pro tips：https://codeforces.com/blog/entry/113785；https://www.luogu.com.cn/blog/119621/ni-xu-yao-lao-ji-di-shi-qing-post；https://codeforces.com/blog/entry/98806。

看题时：仔细读题，无关的背景跳过，有形式化描述之后的部分要一字一句的读。

看完题后：关注数据范围，留意题面里的关键信息。

想题的时候：

- 从题面里的关键部分出发，有逻辑的寻找突破口，要有思维的逻辑链。
- 不要急于解决整个题，很多时候一个题的几个性质是要一个一个慢慢发掘的。
- 很多时候一个题需要尝试很多种思路才能找到合适的，不要卡死在一个思路里。比如 dp 不行就想贪心，直接数不方便就要想容斥。再不行就继续枚举。有时候你甚至可以直接开始枚举所有你学过的内容，看看能不能 apply 到这个题目上。切忌认定了一个题就只能怎么做。
- 遇到自己的弱项不要害怕，要保持自信。
- 不要轻言放弃，在彻底失去新思路之前不要看题解。
- 当一个题的结构太过复杂到你摸不清头脑的时候，尝试化简题目。对有部分分的题目可以对着部分分想，如果部分分还不会就可以自己编一个更简单的部分分出来想。图太复杂就想树，树太复杂就想链或者菊花，多次询问太难就先想单次询问，计数太难就先想判定。一般来说，从对这种特殊性质或者化简了的问题的研究中你总会得到一些对解决整个题都很有用的关键信息。
- 还有一种时候，比如题目给了一个具体的过程的时候，手动模拟一些很小的 case 会帮助你熟悉这个过程，让你对其有更好的感知，也会比较有用。但模拟的时候要注意思考和观察，不要想着要尽快手模完然后只是进行机械的计算。
- 也别忘记打表或者找规律。这也是化简题目的方式。
- 如果觉得思路太绕，可以写在纸上或者打字打出来
- 不要轻言放弃，在做完上述的所有事情之前不要看题解。
- 如果觉得怪怪的，再确认一遍题有没有读对。

写代码之前：捋清思路！Think Twice, code once. 例如：dp 状态的设计和转移，使用容斥的条件，数据结构题的每个模块内部和模块之间的关系等等。可以把以下这些东西都写在纸上记下来（其实在打草稿的时候最好就记好）。切忌要开写的时候连自己到底要做什么、写什么都还没想清楚。

- What's the complexity?
- Do I understand the problem correctly?
- What functions do I need?
- Which places are tricky? What do I need to remember to implement them correctly?
- Which place is the heaviest by implementation? Can I do it simpler?
- Which place is the slowest? Where do I need to be careful about constant factors and where I can choose slower but simpler implementation?

- **OI 赛制中的比赛技巧**：将已经会了的几档部分分以及你预估代码实现的时间罗列出来，评估一下性价比，选择一个最优的去做。也可以从这些 subtask 里寻找进一步突破的启发。

写代码时：

- 有时候容易陷入"刷牙状态"的自动档里去，这时候就很容易出错。保持专注，不要分神。比赛中其实挺难做到时时都专注，不要让自己太紧张或者有过大的压力，保持放松且专注的状态写代码是最好的。
- **代码技巧**：把你觉得难以实现的部分先用暴力写，之后一部分一部分再改成正解。这个方法尤其在 OI 比赛中好用，有时候暴力写着写着会发现简单的正解写法，甚至直接能会了正解。在非 OI 比赛里也可以用。
- 养成良好的代码习惯，该避的坑都要避掉。多学习别人代码的优秀实现。

调代码：

- 不要一股脑地调试输出。也可以仔仔细细地瞪眼调。
- （Linux 的 fsanitize / Debug）
- 不要卡死在一个地方。有些时候，你觉得最容易出错的地方实际上都没写错，而错误往往藏在你忽略了的地方。
- 冷静，不要急，不要做无意义的输出调试。
- 我有写一段代码瞪一段，看看有没有写错的习惯。我其实也不太擅长调试，一遍写对或者尽量少写错是最重要的。
- 对拍：如果可以的话，造比直接随更强的数据（NOIP2022 T2）。

AC 之后的反思与总结：

- 前面几部分里的什么环节我可以做得更好？
- 学习别人对这道题的优秀实现
- 可以适当将总结文字化记录下来

P4818：总收益只和（设定的学费，足够富的人数）有关。最大化两者的乘积，一种经典做法是枚举其中一项，算出另一项的最大值。

- long long
- 答案的选取

P4819：

一眼：怎么感觉有点难？但大体模型是"定长区间覆盖"的问题。

- 只有一种牛怎么做？经典的"定长区间覆盖"问题，是从左往右贪心的经典结构（经验的积累）。
- 位置可以重叠怎么做？
- 这个贪心是从左往右的，这个从左往右考虑的顺序可以被用来 dp 吗？
- ...等等，位置真的会重叠吗？
- 只有可能在 n 处出问题，怎么解决？试试 $n-1$？

P4820：

这个过程意味着什么？怎么判定给定的一个 if else 序列是否合法？

什么样的输入，我知道它一定不可以？如果我知道至少能做一步，之后又该怎么办？

诶，好难啊。这是铜组啊兄弟！不要想复杂了。

P4815：

这个题有经验的同学可能一眼就会做。如果没有做过类似的题，第一次做的同学该怎么想呢？

- 什么时候一定存在合法的序列？只要能整除就可以了，因为我可以任意地实现 h_u 减 1，h_v 加 1。
- 答案是什么？
  - 特殊情况：菊花/链
- or 冷静思考一下，我们考虑某一条边：
  - 运输一定是单向的
  - 方向、以及具体的值是可以确定的！
  - 只要这条边能做就做，做完就可以分裂成两个了！

    - 只要一个点能做就做。可以用 bfs 的形式维护。
    - 更有经验者：不妨连边表示，变成有向图，拓扑排序即可

做完这个题之后：我能怎么推广这个题？

- 给定初始序列和目标序列
- 环？图？

P4816：

博弈题？乍一看没有思路啊

- $n=1$？11101110？可以简单地归纳证
- $n>1$？对于每个 a_i，能控制这个位置的人一定不会将控制权拱手相让。在此基础上，他想速战速决，另一个人想拖！
- 因此要求出一个最短路状物。怎么做？显然可以 dp，但复杂度太高了。
- 遇事不决先打个表？或者对观察力很自信，直接考察其性质。

P4817：

- 给我的信息看上去很多的样子。我能很容易地知道什么？
- 好像把一个合法的序列全取相反数也满足条件。
- 整个序列能被唯一的确定吗？（除了取相反数）
- 没有什么头绪。不如先直接考虑简单情况。
- $n=2$？简单
- $n=3$？好像需要分 b[1] 是否和 b[2] 相等讨论。然后用 [1,3] 的极差判断就可以了
- $n=4$ 呢？好像可以用 $n=3$ 的类似的方法诶！
- 好像可以一直这样做，直接确定整个序列！
- 诶，既然这样不如把相等的段先缩起来吧（可以想一想怎么实现这个过程——优秀简洁的代码实现能力），这样相邻的两个数都互不相等了。
- 然后只需要随便钦定前两个数的值，然后三个数三个数的做！

P4812：

- 要是 X_i 都无限大呢？直接背包就行
- 有 X_i 的话，似乎要做二维背包的样子。但又不可能弱于背包，所以要努力去掉一维

- 但如果已经确定了有哪些人要收买，尽可能用 X_i 小的是更优的（贪心的直觉）
- 所以我们不妨考虑判定，给定一些人，怎么做，显然是按 X_i 从小到大贪心
- 贪心能对我 dp 的改进带来什么启发？我不妨按 X_i 排序。然后呢？
- 考虑刚刚的判定，一定是一个先全用 X_i，等到用完了再用 C_i。所以两维是不用同时记的！

P4814：答案只和（最小度数，点集大小）有关。和 P4818 是一个道理。

- 先将每个连通块分开考虑。
- 对于一个连通块，怎么做？

（和第一题是一样的，可以画出（最小度数，点集大小）函数图像）

- 显然点集大小最大的答案候选项就是整个块。
- 如果答案不是它自己呢？要删去一些点，会让点数变小。那肯定得让最小度数增大！

因此达到最小度数的那个点是一定要被删去的。删去之后，再依次考虑分裂出来的几个连通块即可。

- 进一步，各个连通块可以合在一起做。
- 删点序列是好求的，连通块分裂不好维护。可以倒着合并。
- 可以用桶排序代替优先队列去 log。
- 答案不用开 long long。

P4811（$O(n^3)$）：

先考虑算一个串答案：

- 两个 01 序列从一个变到另一个的步数要怎么算？（经典问题的积累）
  - 考虑一种数字的移动
  - 考虑前缀和（P4815）
- 一个 01 串构成回文串的条件是什么？
  - 考虑一种数字，出现的位置必须成对

  因此采用“用一种数字的所有出现位置”代表整个串。比如只考虑 1。
- 将所有 1 首尾配对，先考虑 1 的个数为偶数的情况。每对 1 最终的位置 x, y 一定满足 $x + y = n + 1$。

- 考虑数形结合。二维平面！
- 咦！我会了！

学习资源: 算法竞赛进阶指南、ix35（https://www.luogu.com.cn/blog/ix-35/）、command_block（https://www.luogu.com.cn/blog/command-block/blog-suo-yin-zhi-ding-post）、OI 中转站（https://yhx-12243.github.io/OI-transit）、Alex_Wei（https://www.cnblogs.com/alex-wei）、atcoder(https://kenkoooo.com/atcoder/#/table/)、qoj（qoj.ac）

提升思维能力：

- 唯一的途径就是**独立思考**，不要把一天过了多少题作为 KPI，而是**专注且高效的思考时间。**（因此或许很难规定自己每天要把什么题做好，至少我从来没有做到）。如果比作健身的话，看题解像是学会一个新的动作，自己独立思考的时候才是在真正锻炼的时候。（讲讲独立思考的意义和自己在这方面的转变，思考时保持自信真的很重要）
- 做思维含量高的题：日本题
- 更 ad-hoc 一些：AGC、https://codeforces.com/blog/entry/113093。

刷什么题：CCF官方真题（不一定好但是价值最大）、ATC/CF、JOISC/JOI

模拟赛资源的推荐：官方真题、pjudge(pjudge.ac)、uoj(uoj.ac)（OI 赛制）、USACO、**JOISC、JOI**、IOI、CEOI、PA（这些在 loj.ac 上都有） 等等。机构的题不一定好，联考其实也并没有很好。不过机构的好处是为你提供了一群同学。想一想，打模拟赛是为了什么？

- 模拟全真环境，锻炼比赛心态（打 cf/atc 也可以有帮助，主要还是靠真实的比赛经验积累）。只要认真 vp 上述比赛也是可以做到的。
- 提升临场策略的设计能力：需要有合理、精细的部分分设计，大部分机构和联考的模拟赛做不到这一点。可以拿一些真题，用讲课最开始分析一下每个题部分分。

如果大家觉得信息比较闭塞：可以尝试更关注算法竞赛社群一些，比如在洛谷上多关注一些竞赛选手，加一些活跃的算法竞赛讨论群（uoj，luogu），cf 上的 blogs 等。或者多关注一些人的博客，luogu/cnblogs。如果在现实生活中没有一起做题的同伴，可以找网友作伴。如果没有可以请教问题的学长，可以大胆向高水平选手问问题。

文化课与 OI 的平衡：讲讲自己初三的经历，可能最重要的还是热情。如果有很多作业要写的话确实比较麻烦，试试看能不能糊弄过去或者向学校申请一下。

如果时间不够的话，可以考虑每天 vp 一场 ATC/CF（vp 的好处：提升专注思考的时间），下一天把上一天没过的题补了。

我去 WC 吗？去的！

天赋：我感觉我是后来慢慢开窍的。最主要的可能还是喜不喜欢。停一段时间的课是问题不大的，也可以尝试申请选择性听课/不写作业等。

抽代、高代？我抽代是一点不会，高代大概就会到行列式/LGV/矩阵树/BEST。参考考纲就行，不感兴趣的话就不用学，比如多项式也不用学（但例如插值还是要会）。OI 里的数学最重要的还是组合数学。

题单推荐：上面已经说了一些。有针对性的刷题，让自己少一些非常害怕的题目类型，CF 的 filter 挺好用的。但是不要在已经知道一个题怎么做的情况下去做。

比赛前的心态调整（个人经验，仅供参考）：不要对自己要求过高，放低期待。

放平心态！OI 不是一切（文化课也不是），只是人生经历中的一小部分。和”上一个好大学“同样很重要的事情还有：尽量让自己开心和保持健康的心态（感觉一直全停课的话很难做到这一点）、**交朋友**、玩和**休息**、向内探索（认识自身的性格上的优缺点、感知自己内心情感的运作方式、锻炼处理负面情绪的能力、找到适合自己的学习/生活模式、让自己在心智上不断成长）、向外探索（感知世界、思考人生（雾）、探索自己感兴趣的事情）。（这些都是我觉得自己本来可以和接下来应该做得更好的地方。总之不要把所有事情都押在 OI 上，不要给自己太大的压力！）