这篇文章最早发在团队的文档中。我的团队鼓励每个同学都在业余时间多做算法题，特别是新人。个人认为在编程能力的提升上比做 side project 更有用，对职业发展也是如此。当然出于兴趣做的 side project 另当别论。转到这里，希望对各位读者有用。

**如何有效地写算法题**

**目的**

持续做算法题的目的仍然是自身能力提升。可以继续细化成三点:

保持思维敏捷。非常重要，状态好才能保持对编程的热情。  
对基础的数据结构、查找和排序保持熟练。能解决日常开发中的性能相关问题。

积累对问题域的探索。只有对问题域有足够的探索，才可能举一反三，迸发灵感。

**方法**

为了更有效地实现上面的目标。推荐用下面的方式来做题：

严格使用番茄时钟进行规划

在刷题的过程中非常最容易产生挫败感，无法坚持。原因是，长时间的思考导致疲倦，多次积累的疲倦使得自己产生了 抵触记忆。以至于会下意识觉得做题就是 刻苦。

推荐大家在开始之前看看《意志力》。里面指出 喜好 是会被记忆操控的，如果每次做一件事最后留下的映像都是轻松愉快的，那么人就会越来越喜欢做此事，反之厌恶。所以为了能保持做题的兴趣，务必每次要主动给自己留下好的记忆。

番茄时钟能够很好地保障不会出现 长时间 的思考，同时也能保障不容易 疲倦。如果你已经能很熟练的使用番茄时钟，请跳过。如果你对番茄时钟的印象仍然只是20分钟休息一次。那么请继续阅读。

番茄时钟有两个重点，一是通过长期的训练，让大脑习惯在一段时间内保持高效。二是通过要求每次在开始前有规划和每次结束后有总结，保障产出。当把这两点应用到做算法的过程中时，应该采取以下的方式：

用一个番茄时钟对题目进行彻底的分析

目前 leetcode 上的题大致可分为几种类型：

* 对某种复杂规则的彻底解析，很有可能要构造状态机，充分考虑边界情况。
* 对某种数据结构及算法的应用。
* 对数学概念、遍历、动态规划等的综合应用。

在这个分析过程中首先要大致判断出属于哪一类。在掌握了基本的数据结构和算法后，应该能很好的判断是不是属于前两类。如果判断不出说明需要回头先重新复习基本数据结构。推荐《算法》一书。不要强行刷题。算法书的每种数据结构及算法的大概思路、解决的问题以及相应的时间和空间复杂度了解之后可以再回来。

第一种情况

例子：https://leetcode.com/problems/valid-number/description/

这个番茄时钟内的目标是：

* 理清题目背后解法要用的技术
* 充分收集可能涉及到的边界

完成后应该有的总结是：

* 是否理清了要用的技术
* 是否有不确定的地方
* 收集到的边界是否能覆盖所有情况

如果发现在要用的技术中有不熟悉的地方，应该立即中断，开启另一个番茄时钟进行学习。切忌盲目尝试。当发现有不确定的地方时，重新开启一个番茄时钟，按照当前思路把不确定地方当成一个单独的算法问题进行解决。

第二种情况  
例子:https://leetcode.com/problems/reverse-pairs/

这一类题目通常采取遍历的方法一定都能找到解法。重点是找到最优解，因此需要提前有足够的数据结构的知识。数据结构可大致分为链(数组、栈、队列)、树、图。在这三类数据中要分别掌握排序和查找算法。特别是相应的时间复杂度。

这类题目很好判断，通常题目中会描述了几个数据或者状态的关联的关系，然后需要你找出符合条件的某些数据。那么将题目中的关联关系转换成相应的数据结构，再使用对应算法就够了。要对数据结构的足够熟悉，才能知道如何转化。

这种情况下番茄时钟的目标是：

将问题转化为对相应数据结构的问题。

总结是：

* 需不需要分情况讨论，需要一种数据结构还是多种
* 相应数据结构是否能完全覆盖题目问题中的所有情况

第三种情况  
例子：https://leetcode.com/problems/minimum-window-substring/

这一类情况最好用排除法，发现不是第一种或者第二种，那么再往这种情况下考虑。这类题的特点是通常是发散性质，刚看到题目容易有思路，但不太容易找到最优解。这种情况下，也要先判断题目子类型。

* 如果发现题目能从遍历的角度解决问题，那么可以往遍历的优化上去想。例如是否在遍历的时候能够排除掉一些情况。或者通过排序等手段之后，能实现遍历时排除某些情况。
* 如果发现题目中存在多种约束关系，然后求某个值，那么可以往数学方程组上去想。
* 如果发现问题可以被递归解决，并且能够将递归方式转化成顺序方式，可以往动态规划上去想。

在这种情况下，番茄时钟的目标：

判断出题目类型。

总结：

* 是否有其他类型更适合。
* 是否需要多种手段结合。

**执行时的番茄时钟**

当分析完之后，建议不要开始写代码，一定要休息片刻。执行阶段是对我们平时写代码状态的一种锻炼，应该非常珍惜。如果一个番茄时钟执行不完，应该拆分成多个。在这段时间中，设定的番茄时钟目标应该是：

**高效地验证分析阶段的思路**

要实现执行高效，最重要的是养成良好的编码习惯，不要犯小错误。要始终朝着只要想清楚了，一次写好，不要调试的状态要求自己。这里常见的小错误有：

* 拼写错误。变量命名要足够清楚，不要用单个字母或者语意不明的单词。
* 数组边界未考虑。
* 空值未考虑。
* 用 Math.ceil 之类函数时未考虑清楚上下界。

调试超过写代码时间 30% 时说明状态非常有问题。在这个阶段的总结是：

* 是否完成了对分析的验证
* 编码过程是否足够高效

如果中间发现了分析阶段的错误或者疏漏。应该立即结束编码，休息。并且重新开启分析阶段的时钟。切忌边写边改方案。如果发现编码过程状态不够好，应该加长休息时间，或者干脆结束掉。不要给自己留下低效的映像。将任务留到第二天其实也可以检验自己第一天的思路是否足够系统化，如果是，那么第二天应该能很快的重新找回思路。

**任一番茄时钟结束时**

一定要做好总结，特别是当没有解出题来，没有思路的时候，一定要通过结束阶段的总结来反思犯了什么错误。解出来了也一定要总结题目的特点，题目中哪些要素是解出该题的关键。不做总结的话，花掉的时间所得到的收获通常只有 50% 左右。

在题目完成后，要特别注意总结此题最后是归纳到哪种类型中，它在这种类型中的独特之处是什么。经过总结，这样题目才会变成你在此问题域中的积累。

做好总结，让每道题都有最大的收获。一个月之后自己的状态应该会有很大变化。

**如何分享**

在这个仓库中进行解题分享时，建议大家就把自己番茄时钟的执行记录进行分享。最后标准的解法以及思路其实在 discussion 中都有。对他人有用的分享不是结果，而是：

* 你在番茄时钟中是如何规划的，也就是番茄时钟的目标。
* 你是如何分析，也就是思路。
* 你的结论是什么，或者是你在执行时除了什么问题。
* 你所总结出的题目的关键部分。也就是对问题域进行探索的经验。