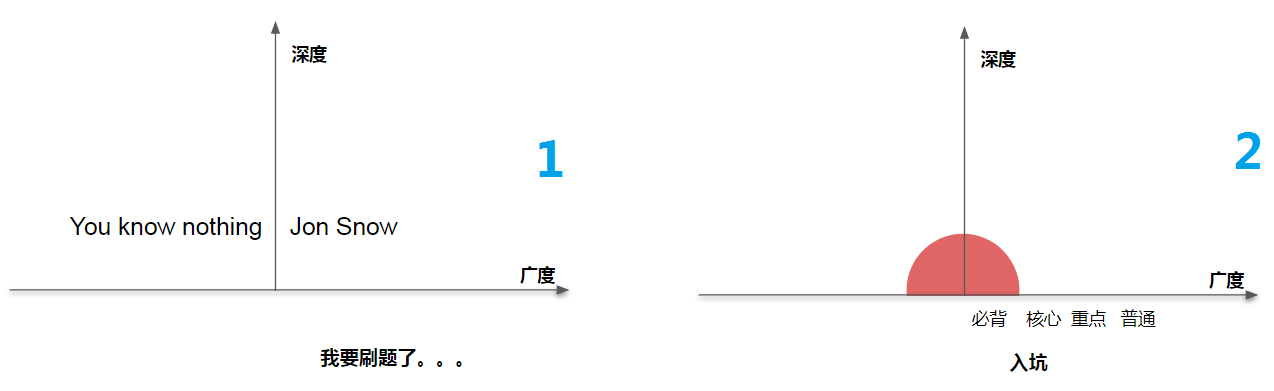
就像背单词一样，每个单词花5分钟读300遍未必是最好办法，间隔多天的反复学习反而效果会更好，因为我们需要尊重大脑的工作方式和记忆方式。

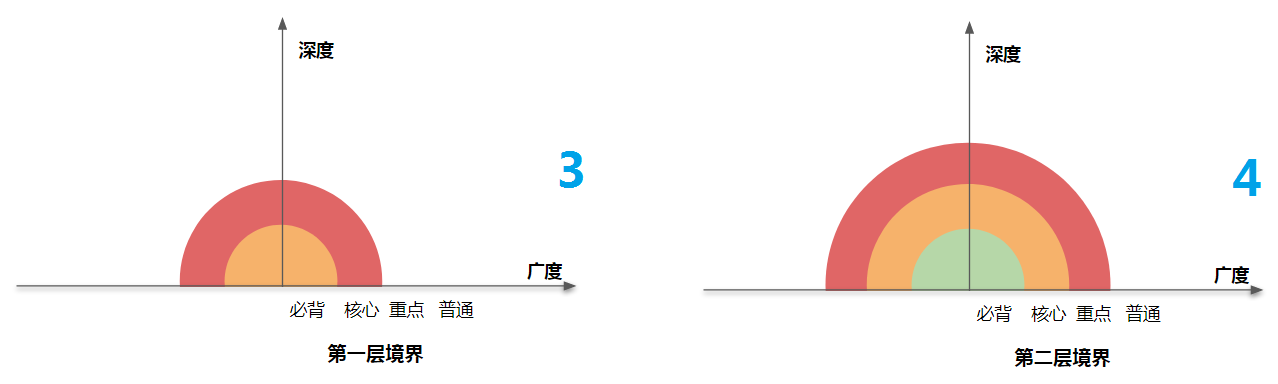
所以坐着不但强调刷题要分类，分重点，同时也要分阶段使用不同的方法，实现螺旋形上升的学习过程。

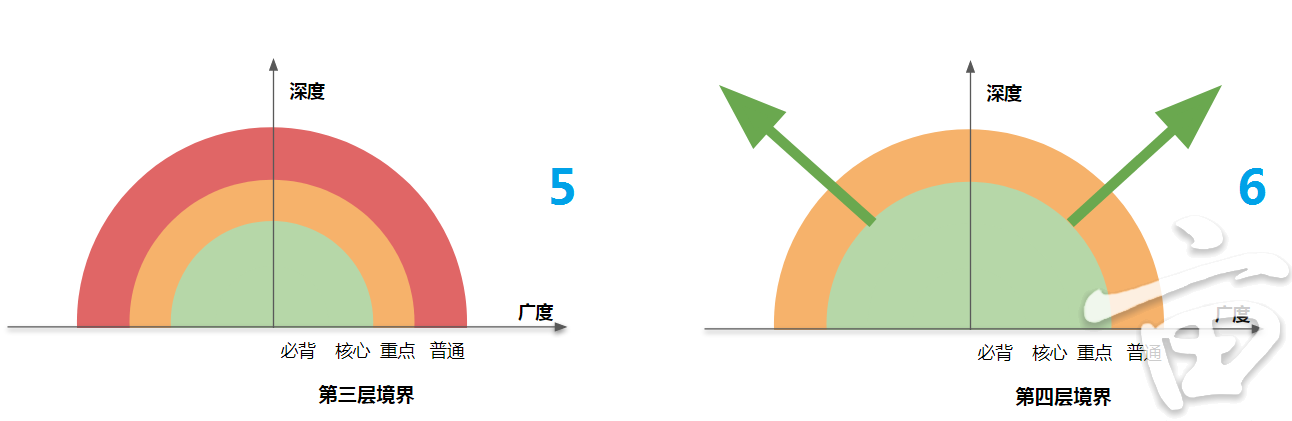
简单的说，就是新手状态只需要囫囵吞枣，不求甚解；稍微有些经验之后再深入思考，恍然大悟；重点题目都基本理解之后再扩展范围，反复训练；最后在不断的、轻重结合的反复训练中实现融会贯通，能做能说。

也就是相比于简单粗暴的依次刷题，循序渐进的循环刷题效率可能更高。

以下图为例，就是越核心的题目重复的次数越多，掌握的程度越深，熟练度也越高。







从刷题频次上来看，更接近于 30 \* 10 + 100 \* 8 + 300 \* 5 。

这样不仅学习效果更好，对于面试准备也更友善。

因为即便你只处于第二层境界的状态，也已经拥有了相对完整的知识体系，可以应付一些要求不那么高的面试。

## 刷题模式

刚才提到轻重结合就是指刷题力度不同，这个概念在序言里面已经提到过，这里再重复一下（稍作修改）不同刷题模式：

### 初刷：

新手期遇到新题，如果短时间内（五分钟）没有思路没关系，不要死缠烂打，早点看答案。

找到最优的答案并尽力去理解，用自己的面试语言和规范的代码风格进行“抄写”，不完全懂也没关系，标注一下，以后再看。

每题时长控制在半小时之内。

### 精刷：

针对初刷过还未充分理解的题。

尽量全面理解答案的要义，并且至少在能自己“默写”出来，并调试通过。

将解题思路和形成的标准答案加入到笔记当中。

如果实在不能完全领悟，需要在笔记中用显著符号标注该题，写下困惑的地方，过一段时间再试试看。

每题时长控制在一小时之内。

### 复刷：

针对做过的题进行复习。

自言自语clarify题意，讲请楚自己的思路，写题，检查，跑case，争取bug free一遍过。

只有参考笔记和标准答案复盘自己的错误和不足。

每题时长控制在20分钟内。

### 回顾：

按照笔记的标签（比如按分类，或者按优先级）大量复习题目。

针对做过的题进行复习，如果对这一题还有映象，则在大脑中过一遍解题思路及实现中的细节要点。

如不记得，则进行主动思考寻求解法，之后对比之前的答案和笔记，看是否正确。

如果不正确则需要强化思考并记忆，不需要写题。

每题时长控制在5-10分钟内。

### 讲题：

在对此题解题思路正确的情况下，尝试在白板上画图并讲解自己的解题思路，看是否能够流畅完成此过程。

找到一个让自己满意的描述方式，将图表和讲题思路/要点加入到笔记当中。

### 模拟：

抽取一道新题，看能不能做到复刷中的效果目标。

把题目记入笔记。

每题时长控制在30分钟内。

## 刷题过程

每个人都可以根据自己的情况定制具体计划，这里给一个例子，是一个对于新手相对详细全面的螺旋型学习过程，可以根据自己的状态进行步骤的增减。

其中必背题大约30道，核心题大约100+，重点题大约200-300道。

### （1）基本知识学习

要刷题首先得学会语言，新手起步一般是 Python 或者 Java。（当然刷题语言用什么这又是个引战无数的问题，没必要展开，因为各有优劣，所以自己爱用啥用啥，反正面试官一般都看得懂。作者自己一开始用的Java后来转用Python，因为一般来说大部分算法题的情况下 Python 会相对短一些，面试的时候省一些时间。）

语言学习需要理解这种语言的基本数据结构（Array, Linked List, Set, Map等等）、逻辑操作（if,else, for, while, continue,break, return 等等），然后再学习一些稍微复杂一些的数据结构（Stack, Queue, Tree, Heap等等），以及一些简单的 OOD 知识，比如关于Class的最基本的概念。

一般来说很多网站都有详细或者简单的教程。建议从简单好玩可互动的入门级教程开始上手，然后再去读相对复杂枯燥的讲解，最后也可以适当做一些入门题，就是Lintcode上Naive标签的题（LeetCode好像没有这个难度），熟悉一下最基本的写代码的感觉。

目标：知道if语句，for循环怎么写，基本数据的增删改查知道怎么处理。

### （2）初刷必背+核心

分类学习各算法分类的基本概念和思路，以初刷模式刷一遍必备+核心题目（30 + 100左右）。

其中概念和思路要做到基本理解，但是题目一时半会无法完全理解很正常，把题号+题目+参考答案 按照第二章说的方式记在笔记里即可，刷题记录可以标记为深红（尚未理解）。

目标：知道算法大概有哪些门类，题目大概长什么样，都是在问些什么，总结100+题的答案，并收录在笔记里。

### （3）精刷必背题

按分类尝试深入理解每个分类下面的必背题。

对于核心题可以按照初刷的方式再过一遍，看不懂就跳过。

目标：基本理解了必背题的解题思路。

### （4）精刷核心题

尝试默写出必背题，对核心题使用精刷操作。

目标：能够基本写出必背题的答案，基本理解核心题的解题思路。（第一层境界）

### （5）初刷重点题

按照分类对重点题使用初刷模式。

每道题自己先思考几分钟，有思路就先自己试一试，没思路就看答案，不管是否成功写对，最后都要找到标准答案。

目标：把重点题过一遍，至少知道前400题长什么样。

### （6）复刷必背+核心

继续熟练默写必背题，并尝试讲题，包括clarify，描述思路，过一些test case。

尝试自己写出核心题。

目标：能都随手写出30+必背题；碰到核心题，都能有思路，虽然可能有小问题，但基本都能写出来。（第二层境界）

### （7）精刷重点题 + 回顾必背、核心题

按照高频顺序（不再是分类）精刷重点题，如果觉得比较费时间，就从前往后刷到哪是哪。

同时回顾（过题）必备题和核心题，做过时间的比较久的重点题也可以放在回顾计划中。回顾时主要关注思路和实现要点、易错点。

目标：对于前300 ~ 400题形成相对完整的知识体系、笔记体系。

### （8）复刷必背、核心题 + 回顾重点题

做到必背、核心题基本都能一次写对，尝试讲题。

目标: 能秒杀100+核心题。

### （9）复刷重点题 + 回顾所有做过的题

如果重点题能直接做出来（可能只是一小部分），就对照一下之前的答案对比一下小的细节差异，如果五分钟没有思路，或者三十分钟没写对，直接看之前笔记的答案修正问题。

目标：对于300+重点题，也基本看到就都有思路。

### （10）回顾所有做过的题 + 模拟

按频率或者面经，以模拟的形式尝试还没做过的题。

包括clarify，解法描述，写题，跑case等等。

不管有没有做对，都要看答案形成笔记。

目标：对于必背、核心题都可以秒杀并详细讲解；对于重点题可以在没有太大逻辑错误的情况下写对；对于400道以外的题目也做过一些，并且看到大部分新题都至少有思路。（第三层境界）

### （10 +）回顾旧题 + 做新题

进入了一个开放性的成长阶段，面试能用到的理论体系已经完整建立，主流高频题都能搞定。

通过各种新题磨炼自己的临场技能，补充知识体系中的细节。

目标：前往第四层境界的路上。

刷题计划表

也可以定制一个自己的刷题计划表（不同于刷题记录表），方便计划和打卡。

一般来说，刷题这样的高脑力活动，一天4~6个小时效率会比较高。

超出的时间做一些知识学习、回顾旧题这样相对较轻松的任务会比较合适。

如果经常无法按时完成，需要及时调整。

注意劳逸结合。大脑和肌肉一样，练多了效率会降低，特别是对于二十多岁的中老年人。

把每天有限的心力及时用在最困难的任务上。



一些重复三遍的小建议：

学不明白不是你的错，是题目确实难

不要和某一题死磕太久

劳逸结合

能讲和能写一样重要；但是在会讲题之前，得先会写；会写之前，得先理解

按照现在的市场价格，学会一题相当于赚 $500，而且是每年

过程是艰难的，结果是美好的

到这里，前三章的基本方法和思路就差不多讲完了。

第四章开始总结分类列表。

# 第四章 程序基础

从本章开始，将按分类分享题目列表和优先级。

优先级用不同颜色标注：必背：紫色；核心：蓝色；重点：绿色；普通：黄色。

另外对于必背和核心题，题目本身的难度级别反而不重要（不管简单还是难你都得会），所有就不标注了。

至于题目的答案，因为作者自己的题解未必是最好的，还希望大家在网上学习总结出适合自己的模板。

LeetCode的Solution和discussion以及中文网站的各种博客已经足够查找到需要的所有信息了。

另外也会针对每类问题列出一些比较重要的问题，供大家在自学中有一个大致的方向。

Python 基本入门的网站（现在基本都用Python3）：

W3schools:

这个自带iteractive的在线编译器，比较方便。

https://www.w3schools.com/python/

Tutorialspoint :

这个也差不多，没有编译器。

https://www.tutorialspoint.com/python/index.htm

官方文档：

用来查询API的用法，很实用。

https://docs.python.org/3/

Google 代码风格规范：

一开始会看不明白，几十题之后再回来读，然后自觉向规范靠拢。

https://google.github.io/styleguide/pyguide.html

## 基本问题：

（1）变量，基本数据类型，逻辑运算符号

（2）控制语句，if else

（3）循环结构，for, while, continue, break

（4）基本数据操作，int，float和常用的数学运算，不同数据类型之间的转换

（5）基本数据结构，理解hashmap，set，tree,  linked list的概念，理解Python 中 string，list，tuple，dict, set，linked list的增删改查

（6）高级数据结构，Counter, defaultdict, OrderedDict, queue, deque, stack, heap, bisect的概念和在python中的用法

（7）其他常见方法，sort, enumerate, map, filter, lambda

（8）函数的使用，参数的传递，引用的传递，递归的基本概念

（9）基本的面向对象，Class, Object，理解引用，deep copy的概念，通过自定义\_\_lt\_\_,  \_\_gt\_\_, \_\_eq\_\_来控制数据结构的排序。

## 语法入门题目列表：

因为Leetcode没太多基础的题，所以这个章节基本用的是Lintcode上的Naive。

这些题主要是帮助熟悉语法，所以没有用特别高的优先级。

**Lint-37. Reverse 3-digit Integer**

https://www.lintcode.com/problem ... integer/description

**Lint-214. Max of Array**

https://www.lintcode.com/problem/max-of-array/description

**Lint-283. Max of 3 Numbers**

https://www.lintcode.com/problem/max-of-3-numbers/description

**Lint-146. Lower to uppercase II**

https://www.lintcode.com/problem ... case-ii/description

**Lint-241. String to Integer**

https://www.lintcode.com/problem/string-to-integer/description

**Lint-449. Char to Integer**

https://www.lintcode.com/problem/char-to-integer/description

**Lint-463. Sort Integers**

https://www.lintcode.com/problem/sort-integers/description

**Lint-484. Swap Two Intergers in Array**

https://www.lintcode.com/problem ... n-array/description

**Lint-485. Generate ArrayList with Given Size**

https://www.lintcode.com/problem ... en-size/description

**Lint-225. Find Node in Linked List**

https://www.lintcode.com/problem ... ed-list/description

**Lint-466. Count Linked List Nodes**

https://www.lintcode.com/problem ... t-nodes/description

**Lint-483. Convert Linked List to Array List**

https://www.lintcode.com/problem ... ay-list/description

**Lint-454. Rectangle Area**

https://www.lintcode.com/problem/rectangle-area/description

**Lint-478. Simple Calculator**

https://www.lintcode.com/problem/simple-calculator/description

**Lint-366. Fibonacci**

https://www.lintcode.com/problem/fibonacci/description

**Lint-632. Binary Tree Maximum Node**

https://www.lintcode.com/problem ... um-node/description

**Lint-40. Implement Queue by Two Stacks**

https://www.lintcode.com/problem ... -stacks/description

**Lint-492. Implement Queue by Linked List**

https://www.lintcode.com/problem ... ed-list/description

**Lint-494. Implement Stack by Two Queues**

https://www.lintcode.com/problem ... -queues/description

**Lint-495. Implement Stack**

https://www.lintcode.com/problem/implement-stack/description

# 第五章 二分法

## 基本问题：

（1）基本思想？（有序的数据，每次通过判断逻辑排除掉一部分的答案，直到触发终止条件）

（2）二分法实现模板（可以递归，可以迭代；一般以迭代为主）

（3）移动两个指针（start与end）的含义？移动条件是什么（筛选掉一部分数据的依据）？循环的截止条件？

（4）数据中是否有重复数字？对结果有什么影响？

（5）为什么你选择的模板中使用start < end 或者 start <= end 或者 start + 1 < end 作为终止条件？这样写是如何避免死循环的？不这么写在什么情况下会出现死循环？

（6）在处理逻辑中，当前结果>, <, = 目标值时分别如何处理？移动指针的依据是什么？

（7）循环退出后是否需要额外处理？

（8）如果遇到corner case根本没进主循环，你的代码是否能正常工作？

（9）为什么Java需要写 mid = start + (end - start) / 2 而 Python可以直接写 mid = (start + end) // 2 ？

（10）如何理解从基本的朴素二分，到相对复杂的条件二分，到更加抽象的答案二分？（在一个显性有序数组中一次砍掉一部分 -->  在一组有规律的数据上利用判断条件逐步缩小范围  -->  在一个有序的抽象模型里，利用不断的"猜测+检验"逐步逼近最终结果）

## 二分法题目列表：

（必背：紫色；核心：蓝色；重点：绿色；普通：黄色；默认是LeetCode，如果是LintCode会以Lint开头）

### 朴素二分法：

**704. Binary Search**

https://leetcode.com/problems/binary-search/

**34. Find First and Last Position of Element in Sorted Array**

https://leetcode.com/problems/fi ... nt-in-sorted-array/

**702. Search in a Sorted Array of Unknown Size**

https://leetcode.com/problems/se ... ay-of-unknown-size/

**153. Find Minimum in Rotated Sorted Array**

https://leetcode.com/problems/fi ... tated-sorted-array/

**154. Find Minimum in Rotated Sorted Array II**

https://leetcode.com/problems/fi ... ed-sorted-array-ii/

**278. First Bad Version**

https://leetcode.com/problems/first-bad-version/

**658. Find K Closest Elements**

https://leetcode.com/problems/find-k-closest-elements/

### 条件二分法：

**33. Search in Rotated Sorted Array**

**(81. Search in Rotated Sorted Array II， follow up)**

https://leetcode.com/problems/search-in-rotated-sorted-array/

https://leetcode.com/problems/search-in-rotated-sorted-array-ii/

**4. Median of Two Sorted Arrays**

https://leetcode.com/problems/median-of-two-sorted-arrays/

**74. Search a 2D Matrix**

https://leetcode.com/problems/search-a-2d-matrix/

**162. Find Peak Element**

https://leetcode.com/problems/find-peak-element/

**302. Smallest Rectangle Enclosing Black Pixels**

https://leetcode.com/problems/sm ... osing-black-pixels/

**852. Peak Index in a Mountain Array**

https://leetcode.com/problems/peak-index-in-a-mountain-array/

### 答案二分法：

**875. Koko Eating Bananas**

https://leetcode.com/problems/koko-eating-bananas/

**1283. Find the Smallest Divisor Given a Threshold**

https://leetcode.com/problems/fi ... -given-a-threshold/

**69. Sqrt(x)**

**(Lint-586. Sqrt(x) II, follow up)**

https://leetcode.com/problems/sqrtx/

https://www.lintcode.com/problem/sqrtx-ii/description

**Lint-183. Wood Cut**

https://www.lintcode.com/problem/wood-cut/description

**Lint-437. Copy Books**

https://www.lintcode.com/problem/copy-books/description

**Lint-438. Copy Books II**

https://www.lintcode.com/problem/copy-books-ii/description

# 第六章 多指针

## 基本问题：

（1）多指针是一个非常广泛的概念，并不是一个固定的算法。但基本上是通过一些变量的控制与循环把问题的复杂度控制在一两层for循环之内。可以用在数组、链表、区间、滑动窗口、流、回文串、和差问题等多个场景。（前项和其实并不完全是指针问题，但也归并在这里）。

（2）Quick Sort和Merge Sort的基本原理与实现，排序的稳定性问题

（3）Quick Select的实现与复杂度

（4）同向指针与相向指针的使用场景

（5）不同场景下循环终止条件？

（6）两数之和，之差，特定条件下（大小于某值等）的计数问题

（7）三数或三数以上之和的通用写法（两数之和+搜索）

（8）数组有没有排序？是否需要排序？

（9）数组有没有去重？是否需要去重？

（10）离线数据（内存中，有限长）还是在线数据（无法放入内存，长度未知）？

（11）链表操作中dummy node与previous node的使用技巧

（12）链表的中点，判断是否有环，寻找环的交叉点

## 多指针题目列表：

（必背：紫色；核心：蓝色；重点：绿色；普通：黄色；默认是LeetCode，如果是LintCode会以Lint开头）

### 数组：

**912. Sort an Array (Quick Sort and Merge Sort)**

<https://leetcode.com/problems/sort-an-array/>

**75. Sort Colors**

<https://leetcode.com/problems/sort-colors/>

**26. Remove Duplicates from Sorted Array**

https://leetcode.com/problems/re ... -from-sorted-array/

**80. Remove Duplicates from Sorted Array II**

https://leetcode.com/problems/re ... om-sorted-array-ii/

**88. Merge Sorted Array**

https://leetcode.com/problems/merge-sorted-array/

**283. Move Zeroes**

https://leetcode.com/problems/move-zeroes/

**215. Kth Largest Element in an Array**

https://leetcode.com/problems/kth-largest-element-in-an-array/

**347. Top K Frequent Elements**

https://leetcode.com/problems/top-k-frequent-elements/

**349. Intersection of Two Arrays**

https://leetcode.com/problems/intersection-of-two-arrays/

**350. Intersection of Two Arrays II**

https://leetcode.com/problems/intersection-of-two-arrays-ii/

**845. Longest Mountain in Array**

https://leetcode.com/problems/longest-mountain-in-array/

**42. Trapping Rain Water**

https://leetcode.com/problems/trapping-rain-water/

**43. Multiply Strings**

https://leetcode.com/problems/multiply-strings/

**969. Pancake Sorting**

https://leetcode.com/problems/pancake-sorting/

**Lint-31. Partition Array**

https://www.lintcode.com/problem/partition-array/description

**Lint-625. Partition Array II**

https://www.lintcode.com/problem/partition-array-ii/description

**Lint-143. Sort Color II**

https://www.lintcode.com/problem/sort-colors-ii/description

**Lint-461. Kth Smallest Numbers in Unsorted Array**

https://www.lintcode.com/problem ... d-array/description

**Lint-544. Top k Largest Numbers**

https://www.lintcode.com/problem ... numbers/description

### 链表：

**21. Merge Two Sorted Lists**

<https://leetcode.com/problems/merge-two-sorted-lists/>

**86. Partition List**

https://leetcode.com/problems/partition-list/

**141. Linked List Cycle**

https://leetcode.com/problems/linked-list-cycle/

**160. Intersection of Two Linked Lists**

https://leetcode.com/problems/intersection-of-two-linked-lists/

**234. Palindrome Linked List**

https://leetcode.com/problems/palindrome-linked-list/

**328. Odd Even Linked List**

<https://leetcode.com/problems/odd-even-linked-list/>

**142. Linked List Cycle II**

https://leetcode.com/problems/linked-list-cycle-ii/

**287. Find the Duplicate Number**

https://leetcode.com/problems/find-the-duplicate-number/

**876. Middle of the Linked List**

<https://leetcode.com/problems/middle-of-the-linked-list/>

### 区间：

**Lint-391. Number of Airplanes in the Sky**

https://www.lintcode.com/problem ... the-sky/description

**56. Merge Intervals**

https://leetcode.com/problems/merge-intervals/

**57. Insert Interval**

https://leetcode.com/problems/insert-interval/

**252. Meeting Rooms**

https://leetcode.com/problems/meeting-rooms/

**253. Meeting Rooms II**

https://leetcode.com/problems/meeting-rooms-ii/

**986. Interval List Intersections**

<https://leetcode.com/problems/interval-list-intersections/>

### 回文串：

**5. Longest Palindromic Substring**

https://leetcode.com/problems/longest-palindromic-substring/

**345. Reverse Vowels of a String**

https://leetcode.com/problems/reverse-vowels-of-a-string/

**680. Valid Palindrome II**

<https://leetcode.com/problems/valid-palindrome-ii/>

**125. Valid Palindrome**

<https://leetcode.com/problems/valid-palindrome/>

### 滑动窗口：

**3. Longest Substring Without Repeating Characters**

[https://leetcode.com/problems/lo ... peating-characters/](https://leetcode.com/problems/lo%20...%20peating-characters/)

**11. Container With Most Water**

https://leetcode.com/problems/container-with-most-water/

**76. Minimum Window Substring**

https://leetcode.com/problems/minimum-window-substring/

**209. Minimum Size Subarray Sum**

https://leetcode.com/problems/minimum-size-subarray-sum/

**239. Sliding Window Maximum**

https://leetcode.com/problems/sliding-window-maximum/

**713. Subarray Product Less Than K**

<https://leetcode.com/problems/subarray-product-less-than-k/>

**395. Longest Substring with At Least K Repeating Characters**

https://leetcode.com/problems/lo ... peating-characters/

**480. Sliding Window Median**

https://leetcode.com/problems/sliding-window-median/

**567. Permutation in String**

https://leetcode.com/problems/permutation-in-string/

**727. Minimum Window Subsequence**

https://leetcode.com/problems/minimum-window-subsequence/

**Lint-604. Window Sum**

<https://www.lintcode.com/problem/window-sum/description>

### 流：

**295. Find Median from Data Stream**

<https://leetcode.com/problems/find-median-from-data-stream/>

**346. Moving Average from Data Stream**

https://leetcode.com/problems/moving-average-from-data-stream/

**352. Data Stream as Disjoint Intervals**

https://leetcode.com/problems/data-stream-as-disjoint-intervals/

**703. Kth Largest Element in a Stream**

<https://leetcode.com/problems/kth-largest-element-in-a-stream/>

### 前项和：

**53. Maximum Subarray**

<https://leetcode.com/problems/maximum-subarray/>

**238. Product of Array Except Self**

https://leetcode.com/problems/product-of-array-except-self/

**303. Range Sum Query - Immutable**

https://leetcode.com/problems/range-sum-query-immutable/

**325. Maximum Size Subarray Sum Equals k**

https://leetcode.com/problems/maximum-size-subarray-sum-equals-k/

**528. Random Pick with Weight**

https://leetcode.com/problems/random-pick-with-weight/

**560. Subarray Sum Equals K**

<https://leetcode.com/problems/subarray-sum-equals-k/>

### 和差问题：

**1. Two Sum**

<https://leetcode.com/problems/two-sum/>

**15. 3Sum**

https://leetcode.com/problems/3sum/

**18. 4Sum**

https://leetcode.com/problems/4sum/

**Lint-382. Triangle Count**

<https://www.lintcode.com/problem/triangle-count/description>

**167. Two Sum II - Input array is sorted**

https://leetcode.com/problems/two-sum-ii-input-array-is-sorted/

**170. Two Sum III - Data structure design**

https://leetcode.com/problems/two-sum-iii-data-structure-design/

**653. Two Sum IV - Input is a BST**

https://leetcode.com/problems/two-sum-iv-input-is-a-bst/

**1099. Two Sum Less Than K**

https://leetcode.com/problems/two-sum-less-than-k/

**259. 3Sum Smaller**

https://leetcode.com/problems/3sum-smaller/

**Lint-57. 3Sum Closest**

https://www.lintcode.com/problem/3sum-closest/description

**Lint-443. Two Sum - Greater than target**

https://www.lintcode.com/problem ... -target/description

**Lint-533. Two Sum - Closet to target**

https://www.lintcode.com/problem ... -target/description

**Lint-587. Two Sum - Unique pairs**

https://www.lintcode.com/problem/two-sum-unique-pairs/description

**Lint-609. Two Sum - Less than or equals to target**

https://www.lintcode.com/problem ... -target/description

**Lint-610. Two Sum - Difference equals to target**

https://www.lintcode.com/problem ... rget/my-submissions

# 第七章 宽度优先搜索

## 基本问题：

（1）如果复杂程度类似， 面试中尽量优先使用BFS

（2）BFS主要几种场景： 层级遍历，拓扑排序，图上搜索（包括二叉树，矩阵）

（3）Queue的使用技巧，BFS的终止条件？

（4）什么时候使用分层？什么时候不需要？实现的时候的区别在哪里？

（5）拓扑排序的概念？如何判断是否存在拓扑排序？是否存在唯一的拓扑排序？找到所有拓扑排序？

（6）什么时候需要使用set记录访问过的节点？（为什么二叉树上的BFS往往不需要set？）什么时候需要map记录到达过的节点距离？

（7）如何在矩阵中遍历下一步的所有节点？如果每次可能走不止一步怎么办（Maze II）？

（8）为什么BFS解决的基本都是简单图（边长为1）问题？如果边长不为1，该怎么办？

（9）BFS的时空复杂度估算？

（10）如何使用双向BFS进行优化？

## BFS题目列表：

（必背：紫色；核心：蓝色；重点：绿色；普通：黄色；默认是LeetCode，如果是LintCode会以Lint开头）

### 二叉树:

**297. Serialize and Deserialize Binary Tree**

https://leetcode.com/problems/se ... ialize-binary-tree/

**102. Binary Tree Level Order Traversal**

https://leetcode.com/problems/binary-tree-level-order-traversal/

**103. Binary Tree Zigzag Level Order Traversal**

https://leetcode.com/problems/bi ... el-order-traversal/

**107. Binary Tree Level Order Traversal II**

https://leetcode.com/problems/bi ... order-traversal-ii/

**513. Find Bottom Left Tree Value**

https://leetcode.com/problems/find-bottom-left-tree-value/

**Lint-242. Convert Binary Tree to Linked Lists by Depth**

https://www.lintcode.com/problem ... y-depth/description

### 拓扑排序:

**Lint-127. Topological Sorting**

https://www.lintcode.com/problem/topological-sorting/description

**207. Course Schedule**

https://leetcode.com/problems/course-schedule/

**210. Course Schedule II**

https://leetcode.com/problems/course-schedule-ii/

**269. Alien Dictionary**

https://leetcode.com/problems/alien-dictionary/

**444. Sequence Reconstruction**

https://leetcode.com/problems/sequence-reconstruction/

### 矩阵:

**200. Number of Islands**

https://leetcode.com/problems/number-of-islands/

**490. The Maze**

https://leetcode.com/problems/the-maze/

**505. The Maze II**

https://leetcode.com/problems/the-maze-ii/

**542. 01 Matrix**

https://leetcode.com/problems/01-matrix/

**733. Flood Fill**

https://leetcode.com/problems/flood-fill/

**994. Rotting Oranges**

https://leetcode.com/problems/rotting-oranges/

**305. Number of Islands II**

https://leetcode.com/problems/number-of-islands-ii/

**773. Sliding Puzzle**

https://leetcode.com/problems/sliding-puzzle/

**Lint-573. Build Post Office II**

https://www.lintcode.com/problem/build-post-office-ii/description

**Lint-598. Zombie in Matrix**

https://www.lintcode.com/problem/zombie-in-matrix/description

**Lint-611. Knight Shortest Path**

https://www.lintcode.com/problem/knight-shortest-path/description

**Lint-794. Sliding Puzzle II**

https://www.lintcode.com/problem/sliding-puzzle-ii/description

### 图:

**133. Clone Graph**

https://leetcode.com/problems/clone-graph/

**127. Word Ladder**

https://leetcode.com/problems/word-ladder/

**261. Graph Valid Tree**

https://leetcode.com/problems/graph-valid-tree/

**841. Keys and Rooms**

https://leetcode.com/problems/keys-and-rooms/

**323. Number of Connected Components in an Undirected Graph**

https://leetcode.com/problems/nu ... n-undirected-graph/

**1306. Jump Game III**

https://leetcode.com/problems/jump-game-iii/

**Lint-531. Six Degree**

https://www.lintcode.com/problem/six-degrees/description

**Lint-618. Search Graph Nodes**

https://www.lintcode.com/problem/search-graph-nodes/description

**Lint-624. Remove Substrings**

<https://www.lintcode.com/problem/remove-substrings/description>

# 第八章 二叉树与递归

## 基本问题：

（1）理解二叉树、平衡二叉树、二叉搜索树的关系和概念。

（2）理解递归的概念和方法，递归三要素。

（3）在解决递归问题的时候，有时可以返回多个值（Python），或者用一个额外的class包装多个值（Java）。

（4）熟练掌握用递归和非递归的方式分别前序、中序、后序遍历二叉树的方法。

（5）理解掌握分治和遍历的区别和联系。

（6）理解掌握top-down, buttom-up的思路。

（7）理解掌握二叉树上的Iterator。

## 二叉树与递归 题目列表：

（必背：紫色；核心：蓝色；重点：绿色；普通：黄色；默认是LeetCode，如果是LintCode会以Lint开头）

因为二叉树上的递归很多时候既可以用分治，也可以用遍历，并不是哪一种方法总能最优。

所以我们按相似题目分类，而不是按解法分类。

### 二叉树前中后序遍历（需要熟练掌握非递归方式）：

**94. Binary Tree Inorder Traversal**

https://leetcode.com/problems/binary-tree-inorder-traversal/

**144. Binary Tree Preorder Traversal**

https://leetcode.com/problems/binary-tree-preorder-traversal/

**145. Binary Tree Postorder Traversal**

https://leetcode.com/problems/binary-tree-postorder-traversal/

### 反向复原二叉树：

**105. Construct Binary Tree from Preorder and Inorder Traversal**

https://leetcode.com/problems/co ... -inorder-traversal/

**106. Construct Binary Tree from Inorder and Postorder Traversal**

https://leetcode.com/problems/co ... ostorder-traversal/

**889. Construct Binary Tree from Preorder and Postorder Traversal**

https://leetcode.com/problems/co ... ostorder-traversal/

### Iterator相关：

**173. Binary Search Tree Iterator**

https://leetcode.com/problems/binary-search-tree-iterator/

**230. Kth Smallest Element in a BST**

https://leetcode.com/problems/kth-smallest-element-in-a-bst/

**285. Inorder Successor in BST**

https://leetcode.com/problems/inorder-successor-in-bst/

**270. Closest Binary Search Tree Value**

https://leetcode.com/problems/closest-binary-search-tree-value/

**272. Closest Binary Search Tree Value II**

https://leetcode.com/problems/cl ... arch-tree-value-ii/

**510. Inorder Successor in BST II**

https://leetcode.com/problems/inorder-successor-in-bst-ii/

**Lint-915. Inorder Predecessor in BST II**

https://www.lintcode.com/problem ... -in-bst/description

### 判断树的形态：

**98. Validate Binary Search Tree**

https://leetcode.com/problems/validate-binary-search-tree/

**100. Same Tree**

https://leetcode.com/problems/same-tree/

**101. Symmetric Tree**

https://leetcode.com/problems/symmetric-tree/

**110. Balanced Binary Tree**

https://leetcode.com/problems/balanced-binary-tree/

### 子树相关问题：

**111. Minimum Depth of Binary Tree**

https://leetcode.com/problems/minimum-depth-of-binary-tree/

**104. Maximum Depth of Binary Tree**

https://leetcode.com/problems/maximum-depth-of-binary-tree/

**333. Largest BST Subtree**

https://leetcode.com/problems/largest-bst-subtree/

**Lint-596. Minimum Subtree**

https://www.lintcode.com/problem/minimum-subtree/description

**Lint-597. Subtree with Maximum Average**

https://www.lintcode.com/problem ... average/description

### 路径相关问题：

**112. Path Sum**

https://leetcode.com/problems/path-sum/

**113. Path Sum II**

https://leetcode.com/problems/path-sum-ii/

**124. Binary Tree Maximum Path Sum**

https://leetcode.com/problems/binary-tree-maximum-path-sum/

**Lint-475. Binary Tree Maximum Path Sum II**

https://www.lintcode.com/problem ... -sum-ii/description

**298. Binary Tree Longest Consecutive Sequence**

https://leetcode.com/problems/bi ... nsecutive-sequence/

**549. Binary Tree Longest Consecutive Sequence II**

https://leetcode.com/problems/bi ... cutive-sequence-ii/

**Lint-619. Binary Tree Longest Consecutive Sequence III**

https://www.lintcode.com/problem ... nce-iii/description

### LCA问题：

**236. Lowest Common Ancestor of a Binary Tree**

https://leetcode.com/problems/lo ... r-of-a-binary-tree/

**Lint-474. Lowest Common Ancestor II**

https://www.lintcode.com/problem ... stor-ii/description

**Lint-578. Lowest Common Ancestor III**

https://www.lintcode.com/problem ... cestor-iii/solution

### 其他：

**199. Binary Tree Right Side View**

https://leetcode.com/problems/binary-tree-right-side-view/

**513. Find Bottom Left Tree Value**

https://leetcode.com/problems/find-bottom-left-tree-value/

**331. Verify Preorder Serialization of a Binary Tree**

https://leetcode.com/problems/ve ... n-of-a-binary-tree/

**449. Serialize and Deserialize BST**

https://leetcode.com/problems/serialize-and-deserialize-bst/

**114. Flatten Binary Tree to Linked List**

<https://leetcode.com/problems/flatten-binary-tree-to-linked-list/>

# 第九章 深度优先搜索

## 基本问题：

（1）DFS中递归的基本要素

（2）终止条件的选择；回溯；剪枝

（3）什么时候需要排序？

（4）如何去除重复元素？一个元素允许使用多次的情况？

（6）在图上进行DFS如何避免回到重复节点

（5）识别一个隐式图，并使用DFS

（6）在某些情况下，利用记忆化搜索进行优化

## 深度优先搜索 题目列表：

（必背：紫色；核心：蓝色；重点：绿色；普通：黄色；默认是LeetCode，如果是Lint会以Lint开头）

### 排列组合：

**39. Combination Sum**

https://leetcode.com/problems/combination-sum/

**40. Combination Sum II**

https://leetcode.com/problems/combination-sum-ii/

**46. Permutations**

https://leetcode.com/problems/permutations/

**47. Permutations II**

https://leetcode.com/problems/permutations-ii/

**77. Combinations**

https://leetcode.com/problems/combinations/

**78. Subsets**

https://leetcode.com/problems/subsets/

**90. Subsets II**

https://leetcode.com/problems/subsets-ii/

**17. Letter Combinations of a Phone Number**

https://leetcode.com/problems/le ... -of-a-phone-number/

**22. Generate Parentheses**

https://leetcode.com/problems/generate-parentheses/

**51. N-Queens**

https://leetcode.com/problems/n-queens/

**254. Factor Combinations**

https://leetcode.com/problems/factor-combinations/

**301. Remove Invalid Parentheses**

https://leetcode.com/problems/remove-invalid-parentheses/

**491. Increasing Subsequences**

https://leetcode.com/problems/increasing-subsequences/

**37. Sudoku Solver**

https://leetcode.com/problems/sudoku-solver/

**52. N-Queens II**

https://leetcode.com/problems/n-queens-ii/

**93. Restore IP Addresses**

https://leetcode.com/problems/restore-ip-addresses/

**131. Palindrome Partitioning**

https://leetcode.com/problems/palindrome-partitioning/

**Lint-10. String Permutation II**

https://www.lintcode.com/problem ... tion-ii/description

**Lint-570. Find the Missing Number II**

https://www.lintcode.com/problem ... mber-ii/description

**Lint-680. Split String**

https://www.lintcode.com/problem/split-string/description

### 二叉树：

**113. Path Sum II**

https://leetcode.com/problems/path-sum-ii/

**257. Binary Tree Paths**

https://leetcode.com/problems/binary-tree-paths/

**Lint-246. Binary Tree Path Sum II**

https://www.lintcode.com/problem/binary-tree-path-sum-ii/solution

**Lint-376. Binary Tree Path Sum**

https://www.lintcode.com/problem/binary-tree-path-sum/solution

**Lint-472. Binary Tree Path Sum III**

https://www.lintcode.com/problem ... sum-iii/description

### 图：

**140. Word Break II**

https://leetcode.com/problems/word-break-ii/

**494. Target Sum**

https://leetcode.com/problems/target-sum/

**1192. Critical Connections in a Network**

https://leetcode.com/problems/critical-connections-in-a-network/

**126. Word Ladder II**

https://leetcode.com/problems/word-ladder-ii/

**290. Word Pattern**

https://leetcode.com/problems/word-pattern/

**291. Word Pattern II**

https://leetcode.com/problems/word-pattern-ii/

# 第十章 数据结构

## 基本问题：

(1) 本章按照数据结构分类一些问题，和之前按算法分类的题目相比可能会有重复，因为一道题可能有多个标签。

(2) 对于每种数据结构，需要先学习掌握其基本原理，优缺点，复杂度，和对应语言中的API用法。对于其基本的实现方式也要了解。

(3) Array，Matrix，String，Hash都是一些常用的数据结构，一般在各种题里都会用到，这里主要列举一些没有涉及到其他算法的题目。

(4) Linked List往往自成一类，会涉及到一些pointer操作，需要细心。

(5) Queue一般用在BFS里面比较多，这里不单独列举了。

(6) Heap， Stack往往和其他知识点混用，但自己单独出题也可以。

(7) Trie，Union Find， Sweep Line的套路比较明显，需要记住模板。

(8) Binary Index Tree 和Segment Tree涉及到的题目有限，需要记住模板。Segment Tree解法一般来说可以覆盖BIT能解决的问题，但是BIT写起来短一些。

(9) 复合数据结构里面LRU和LFU相对比较重要。其他的在掌握基本数据结构即复杂度之后，可以随机应变。

## 数据结构 题目列表：

（必背：紫色；核心：蓝色；重点：绿色；普通：黄色；默认是LeetCode，如果是LintCode会以Lint开头）

### Array & Matrix:

**442. Find All Duplicates in an Array**

https://leetcode.com/problems/find-all-duplicates-in-an-array/

**48. Rotate Image**

https://leetcode.com/problems/rotate-image/

**54. Spiral Matrix**

https://leetcode.com/problems/spiral-matrix/

**73. Set Matrix Zeroes**

https://leetcode.com/problems/set-matrix-zeroes/

**289. Game of Life**

https://leetcode.com/problems/game-of-life/

### String:

**6. ZigZag Conversion**

https://leetcode.com/problems/zigzag-conversion/

**13. Roman to Integer**

https://leetcode.com/problems/roman-to-integer/

**14. Longest Common Prefix**

https://leetcode.com/problems/longest-common-prefix/

**68. Text Justification**

https://leetcode.com/problems/text-justification/

**443. String Compression**

https://leetcode.com/problems/string-compression/

### Linked List:

**2. Add Two Numbers**

https://leetcode.com/problems/add-two-numbers/

**21. Merge Two Sorted Lists**

https://leetcode.com/problems/merge-two-sorted-lists/

**25. Reverse Nodes in k-Group**

https://leetcode.com/problems/reverse-nodes-in-k-group/

**82. Remove Duplicates from Sorted List II**

https://leetcode.com/problems/re ... rom-sorted-list-ii/

**83. Remove Duplicates from Sorted List**

https://leetcode.com/problems/remove-duplicates-from-sorted-list/

**86. Partition List**

https://leetcode.com/problems/partition-list/

**92. Reverse Linked List II**

https://leetcode.com/problems/reverse-linked-list-ii/

**138. Copy List with Random Pointer**

https://leetcode.com/problems/copy-list-with-random-pointer/

**141. Linked List Cycle**

https://leetcode.com/problems/linked-list-cycle/

**148. Sort List**

https://leetcode.com/problems/sort-list/

**160. Intersection of Two Linked Lists**

https://leetcode.com/problems/intersection-of-two-linked-lists/

**203. Remove Linked List Elements**

https://leetcode.com/problems/remove-linked-list-elements/

**206. Reverse Linked List**

https://leetcode.com/problems/reverse-linked-list/

**234. Palindrome Linked List**

https://leetcode.com/problems/palindrome-linked-list/

**328. Odd Even Linked List**

https://leetcode.com/problems/odd-even-linked-list/

**445. Add Two Numbers II**

https://leetcode.com/problems/add-two-numbers-ii/

**142. Linked List Cycle II**

https://leetcode.com/problems/linked-list-cycle-ii/

**876. Middle of the Linked List**

https://leetcode.com/problems/middle-of-the-linked-list/

### Hash:

**706. Design HashMap**

https://leetcode.com/problems/design-hashmap/

**49. Group Anagrams**

https://leetcode.com/problems/group-anagrams/

**128. Longest Consecutive Sequence**

https://leetcode.com/problems/longest-consecutive-sequence/

**560. Subarray Sum Equals K**

https://leetcode.com/problems/subarray-sum-equals-k/

**953. Verifying an Alien Dictionary**

https://leetcode.com/problems/verifying-an-alien-dictionary/

**290. Word Pattern**

https://leetcode.com/problems/word-pattern/

### Heap:

**23. Merge k Sorted Lists**

https://leetcode.com/problems/merge-k-sorted-lists/

**295. Find Median from Data Stream**

https://leetcode.com/problems/find-median-from-data-stream/

**347. Top K Frequent Elements**

https://leetcode.com/problems/top-k-frequent-elements/

**692. Top K Frequent Words**

https://leetcode.com/problems/top-k-frequent-words/

**767. Reorganize String**

https://leetcode.com/problems/reorganize-string/

**973. K Closest Points to Origin**

https://leetcode.com/problems/k-closest-points-to-origin/

**480. Sliding Window Median**

https://leetcode.com/problems/sliding-window-median/

**703. Kth Largest Element in a Stream**

https://leetcode.com/problems/kth-largest-element-in-a-stream/

### Stack:

**155. Min Stack**

https://leetcode.com/problems/min-stack/

**20. Valid Parentheses**

https://leetcode.com/problems/valid-parentheses/

**85. Maximal Rectangle**

https://leetcode.com/problems/maximal-rectangle/

**224. Basic Calculator**

https://leetcode.com/problems/basic-calculator/

**227. Basic Calculator II**

https://leetcode.com/problems/basic-calculator-ii/

**394. Decode String**

https://leetcode.com/problems/decode-string/

**1249. Minimum Remove to Make Valid Parentheses**

https://leetcode.com/problems/mi ... -valid-parentheses/

### Monotonic Stack:

**300. Longest Increasing Subsequence (Patience Sort)**

https://leetcode.com/problems/longest-increasing-subsequence/

**84. Largest Rectangle in Histogram**

https://leetcode.com/problems/largest-rectangle-in-histogram/

**239. Sliding Window Maximum**

https://leetcode.com/problems/sliding-window-maximum/

**1019. Next Greater Node In Linked List**

https://leetcode.com/problems/next-greater-node-in-linked-list/

### Trie:

**208. Implement Trie (Prefix Tree)**

https://leetcode.com/problems/implement-trie-prefix-tree/

**211. Design Add and Search Words Data Structure**

https://leetcode.com/problems/de ... rds-data-structure/

**1032. Stream of Characters**

https://leetcode.com/problems/stream-of-characters/

### Union Find:

**200. Number of Islands**

https://leetcode.com/problems/number-of-islands/

**305. Number of Islands II**

https://leetcode.com/problems/number-of-islands-ii/

**323. Number of Connected Components in an Undirected Graph**

https://leetcode.com/problems/nu ... n-undirected-graph/

### Sweep Line:

**Lint-391. Number of Airplanes in the Sky**

https://www.lintcode.com/problem ... the-sky/description

**252. Meeting Rooms**

https://leetcode.com/problems/meeting-rooms/

**253. Meeting Rooms II**

https://leetcode.com/problems/meeting-rooms-ii/

### Binary Index Tree & Segment Tree:

**307. Range Sum Query - Mutable**

https://leetcode.com/problems/range-sum-query-mutable/

**327. Count of Range Sum**

https://leetcode.com/problems/count-of-range-sum/

**715. Range Module**

https://leetcode.com/problems/range-module/

**315. Count of Smaller Numbers After Self**

https://leetcode.com/problems/co ... numbers-after-self/

**493. Reverse Pairs**

https://leetcode.com/problems/reverse-pairs/

### Complex Data Structure:

**146. LRU Cache**

https://leetcode.com/problems/lru-cache/

**460. LFU Cache**

https://leetcode.com/problems/lfu-cache/

**211. Design Add and Search Words Data Structure**

https://leetcode.com/problems/de ... rds-data-structure/

**380. Insert Delete GetRandom O(1)**

https://leetcode.com/problems/insert-delete-getrandom-o1/

**528. Random Pick with Weight**

https://leetcode.com/problems/random-pick-with-weight/

**588. Design In-Memory File System**

https://leetcode.com/problems/design-in-memory-file-system/

**981. Time Based Key-Value Store**

https://leetcode.com/problems/time-based-key-value-store/

**1396. Design Underground System**

<https://leetcode.com/problems/design-underground-system/>

# 第十一章 动态规划

## 基本问题：

(1) 动态规划更准确的说是一种数学思想，而不是一种算法。学习曲线相对于前面的算法会比较陡峭，如果是有天赋的大佬，可能可以很快领悟。但是对于大部分平均水平的同学，可能需要前后间隔几个礼拜甚至几个月，反复思考两三遍才能顿悟并运用。所以作为初学者，一时半会想不明白没关系，隔几天回来再多看几次就能渐渐理解了。

(2) 不过针对目前的面试，除了少数那几家公司之外，动态规划的出现频率其实没有那么高，而且主要也都是中等难度的题目。所以如果准备时间有限，建议优先把时间放在前面的算法上，动态规划可以先看几道中等难度经典题，其他的题目后面有时间再看。

(3) 关于一道题是用动态规划还是用贪心法，一般来说时间复杂度类似的时候优先用动态规划，因为通用性、可解释性都比较强。而自己凭空想出来的贪心法，不但不容易解释，而且很容易是错的，面试风险相对比较高。不过有一些题目确实是贪心法最优，作者在后面也列出了几题，如果碰到原题或者类似题，可以参考。

(4) 对于新手而言，在学习动态规划的时候，看懂题目在问什么之后就可以在网上找答案了，别自己瞎折腾。网上各种大佬的博客有详细的图文解释，慢慢揣摩理解。

(5) 动态规划的一般思路是数学归纳法，就是用递推的方式把大问题（最终问题）分解为小问题（前置问题），然后一路倒推到边界；在边界附近计算出初始状态后，再原路反向往回计算，最后得到所求解。所以对于绝大部分题目，都需要遵循：分解子问题，写出转移方程，描述边界条件，计算出最终解这几个步骤。

(6) 有些动态规划问题，可以通过滚动数组的方式优化空间复杂度，一般可以降一个维度。但是要注意运算的方向，需要避免前序的结果在被用到之前就被覆盖掉的情况。

(7)大部分动态规划都是求解“可行性”，“最值”问题，如果有些题目要求输出结果，也可以考虑用“打印路径”的方式。

(8) 很多问题，通过细微的改一些条件，就会变成另外一道题，解法思路会产生明显差异，所以审题要小心。比如背包类问题，是否可以重复选同一个物品，是否有重复物品，求解最大重量还是最大价值， 背后的原理可能会产生变化。有时候是组合问题，有时候是排列问题，还叠加了是否可以重复的情况，需要透彻的理解。另外在解法上，比如说，正着走一遍循环和倒着走一遍循环可能代表的是两种不同的思考方式，这些往往需要反复细致的理解才能完善自己的思维体系。

(9) 有些问题需要求“所有可行解”，这时候往往会使用搜索（DFS，BFS）的方法。但为了进行时空优化，记忆化搜索也会常常被用到。其实DFS记忆化搜索和常规动态规划写法常常是一个思维的两种实现方式，在不同的题目中各有优劣。

(10) 在面试动态规划的时候，重点在于能够比较清晰地画图描述并解释清楚所写的动态方程，让面试官理解你的思路，注意初始化以及for循环的起始条件。至于代码本身，往往是for循环为主，一般也不长。

## 动态规划 题目列表：

（必背：紫色；核心：蓝色；重点：绿色；普通：黄色；默认是LeetCode，如果是LintCode会以Lint开头）

### Backpack:

**Lint-92. Backpack**

https://www.lintcode.com/problem/backpack/description

**Lint-125. Backpack II**

https://www.lintcode.com/problem/backpack-ii/description

**Lint-440. Backpack III**

https://www.lintcode.com/problem/backpack-iii/description

**Lint-562. Backpack IV**

https://www.lintcode.com/problem/backpack-iv/description

**Lint-563. Backpack V**

https://www.lintcode.com/problem/backpack-v/description

**Lint-564. Backpack VI (Combination Sum IV)**

https://www.lintcode.com/problem/combination-sum-iv/description

**Lint-971. Surplus Value Backpack**

https://www.lintcode.com/problem ... ackpack/description

**474. Ones and Zeroes**

https://leetcode.com/problems/ones-and-zeroes/

### Single Sequence:

**139. Word Break**

https://leetcode.com/problems/word-break/

**121. Best Time to Buy and Sell Stock**

https://leetcode.com/problems/best-time-to-buy-and-sell-stock/

**122. Best Time to Buy and Sell Stock II**

https://leetcode.com/problems/best-time-to-buy-and-sell-stock-ii/

**123. Best Time to Buy and Sell Stock III**

https://leetcode.com/problems/be ... and-sell-stock-iii/

**188. Best Time to Buy and Sell Stock IV**

https://leetcode.com/problems/best-time-to-buy-and-sell-stock-iv/

**256. Paint House**

https://leetcode.com/problems/paint-house/

**265. Paint House II**

https://leetcode.com/problems/paint-house-ii/

**Lint-843. Digital Flip**

https://www.lintcode.com/problem/digital-flip/description

### Double Sequences:

**10. Regular Expression Matching**

https://leetcode.com/problems/regular-expression-matching/

**44. Wildcard Matching**

https://leetcode.com/problems/wildcard-matching/

**72. Edit Distance**

https://leetcode.com/problems/edit-distance/

**97. Interleaving String**

https://leetcode.com/problems/interleaving-string/

**115. Distinct Subsequences**

https://leetcode.com/problems/distinct-subsequences/

**1143. Longest Common Subsequence**

https://leetcode.com/problems/longest-common-subsequence/

### Sections:

**312. Burst Balloons**

https://leetcode.com/problems/burst-balloons/

**516. Longest Palindromic Subsequence**

https://leetcode.com/problems/longest-palindromic-subsequence/

**87. Scramble String**

https://leetcode.com/problems/scramble-string/

### Matrix:

**62. Unique Paths**

https://leetcode.com/problems/unique-paths/

**63. Unique Paths II**

https://leetcode.com/problems/unique-paths-ii/

**64. Minimum Path Sum**

https://leetcode.com/problems/minimum-path-sum/

**85. Maximal Rectangle**

https://leetcode.com/problems/maximal-rectangle/

**221. Maximal Square**

https://leetcode.com/problems/maximal-square/

**361. Bomb Enemy**

https://leetcode.com/problems/bomb-enemy/

### Others：

**91. Decode Ways**

https://leetcode.com/problems/decode-ways/

**Lint-394. Coins in a Line**

https://www.lintcode.com/problem/coins-in-a-line/description

**132. Palindrome Partitioning II**

https://leetcode.com/problems/palindrome-partitioning-ii/

**279. Perfect Squares**

https://leetcode.com/problems/perfect-squares/

**639. Decode Ways II**

https://leetcode.com/problems/decode-ways-ii/

**Lint-395. Coins in a Line II**

https://www.lintcode.com/problem/coins-in-a-line-ii/description

**Lint-396. Coins in a Line III**

https://www.lintcode.com/problem/coins-in-a-line-iii/description

### Greedy:

**55. Jump Game**

https://leetcode.com/problems/jump-game/

**45. Jump Game II**

https://leetcode.com/problems/jump-game-ii/

**763. Partition Labels**

https://leetcode.com/problems/partition-labels/