《PAT | 蓝桥 | LeetCode学习路径 & 刷题经验 by 柳婼 》

目录

计算机系学生为什么要刷算法

很多人（尤其是一些初入门开发的计算机学生）可能会觉得算法在开发过程中的用处不大，甚至很多没有认真学过算法的老程序员都会这样认为，然而并不是这样～

首先，刷算法可以培养清晰的逻辑思维能力，改变人的思维方式，使人对待复杂的代码问题更容易看透本质～在未来做工程项目的过程中也会慢慢发现，当年认真学过的算法也并非当初想得那样毫无用处，反而能提高解决问题的效率，使自己能用更简单的方法、更精简的代码解决实际问题～

其次，看过很多IT公司招聘信息的人就会发现，他们都会优先选择有一定算法基础的人，尤其是一些大公司在面试过程中，对于应届生，更多考察的是数据结构与算法、计算机网络、操作系统等这些计算机基础课程，而并非都是在考察实际应聘岗位的相关技术、工具、知识等～

再者，提高算法能力不是一蹴而就的，需要静下心来用大片的时间去看书、学习、刷题、找bug或者参加竞赛，也许市面上有很多开发类的培训班，如前端、JavaWeb、安卓之类，但鲜有算法类的培训，因为学习算法是实实在在、需要自己付出努力的过程，自己对题目的思考、学习、钻研，别人无法替代，也不可能通过培训速成。所以这也是为什么很多公司会在应聘的笔试面试环节考察算法能力的原因，也许语言的特性、工具的使用、开发的技巧可以短期培训，但数据结构和算法的能力足以证明应聘者在计算机知识方面的硬实力～而且很多大公司在考察算法时都是白板编程（关于白板编程我

会在文章最后讲到），这比写代码更能考验应试者的水平～

最后，如果从短期角度来说，可能刷算法能带来的一些看得见的收益会给自己更多的动力，比如：PAT乙级刷完可以让自己在C语言二级或者C语言期末考试中获得一个高分；PAT甲级能让自己在考浙大时候直接抵复试中机试的成绩；PAT甲级考高分能让自己收到一些大公司的面试邀请；LeetCode能够让自己在面试很多大公司尤其是国外一些公司（Google、微软、Facebook、Amazon等）时正确解答出笔试和面试的算法问题等…无论是长期还是短期来看，刷算法对于有追求的计算机学生都是必经之路～

虽然刷算法是需要自己一道道题目亲力亲为的，但我却可以通过总结和讲解我大学生涯刷算法过程中的经验，帮助学习算法路上的小可爱们节约时间、少踩一些坑、多走一些捷径，或者通过blog分享代码的方式为某道具体的算法题目提供自己的题解思路和清晰简短的代码～

什么是OJ，以及对AC、WA、TLE、CE、RE、MLE、PE等状态术语的解释

OJ（Online Judge）是指在线判题系统，将代码提交给OJ后，OJ会在线检测程序源代码的正确性，并返回结果～国内著名的OJ系统有POJ（北京大学OJ）、杭电OJ（参加过ACM的人都知道）等，PAT的官方题库、蓝桥杯题库和LeetCode也是OJ系统，都可以在线提交代码并得到返回结果～PAT考试过程中使用的就是和平时刷题题库一样的OJ判题系统，而蓝桥杯在考试的时候只能提交答案，不能实时看到提交的答案是否正确，但蓝桥杯平时刷题练习的题库是OJ～相比较而言，PAT和蓝桥需要的都是完整的有输入输出的代码段，而LeetCode并不需要输入输出，而是将需要填写的代码包在一个Solution类里，甚至会给好Solution类中的函数，涉及到树的题目还会给出树的定义，我们只需要在这个函数中填充代码并按照题目的要求返回相应类型的值即可～

刷算法OJ过程中提交答案后会返回的一些状态术语，`AC`（Accepted = 答案正确），`WA`（Wrong Answer = 答案错误），`TLE`（Time Limit Exceeded = 运行超时 / 时间超限），`CE`（Compile Error = 编译错误），`RE`（Runtime Error = 运行时出错），`MLE`（Memory Limit Exceeded = 内存超限），`PE`（Presentation Error = 格式错误），如果刷过LeeTCode的人对这些应该比较熟悉～所以我们经常会说`AC`了几道题、`TLE`了或者`WA`了之类的～而PAT的OJ里返回的是中文反馈信息，例如答案正确、格式错误、内存超限、运行超时、段错误等～（后面我会讲解PAT中遇到运行超时、段错误、格式错误等不同反馈信息时应该排查bug的方向）

PAT乙级和甲级是什么

和其他算法竞赛考试比起来，知道PAT的人确实不多～所以我在这里简单介绍一下～

PAT是一场算法考试，考试的分数只要不是0分就会获得PAT相应等级的证书～

PAT有三个等级：乙级（Basic）、甲级（Advanced）、顶级（Top），考试时间都是3小时，满分都是100分，每年组织三次考试，分别在3月、9月、12月～一般安排在月初或月中的周六，具体比赛时间在报名前才会公布～全国有很多考点，每个考点所拥有的开发环境和版本也不一样，可以在官网（https://www.patest.org/location）这里看到各考点的开发环境～一般情况下我们会选择最近的那个考点，不过也不一定，比如离你最近的那个考点提供的`C/C++`语言编译环境没有`Dev-C++`只有`Visual Studio`，而你又非常喜欢`Dev-C++`写代码，那也可以选有`Dev-C++`的考点～比如在我第一次参加PAT的时候，南京当时还没有考点，我就去了提供`Dev-C++`的浙江大学紫金港校区参加考试～还有一点就是，报名要趁早，有的时候某个考场报满了就无法在该考场报名只能去其他考场报名啦（比如我一个小伙伴报名晚了，南京考场都报满了，只能报了苏州，做高铁去苏州参加PAT考试…）报名费之前是200元，现在是256元～（比蓝桥杯竞赛报名费便宜耶- -）

整个PAT考试的流程是：等到开放报名入口的时候在官网报名 -> 自己把网页上的准考证打印一下 -> 考试当天带着身份证、准考证和笔去考场 -> 在电脑上做题目（和平时刷题一样提交后就能看到这道题的得分）-> 等到3小时考试时间到（考100分就可以提前交卷出考场啦） -> 考完等大概半小时，考点当场打印证书，拿着证书回家～

乙级是5道中文题目，分别是15分、20分、20分、20分、25分，官网题库链接：https://pintia.cn/problem-sets/994805260223102976/problems

甲级是4道英文题目，分别是20分、25分、25分、30分，官网题库链接：https://pintia.cn/problem-sets/994805342720868352/problems

顶级是3道英文题目，由于顶级题目偏难，一般ACM大佬才参加顶级，加上我自己也没有参加过，所以就不多介绍啦～（相信参加顶级的一定对PAT比较了解不需要我来介绍啦= =）

最近PAT题库中还推出了基础编程题目集，比乙级还要更基础（简单）一些，如果觉得乙级的难度还不够简单可以去体验一下题库～链接：https://pintia.cn/problem-sets/14/problems （一般情况下我觉得直接从乙级or甲级开始刷题就ok啦～）

刷题的话还是在官网pintia（PTA）的题库刷比较好，我听说有些人是从牛客网或者其他地方刷题的，然而很多时候虽然可能题目相似，但内部具体的测试样例还是有些不一样的，所以同样一段代码提交上去可能会产生不同的结果，比如我博客某篇题解代码，很多人来评论说在牛客网上无法通过，但在官网能通过，说我的代码没有考虑周全，最后发现是牛客网的内部测试样例没有按照题意来设计…当然有的时候也会发现官网通过了牛客网没通过时确实是自己的代码有疏漏，没有考虑到一些边界情况，正好官网的判题样例也没有考虑到…（这个时候我会把自己的代码完善一下）～官网的题库从上到下的顺序就是每一次PAT考试产生的题目，在考试过程中也使用的是和官网OJ题库中同样的判题系统，相比较而言更真实可靠～所以还是推荐用官网PTA（https://pintia.cn/）直接刷题吖～

考PAT对学习 | 考研 | 工作的作用是什么

这也是很多人会在刚知道PAT的时候问我的问题～

学习方面：刷算法题当然对提高算法有很大帮助呀～刷完乙级，基本上就理解和掌握了基础的数据结构（数组、链表、字符串）～也会了一些比较基础的算法（比如排序、查找、递归），最重要的是，很多人刚入代码的坑，自己的思维无法准确的用代码来描述，想到的解决思路却无法用代码表达出来，乙级刷完就完全可以解决这个问题啦～刷完甲级，首先已经能看懂英文的算法题目描述了～然后还能理解和熟练使用一些经典的算法，比如最短路径、并查集、树、图、深度优先搜索、广度优先搜索、贪心算法、拓扑排序、哈希等等～对自己的算法能力的提高是显著的～

考研方面：首先说浙大计算机院，浙大计算机系的考研是分为初试和复试，复试又分为机试和面试，机试的题目就是类似于PAT甲级考试的题目（每年机试过后的题目也会增加到甲级题库后面），而且机试可以申请免试，只要具有前一年的3月、9月、12月的任意一次PAT甲级的证书即可～要知道把所有的机会都压在机试那一次是比较冒险的，一般情况下大家都会在3月复试之前拿着前一年3次PAT考试中最高的那次成绩申请浙大计算机系考研的复试机试免考，这样就可以在复试中更加稳妥啦～不仅是甲级的成绩可以免机试，顶级的成绩也可以，顶级成绩乘以1.5可以充当机试成绩（最高100分），例如顶级考了67分及以上，就可以和甲级100分一样～我认识的很多其他学校的考研or保研的学生也会参加PAT考试，为了得到一张满意成绩的证书，这样就能在考研or保研的面试过程中更加具有竞争力～

工作方面：首先说PAT企业联盟，PAT和很多企业合作，这也是为什么PAT定在3月（春招）9月（秋招）12月（秋招补招or寒假实习）的原因～在PAT报名的时候填写自己的个人信息，如果是有求职意向的，可以对自己的个人简历写详细些，让企业知道你是打算找工作的并且更加了解你拥有的技能和想应聘的职业，这样在考完后会收到很多企业的邀请面试的邮件～也可以主动拿着PAT的证书在有效期内（有的是一年，有的是两年，不同公司规定不同）去这些和PAT联盟的企业，一般都可以免笔试并优先录取，比如Google中国的要求是：甲级前15名、顶级前20名且80分以上，2年有效，可以免除笔试，直接进入Google的面试阶段；微软的要求是：甲级90分以上，并且排名在当场考试前30名，同时符合微软项目招聘的要求（毕业时间和专业等），可免技术笔试进面试...还有其他一些公司，例如小米、华为、网易、美团、雅虎、阿里、淘宝、腾讯等...都对PAT的成绩以及能够给予的优惠有详细说明，具体企业联盟详情在官网的这里可以看到：https://www.patest.org/company 。比如我收到过百度、网易雷火的面试邀请邮件，还接到过一些企业的电话，但是由于当时我才大二大三没有打算找工作所以也就没有去参加面试...而且我的个人简历里面也明确写了目前不打算找工作，所以打电话找我的企业不算多，不过身边有朋友通过PAT进了网易，接到微软的面试邀请等～所以对于求职还是很有帮助的～而且身边很多考PAT的都是非科班打算转计算机的，如果想要找计算机类的工作，考PAT通过企业联盟应聘确实是个可行的捷径～即使不是PAT企业联盟中的企业，如果公司知道PAT（我刚知道PAT时候PAT名气和影响力还很小，这些年逐渐扩大）对丰富简历or面试应聘也是很有帮助的～

刷PAT乙级和甲级需要掌握哪些知识

1. 官网对于PAT乙级考试应具备的能力是这样描述的：

> 1. 基本的C/C++的代码设计能力，以及相关开发环境的基本调试技巧；
> 2. 理解并掌握最基本的数据存储结构，即：数组、链表；
> 3. 理解并熟练编程实现与基本数据结构相关的基础算法，包括递归、排序、查找等；
> 4. 能够分析算法的时间复杂度、空间复杂度和算法稳定性；
> 5. 具备问题抽象和建模的初步能力，并能够用所学方法解决实际问题。

概括来说乙级的要求就是，一门编程语言+最基本的数据结构～

2. 官网对于PAT甲级考试应具备的能力是这样描述的：

> 1. 具有充分的英文阅读理解能力；
> 2. 理解并掌握基础数据结构，包括：线性表、树、图；
> 3. 理解并熟练编程实现经典高级算法，包括哈希映射、并查集、最短路径、拓扑排序、关键路径、贪心、深度优先搜索、广度优先搜索、回溯剪枝等；
> 4. 具备较强的问题抽象和建模能力，能实现对复杂实际问题的模拟求解。

概括来说甲级的要求就是，在乙级的基础上，能阅读英文题意+基础数据结构+一些经典算法～

其实我们在准备PAT的过程中无需关心乙级和甲级考察哪些知识点，因为目前PAT的题库中题目已经足够的多，以上知识点在题库中都已经完全涉及，我们只需要刷完题库、熟练掌握题库中的那些题型就能掌握到PAT需要考察的所有知识点～甚至能根据题库中不同类型题目出现的频率推断常考的题型是哪些、复习的重点应该放在哪方面等（关于常考题型和复习重点我在下文会提及）～

关于PAT刷题中编程语言选择和开发环境IDE选择的一些愚见

虽然PAT不规定编程语言，在刷题的时候也可以看到支持31种编译器，语言包括 C 、 C++ 、 Java 、 Python 、 Ruby 、 Go 、 JavaScript 、 PHP 等，但是根据考点提供的开发环境可以看到，基本上只有支持 C/C++ 的 Dev-C++ 、 CodeBlocks 、 Visual Studio 、 VC++6.0 等，和支持 Java 的 Eclipse 、 NetBean 、 MyEclipse 等，少部分考场会有支持 Python 的 PyCharm ...虽然之前我在考场遇到过直接用记事本写 Python 代码的大佬（因为我当时的那个考场没有 Python 编译器），也可以直接通过网页OJ提交看报错信息，然而如果要调试查看变量的值的话还是非常的不方便...所以那些不是主流的语言应该很少会有人使用它们刷PAT吧（毕竟不像LeetCode大家都是在网页上直接在线写代码提交），刷PAT用的最多的编程语言还是 Java 和 C/C++ ，那关于 C 、 C++ 、 Java 如何选择呢?

C语言作为一门面向过程语言，是很多计算机初学者必学的一门编程语言，绝大多数计算机系大一就会教C语言，所以大多数人对C语言还是很熟悉的（不熟悉没学过也没有关系，书籍推荐部分我会讲到如何高效学习C语言）～C语言也确实是一门执行效率高的语言，然而最大的缺陷是，它缺少一些必要的、封装好的、可以直接使用的数据结构，比如我们知道数据结构里面除了数组链表还有栈、队列、集合、映射等，做题的时候会需要用到这些数据结构，如果仅仅使用C语言，可能需要手动用数组或者链表实现它再使用，而一些比C语言高级的语言（这里的高级是指高度封装、对程序员更加友好）已经封装实现好了一些常用的数据结构，需要用的时候直接拿来用就可以～比如排序，使用C语言可能要手写一个是冒泡排序或者快排，而更高级的语言已经封装好了较为高效的排序算法，需要用的时候加个 #include <algorithm> 头文件然后直接一行 sort(begin(a), end(a); 代码即可实现排序，这样能让我们在刷算法过程中把更多的精力集中在如何解决算法逻辑而非如何用语言实现一个常用的数据结构方面，大大提高做题的速度、降低代码的错误率～

我知道大学计算机系或者想要做开发的必学 Java ，包括绝大多数人面向对象的入门语言也是 Java ，所以很多人就会想用 Java 刷算法，毕竟是将来工作和开发需要用到的语言～ Java 中确实有很多好用的集合框架封装好了各类常用数据结构，然而 Java 作为一门优秀的工程开发语言，在刷算法的时候却有些显得力不从心。我在刚刷算法的时候也是使用的 Java ，一开始PAT乙级的题目还好，勉强只有几题无法AC，都是因为运行超时...而刷甲级的时候就明显感觉出 Java 在运行时间方面的劣势了，有些题目连 C/C++ 都需要优化算法才能保证不超时，更何况用 Java ...而PAT对于同一道题目不同语言的时间限制要求是统一的，如果用 C/C++ 没有超时的题目一样的算法思路换了 Java 描述就超时了...岂不是很心塞...（虽然官方说可以修改 Java 输入方式提高运行效率，然而我试过了，还是会有一些30分的题目始终无法完全AC...）如果你不介意不能在PAT甲级中获得一个满意的成绩，只是希望通过PAT题库来锻炼自己的代码能力和编程语言运用能力，倒是可以考虑全程使用 Java 刷题...（当然我感觉如果单纯想提高代码能力和算法水平，刷LeetCode会更节约时间）～如果执意想用 Java 学算法，而且是0基础，可以参考下文从0基础到蓝桥杯省赛一等奖的高效学习路径 & 书籍推荐部分中关于 Java 组如何备考蓝桥杯的学习路径和参考书籍～

接下来说 C++ 。可能对于没有接触过 C++ 的人来说这是一门陌生的语言，毕竟很多人对它的第一印象是：这是一门全新的和面向过程的C语言完全不一样的面向对象编程语言，进而产生畏惧感，其实不然。下面是一段关于 C++ 的简介：

> C++是C语言的继承，它既可以进行C语言的过程化程序设计，又可以进行以抽象数据类型为特点的基于对象的程序设计，还可以进行以继承和多态为特点的面向对象的程序设计。C++擅长面向对象程序设计的同时，还可以进行基于过程的程序设计，因而C++就适应的问题规模而论，大小由之。
>
> C++不仅拥有计算机高效运行的实用性特征，同时还致力于提高大规模程序的编程质量与程序设计语言的问题描述能力。

所以说， C++ 是一门可以兼容C语言语法的面向对象语言。其实对于刷算法来说，它是否是面向对象语言并不重要，它关于面向对象的部分（继承封装多态之类）我们也可以完全不学习，而且对于已经懂C语言的人来说，想要用 C++ 刷算法，几乎没有多少学习成本。也就是说，你完全可以在 C++ 的文件里面使用C语言的语法，这不会产生报错。而那些好用的 C++ 特性又可以随意的使用，像是增强版功能的C语言。对于刷算法来说， C++ 最大的好处是拥有 STL （标准模版库），就像 Java 的集合框架一样， C++ 里面封装了很多常用的数据结构，而只需掌握STL和C++的一些和C语言稍有区别的基本语法，就完全可以使用 C++ 来刷PAT、LeetCode和蓝桥杯，这对算法之路是大有裨益的～下面是一段关于 STL 的简介：

> STL是Standard Template Library（标准模板库）的简称，从根本上说，STL是一些“容器”的集合，这些“容器”有list，vector，set，map等，STL也是算法和其他一些组件的集合。STL的目的是标准化组件，这样就不用重新开发，可以使用现成的组件。STL现在是C++的一部分，因此不用安装额外的库文件。

什么意思呢，比如说刷算法过程中需要用到集合 set ，我们知道 set 的特点是集合内的元素不重复（特别地，在 C++ 里面， set 是会自动排序的集合， unordered_set 是不会自动排序的集合），比如在set里分别放进 4、1、1、2、3 这几个元素，set里会自动变成 1、2、3、4 ，如果我们只会C语言，可能需要一个个把数据放到数组里面，然后手动编写一些代码，检查进来的每一个元素在数组中是否已经存在，如果存在就不要再放进数组，最后对整个数组进行排序，才能达到set的效果。但是如果直接使用 C++ ，我们就可以在头文件里面加个 #include <set> ，然后直接把 set 当作一个类似于数组的容器，把所有元素直接丢到自己定义的这个set类型的容器里，就会自动实现去重（去除重复元素）和排序的步骤～比如说题目要求将最后所有的答案去重后按从小到大的顺序输出，就可以直接将所有的答案元素放到set里面输出即可～这样我们在刷题过程中就能更好地集中精力解决代码思路、算法方面的问题，而不是一个简单的答案输出或语法方面的问题，在考试过程中也能大大地节省时间，降低代码的错误率～

更重要的是，使用 C++ 能写出更加精简的代码，简短的代码对于刷算法过程中思路的梳理和把握是非常有好处的～看过我blog代码的人应该都会发现，我每道题的代码绝大多数都比较简短，这并非是我做完题目后将代码修修改改有意为之，你们所看到的代码就是我当时做这道题的过程中产生的代码，可能是因为曾经刷LeetCode养成的好习惯，因为LeetCode评论区有很多优秀简洁的代码，我会不断地向他们学习，写代码过程中也会想好思路再写，写的过程中也会不断反问自己是不是一定要这么复杂，进而不断优化自己的思路。因为代码写长了就容易产生错误，更容易使自己思维混乱，一旦题目没能AC，找bug和调试的过程也是非常地麻烦～而且我已经在blog上尽可能还原了当时写完题目后产生的代码原样，我不希望通过后期经过人工修改刻意简化的代码让很多初学者接触学习，这对初学者理解代码思路也是十分不利的。曾经 Github 上有很多大佬帮我优化过一些代码，我只接受了其中部分代码，如果我觉得这不是刚看到题目写代码时候就能想到的思路，就不会接受这段代码～

我之前写的一篇教程 《从放弃C语言到使用C++刷算法的简明教程 by柳婼》 （https://www.liuchuo.net/333-3）很多人已经阅读过，里面也详细讲述了C++在C语言基础上扩充的一些新特性（如名称空间、 cin 和 cout 、 bool 变量、 const 常量、引用、 sort 排序、 auto 声明、 to_string() 、 stoi 和 stod 等）以及刷算法过程中常用的STL（如动态数组 vector 、集合 set 、映射 map 、栈 stack 、队列 queue 、在超时的时候很好用的 unordered_map 和 unordered_set 等），学习C++的成本已经比绝大多数踩过算法坑的人降低很多～

接下来说我关于开发环境 IDE 如何选择的看法～

经常有一些人会给我blog评论说，我的代码无法通过编译（大部分人都很和善，只是好奇地问我的代码为什么能通过OJ但是无法在他的编译器上运行，而有的人直接是以理直气壮的质问和嘲讽的语气说代码根本是错的少了头文件or错的代码还放blog上误导人之类的- -(⊙o⊙)我也是非常的无奈～）其实是这样的，编译器的种类、版本不同，所得到的结果自然不同。如果你使用的是 Visual Studio ，可能会有部分C99标准的库函数或语法是不被支持的，但是这在 C++11 中是支持的，所以会导致一段通过OJ的代码在VS上无法编译通过的情况。如果你使用的是 Dev-C++ ，可能忘记了在设置中包含 -std=c++11 让Dev支持C++11特性（我在简明教程的末尾也有特别提到如何让 Dev 支持 C++11 特性），否则一些 C++11 中的好用的函数可能会产生编译无法通过的情况～如果你用的是 Xcode 或 CLion 等，因为它们内置了较多的常用库函数，很多变量名可能在OJ中不是关键字但是在自己IDE上却是，更严格的标准和更智能的 IDE 确实会引起一些Warning甚至Error～如果你使用的是 VC++6.0 或者 Turbo C++ ，emmmmm人生苦短，为什么不考虑换个好用的IDE呢?

有的时候也会产生自己IDE上好好的，OJ上却编译错误的情况，每个OJ的判题系统有自己的编译环境，这个编译环境的版本配置应该会在它的OJ系统中阐述介绍，比如PAT编译 C/C++ 时用的是 gcc/g++ 4.7.2 ， Java 用的是 Javac 1.6.0 等，而每个 IDE 自带的编译器版本不同，我推荐的是 Dev-C++ ，因为绝大多数在 Dev-C++ （使用时请在设置中包含 -std=c++11 ）通过的代码基本都能顺利通过PAT的OJ，但也不是绝对，比如很多人就是习惯 Visual Studio （记得要注意考场上 Visual Studio 的版本，可能和你平时自己用的新版本的 VS 差别很大）或者 Code::Blocks ，我认为选什么编译器是次要的（只要不是用 VC++6.0 就好...），最重要的是选自己考场上提供的、熟悉的 IDE 进行平时的代码刷题练习～

因为我之前在网页上直接刷LeetCode养成了不使用IDE的习惯，所以我自己平时刷PAT题的时候都是直接打开 Sublime Text 就开始写，写完直接提交给PAT OJ，一般情况下就能AC了...如果没能通过OJ，我会再细看一遍修改一下代码，基本上如果是小问题就能直接解决了...如果实在遇到了难以解决的bug，我会用 XCode 运行调试～但是由于考场上没有 XCode 这么智能方便的IDE（如果考场上有，我的做题效率肯定翻倍提高...所以说考试时候用自己平时习惯的IDE是多么重要...极大提升AC题的效率...），我会在考前几天熟悉一下 Dev 的使用和调试等技巧，避免考场上出现慌乱不会用IDE的情况（我第一次参加PAT就因为在考场上不会使用 Dev 只能用记事本敲代码...最后竟然结果还不错...emmmm）

从0基础到PAT甲级100分的高效学习路径 & 书籍推荐

根据上文关于编程语言如何选择的分析，我就默认正在阅读的你接受了我的安利，决定用 C++ 愉快地刷PAT并且打算在PAT甲级中获得高分甚至满分啦～

首先，得学C语言～大多数人在大学里都是学过C语言的，如果你是刚入计算机坑的萌新，不会也没有关系，可以现在开始学嘛～ 《C语言入门经典》 （霍顿 / 清华大学出版社）是入门C语言的好书～因为作者很厉害～书在豆瓣的评分也很高～作为初学者，如果你对这种巨佬作者写的500多页的书比较抵触和畏惧，可以看 《啊哈C语言！逻辑的挑战（修订版）》 （啊哈磊 / 电子工业出版社），是关于C语言入门的风趣的编程启蒙书～豆瓣的评分8.3，也很不错～通过这本幽默风趣的书也能掌握C语言～

其次，得了解数据结构和一些基础的算法～我所说的了解，是指知道有哪些常用的数据结构，了解如何分析和比较不同算法的算法复杂度，不求能够像我当初学习一样直接用C语言手写实现每一个数据结构（当然这也花费了我大量的时间，对于刷算法来说暂时没有必要，代码能力可以通过大量刷OJ算法题目培养，而我只是提早培养了自己的代码能力而已），但求能看懂明白这些数据结构的概念含义～可能《数据结构》（严蔚敏 / 清华大学出版社）这本书让很多人产生了怀疑人生甚至想要放弃编程的想法，所以在了解数据结构这方面，我推荐阅读的是 《大话数据结构》 （程杰 / 清华大学出版社），至少不会像严奶奶的书那样大量使用很多伪代码让人难以理解，这本书的好处是能让人看得下去，使用C语言进行讲解示例，而且讲述了每种数据结构如何运用在我们熟悉的生活中，对数据结构的理解很有帮助～

接着，应该了解一下C++语言～我上面已经安利过 C++ 有哪些好处，但是我本人在学习算法过程中接纳并学习 C++ 经历了很大的波折（辛酸～），一开始我会了C语言，开始学 Java 做开发，当时觉得相比较而言 Java 真的比C更人性化更高级，里面封装了很多好用的函数，所以打算用 Java 刷算法，后来发现大量超时，又换回C语言刷算法，在刷题过程中看其他博主的博客发现很多使用了C++的语法看不懂，想找到一篇纯C语言实现的很难，而且大多数代码并不优雅，反而冗长地让人不想去了解～所以只能被迫学习C++语言，一开始买了《C++ Primer Plus》中文版，900+页的厚书，虽然很多和C语言内容重合看起来并不累，但看着很难挑出重点，书中有很多内容和特性都是更适合于做C++开发的人去学习，而对刷算法这件事儿并没有多大帮助，但也没有办法，毕竟对于一个算法路上的初学者来说，我并不了解哪些有用哪些没有用，每天带着那么厚的书去自习室很累，最后书被我用刀分成了3份每次带一小半减轻重量…也许正是经历过这些，我才会去写短短15页内容的 《从放弃C语言到使用C++刷算法的简明教程 by 柳婼》 （https://www.liuchuo.net/333-3），这个教程两三个小时就可以看完，编程语言更重要的是练习，将教程里面讲过的那些特性有些印象，在刷乙级or甲级的过程中能够想起并主动多写多用多练，希望可爱的你们不要像我一样一开始因为怕麻烦和畏惧，错过了这么好用的刷算法的语言～

最后，就是开始刷题啦～可能有些人会觉得，不是应该在刷题之前先补充点算法理论知识么，比如看些算法书之类的～确实，在我学习算法的过程中，刷题之前我阅读了一些算法类的书籍，比如《啊哈！算法》、《数据结构与算法分析：C语言描述》和《算法导论》，甚至还为了准备蓝桥杯竞赛读了《算法竞赛入门经典》和《大学程序设计课程与竞赛训练教材：算法设计编程实验》，阅读这些书确实让我补充了很多关于算法的理论知识，但也花费了我大量的时间～而且对于在PAT甲级中获得一个高分并没有太大的帮助～因为理论终究只是理论，想要在PAT这样的算法考试中获得好成绩还是需要大量的代码训练、针对题库的题型了解和锻炼等，而且PAT甲级中涉及的算法其实并不多，可能你打开一本算法书并不能很快找到这些在PAT考试中考察的知识点对应的章节，因为他们零散的分布在各类理论书籍里，而针对刷题过程中遇到的题目直接去针对性搜索学习相关理论知识也能很快地掌握～重要的是，你要克服在面对这些陌生的高大上的算法名词时的畏惧心理，遇到不懂的理论主动去网上搜索，有很多优秀的博客会详细讲解这些算法的理论知识，在懂了基本的数据结构的基础上一定能很快地看懂哒不用担心～

刷题过程中如何平衡自己写代码和看他人代码的小纠结

可能很多人在刷算法过程中，会觉得自己写不出来的时候看了别人写的代码就不是自己的了，感觉像是抄了一遍别人的代码，觉得不是自己想出来的印象不够深刻，可能因为小时候做数学题的时候，发现老师讲了一遍的数学题自己没记住，但是自己独立思考然后做出来的却印象很深刻…所以觉得写代码的过程中也应该尽量保持独立自主完成～其实我觉得这样的想法是不太对的哦～

算法这个神奇的东东，有它自身的一些特点，比如一道PAT题目，可能你看了题目后觉得自己有一点思路了，毕竟只是给个输入要求你给出正确的输出嘛，或多或少还是有些自己的想法的，就开始自己写，结果没能AC，修修补补改改也勉强最后AC了，但是代码却冗长繁琐，过阵子让你再做一遍这道题又没有思路了…算法题就是这样，总给你一种好像也不是太难的感觉，而且这种提交后会看到自己得分的真题题库总会让人产生一种当作一次正式考试测试一下自己的水平的想法，导致很多人刷算法完完全全就是在把自己仅有的思维和编程语法知识完全倒出来展现在代码里，如果这个人是个竞赛高手倒也没什么关系，但是如果基础不太好，直接自己写而排斥看他人代码的想法是对自己的算法提升是非常不利的，可能你冗长而思路不够清晰的代码确实AC了这道题，但是你可能也错过了向更优秀思路的代码学习的机会。反而那些没什么思路的人，可能去看了别人优秀的代码，让自己学会遇到这类算法题的清晰思路，还学了一些下次能用得到的编程语言技巧（比如18年12月PAT考试结束后，一位可爱的小学弟来感谢我学了我代码中的s.substr()的用法，让他考试的时候直接AC了一道题，增强了考试时的信心，考试的后半段时间做题状态很好多拿了很多分）所以我建议不管这道题你写出来的代码是AC

了还是做错了找不到bug，都应该看一看别人解这道题的代码是不是和你思路相同～在我刷题的时候，如果自己的代码和别人思路方法完全不同，那我会思考，我所写的方法是不是比别人写的代码优秀？很多时候会发现，并不能找到错误原因的那段代码本身逻辑就较为混乱，所以我的建议是直接删掉原来写的，对自己写的代码用更好的方法进行重构，因为即使这段代码勉强调试写出来了，下一次见到它还是难以理解，对自己的考前复习也是一种打击，会让自己看到这段代码就想要跳过不看，而且还让自己错过了一次学习他人优秀方法的机会，要知道刷题的真正意义是学到知识呀～

PAT乙级和甲级题库的正确刷题顺序

我在PAT题库的离线版中对每道题都标注了这道题所属的分类，比如PAT乙级：

> 1001.害死人不偿命的(3n+1)猜想(15) [模拟]
>
> 1002.写出这个数 (20) [字符串处理]
>
> 1003.我要通过！(20) [数学题]
>
> 1004.成绩排名 (20) [查找元素]
>
> 1005.继续(3n+1)猜想 (25) [Hash散列]
>
> 1006.换个格式输出整数 (15) [字符串处理]

虽然看上去PAT乙级涉及了很多不同种类的题目，但是由于PAT乙级题库整体属于比较简单的难度，所以在PAT乙级中不同分类题目之间区别并不是很大，都是一些简单的逻辑方面的处理和编程语言语法的运用等，所以我建议PAT乙级就直接按照题库所给出的题号顺序或者自己喜欢的顺序刷即可～刷题过程中如果觉得自己对于某一分类的题目做的效果特别不理想，可以多刷一些和这个分类相同的题目，加深对这个类型题目的理解～

不管是为了准备PAT乙级的考试，还是仅仅是为了通过PAT乙级锻炼一下代码能力和编程语言能力，刷的过程中如果发现这些题目实在是太简单了，可以不用完全刷完整个题库，挑一些自己做的不理想的题型多加练习即可，如果觉得题目对于自己来说还是有一定难度，为了在PAT乙级考试中获得一个满意的分数，还是建议把整个题库都刷一遍的～乙级的题目没有甲级那么多，题目又偏简单，刷一遍不需要多少时间的～

相比较而言，PAT甲级的分类就显得非常具有参考意义。PAT甲级中的每一道题我也标注了分类，如：

> 1001.A+B Format (20) [字符串处理]
>
> 1002.A+B for Polynomials (25) [模拟]
>
> 1003.Emergency (25) [Dijkstra算法]
>
> 1004.Counting Leaves (30) [BFS，DFS，树的层序遍历]
>
> 1005.Spell It Right (20) [字符串处理]
>
> 1006.Sign In and Sign Out (25) [查找元素]
>
> 1007.Maximum Subsequence Sum(25) [动态规划，最大连续子序列和]

原则上为了准备PAT甲级的考试，最好是需要将整个题目完完全全刷一遍的～甚至有一些准备考试的学霸会刷两三遍...当然我本人只刷过一遍...很佩服大佬们的毅力～不过也不是完全没有侧重点地刷～

首先需要明确的是，如果是为了针对PAT甲级的考试，有一些题型是不会再考察的，所以刷题的过程中如果觉得这些不会再考的题型有些难以理解，可以选择跳过不做～当然如果是想要通过题目锻炼自己的算法能力，还是可以尝试做一下的～那为什么以前考过现在不考了呢，主要原因是，PAT考试一开始是没有顶级的，有一些稍微复杂的应该放在PAT顶级里的题目也被混在了甲级里面，而自从有了PAT顶级考试之后，这些稍微复杂的题目就不会再出现在之后的PAT考试中了～而且PAT甲级一开始是作为浙江大学计算机系考研复试的上机题目出现的，在题目量不够多的时候，题型涉及面较广，有了PAT考试之后，题目类型越来越趋于稳定，有一些题号较早的题目可能确实出过某类题型，但是在之后不会再出现，甚至在考纲中也移除了相关知识点的要求～这些题目主要是动态规划和大模拟题，我将这些可以暂时刷题过程中跳过的题目列在了下方：

> 1007.Maximum Subsequence Sum(25) [动态规划，最大连续子序列和]
>
> 1014.Waiting in Line (30) [queue的应用]
>
> 1017.Queueing at Bank (25) [模拟]
>
> 1026.Table Tennis (30) [模拟，排序]
>
> 1040.Longest Symmetric String (25) [动态规划]
>
> 1045.Favorite Color Stripe (30) [动态规划，LIS / LCS]
>
> 1068.Find More Coins (30) [01背包，动态规划]

其次，有一些题目虽然在题库中出现过，但是出现的次数很少，近两年也没考过，或者我觉得将来不会考、研究了也没有意义的，最重要的是可能研究起来也不是一时半会就能学会记住的，所以在备考过程中这些题目不是复习的重心，遇到这样的题目可以考虑暂缓复习，先去复习那些经常考察的题型，等到复习完常考的类型后还有充裕的时间时再去钻研这类题目。我把这一级别的题目列在了下方：

> 1010.Radix (25) [二分法]
>
> 1016.Phone Bills (25) [排序]
>
> 1033.To Fill or Not to Fill (25) [贪心算法]
>
> 1056.Mice and Rice (25) [queue的用法]
>
> 1057.Stack (30) [树状数组]
>
> 1066.Root of AVL Tree (25) [平衡二叉树(AVL树)]
>
> 1123.Is It a Complete AVL Tree (30) [AVL树]

剩下的题目就是我们真正要集中精力刷的题目啦，我把它们分为两类，一类是比较简单的题目，难度和PAT乙级差不多（每一场PAT考试，PAT乙级的最后两道题，一般都是同场PAT甲级考试的前两道题，只不过乙级是中文的题目描述，甲级是对应的英文题目描述，而题目、测试样例和需要提交的代码都是一样的），还有一类是涉及基础算法的（很有可能是你没有了解过或者只听说过但不会用的算法，一般出现在PAT甲级考试的最后两道题），每场考试PAT甲级就4道题，而前两道又是乙级难度的简单题，这两道题占了45分～而后两道25分和30分的题目又不都是难题（不要觉得30分的题目一定比25分

的难...如果你复习完整个题库，你会发现，有的时候30分题目很简单，甚至短短三十行代码就能解决，但是往往25分题目有一些出题人挖好的坑，有一些测试点总是无法通过不知道错在哪里，或者最后一两个测试点总是超时这样的情况...）所以对于前45分，在复习的时候给自己的目标是要做到完全AC，后面两道题可能没法完全AC但至少也要做出绝大部分测试点（要是考场上运气好or自己思考这道题的脑回路完美避开了测试点给挖的坑or那次PAT考试的题型正好被你考前复习到了就可以满分啦～）

对于那类比较简单的题目，如果你在刷甲级之前已经刷完乙级，这类题目对你来说应该能够轻松驾驭（毕竟只是乙级的题目一样的英文翻译过来嘛），唯一的难点可能就是英文题意的理解方面，我觉得大可不必担心自己的英文基础比较菜导致甲级的题目会看不懂（比如什么四级六级没考过或者从小英语不好呀之类的），出现在题目中的词汇和平时我们英语考试的词汇还是有一些区别的，一些固定的表述语句、计算机专业词汇（比如 vertex 顶点、 eulerian cycle 欧拉回路、 clique 团）对于刚接触算法的人不管英语好不好都是陌生的，我当时的做法是将甲级题目中不会的单词或者固定搭配的中英文都记了下来整理成了笔记，有空的时候看一看或者见多了也就记住了，很多题目当中都会重复出现一些表述～何况题意看不太明白还能通过测试样例的输入输出猜测题意，基本上都是能明白题目的意思哒～emmm绝大多数时候都能猜出来，有的时候也会翻车，我曾经一次考试就因为没有看懂题目中的一句话（因为里面有我两个不认识的词汇，确实是之前没有复习整理的词汇导致的基础不扎实），就没能看出题目的意图，更惨的是我看着测试样例的输入输出也没能猜出来题目的意思，尽管那是一道很简单的题目...而我对着测试样例把它想复杂了（谁让它放在了倒数第二题这么一个看着就应该很难的题目位置...都怪自己当初太年轻...）整个考试那道题我就一直在对着测试样例的输入推测如何得到的输出...结果到考试结束都没有猜出来，其他题目都做完了就那道题一行代码都写不出来...所以平时看题目时候，还是应该自己去主动读一遍题意，把不会的单词和语句记下来平时看看，不要直接看别人翻译好的题目大意，看懂了题目大意做出来了真的不代表你在考场上看着题目就能明白题意...如果之前没有刷过乙级，编程基础又恰好比较弱，这类题目直接上手可能一开始会觉得思路不够清晰，可以考虑从乙级开始刷降低自己的心理压力，也可以直接从甲级入手，我建议是考甲级就直接从甲级入手，就像之前建议的懂了数据结构就直接开始刷甲级遇到不会的再去现学现用，不要畏难，只要你心里告诉自己，很简单的很简单的很简单的不信你打开排行榜看看那么多人都全刷完了排第一了～

对于那类涉及基础算法的题目，是我们学习的重点和难点～这类题目包括：树、图、并查集、堆等等，PAT和我们平时参加的大型考试不一样，大型考试命题可能前两年刚刚考过的知识点今年就不会再考了，PAT考试不是，反而是一种考察的内容趋于稳定的状态，比如树近期考的很多，那下次还可能继续考，甚至有的时候题目出现了重复考察的情况（我说的重复不是题目完全一样，题干部分说法还是有差别的，但是实现的代码都差不多），比如我有一次PAT甲级考试之前复习的时候，我认为树的遍历这种题目，尤其是前序中序后序互相转化的题目都出现过三次了，已经没有别的花样可以变了，应该不会再考，所以打算不复习树的遍历了（by the way，算法考试前的复习很重要，因为算法题虽然都叫算法其实都是一些毫无关联的解决问题的思想方式集合，一个人算法再好过了几个月大半年的也有可能忘记一类题目的解法，只能有个大概的印象，考前的复习能绝大程度调动当年刷这类题目的思路回忆，对考试非常有帮助，很有可能平时那类题目刷的特别6，结果过了段时间考前没复习就完全不记得怎么做了...），结果考前半天一个正在刷算法找工作的研究生学长给我发红包请我讲解一下树的遍历到底是如何转化的，他说研究了好久还是没明白，我只能“被迫”复习了一下相关内容然后讲给他听，结果下午的考试最后一题真的就考了一样的题目...之后几次考试也真的发现PAT喜欢考上几次才考过的题目类型，所以复习的时候还是建议面面俱到（就是有时间的话整个题库都刷一遍的意思）的同时重点关注一下近两三年考的多的题目类型，因为还有可能会继续考，而且概率很大...为了保证在短的复习时间内能最大程度地抓住考试重点，所以PAT甲级的题库要 从！后！往！前！ 刷～～～不仅要从后往前，还要 按！照！分！类！ 刷～～接下来我讲解一下为什么～

因为刷一道算法题需要花一两个小时甚至半天，平时我们还要上课做别的事情，你在一段时间内刷算法如果只按照顺序，可能今天遇到了一道最短路径的题目，弄了半天好不容易看懂了别人的代码，以为自己懂了，结果一周后又遇到了最短路径的题目，此时已经忘记上一次怎么做出来的了…这里就要提到刷题看别人代码的时候要避免看的每道题几乎都是不同人写的代码，我觉得至少同一个分类的题型要看同一个人写的代码，如果这个分享代码的人是一个思路清晰的人，他对待同一类题目的解法是有固定的思路和解题模式的，这样你在学习他的代码的时候也能形成固定的解题模式，有利于你在考场中遇到同类题目能够直接产生稳定可靠的思路～如果你只是针对一道算法题AC了就万事大吉，下次遇到同类题目又仅仅针对这道题进行解决，就永远无法形成自己对待这类题目的固定思路，那么考场上遇到同类型题目能不能AC就只能看运气啦…

所以刷PAT甲级的算法题应该是这样的顺序，打开题库从后往前整理一下每次考试（每一个由20分、25分、25分、30分形成一组的四道题为一次考试）的后两道题的题目类型，比如并查集、最短路径、图的遍历、树的遍历、DFS、BFS、图的连通分量这些，然后挑选分类从考察的多的开始逐个击破，比如发现最近树的遍历考的特别多，近期想和“树的遍历”这个题型来杀个痛快，就从题库的从后往前顺序把树的遍历的题目都挑出来开始做，一开始可能连树的遍历的概念都得复习（预习）一下，那就打开百度搜树的遍历的概念，然后找点博主写的文章或者百科看看学学，懂了基础概念的大概之后，就开始刷题，一开始刷题可能还需要看别人的代码（比如可爱的小柳柳的代码），完全没有思路甚至只能对着别人的代码抄一遍（甚至抄都有可能抄错了没AC自己还找不到bug…），边写边学习思路，接下来再挑一道树的遍历的题目，可能这一次自己能写出个大概，很有可能还是一下子写不对，继续看别人的代码学习，学个两三题很快就会形成自己对“树的遍历”这类题目的解题模式，这样自己就能慢慢地独立解决“树的遍历”这个类型的所有题目啦～接着再去会会其他常考的算法题型～

PAT刷题过程中应该如何记笔记

我在PAT刷题的过程中整理了大量的笔记，这也是我考前复习的重要参考资料，除非你有把握或者只打算参加一次PAT考试，而且能在考前短时间复习所有的内容保证考试那一天全都记得，否则每次考前都应该认真看一下自己平时刷题记的笔记～我记的所有笔记都是电子版的（如果你对我打字速度不算慢这件事儿有兴趣，可以看我写的这篇入门教程 `《双拼输入法入门指南》` https://www.liuchuo.net/archives/2786 和进阶教程 `《【双拼输入法】自然码辅助码入门教程（辅助码表）》` https://www.liuchuo.net/archives/2847，这两篇也是我博客中访问量比较高的两篇文章），毕竟写代码写着写着去拿本子和笔很有违和感，而且电子版的笔记易于检索、不容易丢失，内容形式也更加丰富，比如可以在看官方文档、电子书或视频的时候直接对内容进行截图粘贴在笔记里，或者直接将电脑上写的代码粘贴在笔记中，我一般会把笔记写在印象笔记或者 `MarkDown` 文件里（如果你对 `Markdown` 语法不了解却有兴趣，可以看我曾经写的这篇 `《Markdown语法说明笔记》` ： https://www.liuchuo.net/archives/401），同时我也将我刷题时候记的笔记整理在了这篇经验中～不过笔记还是自己整理的自己看比较好～毕竟每个人的知识盲区是不一样的～我认为记的笔记应该有这些类别：

1. 刷PAT甲级过程中整理的不认识的单词or词组or语句，记录方式应该是英文+中文翻译，示例如下：

> threshold 起征点，下限，开端
>
> M by N matrix M行N列矩阵
>
> decimal system 十进制
>
> for the sake of 为了
>
> scattered cities 分散的城市

one-way is 1 if the street is one-way from V1 to V2, or 0 if not

如果该道路是从V1到V2的单行线，则one-way为1，否则为0

fewest intersections 最少的结点

radix 基数，进制，根

2. 对于某一类题型或者某个特殊的有技巧套路的题型的解题模式，可以是文字描述形式的，也可以直接贴代码，比如我的这篇文章 《【最短路径】之Dijkstra算法》 （https://www.liuchuo.net/archives/2357）就是我在刷最短路径的题目时整理的笔记，有文字解释也有代码形式，示例如下：

对于递归边界而言，如果当前访问的结点是叶子结点（就是路径的开始结点），那么说明到达了递归边界，把 `v` 压入 `temppath` ， `temppath` 里面就保存了一条完整的路径。如果计算得到的当前的 `value` 大于最大值，就让 `path = temppath` ，然后把 `temppath` 的最后一个结点弹出， `return ;`

```
// 边权之和
int value = 0;
for(int i = tempptah.size() - 1; i > 0; i--) {
  int id = temppath[i], idnext = temppath[i - 1];
  value += v[id][idnext];
}

// 点权之和
int value = 0;
for(int i = temppath.size(); i >= 0; i--) {
  int id = temppath[i];
  value += w[id];
}
```

3. 关于编程语法方面的小技巧，或者自己容易忘记的好用的函数写法，或者提醒自己做题应该注意的点，示例如下：

1. 如果对数组进行sort排序： `sort(a, a + n, cmp1);`

   如果对vector v或者字符串v进行sort排序： `sort(v.begin(), v.end(), cmp1);`

2. 辗转相除法求最大公约数：

```
int gcd(int a, int b) {
    return b == 0 ? a : gcd(b, a % b);
}
```

3. `#define max(a,b)` a逗号后面不要加空格 define定义时候前面变量不要加空格 最后面不要加分号

4. `printf` 中的 “%%” 能够输出一个%符号～

5. `cctype` 里面： `isalnum` 是字母数字； `isalpha` 是字母； `isdigit` 是数字； `isupper` 是大写字母； `islower` 是小写字母； `isblank` 是空格/tab键； `isspace` 是空格/tab键/回车

有了这些笔记，考前复习的时候只需要看自己写的笔记+拿两道题做一下培养一下写代码的feeling就可以啦～

PAT刷题过程中的一些小技巧 | 我的PAT刷题笔记

之前有很多人问我，为什么别人blog的代码是处理一组测试数据就输出一组？不是应该全部读入之后再输出吗？这是新手或者第一次发现别人blog上的“貌似错误的代码”竟然能够AC经常会问的一个问题～大多数OJ系统都是支持“处理一组测试数据就输出一组”的做法的，由于PAT的OJ系统是在你的程序运行结束后开始检查输出是否是正确的，而且虽然在你的编译器上输入输出都显示在一个框框里，但是在OJ上却是输入输出用文件分开的，所以对于有多组测试数据的输入，可以全部读入之后再输出，也可以处理一组测试数据就输出一组～这样做是没有任何问题的不要担心哦摸摸头～（而且更省事了呀不用把答案保存下来再输出了呀～O(∩_∩)O～）

还有一个小问题就更好玩了，有的人问我遇到无法粘贴的黑框命令行（尤其是考试的时候）怎么在里面粘贴输入数据...可能现在的小可爱平时都直接用的 `win10` 的电脑， `win10` 的系统一般都是可以直接在运行之后的黑框里直接 `Ctrl + C` 、 `Ctrl + V` 复制粘贴的，但是考场上发现有的考点机房装的是 `win7` ，不能 `Ctrl + C` 、 `Ctrl + V` ...当然还是有一小半的小可爱知道可以直接通过鼠标右击黑框最上面的菜单栏选择复制和粘贴的...所以如果有办法直接复制粘贴能不用文件输入就不要用啦，觉得用文件输入挺麻烦的呢～

接下来就是一些刷题过程中关于代码方面的笔记整理～我根据我刷过的算法题整理的一些容易忘记的知识点、找了好几个小时找到bug后总结出来的注意事项整理成如下笔记，希望能最大程度减轻你的刷题负担和找bug的痛苦～

dev中查看 `vector` 的值： `*(&v[0])@3` ——必须对 `v[0]` 变换，先取地址再取值， `@` 后面的数字 `3` 能让dev显示数组前3个的值，在用这个之前要先在 编译选项-编译器里面 `-std=C++11`

`INT_MAX` 、 `INT_MIN` 、 `LLONG_MAX` 在 `#include <climits>` 头文件里面，int最大值为2147483647，共10位数字； `LLONG_MAX` 最大值有19位数字，以9开头。所以说储存13位的学号可以用 `long long int` ，输出的时候使用 `%013lld`

对 `char c[15]` 进行 `sort` 排序的 `cmp(char a[], char b[])` 函数要这样写： `strcmp(a, b) < 0` ，因为 `strcmp` 返回的不是 `-1 0 1` 而是 `等于0 小于0 大于0` 的某个值， `strcmp` 在头文件 `#include <string.h>` 里面

对 `vector v` 或者 `string v` 进行 `sort` 排序： `sort(v.begin(), v.end(), cmp1);` ；对数组a进行sort排序： `sort(a, a + n, cmp1);`

将十进制 `a` 转换为 `b` 进制数，当 `a` 不等于 `0` 时，将 `a%b` 从后往前倒序保存下来，每次保存后将 `a/b` 。这样倒序保存的数就是十进制 `a` 在 `b` 进制下的结果。

```
// 判断n是否为素数
bool isprime(int n) {
    if (n <= 1) return false;
    int sqr = int(sqrt(n * 1.0));
    for (int i = 2; i <= sqr; i++)
        if (n % i == 0) return false;
    return true;
}
```

比较有学号、姓名、分数的学生成绩排名，用结构体数组存储，用sort函数对 `node a, node b` 进行sort排序， `cmp` 函数根据情况写： `cmp` 函数中：升序（or非降序）： `a < b` ；降序（or非升序）： `a > b` ；

> 注意： `sort` 函数的 `cmp` 必须按照规定来写，即必须只是 `>` 或者 `<` ，比如： `return a > b;` 或者 `return a < b;` 而不能是 `<=` 或者 `>=` ，因为快速排序的思想中， `cmp` 函数是当结果为 `false` 的时候迭代器指针暂停开始交换两个元素的位置，当 `cmp` 函数 `return a <= b` 时，若中间元素前面的元素都比它小，而后面的元素都跟它相等或者比它小，那么 `cmp` 恒返回 `true` ，迭代器指针会不断右移导致程序越界，发生段错误～

所谓two pointers问题，就是用 `p` 、 `q` 两个指针，使用 `while` 语句处理链表问题，链表什么的用 `vector` 存储就好了～

输出语句如果有 is / are、加s / 不加s的区别的话一定要当心，比如 `"There is 1 account"` 和 `"There are 2 accounts"` 是不同的输出语句

移除字符串s的第一个字符： `s.earse(s.begin());` 在字符串 `s` 前面增加 `n` 个 `0` ： `s.insert(0, n, '0');`

给一些数字字符串，求这些字符串拼接起来能够构成的最小的数字的方式：用 `cmp` 函数就能解决： `bool cmp(string a, string b) {return a + b < b + a;}`

学生姓名和分数，如果不需要将姓名输出的话，不仅可以用 `struct` 结构体中 `string` 或者 `char` 存储姓名，还可以将姓名转化为 `int` 类型，比如 `ABC4` 可以转换为 `((((A * 26) + B) * 26) + C) * 10 + 4 = ((((1 * 26) + 2) * 26) + 3) * 10 + 4`

用数组 `hash[26]` 或者 `hash[10]` 保存某个字母或者数字出现的次数/是否曾经出现过；用 `hash[256]` 保存某个 `ASCII码` 字符是否出现过， `exist[10000]` 、 `cnt[10000]` 同理

题目输入中所给的a b区间可能a和b给的顺序反了，要比较一下a和b的大小，如果 `a>b` 记得swap回来～

`char a[]` 的长度要用 `strlen(a)` ; 得到的长度是里面真实存储了字母的长度，不包括 `\0`

要输入一行的数据的话：

如果是 `string s` ，则用 `getline(cin, s);` 在头文件 `#include <string>` 里面；

如果是 `char str[100]` , 则用 `cin.getline(str, 100);` 在头文件 `#include <iostream>` 里面，也可以用 `gets(str);`

遇到链表中有无效结点还要排序的，在结构体里面加一个 `bool flag = false;` 变量，将有效结点的 `flag` 标记为 `true` ，并统计有效节点的个数 `cnt` ，然后在 `cmp` 函数中这样写：

```
bool cmp(node a, node b) {
    if(a.flag == false || b.flag == false)
    return a.flag > b.flag;
else
    return a.value < b.value;
}
```

这样 `flag == false` 的会自动变换到最后面去，到时候输出前 `cnt` 个有效的结点就可～

如果问年龄区间内的前100个，而数据量很庞大的话，在处理数据的时候就可以把每个年龄的前100个选出来，然后再遍历，因为无论如何也不会超过100个，最极端的情况就是当前区间只包含一个年龄～

```cpp
// 素数表的建立
// 首先都标记为1，外层循环i从2到√n，内层循环j从2开始到i*j<n 把j的i倍都标记为0
vector<int> isprime(50000, 1);
    for (int i = 2; i * i < 50000; i++)
        for (int j = 2; j * i < 50000; j++)
            isprime[j * i] = 0;
```

集合s或者映射s中寻找是否存在某一数字： `s.find(num) != s.end()`

在迭代遍历集合s或者映射s时， `auto it = s.begin(); it != s.end(); it++`

`set` 获取元素值： `*it`

`map` 获取元素值： `it->first it->second`

```cpp
// 关于pair的用法
// pair是一个变量类型，两两一对组成一个变量，可以自定义pair中两个元素的类型，比如string和int，string和double，double和string等
// pair<string, int>就是string和int类型的一对pair类型，为了简便书写，我们一般会将pair<string, int>使用typedef重命名为p之后再使用，避免每次用的时候都写很长的pair<string, int>
#include <iostream>
#include <vector>
using namespace std;
int main() {
    typedef pair<string, int> p; // 建立string和int类型的一对pair类型，并将类型命名为p
    vector<p> v; // 建立p类型的数组vector v
    p temp = make_pair("abc", 123);// "abc"和123组成的pair赋值给pair类型的临时变量temp
    v.push_back(temp);// 将临时变量temp放入v数组中
    cout << temp.first << " " << temp.second << endl;// 输出temp的第1个元素和第2个元素
    cout << v[0].first << " " << v[0].second << endl;// 输出v[0]的第1个元素和第2个元素
    return 0;
}
```

当心两个数累加或者累乘过程中的超出定义的类型范围，如果是分子分母都是 `long long int` 类型的分数，(分子+分子) / 分母的累加，记得在累加过程中约分，以保证不超出 `long long int` 范围～

`abs()` 在头文件 `#include <stdlib.h>` 里面

`% 1000000007` 如果怕溢出可以多取余几次，比如 `result = (result + (countp * countt) % 1000000007) % 1000000007;`

```cpp
// 树的遍历-前序中序转后序：
void post(int root, int start, int end) {
    if (start > end) return;
    int i = start;
    while (i < end && pre[root] != in[i]) i++;
    post(root + 1, start, i - 1);
    post(root + 1 + i - start, i + 1, end);
```

Mail: 1051276278@qq.com UUID: 047a855b-d070-38c9-9eb5-eb7a59cc850f

```
        printf("%d ", pre[root]);
    }

    // 树的遍历-后序中序转前序:
    void pre(int root, int start, int end) {
        if (start > end) return;
        int i = start;
        while (i < end && post[root] != in[i]) i++;
        printf("%d ", post[root]);
        pre(root - 1 - end + i, start, i - 1);
        pre(root - 1, i + 1, end);
    }
```

树的遍历-后序中序转层序，只需在转前序的时候，加一个变量index，表示当前的根结点在二叉树中所对应的下标，根结点为 `0` ，左子树根结点为 `2 * index` ，右子树根结点为 `2 * index + 1` ，将转的过程中产生的 `post[root]` 的值存储在 `level[index]` 中， `level` 数组一开始都为 `-1` ，这样将所有不是 `-1` 的点输出即为层序的输出～

树的遍历-后序中序转之字形，用 `level` 上面那个方法就比较不方便，可以在转前序的时候，加一个变量 `level` ，一开始为 `0` ，每一次 `level+1` ，建立一个 `vector<int> v[10000]` ，则每次将得到的前序的值放入 `v[level].push_back()` ，得到的就是一个每一行代表树的一层的二维数组，然后对每一行之字形输出～

建立树的方式也可以用邻接表，建立结构体 `struct TREE` ，包含 `left` 和 `right` 两个变量，然后以建立 `TREE` 的二维数组 `tree` ： `vector<TREE> tree[n];` ，然后将 `tree[i]` 的 `left` 和 `right` 赋值为i的结点的左右子树的下标： `tree[i].left= ;tree[i].right = ;`

```
    // 二叉搜索树的递归建树方法:
    node* build(node *root, int v) {
        if(root == NULL) {
            root = new node();
            root->v = v;
            root->left = root->right = NULL;
        } else if(v <= root->v)
            root->left = build(root->left, v);
        else
            root->right = build(root->right, v);
        return root;
    }
    for(int i = 0; i < n; i++) {
        scanf("%d", &t);
        root = build(root, t);
    }
```

求最短路径过程中，由dfs求得的路径path是逆的，需要倒置输出～

`string` 类型转 `int` 、 `long` 、 `long long` 、 `float` 、 `double` 、 `long double` // 头文件 `#include <string>` ： `stoi` 、 `stol` 、 `stoll` 、 `stof` 、 `stod` 、 `stold`

`char c[10]` 类型转 `int` 、 `long` 、 `long long` // 头文件 `#include <cstdlib>` ： `atoi` 、 `atol` 、 `atoll`

`int` 、 `long` 、 `long long` 、 `float` 、 `double` 、 `long double` 转 `string` // 头文件 `#include <string>` ： `to_string`

使用 `reverse` 倒转vector v或者数组v里面的值 // 头文件 `#include <algorithm>` ，使用方法： `reverse(begin(v), end(v));`

`string` 可以使用 `push_back('a');` 和 `pop_back();` 移除一个字符 // 头文件 `#include <string>`

`is_permutation` 可以用来判断v2是否是v1的一个序列，返回值是0或1 // 头文件 `#include <algorithm>` ，使用方法： `is_permutation(v1.begin(), v1.end(), v2.begin());`

判断是否是不降序列： `is_sorted(begin(a), end(a));`

`next_permutation()` 和 `prev_permutation()` // 头文件 `#include <algorithm>` ，使用方法如下：

```
string s = "12345";
do {
    cout << s << endl;
}while(next_permutation(s.begin(), s.end()));

do {
    cout << s << endl;
}while(perv_permutation(s.begin(), s.end()));
```

`fill` 在 `#include <algorithm>` 头文件中，使用方法： `fill(e[0], e[0] + 500 * 500, inf);`

大数组必须开全局～

`set` 中可以使用 `rbegin` 访问最后一个元素，使用方法： `cout << *s.rbegin();`

`getline(cin, s);` 前面如果有换行的输入，一定要在前面加上 `getchar();` （用来读取空格），否则会直接只读入要读的字符串前面的 `\n` ～

关于质因数的解释：质因数就是一个数的约数，并且是质数。除了1以外，两个没有其他共同质因子的正整数称为互质。因为1没有质因子，1与任何正整数（包括1本身）都是互质。任何正整数皆有独一无二的质因子分解式。只有一个质因子的正整数为质数。

`queue q` 只有 `q.push(i);` 和 `q.pop();` ，不是 `push_back` 和 `pop_front` ....但是可以 `q.back();` 和 `q.front();` 访问队列q的最后一个元素和第一个元素～

```
// vector的erase与insert要用迭代器:
vector<int> v{1, 2, 3, 4};
v.insert(v.begin()+2, 5); // {1, 2, 5, 3, 4}
v.insert(v.begin(), 2, -1); // {-1, -1, 1, 2, 5, 3, 4}

// string的erase和insert无须迭代器:
string s;
s.insert(0, "3")   // 第一个参数为0~s.length()包括len
s.insert(0, 2, "3");
s.erase(0, 5); // 下标0开始，删除5个字符
s.erase(2); // 删除下标2开始的所有字符
```

`switch()` 括号里面只能是 `int` 型或者 `char` 型

在 `string s` 后面加字符 `s += b;`

在 `string s` 前面加字符 `s = a + s;` // a可以是 `char` 型也可以是 `string` 类型

`substr` 只有两种用法：

`string s2 = s.substr(4);` // 表示从下标4开始一直到结束

`string s3 = s.substr(5, 3);` // 表示从下标5开始，3个字符

如果树的某个结点是 `NULL` ，而 `NULL` 是没有值 `val` 的，所以不能够 `NULL->val` 更不能 `NULL->next` ，否则会导致 `runtime error` ，可以利用 `&&` 和 `||` 的短路的性质，把 `NULL` 的判断提前～

```
// 求数组中最大值，可以使用algorithm头文件中的max_element()，返回的是指针
int *b = max_element(a, a + 5);
cout << *b;
```

建立一个 `node` 结构体，在里面重载小于号，将里面的 `node` 按照 `cnt` 从大到小排列，如果 `cnt` 相等就按照 `value` 从å小到大排列：

```
struct node {
    int value, cnt;
    bool operator < (const node &a) const {
        return (cnt != a.cnt) ? cnt > a.cnt : value < a.value;
    }
};
```

```
// 关于并查集(Union Find)的笔记整理
// 并查集中findFather里压缩路径的代码
int findFather(int x) {
    int t = x;
    while(x != fa[x])
        x = fa[x];
    while(t != fa[t]) {
        int z = t;
        t = fa[t];
        fa[z] = x;
    }
    return x;
}
// 并查集的并（Union）代码：
void Union(int a, int b) {
    int faA = findFather(a);
    int faB = findFather(b);
    if(faA != faB) fa[faA] = faB;
}

// 统计有多少个团体(numTrees)，每个团体多少人(cnt[i])，一共有多少个人(numBirds)
for(int i = 1; i <= maxn; i++) {
    if(exist[i] == true) {
        int root = findFather(i);
        cnt[root]++;
    }
}
int numTrees = 0, numBirds = 0;
for(int i = 1; i <= maxn; i++) {
    if(exist[i] == true && cnt[i] != 0) {
```

```
            numTrees++;
            numBirds += cnt[i];
        }
    }
```

```
// sscanf() 从一个字符串中读进与指定格式相符的数据
// sprintf() 字符串格式化命令，主要功能是把格式化的数据写入某个字符串中
// 在头文件 #include <string.h>，使用方法:
char a[50], b[50];
double temp;
sscanf(a, "%lf", &temp);
sprintf(b, "%.2lf",temp);
```

如果觉得数组从下标0开始麻烦，可以考虑舍弃0位，从1下标开始存储数据～

如果对于不同的结果输出不同的字母，最好考虑用字符数组或者字符串数组将这些字符预先存储好，避免多个printf语句导致的错误率增加～ `string str[10] = {"Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"};`

处理最后一个结果末尾没有空格的情况，考虑在for语句里面添加if语句的方法处理： `if(i != n - 1) printf(" ");`

如果输入一组数据和输入好多个组数据的得到的同一组答案不一样，考虑下是不是int n后忘记写了输入语句...

```
// 把字符串string变成字符数组char *c，在头文件<string>里面
string s = "abc";
char *c = s.c_str();
```

如果 `name` 为10个字符，注意 `char name[]` 要开至少11个字符～

`string.find` 返回的是下标的值，如果没有，用 `== string::npos` ，如果找到了某个字符，就将这个字符的位置赋值给index，可以这样写代码：

```
if(s.find('a', 5) != string::npos)
    int  index = s.find('a', 5);
```

解释：

`int index = s.find('a');` //返回a这个字母第一次出现的下标（从0开始）

`int index = s.find('a', 5);` //从下标5开始，寻找a第一次出现的下标

按照名称的升序排序，因为 `strcmp` 比较的是 `ACSII码` ，所以 `A < Z` 。写cmp函数的时候 `return strcmp(a.name, b.name) <= 0;` ，因为 `cmp` 的 `return` 语句需要返回的是 `true` 或者 `false` 的值，所以要写 `<= 0` 这样的形式。比较 `ACSII码` 的大小， `strcmp('a', 'z')` 返回负值，因为 `a<z` ， `a - z < 0`

不能 `char *str; puts(str);` 必须 `char str[100000]; puts(str);`

`bool` 变量在 `main` 函数里面记得要初始化， `bool flag[256] = {false};` 在main函数外面（全局变量）会被自动初始化为 `false` 就可以不写～

`unordered_map` 和 `multimap` ： 头文件分别为 `#include <unordered_map>` 和 `#include <map>`

`unordered_map` 是不排序的map，主要以key，下标法来访问

在内部 unordered_map 的元素不以键值或映射的元素作任何特定的顺序排序，其存储位置取决于哈希值允许直接通过其键值为快速访问单个元素（所以也不是你输入的顺序，别想太多...）

multimap 是可重复的map，因为元素可重复，所以一般用迭代器遍历

段错误：数组越界，访问了非法内存 段错误也有可能是因为数组a没有初始化，导致 b[a[2]] 这种形式访问了非法内存

vector<int> v[n] ——建立一个存储 int 类型数组的数组，即n行的二维数组；也可以用 vector<vector<int>> v 建立二维数组～

指针是个 unsigned int 类型的整数～

两个 int 类型的指针相减，等价于在求两个指针之间相差了几个 int 。如 &a[0] 和 &a[5] 之间相差了5个int，会输出5～

浮点数的比较不能直接用 == 比较，因为根据计算机中浮点数的表示方法，对于同一个小数，当用不同精度表示时，结果是不一样的，比如 float 的 0.1 和 double 的 0.1 ，直接用 == 比较就会输出不相等，所以可以使用自定义 eps = 1e-8 进行浮点数的比较：

```
const double eps = 1e-8;
#define Equ(a, b) ((fabs((a) - (b))) < (eps))
#define More(a, b) (((a) - (b)) > (eps))
#define Less(a, b) (((a) - (b)) < (-eps))
#define LessEqu(a, b) (((a) - (b)) < (eps))

if(Equ(a, b)) {
    cout << "true";
}
```

```
// 使用while接收输入的两种方式
while(scanf("%d", n) != EOF) {

}
// 等价于下面这种：
//因为EOF一般为-1，所以~按位取反-1正好是0，就可以退出循环了
//所以也写成下面这种情况
while(~scanf("%d", &n)) {

}
//此处有的小可爱会问我，为什么自己的命令行写这样的代码就会得不到输出结果，但是OJ却能AC，因为计算机一直在等你的输入结束呀，你要在输入完所有数据之后按输入结束符（ctrl+d还是ctrl+c不同操作系统不一样...）它才知道你已经输入结束了才会输出你的结果呀～但是提交到OJ就不一样啦，他会自己判断输入文件有没有已经读取完，所以在OJ上直接能AC～
```

queue 、 stack 、 priority_queue 是用 push 和 pop

vector 、 string 、 deque 是用 push_back

push 和 pop 只是一个动作，而 queue 是用 front 和 back 访问第一个和最后一个元素的， stack 使用 top 访问最上面的一个元素

stack 、 vector 、 queue 、 set 、 map 作为容器，所以都有 size

传参是拷贝，所以用引用的话更快，传参拷贝可能会超时～

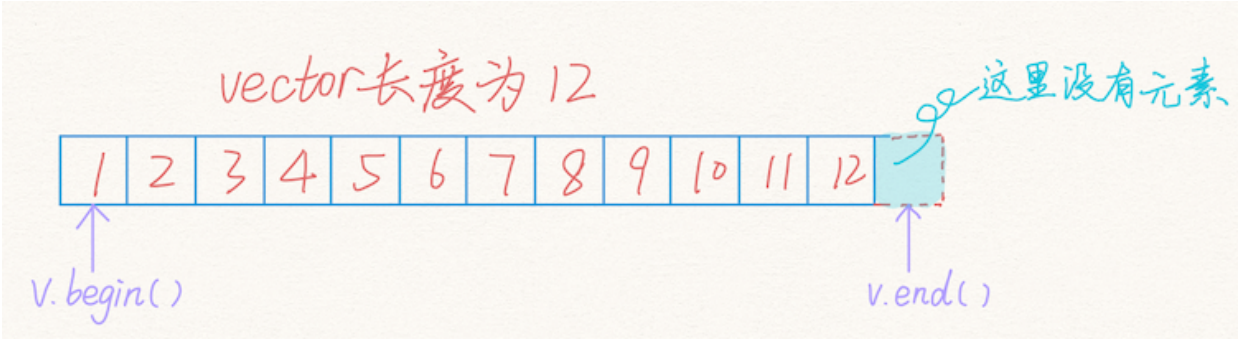
`%c` 是会读入空格和回车的，如果真的想要获取到输入中的字符 `%c` ，可以通过在scanf里面手动添加空格的方式避免： `scanf("%d %c %d", &a, &b, &c);`

在 `#include <algorithm>` 头文件里面，有 `reverse` 函数， `reverse(s.begin(), s.end());` ， `reverse` 是直接改变字符串本身的，并没有返回值，不能 `reverse` 之后赋值给一个字符串，所以 `string t = reverse(s.begin(), s.end());` 这样是不对的～

使用 `fill` 初始化二维数组是 `fill(e[0], e[0] + 510 * 510, 99999);` ，不是 `fill(e, e + 510 * 510, 99999);` ，因为二维数组在内存里面存储是线性的连续的一段...从 `e[0]` 的内存开始一直到 `n * n` 大小的内存～

循环无限输出考虑一下是不是 `i--` 写成了 `i++`

容器 `vector` 、 `set` 、 `map` 这些遍历的时候都是使用迭代器访问的， `c.begin()` 是一个指针，指向容器的第一个元素， `c.end()` 指向容器的最后一个元素的后一个位置，所以迭代器指针it的for循环判断条件是 `it != c.end()` ，我再重复一遍～ `c.end()` **指向容器的最后一个元素的后一个位置**，这是一个重点和难点，我画个图加深一下小可爱们的记忆（再biu你们一下，biubiubiu～这下总该记得了吧～）



关于找不到bug该怎么办 & PAT刷题过程中的一些经验总结

我在刷题过程中也不可避免的会遇到测试点无法通过的情况，我以及身边各位算法大神的做法是：对照着其他人的能够AC的代码，对比自己的代码，一点点更改，慢慢缩小目标区域，直到最后找到答案～

鉴于我每天都会遇到很多向我求助代码错误点的朋友，所以我花了数天时间，重读了我过往刷过的几千道算法题和一些刷题过程中的笔记，从报错信息方面总结了一些代码错误的常见原因，希望能帮助你成功找到不能AC的真正原因～

1. **所有测试点都是段错误**

   段错误一般是由数组越界、堆栈溢出（比如，递归调用层数太多）等情况引起

   段错误是非常喜闻乐见的一种报错方式，因为段错误锁定问题代码区域的方式较为简单：首先对可疑的代码段进行注释掉，然后提交到OJ上看是否会发生段错误，如果确实是自己注释掉的这段代码发生了段错误，那么对应的OJ判题结果就会变为答案错误而不再是段错误，说明注释掉的代码正是引起越界溢出的原因所在～以下是一些可能引起段错误的常见原因：

   1.1 多层 `for` 循环中内层循环本来打算写 `j` 或者 `k` ，却因为习惯或忘记误写成了外层循环的变量 `i` 或 `j` ，导致数组访问 `i` 或 `j` 下标的时候发生了越界

```
for(int i = 0; i < m; i++) {

    for (int j = 0; j < 100; j++) {
        ……
        ……
        …… // 可能第二层循环j离的太远写着写着忘了……

        for (int j = 0; j < 500; j++) {// 如果这里因为习惯或忘记误写成了j
            v[j] = 1; // 就有可能因为数组下标不能超过100而导致数组越界访问
            ……
        }
    }
}
```

1.2 数组开小了导致指针指向了未开辟的数组区域，出现了越界访问

1.3 大数组一定要开全局，而不是写在main函数里面，否则容易发生段错误（因为大数组在 `main` 函数里面的话是存储在栈里，而栈空间是在进程创建时初始化的，有固定的大小，一般为几十KB，所以太大的数组会耗光栈空间。而全局变量占用的堆空间，堆空间中的内存是按需分配，自由增长的，可以非常大，比如32位的系统中可以大到4GB。将大数组放在全局变量中能避免栈溢出～

1.4 递归的出口写的有问题，递归层数太多导致了堆栈溢出，发生段错误

1.5 打算写从后往前遍历的 `for` 循环，本应该写 `i--` ，结果因为习惯误写成了 `i++` ，导致 `i` 下标访问了越界内存，产生了段错误

2. **有几个测试点是段错误**

2.1 是否没有考虑0或者边界值的情况？比如对于一个空数组却访问了 `arr[0]` ，可能会出现段错误的情况

2.2 对于树来说，当前结点如果是 `NULL` ，一定要先判断 `root` 结点是不是 `NULL` 再使用 `root->val` 或者 `root->left` 、 `root->right` 的情况，因为 `NULL` 是没有 `val` 、 `left` 、 `right` 等指针的（这个在LeetCode里面是非常常见的错误）

2.3 `while` 循环的边界条件未判断正确，导致访问了数组的非法内存，产生段错误

3. **所有测试点都运行超时**

如果所有测试点都是运行超时，一般情况是出现了死循环

3.1 明明 `i` 从 `s.length()-1` 开始向前遍历，结果 `i--` 误写成了 `i++` 导致了无限循环

3.2 `while` 循环的边界判断有问题，使得无限循环

3.3 代码的算法需要优化，过于暴力破解的算法在一些运行时间、算法复杂度要求较高的题目中会经常发生运行超时

4. **有几个测试点运行超时**

4.1 `cin` 的输入比 `scanf` 更耗时，如果个别测试点超时可以考虑是否可以通过优化输入将 `cin` 改成 `scanf`

4.2 对数据进行排序的过程中，有很多数据是无效数据，如果对所有数据都排序可能会引起超时，所以可以考虑在放入数组之前就做一些条件判断剔除无用数据，然后再对数组进行排序

4.3 如果写了 `main` 函数之外的其他函数，传引用会比传值的方式更省时，因为传值是拷贝传参，传引用是传入的地址直接在原变量上修改，所以可以考虑优化为 `void func(int& a)` 的方式传递参数减少耗时

4.4 在 `for` 或者 `while` 循环中的及时判断条件，用 `break` 提前退出当前层的循环或者直接 `return 0;` 结束程序，可以降低代码运行时间～

4.5 `for` 循环语句里定义 `int i` 和在外面定义 `int i` 其实有少许区别，在 `for` 循环语句里直接定义的 `i` ，比如 `for(int i = 0; i < 10; i++)` 可能会更耗时，把它改成 `int i; for (i = 0 ; i < 10; i++)` 也是降低代码运行时间的一种可考虑方式哦～

4.6 求最短路径的题目中以用 `Dijkstra` 解决的非要用 `DFS` 强行深搜（毕竟 `DFS` 也是一种暴力破解）获得答案可能会导致超时哦～

4.7 使用了 `v.size()-1` 等方式，如果 `v.size()` 本身就是0，因为 `v.size()` 返回值是无符号整数 `unsigned int` ， `unsigned int` 的0减去1得到的是 `unsigned int` 的最大值而非 `-1` ，这可能会让 `int i` 循环时 `i < v.size() - 1` 超时～

4.8 是不是明明可以用键值对找到对应的值（比如 `bool visit[100]` 寻找对应的 `visit[80]` 是否为 `true` ，复杂度为 `O(1)` ）却用了 `for` 循环一个个找的存储方式（比如邻接矩阵可以一次查找得到，却用了邻接表一个个查找），以空间换时间的方式是常用降低算法时间复杂度的方式～

4.9 使用 `map` 存储如果超时了，而题目又没有要求排序，换成用 `unordered_map` 就不会超时啦～同理，使用 `set` 存储如果超时了，可以考虑换成 `unordered_set` 避免超时～

5. **有几个测试点答案错误**

答案错误的原因太多了，代码思路稍有不对或是小粗心都会出现很多答案错误，甚至有的时候没有完全理解对题意却能得到大部分答案正确、个别答案错误的情况（比如PAT红黑树那道题，我在考场上就未能完全理解题意却得到了大部分分数，但由于对题意把握的不准确，这也影响了我调试找bug），对着代码找半天都不知道错在哪的情况也常有发生，具体问题具体分析，以下是我调试过程中遇到过的一些常见错误～

5.1 输出语句的错误。比如输出的个别字母不正确、输出 `yes` / `no` 的大小写没有按照要求、 `true` / `false` 拼写错误、 `1` 后面单词单数不要加 `s` ， `2` 以上的个数后面单词复数要加 `s` 、除了单复数s还有be动词的 `is` 和 `are` 的区别等问题，有时候不正确的拼写遇上多条输出语句还会引起一半答案错误、一半答案正确的情况还始终找不到原因在哪emmmmmm

5.2 `switch` 后面只能 `int` 或者 `char` 类型 其它类型不可以～用错了 `switch` 也会导致答案错误～

5.3 `memset` 只能赋值 `0` 、 `-1` 和最大值（因为 `memest` 函数是用按位赋值的），如果你想要给数组赋值一个特定的值，请使用 `fill` 函数～

5.4 `getline` 读取的是一整行字符串，如果接下来还要读取字符 `char` ，会读取到上一行字符串末尾的回车换行符，所以为了避免这种情况，要在 `getline(cin, s);` 后面加一句 `getchar();` 才能得到接下来自己想要读取到的 `char` ～

5.5 `int + int` 、 `int * int` 的过程中都有可能出现结果超过了 `int` 的值而产生溢出的情况，这样就会使一些数据较大的测试用例因为溢出变成了别的数字而得到答案错误～（所以不要觉得题目中给出的数据都在 `int` 范围之内就只用 `int` 存储所有变量，因为可能在计算加法或者乘法的过程中超出 `int` 的范围）

5.6 有时候会给一组ab区间，但是a b并没有按照从小到大的方式输入，可能个别数据是 `a > b` ，所以要先判断一下给出的区间ab谁大谁小，如果a大b小就应该使用 `swap` 函数交换一下～

5.7 如果是给id的题目，可能输出中要求按照不满4位在前面补 `0` 的情况，所以要用 `printf("%04d", a);` 的方式输出，否则遇到不满4位的id输出结果可能会导致答案错误（有时题目会显示为格式错误）～

6. **内存超限**

   内存超限是指开的数组或占用的内存过大，超过了要求的内存范围：

   6.1 可以考虑开 `vector` 或者动态开辟数组的方式避免直接开 `1000000` 这样很大的数组

   6.2 如果是为了存储一些键值对，可以避免开 `int hash[100000]` 这样的大数组，而是用 `map` 或者 `unordered_map` 来处理键值对的映射

   6.3 如果你开大数组是为了存储稀疏矩阵，可以用邻接表存储的方式代替邻接矩阵来降低空间复杂度（以时间复杂度换空间复杂度的方式）～

7. **提交第一次超时了，第二次却AC了**

   OJ系统大多有多台服务器，在平时练习过程中，提交那一瞬间如果遇到了新买的高配置服务器可能运行比较快，就能够让你的代码AC，有的时候遇上了普通或者慢的服务器，就有可能会运行超时～但是PAT考试时候不要心存侥幸或是觉得这不公平哦～他们会关闭个别特别快的服务器只留下势均力敌的几个服务器hhhhhhha～

8. **浮点错误**

   程序中出现了除以 `0` 、取余 `0` 的这种非法操作～

   所以发生浮点错误应该考虑程序中：

   是否可能出现了一个数除以 `0` 的情况

   是否可能出现了一个数取余 `0` 的情况

   是否发生了数据溢出而导致的除以 `0` 或者取余 `0` 的情况

9. **返回非零**

   11.1 如果你用的是 `C/C++` ，请使用 `int main` 和 `return 0;`

   11.2 如果你用的是 `Java` ，那么可能原因是你写的代码里面的 `public class` 后面的那个名字不是 `Main` ，改成 `Main` 就可以通过了～

10. **格式错误**

    13.1 可能是换行、空格等出现了问题，输出语句内可能有拼写错误、空格多打少打等情况～

    13.2 行末的不必要的换行或是缺少了必要的换行也有可能引起格式错误～

    13.3 题目要求小写/大写而自己没有按照题目要求输出也有可能格式错误～

    13.4 题目要求输出4位数字不足前面添加 `0` 的时候没有按照要求用 `printf("%04d", a);` 的方式输出，否则遇到不满4位的id输出结果可能会导致格式错误～

PAT考试的注意事项 | 考试期间的心态调整

首先要注意考试日期，提前去官网打印好准考证，考试带身份证、准考证和一两支笔（有一次我带了一支笔结果路上一摔写不出来了断墨了…，考试过程中必然会打一些草稿，比如做树或者图的题目的时候…草稿纸考场上会发不用带），考试时间是3小时，考试开始的时间有时候每个人不统一，坐下后可能有的人提前输入准考证和密码就能登陆进去开始做题了…有的时候有倒计时是统一开始的…不过没关系，对于每个人来说不管有没有提前开始都只有3小时的时间（到了考试正式开始时间还没开始就没3小时了哦），屏幕上会显示考试剩余的时间和你的照片～

考试的时候一般坐在你左边右边的是和你不同类别的PAT考生，比如考PAT甲级的人左右坐的是考PAT乙级的萌新or顶级的大佬，所以考试时候留意自己的考试进度就好，别人有没有在开始考试后看到题目就开始奋笔疾书（或者应该叫噼里啪啦努力敲键盘？）和你没有关系，不用care他/她～你思考你自己的题目就好～

刚在考场座位坐下的时候，可以做以下几件事情：

> 1.打开IDE试试好不好用，如果你是一个又会 `C/C++` 又会 `Java/Python` 的大佬，可以都试一下～毕竟有的时候会发现，一道题用xx语言处理起来有点复杂但是用另一个语言解决起来会更简单…（当然考试绝大多数情况下还是对待各个语言很公平哒），给自己的IDE写个包含输入、输出的 `"hello,world"` ，运行一下看看有没有问题，然后打个断点试试调试好不好用～有问题可以随时找考场老师协商解决～
>
> 2.试一下鼠标键盘好不好用，可能机房的机子个别会有故障，比如我会坐下来在键盘上输入一句 `"The quick brown fox jumps over the lazy dog"` 来检验键盘的26个字母是否好用～（因为这句话包含了所有英文单词…不了解这个梗的可以参考维基百科 `"The quick brown fox jumps over the lazy dog"` 词条： https://en.wikipedia.org/wiki/The_quick_brown_fox_jumps_over_the_lazy_dog ）
>
> 3.提前建好做题的时候需要用到的cpp文件，并写好常用头文件和 `main` 函数和 `return 0` 。如果你用的dev，在直接新建一个 `.cpp` 文件后，保存千万不要把这个 `.cpp` 文件直接保存在盘符里，比如D盘、E盘这些，一定要在盘符里面建立一个文件夹然后将 `.cpp` 文件保存在这个文件夹里，要不然编译调试的时候会出问题～还有就是不要直接保存在桌面，因为桌面一般情况下都在C盘，C盘绝大多数情况是还原盘，也就是机房的这台机子如果一不小心重启了，整个还原盘可能都会被重置，你刚刚保存的代码文件就消失了，如果你正好一道题写到一半…不小心机子重启了…你保存在桌面的话没了…就得重新写这段代码了…所以可以问一下考场的老师哪一个是非还原盘符，一般情况下是D盘E盘…也就是机子重新启动后上一次保存的文件还会在的盘……我的习惯是只建立一个 `.cpp` 文件，每次的题目都写在同一个 `.cpp` 文件里，也有的人喜欢建立4-5个 `.cpp` 文件、一道题目放在一个文件里，做完一道就换一个文件…由于我只用一个 `.cpp` 文件写代码，所以我一般还会打开记事本，保存在刚刚建立的那个文件夹里，用来粘贴我做完题目后的代码，毕竟对考试系统不太信任万一我AC后的代码弄没了呢…或者有题目没AC想换种思路写，但是不确定能不能按照新的思路完美写出来，就把旧的思路写一半没AC的代码先保存在记事本里……头文件一般写以下几个常用的，反正多写了也不会怎样…所以我一般在考场上坐下后会在我新建的 `cpp` 文件里写以下一段代码然后保存，做好开始考试的准备：

```cpp
#include <iostream>
#include <cstdio>
#include <algorithm>
#include <vector>
#include <map>
#include <set>
#include <string>
#include <cctype>
#include <unordered_map>
using namespace std;
int main() {
    int n;

    return 0;
}
```

由于PAT考试是联网的考试，很多时候搜索引擎什么的都能用，不像蓝桥杯那样的考试给的电脑都是只能连局域网的，所以考试过程中需要注意的是，任何时候都不要尝试打开搜索引擎或者其他浏览器（更别用自己随身携带的手机等设备查资料...）因为每个考场的屏幕都有监控程序，如果有违规行为很快就会被识别到，一旦发现就会被逐出考场禁考三年，每次考试都有被强行逐出考场取消考试资格的考生...有的时候或许只是无心之举，比如电脑上的搜狗输入法突然弹窗，你没有及时关闭，或者点了上面的什么新闻标题跳到新闻详情页面了...可能会被监控程序发现后告诉考场老师来提醒警告你...比如有的人在做PAT甲级题目的时候看不懂英文，正好浏览器最上方有一个 `点击翻译` 的按钮...（当年还是直接用浏览器做题目，现在一般考试时候用的是拼题A的客户端，如果拼题A客户端临时出了问题考场才会启用浏览器直接登录做题的方式...）然后他点了...然后就算作违规行为被禁考了...所以千万不要脑壳子一短路随便乱点呀～还有一个禁考的例子就更好玩了，打开 `Visual Studio` 还是 `Eclipse` 的时候，弹出来一串英文，大概意思是你现在用的这个新版的IDE出了很多新的功能要不要深入了解一下之类的，有的人一看全英文的一下子没看懂，就点了个 `Learn More` 按钮，直接跳转到浏览器打开了这个IDE官网的特性介绍页面...然后被监控程序发现了...我所在的考场的监考老师还是比较仁慈的，只是走到那个考生身边提醒他赶紧关闭页面，如果遇到不友善的监考老师就不知道会不会算做作弊啦，所以小可爱们考试的时候一定要注意，考试过程中不要乱点别的～考完了倒是可以在等证书打印好的时候随便倒腾电脑，我一般都会带U盘到考场上等着考完考试的时候把考试时候AC的代码拷贝到U盘里...毕竟回去还要给小可爱们发博客更新题解～

还有一点需要注意的是，有一次考试我旁边的那个考生脚踢到了对面考生的电脑插座，导致对面考生的电脑断电重启了...刚写好的代码全没了...所以我们在考试过程中，首先尽量保证不要影响到其他人的正常做题，还有就是要及时记得保存自己所写的代码，不管是写到一半还是写完了还没找到bug，这段代码如果真的因为机子断电重启突然消失了自己完全重新写是很影响做题效率和心态的...写的时候注意随时 `Ctrl + S` 保存，保存，保存...

考试过程中，如果一道题尤其是简单的题因为一两个测试点没有通过，不要一直纠结导致后面的题目没时间看，注意控制好时间，很多时候后面的题目考的还是挺简单的，如果因为前面一两分错过了后面几十分就很不划算啦～而且这种情况还挺常见的，不信问问那些每次考试考93、95、97分的，基本上都是因为第一道题或者第二道题有某个测试点答案错误...个别人是因为第三第四题的算法没写好导致最后一个测试点运行超时扣了三五分...算法考试，尤其是这种即时可见自己得分甚至还能看到本次考试实时更新的得分排行榜的考试（在考试系统里点排行榜就能看到...大多数人应该在考试的时候没有心情看...我就不一样了...我自己做不出来就去看看有多少大佬100分了...然后告诉自己我也能100的快加油做题...），心态是非常重要的，如果心乱了很有可能什么bug都找不到什么思路都没有了～比如18年3月的那场考试，考试1小时20分钟的时候我已经全部做完题目确认自己得了100分然后交卷出考场了...而和我同考场的男朋友考完后告诉我，他在我考100的时候，自己连一道题目都没能完整的AC...但是看到我这么菜的弱鸡都能全都AC，觉得这次考试的题目一定很简单，瞬间信心倍增，一路AC，最后考了97...（嗯...他就是那种觉得30分的难题很简单，反而会被第一题的一两个测试点卡住或者读不懂简单题目题意的那类人）所以心态很重要，不要觉得别人做出来了自己没AC感到焦虑～要相信别人能AC题目一定不难自己也能AC～

蓝桥杯是什么 | 参加蓝桥杯有什么用

蓝桥杯全称是“蓝桥杯”全国软件和信息技术专业人才大赛， `Java` 和 `C/C++` 项目分为ABC三组，大学A组是重点本科，大学B组是普通本科，大学C组是专科，985、211本科生和所有院校研究生只能报大学A组，其它院校本科生可自行选择报大学A组或大学B组，高职高专院校可报大学C组或自行选择报任意组别～语言又分为 `Java` 软件开发和 `C/C++` 程序设计（所以每次写自己获奖证书的时候，写第九届”蓝桥杯“全国软件和信息技术专业人才大赛 `C/C++` 程序设计大学A组江苏省赛一等奖这么长的名字的时候都觉得很有违和感...），除了 `Java` 和 `C/C++` 项目还有嵌入式设计与开发、单片机设计与开发，由于

我对这两个了解的不多就不多说啦～（我在学校担任了三年的竞赛团队班长，带领大家组织竞赛培训、学习指导和去北京参加决赛考试带队订票之类的，团队里没有人参加嵌入式和单片机的...只有 Java 和 C/C++ 的...所以对嵌入式和单片机组实在不了解...）所以竞赛分为6个组别： C/C++大学A组 ， C/C++大学B组 ， C/C++大学C组 ， Java大学A组 ， Java大学B组 ， Java大学C组 。省赛考4个小时且6个组同时进行，决赛也是4个小时，（软件类比赛是4个小时、电子类比赛是5个小时）分为上午和下午两个场次，具体看准考证上自己是上午考还是下午考...考试过程中是机考，机子会被断网，在局域网内连接到各个考点的竞赛服务器，然后在浏览器输入监考老师给出的局域网地址下载试题和提交答案～题型分为结果填空、代码填空和编程大题，只能提交答案不能看到自己的答案是否正确，也不会知道自己最后得了多少分，只会在公布竞赛结果的时候知道自己获得了几等奖（感觉好不透明啊好担心有暗箱操作.jpg）～

省赛每个组别设置一、二、三等奖，比例分别为10%、20%、30%，总比例为实际参赛人数的60%，零分卷不得奖～省赛一等奖选手获得直接进入全国总决赛资格～省赛报名时间一般为每年的11月、12月，有的学校在省赛报名之前的10月、11月左右还会举办一次校内选拔，选拔出来的学生才可以参加省赛（我们学校是校内选拔赛通过的同学参加省赛和决赛可以报销报名费和旅费，没通过校内选拔的想参加的话可以自费，省赛的报名费是300元）省赛的参赛时间是第二年的3月份（有时候是4月初），省赛的考点准考证上会写，一般情况下都是自己的学校（如果自己的学校是考点的话，不是的话就会被安排在附近别的学校的考点，比如就有别的学校的学生省赛那天来和我们学校的考生一起参加考试）

省赛的成绩大概一两周后就会公布（还是PAT这种边做题就知道自己多少分的感觉比较刺激～），获得省赛一等奖的可以参加全国总决赛，决赛一般是5月底在北京举行，在北京也有很多考点，准考证上会写分配到哪个学校作为考点～我在北京大学和北京科技大学（还是北方科技大学？北京理工大学？北方工业大学？...忘记了...）参加过～决赛的个人赛根据相应组别分别设立特、一、二、三等奖及优秀奖～每个组特等奖1名，一等奖不高于5%，二等奖占20%，三等奖不低于25%，优秀奖不超过50%， 零分卷不得奖～我们学校一般是周五高铁去北京（我学校近两三年比较土豪，给报销地铁票、高铁票、宾馆住宿、吃饭和打车费，之前只报销绿皮火车...），在宾馆休息一下周六去考试，周日早晨去北京大学邱德拔体育馆（或者别的地方）参加颁奖典礼，一般颁奖典礼我就不去了...难得帝都溜一圈，去逛景点名胜了...然后去坐高铁，周日晚上（深夜）到达学校，结束了三天的公费旅游时光...

至于蓝桥杯有什么用，官方是这样解释的：蓝桥杯大赛是全国性比赛，2017年又走出国门举办了国际赛，未来大赛不但从规模还是影响力都会越来越大。能在蓝桥杯大赛上获奖对于学生来说本身是一个荣誉和实力的象征。同时根据各个院校政策的不同，省赛对于保研、就业和院校评三好学生和算学分等都会有一定的作用。

emmmm...至于真正的作用还得看个人，如果你想要得到那两三张证书，或者想通过它丰富一下自己参加算法竞赛的经历，都是可以参加一下的～算法竞赛其实都是相通的，如果之前在一种竞赛中努力准备过，可以顺便在另一种竞赛中稍作准备就直接参赛，有时候也能获得不错的结果（比如就有一些ACM大佬顺便参加蓝桥杯来拿两张证书开心一下）～在大学期间我本着不参加无意义的竞赛不考无意义的证书的原则，最终还是决定参加了蓝桥杯，而且大学四年期间参加了四年，我当初选择参加的原因是希望在专业课老师面前获得关注度和存在感，因为当时蓝桥杯的培训老师都是大学里教我们计算机专业课程的老师，我在蓝桥杯中获得的成绩那些老师们也会知道，所以在大一公共基础课比较多的时候我还是不敢逃课的，后期的大二大三大四专业课多了之后...依靠参加蓝桥杯获得的在计算机系内的存在感（毕竟即使我说我PAT甲级考了100老师也不知道那是什么呀...），我就几乎没有上过课了...有了更多的时间去自习室图书馆写代码看书...这也是我能在贪玩的情况下还有很多时间学习开发知识、看课外书、刷LeetCode和PAT的原因...所以，大学的竞赛、考试、证书那么多，选择几个自己有兴趣的、对自己未来职业发展有帮助的，专心对待，取得一个满意的、将来能写在简历上的成就（简历上蓝桥杯我就不太好意思写...毕竟决赛没拿一等奖...），比多考很多无意义的各种证要好很多～

蓝桥杯省赛和决赛需要掌握的知识

官方考纲对于蓝桥杯考试涉及的知识要求写的很宽泛：

> Java组解题允许使用的特性：JDK1.6 支持的特性
>
> C/C++组解题允许使用的特性：选手可以使用 c 风格或 c++风格或混合风格解答编程大题。允许使用 ANSI C(99) / ANSI C++(98) 特性。允许使用 STL 类库。
>
> 1. Java 大学 C 组 解题所涉及的知识：基本语法、面向对象、网络编程、接口、集合、IO、多线程、内部类、异常与保护，基本数据结构。（不涉及 swing 等图形界面，不涉及 html、JSP、Tomcat、开源框架等 web 开发方面，不涉及 JDBC、SQL 等数据库编程方面）
> 2. Java 大学 B 组 解题所涉及的知识：Java 大学 C 组全部知识 + 数据结构（高校《数据结构》教材中出现的经典结构，及其通过组合、变形、改良等方法创造出的变种）+ 大学程度的基本数学知识（含：解析几何、线性代数、微积分、概率、复平面基本性质）
> 3. Java 大学 A 组 解题所涉及的知识：Java 大学 B 组全部知识 + 设计模式，反射，XML，多核与并发，软件测试。
> 4. C/C++大学 C 组 解题所涉及的知识：结构、数组、指针、标准输入输出、文件操作、递归、基本数据结构（在代码填空中不会出现 c++知识，不会出现 ANSI C/C++ 之外的 windows API 调用）
> 5. C/C++大学 B 组 解题所涉及的知识：C/C++大学 C 组全部知识 + 数据结构（高校《数据结构》教材中出现的经典结构，及其通过组合、变形、改良等方法创造出的变种）、函数指针、位运算 + 大学程度的基本数学知识（含：解析几何、线性代数、微积分、概率、复平面基本性质）
> 6. C/C++大学 A 组 解题所涉及的知识：C/C++大学 B 组全部知识 + 函数模板、宏替换、汇编知识
>
> 除了编程语言的基础知识，大赛很少用到特定领域的知识。比如：电信、医药、地质、银行等特定领域。如果偶尔用到，会详细解释概念，并给出足够的示例。但“数学领域”是个例外。大赛假定选手具有足够的中学数学知识。包括：
>
> - 算数：素数，整出，余数，求模，不定方程 …
> - 代数：函数，方程，多项式，…
> - 解析几何：笛卡尔坐标系，点到直线的举例，极坐标…
> - 复数：模，夹角，矢量的合成和分解

`Java` 和 `C/C++` 组虽然题目不同但是考察的算法都差不多，其实我觉得没有考纲说的那么夸张…尤其对于省赛来说，基本上就是比普通的算法竞赛增加了一点数学知识，而决赛确实会更难一些（省赛和决赛的难度不是一个数量级的，省赛很简单，决赛很难），但是尽可能得做出自己会的题目也能得到一个不错的名次～“蓝桥杯”在前几年的时候被人戏称“暴力杯”，意思是蓝桥杯很喜欢考暴力破解之类的题目，之前省赛可能都是暴力破解能解决的，尤其是解决那些没有超时限制的填空题～什么叫暴力破解呢，简单地说，暴力破解就是想办法把所有可能的情况逐一枚举～复杂地说，就是能不重复、也不遗漏地把所有情况列举出来，然后边筛选边累积处理～最基本的做法是：循环嵌套循环or递归～虽然近年来说暴力破解的题目有所减少，或者变成了带回溯剪枝的DFS题（DFS的本质也是遍历所有情况的暴力破解，只不过在DFS的过程中可能需要对于不可能的情况提前return而已…）

从0基础到蓝桥杯省赛一等奖的高效学习路径 & 书籍推荐

如果你想要愉快地去北京参加蓝桥杯，首先要通过省赛一等奖～否则就没有机会了去北京参加决赛了～（至于决赛该如何备考，我会在下文提到）千万不要把自己的目标定的很低，我之前就有小伙伴来找我问我如何在蓝桥杯省赛中获得二等奖、三等奖…其实每年参加蓝桥杯的人数多达四五万人，其中懂算法并且认真刷题的并不多，而且省赛一等的奖项设置的很宽，成绩在前10%的人都能获得一等奖，很多人报名参加蓝桥杯省赛只是去体验一下都没有认真准备，所以获得一个省赛一等奖其实是很容易的，只要你之前有认真准备～

从0基础到蓝桥杯省赛一等，也分为看书和刷题两大块，刷题是主要工作～首先说看书～

如果你报名的是 `Java` 组的，首先得了解 `Java` 语言，我推荐的 `Java` 语言入门书是机械工业出版社的 `《Java核心技术卷I：基础知识》` 和 `《Java核心技术卷II：高级特性》`，我看的时候还是第9版，现在应该已经出到第10版了～如果仅仅是打算用 `Java` 来刷算法而不是做开发，那这两本书并不需要全看（不过一般情况下不打算用 `Java` 做开发的人应该不会选择用 `Java` 刷算法吧…），只需要了解 `Java` 的基础语言和一些刷算法用得到的集合框架即可，我以第10版为例，把针对算法部分需要和不需要学习的内容对应的章节列在了下方：

> 《Java核心技术卷I：基础知识》**不用看**的内容：
>
> 第10章　图形程序设计 403
>
> 第11章　事件处理 439
>
> 第12章　Swing用户界面组件 469
>
> 第13章　部署Java应用程序 580
>
> 第14章　并发 624
>
> ---
>
> 《Java核心技术卷II：高级特性》**需要看**的内容：
>
> 第2章　输入与输出 39
>
> 第6章　日期和时间API 288

在学习完语言后，还需要学习数据结构，推荐阅读的是 `《大话数据结构》`（程杰 / 清华大学出版社），理由和下方 `C/C++` 组中描述的一样就不赘述啦～

之后还需要学习一点儿算法，如果是 `Java` 组的，建议直接看 `《算法（第4版）》`（[美] Kevin Wayne 著，谢路云 译 / 人民邮电出版社），这本书是算法领域的经典参考书，在亚马逊图书上广受好评，使用 `Java` 实现代码，讲述了排序、搜索、树、图、字符串、正则表达式等经典数据结构和算法，里面有很多插图，讲解内容清晰易懂～

如果你是报名 `C/C++` 组的，就和上面介绍的“从0基础到PAT甲级100分的高效学习路径 & 书籍推荐”里的内容差不多，首先学C语言，然后学习数据结构，最后学一下C++的特性和STL，我把和我上文PAT中所述相同的部分附在了下方：

> 首先，得学C语言～大多数人在大学里都是学过C语言的，如果你是刚入计算机坑的萌新，不会也没有关系，可以现在开始学嘛～ `《C语言入门经典》`（霍顿 / 清华大学出版社）是入门C语言的好书～因为作者很厉害～书在豆瓣的评分也很高～作为初学者，如果你对这种巨佬作者写的500多页的书比较抵触和畏惧，可以看 `《啊哈C语言！逻辑的挑战（修订版）》`（啊哈磊 / 电子工业出版

> 社），是关于C语言入门的风趣的编程启蒙书～豆瓣的评分8.3，也很不错～通过这本幽默风趣的书也能掌握C语言～
>
> 其次，得了解数据结构和一些基础的算法～我所说的了解，是指知道有哪些常用的数据结构，了解如何分析和比较不同算法的算法复杂度，不求能够像我当初学习一样直接用C语言手写实现每一个数据结构（当然这也花费了我大量的时间，对于刷算法来说暂时没有必要，代码能力可以通过大量刷OJ算法题目培养，而我只是提早培养了自己的代码能力而已），但求能看懂明白这些数据结构的概念含义～可能《数据结构》（严蔚敏 / 清华大学出版社）这本书让很多人产生了怀疑人生甚至想要放弃编程的想法，所以在了解数据结构这方面，我推荐阅读的是 《大话数据结构》（程杰 / 清华大学出版社），至少不会像严奶奶的书那样大量使用很多伪代码让人难以理解，这本书的好处是能让人看得下去，使用C语言进行讲解示例，而且讲述了每种数据结构如何运用在我们熟悉的生活中，对数据结构的理解很有帮助～
>
> 接着，应该了解一下C++语言～我上面已经安利过C++有哪些好处，但是我本人在学习算法过程中接纳并学习C++经历了很大的波折（辛酸～），一开始我会了C语言，开始学Java做开发，当时觉得相比较而言Java真的比C更人性化更高级，里面封装了很多好用的函数，所以打算用Java刷算法，后来发现大量超时，又换回C语言刷算法，在刷题过程中看其他博主的博客发现很多使用了C++的语法看不懂，想找到一篇纯C语言实现的很难，而且大多数代码并不优雅，反而冗长地让人不想去了解～所以只能被迫学习C++语言，一开始买了《C++ Primer Plus》中文版，900+页的厚书，虽然很多和C语言内容重合看起来并不累，但看着很难挑出重点，书中有很多内容和特性都是更适合于做C++开发的人去学习，而对刷算法这件事儿并没有多大帮助，但也没有办法，毕竟对于一个算法路上的初学者来说，我并不了解哪些有用哪些没有用，每天带着那么厚的书去自习室很累，最后书被我用刀分成了3份每次带一小半减轻重量…也许正是经历过这些，我才会去写短短15页内容的 《从放弃C语言到使用C++刷算法的简明教程 by 柳婼》（https://www.liuchuo.net/333-3），这个教程两三个小时就可以看完，编程语言更重要的是练习，将教程里面讲过的那些特性有些印象，在刷乙级or甲级的过程中能够想起并主动多写多用多练，希望可爱的你们不要像我一样一开始因为怕麻烦和畏惧，错过了这么好用的刷算法的语言～

之后还需要学习一点儿算法知识，推荐 《啊哈！算法》（啊哈磊 / 人民邮电出版社），这本书里面讲解了栈、队列、列表、DFS、BFS、图的遍历、最短路径、树、堆、并查集、最小生成树等算法内容，而且内容风趣，里面对于具体算法都有代码和注释讲解，很适合刚懂C语言和数据结构的初学者学习～

说完看书就该说刷题了，首先蓝桥有自己的蓝桥OJ练习系统：http://lx.lanqiao.cn/，题库里分为入门训练、基础练习、算法训练、算法提高和历届试题，有一部分题目是vip用户才能看到，如果在学校报名了且学校购买了vip的服务并且把这个服务权限分配给了你的蓝桥杯帐号，你就能看到这些题目…（我们学校购买的名额有限没办法让每个人都有vip，我当年用学校的系账号给自己和一些学习努力的大佬分配了vip权限…）看不到也无所谓啦，把那些能够看到的题目刷一刷就足够啦（也可以直接看我蓝桥杯题解中对于那些vip题的题解）～这些题目有的很简单有的很难…不过对算法的锻炼和提升倒是真的有些帮助～除了这个题库，还有一个比较重要的，就是每一届的省赛和决赛的真题，在网上搜蓝桥杯真题或者在官网进入后的辅导资料模块（http://dasai.lanqiao.cn/pages/dasai/news_detail_w.html?id=644）可以看到这些真题～真题有助于让自己明白在真实的考试情况下考察的是什么内容，对于自己学的不好的地方的针对性复习很有帮助～

蓝桥杯省赛和决赛备考经验 | 备考注意事项

首先，蓝桥杯竞赛 C/C++ 组不支持 C++11 特性，所以不能使用 to_string 、 stoi 、 stol 、 auto 、 unordered_map 、 unordered_set 这些好用的函数啦～

其次，蓝桥杯 Java 组的 JDK 版本为： JDK1.6 ，蓝桥杯 C/C++ 组的编译器版本是： Dev-cpp 5.4.0 ， Java 提交的代码中不可用 package 语句， Java 主类名字必须是 Main ， C/C++ 语言中必须使用 ANSI 标准的函数， C/C++ 语言必须 return 0; ～

由于蓝桥杯平时刷题的时候有OJ系统可以练习，但是考试的时候却只能提交源代码无法知道是否正确，很多人就会对蓝桥杯考试产生很多疑问，例如： STL 能用吗？ bits 能用吗？可以使用 bits/stdc++ 吗？可以用 bits/stdc++.h 头文件吗？官方的回答是，无需问太多，一个简单的标准：你用 dev-cpp5.4 没有做任何额外设置，编译通过了就可以～正好蓝桥考试提供的编译器版本就是 dev-cpp5.4 ，所以不用担心太多啦～备考的时候最好就使用自己考试时候需要用的编译环境来刷题， Java 编程环境、 Dev-cpp5.4.0 和 API 帮助文档可以在官网下载到（每个考点的机房机子上装的也是大赛官网提供的这个编程环境）： http://dasai.lanqiao.cn/pages/dasai/news_detail_w.html?id=644

在每年省赛决赛开始前的两周内， 蓝桥杯的OJ就开始登录不进去刷不出网页...（可以理解，毕竟几万人参加的考试，考前肯定会有很多人去刷OJ的题目，流量太大导致服务器崩溃很正常）所以如果你想要和他们一样在考前的时候留了两周时间专心刷题，结果到时候发现OJ根本没办法刷出网页一定会很无奈的...如果真的想复习、想登录官网的OJ刷题，就一定要趁早呀～

再说一下决赛如何备考～在3月末考完省赛后，大概4月初知道结果，然后距离5月底的决赛还有接近两个月的时间，我觉得最重要的是先熟悉一下决赛的历年真题，了解决赛都考察怎样难度的内容，和省赛很不一样～如果你对自己比较有要求，想在决赛上获得一个不错的成绩，可以刷一刷杭电OJ或者看一下刘汝佳的 《算法竞赛入门经典》 和 《大学程序设计课程与竞赛训练教材：算法设计编程实验》 这两本书～

蓝桥杯省赛和决赛考试期间注意事项

首先要熟悉一下蓝桥杯的评分标准，既然不是可以直接提交代码和答案瞬间出结果的OJ判题考试，就一定要了解它评分的方式和标准，避免因为写对了答案却忽略了细节错失不必要的分数：

> 全程机器阅卷。必要环节有少量人工介入。
>
> 1. 结果填空题： 答案唯一。 只有 0 分或满分（格式错误为 0 分）。
> 2. 程序填空题： 按选手填写的代码代入程序中能否得出正确结果为判据。 测试数据与题面中的数据可能不同。 只有 0 分或满分（格式错误为 0 分） C/C++组选错了编译器类型可能得 0 分。
> 3. 编程大题： 主要以选手所提交的程序的运行结果为依据。特殊情况会参考选手程序的编码风格、逻 辑性、可读性等方面。 多个测试用例单独计分。通过则该用例得分。 C/C++选手选错了编译器类型可能得 0 分 C/C++选手主程序没有 return 0 可能得 0 分。 Java 选手使用了 package 语句按 0 分处理。 Java 选手主类名字不是 Main 按 0 分处理。

比赛前30分钟允许进考场，提前熟悉，有监考老师指导。考试期间不允许上网，也不能上网，该行为属于作弊行为。也不能访问其它考生的机器，只能连接考场服务器。比赛用的电脑系统要看赛点的机器配置，通常是 WINXP、WIN7/8/10 。 Java 选手比赛中只能使用 Eclipse Helios for JavaSE ， JDK1.6 版本， API 帮助文档，C语言选手比赛中只能使用 Dev-cpp 5.4.0 版本， C/C++ API 帮助文档。文档是 chm 格式的中文 API 帮助文档～C组与C++组选手提交答案时，一定要注意选择C或C++（即编译器

类型）。因为使用机器阅卷，很可能会因为选手选择了错误的编译器，而使自己代码无法编译通过～

省赛允许学生带笔，草稿纸统一现场发，用完再领，考完试后草稿纸要写上名字交给老师的～如想带水、吃的需要遵守赛点机房的相关要求，可在赛前一周，在蓝桥网下载的准考证上找到赛点老师信息进行咨询～（比如我们学校考点就是允许带吃的的，毕竟要考四个小时还要提前半小时进考场，而且还要提前出发半小时，如果真的5个小时不吃饭一直思考可能会撑不住…记得我第一次参加蓝桥杯省赛的时候，出考场后缓了好一会儿，感觉饿的有点晕…所以我之后组织系参加省赛的时候，都会提醒大家可以带点吃的比如面包/巧克力/水之类的以备不时之需，当然如果真的做起题目来思如泉涌基本上是来不及写代码也没有空吃的…）

在比赛时间内每道题的答案可以提交多次，并以最后一次提交的结果为准，提交次数不限制（就是说那个答案框框里最后只会保留最后一次的提交结果，后一次提交的会删掉前一次提交的内容），但是考试到点了就没办法提交了哦～

蓝桥杯大赛决赛的时候会有面包、水和牛奶提供～他们每年都要吐槽面包不好吃，说实话，我觉得很好吃呀…

赛点电脑上有的软件都可以使用，比如计算器、 Excel 、 MATLAB 、 Python 或者说参加 C/C++ 组的打开 Eclipse 写段 Java 求答案等等～比如有一次考试，一道填空题，直接打开计算器或者 Excel 的求和公式就能解决，根本不需要写代码，所以就可以直接这么做然后把答案提交上去…

LeetCode是什么 | 为什么要刷LeetCode

LeetCode（https://leetcode.com/）是美国的在线编程网站，虽然最近出了中国版（https://leetcode-cn.com/，一开始叫 领扣 后来叫 力扣 …好奇怪的名字啊…），但我还是建议刷英文网站对自己更有帮助～LeetCode网站收集了各大公司的经典面试题，包括 Google 、 Amazon 、 Facebook 、 Mircosoft 、 Uber 、 Adobe 、 Apple 、 LinkedIn 、 Yahoo 、 Alibaba 、 Baidu 、 Airbnb …LeetCode是一个在线编程的OJ网站，可以在线编写、编译、调试（改变输入样例的测试值），支持的编程语言也很多，包括 C++ 、 Java 、 Python 、 Python3 、 C 、 C# 、 Javascript 、 Ruby 、 Swift 、 Go 、 Scala 、 Kotlin 、 Rust …而且很多大中小公司也会直接或者间接使用LeetCode的题目资源，在面试的时候出算法题考察面试者的能力等～

我比较喜欢LeetCode的原因是：首先，它将所有的题目分为了三个难度等级， Easy 、 Medium 和 Hard ，当年为了锻炼算法能力和编码能力，我几乎刷完了LeetCode上 Easy 和 Medium 的所有题目（不包括付费题目和近期新出的题目）～有了难度等级的分类，就可以在刷题过程中根据自己的水平选择对应难度的题目尝试，更有针对性～其次，它几乎对每个题目都标注了 Tag 标签（现在也叫 Topics ），比如数组 Array 、动态规划 Dynamic Programming 、字符串 String 、树 Tree 、哈希表 Hash Table 、深度优先搜索 Depth-first Search 、二分搜索 Binary Search 、两指针 Two Pointers 、广度优先搜索 Breadth-first Search 、贪心 Greedy 、栈 Stack 、回溯 Backtracking 、链表 Linked List 、位运算 Bit Manipulation 、堆 Heap 、图 Graph 、排序 Sort 、并查集 Union Find 、二叉搜索树 Binary Search Tree 、队列 Queue 、线段树 Segment Tree 、拓扑排序 Topological Sort …可以看到这些标签几乎涉及了所有的基础算法知识，而且分类明确，我们在刷题的时候可以根据自己的薄弱环节进行针对性刷题（比如有些小可爱和我说学习算法的时候动态规划方面掌握的不是特别好，问怎样才能提高动态规划这部分的能力，我就会推荐他去刷LeetCode上 Dynamic Programming 的题目，这样就能针对性地练习很多DP问题，对自己的DP水平的提高很有帮助，比如我在大二的时候也有这样的困扰，当时刚刚接触动态规划，很多概念较为模糊遇到问题也不知道该如何应用，所以我就在LeetCode上刷了很多DP的题目，并且将这些题目由易到难整理在了《LeetCode上Tag为动态规划(Dynamic Programming)的

题目整理》这篇博客文章中：https://www.liuchuo.net/archives/1177，我当时贪心算法也学的不是很好，所以还刷了几道贪心整理在了《LeetCode上Tag为贪心算法(Greedy)的题目整理》这篇博文中：https://www.liuchuo.net/archives/1231，有兴趣的小可爱可以看一下～）再者，LeetCode每道题目都有对应的讨论区（Discuss），里面有全世界程序员提供的针对这道题的优秀代码，这些代码大多很简洁优美，有些代码更是逻辑惊艳得让人拍案叫绝，向这些优秀的代码学习，自己的算法能力也会极大提高～在刷题过程中如果你有了更好地题解，觉得比讨论区的代码还要惊艳，也可以在讨论区发布你自己的题解～我也在讨论区发布过很多自己的题解，有些也广受好评，比如https://leetcode.com/problems/max-chunks-to-make-sorted-ii/discuss/113464/c-9-lines-15ms/203413，收到异国程序员的赞美，对自己的编码信心也会有很大的提高～最后，也是LeetCode的一个优点，就是LeetCode上的每道题题目描述都很简短，没有类似于PAT和蓝桥杯题目那样的情景描述，实现的代码也很简短，所以平均刷一道题只需要15、20分钟，而且是在线编程，无需在电脑上打开编译器或者编辑器，直接在网页上就能写、调试、运行，提交AC后还能看到自己的代码执行效率击败了同等语言的百分之多少的人（这比很多OJ在不同语言维度规定同样的执行时间的行为更公平），也不需要像很多其他OJ一样动辄一道题就要一两个小时甚至半天的时间才能解决，对于学习算法的效率有很大提高～

总得来说，LeetCode不仅仅是为了应聘那些大公司而刷题的一个工具，更可以通过它来锻炼自己的算法和编程能力，很多人在大学中虽然学过了编程语言，也懂了基本的数据结构与算法的概念，但是在代码方面却很薄弱，没有大量的训练算法题提高自己的编码能力，而LeetCode就是非常合适的工具啦～如果是为了面试大公司，那么在考算法的笔试面试中，很大概率会遇到LeetCode的原题或近似题，所以刷LeetCode对找工作很有帮助～

如何高效刷LeetCode | 关于白板编程

既然题目按难度分为了 `Easy` 、 `Medium` 和 `Hard` ，我们可以按照题号顺序根据题目的难度等级练习，先解决 `Easy` 的题目（ `Easy` 是真的简单），再攻克 `Medium` 的题目（ `Medium` 有的难有的简单），最后 `Hard` 可以量力而为（除了特别大的公司，一般很少考 `Hard` ），这样的顺序能够保证你刷题过程中覆盖到了绝大多数算法题型～当然你也可以按照题目每次提交的通过率 `Acceptance` 从高到低刷，这样大致难度也是从易到难，不过还是要参考一下题号的大小，因为题号越靠前刷过的人越多， `Acceptance` 的百分比就越具有客观性，比如你看到一道题 `Acceptance` 很高，但是题号很靠后，那是因为能够坚持到这个题号的都是大佬，萌新进来刷个几十题（还很有可能是靠前的题号）就跑了…所以大佬们刷的多的题目即使很难说不定通过率也很高…所以 `Acceptance` 不是绝对的…但对于题号较小的题目有一定参考价值…

如果你已经刷过别的OJ，对于算法已经了解了大概，只是有个别类型的题型掌握的不是很好，那就可以像我上面所说的那样按照 `Tag` 来刷，针对自己的薄弱环节进行刷题，收效更快～但是 `Tag` 的缺点就是可能会形成惯性思维，比如一道题你没有思路，看到标注的 `Tag` 是 `Hash Table` ，就一下子明白要用 `hash映射` 来做，但是真正面试的时候只会给题目可不会告诉你这道题的 `Tag` 是什么…

以上就是两种刷LeetCode的顺序，根据自己的需求来选择，最好是两种结合～先广度优先地覆盖各类算法题型，顺便培养自己的编码能力和语言能力，再针对自己做的不好的题型深度优先地刷对应的 `Tag` ～

因为LeetCode每道题目的简短特性，你可以利用碎片时间进行刷题，比如课间、晚上没事刷一两题～

如果你是为了准备公司面试，有的公司是纯粹的“白板编程”，比LeetCode这种在线编程的要求还要高，比如给你张A4纸让你手写代码，或者给你一支白板笔直接在白板上手写代码…有些人摸到键盘在有了括号匹配和代码提示的情况下写代码没有任何问题，但是让你徒手写代码就写不出来了…所以在平时练习代码的时候也要注意这方面能力的培养和训练～

感谢阅读，祝小可爱们都能搞定算法，爱你们么么哒～(๑• . •๑)

Mail: 1051276278@qq.com UUID: 047a855b-d070-38c9-9eb5-eb7a59cc850f