量化金融面试刷题笔记

Echo-gram

2026 年 1 月 19 日

前言

这是一份量化金融面试刷题笔记, 希望能帮助读者更有效率地准备面试. 如果想顺利找到工作, 仅仅刷绿皮书还不够, 你最好能比它" 多掌握一点", 本文就是在阐述作者对" 多掌握一点" 的理解.

作者本人是在美国的一个量化金融项目读硕士. 我在一开始准备量化面试的时候, 和大多数人一样, 也是刷所谓的绿皮书, 红皮书. 诚然, 这是一条康庄大道. 不过我在最初的练习中, 我常苦于自己没有能想出这些答案的聪明才智, 于是只能暂时记住这些做法以应付面试. 由于我找工作的经历十分曲折, 相比于同伴们花费了更多的时间. 在这一漫长的过程中, 我逐渐也写了许多市面上常见的其它面试书籍. 随着练习的次数逐渐增多, 当时仿佛自己一辈子也想不出来的解答好像也不过如此.

回头想想, 自己能力的提升不外乎来源于以下两点: 第一是在刷一些新题的时候, 可以联想到旧题的一些思路, 避免了机械回忆答案. 第二是, 在学习其他主题或学科时, 新学习的办法可以用于旧有的题目, 使得我们可以想出比原书更加巧妙的解法.

基于以上的思考, 这份笔记的安排如下, 主体内容将是以绿皮书为线索, 我将绿皮书的一些原题配备上拓展练习的链接, 这样读者在刷题的时候, 可以自行训练举一反三, 事半功倍. 所有这些题目加总起来的范围, 应当足以应付一般公司的面试了. 此外, 如果我发现绿皮书的题目有更好的解法, 我也一并附上, 希望读者能够对这些题目有更好的了解. 在主体内容之前, 我会用一个小节介绍市面上一些优秀的面试材料作为拓展题目的来源. 在这些内容之后, 我将会介绍一些面试书籍没有覆盖的话题以及对应的准备资料.

最后, 我想聊一下刷题的意义. 理科学习的过程中, 做题是必不可少的, 有的时候独立写出一些题目确能产生不少成就感与快乐. 但是从工作的角度来说, 这些题目与实际工作的关联其实并没有那么紧密, 它们更多是作为一种筛选工具存在, 是套在我们每个人身上的枷锁. 但是无论喜欢与否, 掌握这些题目都是我们求职过程中必须要经历的一部分, 是我们上岸的必要条件. 喜欢的话, 我祝愿你享受其中, 不喜欢的话, 我祝愿你早日从枷锁中解脱.

更为重要的是, 扎实的基础知识对你面试状态的提升是有很大帮助的. 心有盈余, 方能气定神闲, 谈笑自若. 如果题目本身难不倒你了, 你就可以把更多的心思放在让自己不要那么紧张, 表达更有逻辑, 做好眼神交流以及表情管理. 就像自己的脑海里已经有答案的小抄了, 将其自然转达出来就好.

当然即便你以上都做得还不错了, 找工作依然是一个极度需要运气, 并不由自己掌握的事情, 精雕细琢往往只能换来草率收场. 我们能掌控的, 是韬光养晦, 静待天时, 在每一次机会来临的时候, 能够以良好的精神面貌和知识储备去迎接; 是相信大数定律, 相信投递样本数的增加终究会使找到工作的频率收敛于真实概率.

限于自身学识以及经验, 这份笔记的疏漏以及偏见在所难免, 我只能做到有力量五分便使尽写出五分. 作者邮箱为 echo80221@gmail.com, 希望各位读者不吝赐教. 祝愿大家求职顺利.

目录

1 量化面试书籍与网站

首先来聊聊最为知名的**绿皮书**，即 Xinfeng Zhou 所著 *A Practical Guide To Quantitative Finance Interviews* [12]. 伟大无需多言. 这本书的优点数不胜数, 具体来说, **绿皮书**的题目与面试高度重合, 即便在 2025 年的面试中仍能频繁遇到原题或类似题. 其次, **绿皮书**解答质量在同类书籍中是最佳的: 作者注重一题多解, 除了讲述特定解法外, 作者还会解释具有推广性的通法, 令人醍醐灌顶. 最后, **绿皮书**解答简洁明了, 大多数不会超过一页, 不像其他书籍四五页冗长的解答.

基于以上的这些特点, 我认为它不仅是准备 Quant 面试的必读第一本书, 也是最值得反复研读的一本.

我觉得**绿皮书**唯一的问题在于, 只写**绿皮书**是不够的, 对于传统的概率题, 特别是一些买方, 面试官往往会问一些更难的问题, 此外**绿皮书**对于面试中的一些其它话题, 例如 stochastic differential equation, linear regression, machine learning, C++ 八股文, Python 八股文等并没有足够的覆盖, 这也是我写这篇笔记的原因.

下面是一些针对于量化面试的其它书籍.

我第二喜欢的面试书是 Dan Stefanica 所著的 *150 Most Frequently Asked Questions on Quant Interviews* [10], 以下简称 **150**, 最新的版本已经出到了第三版. 这本书可以说是绿皮书的绝佳补充, 题目与绿皮书的重合度是比较低的, 而且其中关于 Financial Instrument, C++, Stochastic calculus 的内容都极大补充了绿皮书所没有的部分, 而且解答写的十分详尽. 唯一的缺点在于, **150** 有些题的部分解答会出现没有必要的冗余, 影响理解. 对于这一缺点我的做法是, 依照原答案的思路写出更为精简的解答.

除了**150**, Dan 还出版了两本更难的面试习题书. 分别是 *Probability and stochastic calculus quant interview questions* [6], *Challenging Brainteasers for Interviews* [8], 以下分别简称**DanProb**, **Brainteaser**. 这两本书延续了**150**的风格, 选题普遍偏难, 而且有相当一部分的计算过于繁琐, 并不是面试场合能算出来的. 对于这两本书, 我的想法是如果时间宽裕的话, 优先看**DanProb** 的第一章 Discrete Probability, 里面大部分内容都不会利用太繁琐的计算就可以解答, 而且的确在面试中遇到过和这一章内容非常类似的题.

还有一本值得阅读的是 Mark Joshi 所著的 *Quant Job Interview Questions and Answers* [4], 已经出到了第二版, 由于第一版封面是红色的所以也被人称之为红皮书, 以下简称 **Joshi**. 我主要推荐的部分是第 2 章 option pricing 以及第 4 章 interest rate. 在阅读的时候你可以感受到作者对于 derivative pricing 深厚的理解, 可以学到一些很难在别的地方中接触到的真知灼见, 比较好的例子可以参考书中的 2.29, 2.58, 2.59, 4.4 这些问题. 除此之外, 这本书和绿皮书重合度比较高, 解答的质量却比不上绿皮书. 作者还在每一个问题的答案后附加了一些 related questions, 但是作者却没有提供答案, 这也使得这些 related questions 的价值被大大削弱.

最后一本是 *Heard on the street: Quantitative questions from wall street job interviews* [1], **Heard on the street**. 这本书作者每一年都会更新一次, 现在已经更新到第 26 版

了, 这本书中值得一看的内容有第 1 章中 Purely Quantitative & Logic Questions 的一些前面的书没有囊括的 brain teaser, 以及第 5 章 Non-Quantitative Question 中, 作者给了一个行为面问题的清单, 大家都可以查漏补缺. 其余内容和前面的书均高度重合.

除了书籍以外, 我还曾经在一个网站上刷过一些量化面试题, 这个网站叫做 Quantable, 链接为 https://www.quantable.io/. 这个网站需要每个月 30 多刀的订阅, 订阅后就可以解锁很多题目以供面试准备. 除了这个网站, 市面上还有很多类似的竞品, 我觉得 Quantable 的优点是它对于题目的分类做的比较好, 网站上有各种各样的 question list, 点进去还会有根据知识点进一步的细分, 比如一部分是排列组合, 一部分是条件概率.

以上的所有都不是广告, 我只是列举了我自己买过, 并且从中收益的资源. 大家可以根据自己的时间安排进行选择. 我的建议是先从**绿皮书**, **150**入手, 然后根据具体岗位的要求, 依照剩下的资源进行补充.

2 绿皮书读书笔记与拓展题

2.1 引言

我们现在进入正文部分. 我们将会以绿皮书为索引, 对绿皮书原题, 相关数学背景知识以及绿皮书未囊括但是对面试有帮助的题目, 进行梳理, 并与绿皮书原文顺序一一对应.

在进行正式内容之前, 我们可以先将绿皮书的题目分为三类:

1. **教材型题目:** 这一部分的题目其实就是教材的经典例题, 在本科数学课中大概率接触过的内容, 例如 Gambler's Ruin, Markov Chain First Step Analysis, 动手解 ODE. 读者如果已经学习过这些内容, 最好应该去看当时学习所使用的教材或笔记, 效果会更好. 如果没有学习过, 我会附上一些我在网上搜集到的 notes 作为参考. 这些内容在熟悉数学知识之后只是一些简单计算题.

2. **Brain Teaser 型题目:** 这些题目解答和题目的设定高度绑定, 几乎无法拓展改编, 面试官大概率也只能问你原题, 因此我认为只用做到背题的级别就足够用了. 此外, 我认为准备 brain teaser 的投入产出比非常非常低, 你能否回答出来很大程度是取决于你是否之前见过这道题, 也就是说区分度很低. 我个人觉得这部分的阅读优先级可以放在最后, 这份笔记里面也不会花太多篇幅去介绍拓展题目.

3. **可拓展型题目:** 剩下的题目就是一些可以用通法解决的题目了, 要么是一些排列组合, 条件概率里面的通用技巧, 要么是一些在绿皮书里可以举一反三的技巧, 这些也就是我们值得花时间的题目了. 这些题目我会附上一些拓展题, 希望能提升读者的计算能力以及对概率论中的常见技巧的掌握.

对于绿皮书中的原题, 如果有一些背景知识或者是解法的补充, 我会放在橙色框内. 如果我找到了对应的拓展题目, 我会把题目的链接放在红色框内, 其中如果是特别重要的内容我会标红来进行区分.

本文所提到的所有拓展题目都可以在第4部分找到一个汇总.

2.2 Brain Teasers

Brain Teaser 这一节的大部分拓展题目其实都可以当成图一乐, 如果说有什么相对重要一点的可以看本节最后 Additional Related Questions 里面的题目. 前文里也提到了, 这一块很多题目区分度并不高, 所以有不少内容我是觉得没什么值得拓展的, 但是为了保持笔记的章节编号与绿皮书原书章节编号一一对应, 我加上了一些空白的章节.

2.2.1 Problem Simplification

Screwy pirates

拓展题目

- **Brainteaser**- 1.1 'Aha' Questions- Logical Conundrums- Question 4 (纯图一乐)

2.2.2 Logic Reasoning

Defective Ball

注记

Heard on the street Question 1.6, 1.21 与该题重合, 但是 **Heard on the street** 提供了更为详细的解答, 并且对于问题的背景进行了补充介绍

2.2.3 Thinking Out of the Box

Last Ball

注记

这道题我见过的最优解是利用异或运算 (xor) 的做法, 具体可以见左程云的如下算法视频的前 15 分钟：https://www.bilibili.com/video/BV1LN411z7nu/.
这个做法也是我在看他的视频准备 leetcode 的时候学会的. 他的视频是我认为准备 leetcode 最好的参考资料, 此后一些别的 brain teaser 我也会引用他的讲解.

2.2.4 Application of Symmetry

2.2.5 Series Summation

Glass balls

注记

150- Chapter 2.8 Brainteasers - Question 8 与此题重复, 但是**150**的解答更为清晰. 此外, 这题在 leetcode 也有测试链接, https://leetcode.com/problems/super-egg-drop/description/. 对应的视频讲解可以参考 https://www.bilibili.com/video/BV1vT421Y7Dr/中的第四题, 我觉得这两个讲解比绿皮书更加优秀.

2.2.6 The Pigeon Hole Principle

2.2.7 Modular Arithmetic

2.2.8 Math Induction

2.2.9 Proof by Contradiction

2.2.10 Additional Related Questions

首先是两题非常著名的而且的确会在面试中出现的题目

Josephus Problem

> Imagine n people standing in a circle, and starting from the first person you count off around the circle. Every k-th person is removed, and the counting continues with the next person still in the circle. This process repeats until only one person is left. The puzzle asks: if you know n and k, which position should you stand in to be the last survivor?

对应的测试链接为

- https://www.quantable.io/questions/3712
- https://leetcode.com/problems/find-the-winner-of-the-circular-game/description/

做法可以参考 https://www.bilibili.com/video/BV1Dz2eYTE7T/ 中第三题.

Bash Game

> A pile contains n stones, and players take turns removing stones from the pile. On each turn, a player must remove at least one stone and at most m stones. The player who takes the last stone wins.
>
> The question is: given the starting number of stones n and the maximum number removable per turn m, which player has a winning strategy?

做法可以参考 https://www.bilibili.com/video/BV1T5411i7Gg/第一题

剩下的一些题目大多是经典的 Brain Teaser

拓展题目

- **150**- Chapter 2.8 Brainteasers - Question 11
- **Joshi**- Question 8.6, 8.24
- **Heard on the street**- Question 3.1
- **Brainteaser**- 1.1 'Aha' Questions- Logical Conundrums- Question 16, 17

2.3 Calculus and Linear Algebra

我很少见到面试中直接给一个积分让你进行计算的, 更多利用到微积分还是在算概率或者期望的时候, 所以这一块的拓展并不会特别多.

2.3.1 Limits and Derivatives

2.3.2 Integration

Expected value using integration

拓展题目

- https://www.quantable.io/questions/2882

2.3.3 Partial Derivatives and Multiple Integrals

2.3.4 Important Calculus Methods

Taylor's series

拓展题目

- https://www.quantable.io/questions/3850

2.3.5 Ordinary Differential Equations

注记

ODE 的相关背景知识具体可以参考以下这个 notes, https://www.ucl.ac.uk/~ucahhwi/GM01/ODE_extra.pdf, 这个知识范围已经绰绰有余了. 我个人身边统计学中遇到 ODE 的次数也屈指可数.

2.3.6 Linear Algebra

QR decomposition

注记

绿皮书这一小节是关于 Linear Regression 的，针对面试可以特别注意补充一些额外知识，例如涉及 data duplication 的一些问题，详情可以参考https://xinweizhang.github.io/Duplicating-data-in-linear-regression，又比如 Measurement Error Model，详情可以参考 https://econ.lse.ac.uk/staff/spischke/ec524/Merr_new.pdf，这份笔记最后也会有一部分专门介绍 Linear Regression 的内容.

拓展题目

- 150- Chapter 2.5 Statistics. Machine Learning- Question 5, 6, 8, 10

LU decomposition and Cholesky decomposition

拓展题目

- https://quantinterview.github.io/Random-Point-Circle
- 150- Chapter 1 First Look: Fifteen Questions- Question 7

2.4 Probability Theory

2.4.1 Basic Probability Definitions and Set Operations

2.4.2 Combinatorial Analysis

Hopping rabbit

拓展题目

- https://www.quantable.io/questions/2517

Chess tournament

拓展题目

- **DanProb**- Chapter1 Discrete Probability- Question 11

2.4.3 Conditional Probability and Bayes' formula

Birthday line

拓展题目

- https://www.quantable.io/questions/3669

Dice order

拓展题目

- https://www.quantable.io/questions/3693

Candies in a jar

注记

我觉得这道题可以这样思考, 假设我们把所有的 60 颗糖都抽完, 然后倒过来看这个序列. 其实这道题等价于在问, 在倒过来的序列里, 在抽到第一个红色之前, 至少抽到一个蓝色和一个绿色的概率是多少.

这样其实就只用考虑两种情况, 第一种是第一次抽先抽到了蓝色, 然后在后面的抽签中绿色比红色先出现. 第二种是第一次先抽到了绿色, 然后在后面抽签中蓝色比红色先出现.

针对第一种情况, 第一次抽到蓝色的概率是 $\frac{20}{60}$, 随后我们可以压缩我们的样本空间, 在新的样本空间里面, 我们不关心抽到蓝色与否, 我们只关心绿色和红色哪一个先被抽中, 在被压缩的样本空间里面, 绿色先被抽中的概率是 $\frac{30}{40}$, 所以第一种情况的概率是 $\frac{20}{60} \times \frac{30}{40} = \frac{1}{4}$.

针对第二种情况, 第一次抽到绿色的概率是 $\frac{30}{60}$, 随后我们可以压缩我们的样本空间, 在新的样本空间里面, 我们不关心抽到绿色与否, 我们只关心蓝色和红色哪一个先被抽中, 在被压缩的样本空间里面, 蓝色先被抽中的概率是 $\frac{20}{30}$, 所以第二种情况的概率是 $\frac{30}{60} \times \frac{20}{30} = \frac{1}{3}$.

两种情况是 mutually exclusive 的, 二者概率相加就可以得到最后的答案 $\frac{7}{12}$.

拓展题目

- https://www.quantable.io/questions/2746
- https://www.quantable.io/questions/2545

Gambler's ruin problem

拓展题目

- https://www.quantable.io/questions/3075
- https://www.quantable.io/questions/3076
- https://www.quantable.io/questions/3077

2.4.4 Discrete and Continuous Distributions

Meeting Probability

拓展题目

- https://www.quantable.io/questions/2563

2.4.5 Expected Value, Variance & Covariance

Dice game

拓展题目

- **Heard on the street**- Question 4.25, 4.35, 4.58

Card game

拓展题目

- https://www.quantable.io/questions/3375

Coupon Collection

拓展题目

- **DanProb**- Chapter1 Discrete Probability- Question 13, 14, 15
- https://www.quantable.io/questions/2726
- https://www.quantable.io/questions/3016
- https://www.quantable.io/questions/2527
- https://www.quantable.io/questions/2528
- https://www.quantable.io/questions/3903

2.4.6 Order Statistics

Meeting Probability

拓展题目

- **DanProb**- Chapter1 Discrete Probability- Question 22

2.4.7 Additional Related Questions

剩下的一些题目大多是一些排列组合和条件概率的计算题, 如果感觉不是很熟悉这些计算的话, 可以用这些题目练手.

拓展题目

- https://www.quantable.io/questions/2745
- https://www.quantable.io/questions/2493
- https://www.quantable.io/questions/2618
- https://www.quantable.io/questions/2577
- https://www.quantable.io/questions/2619
- https://www.quantable.io/questions/2480
- https://www.quantable.io/questions/2482
- https://www.quantable.io/questions/2610
- https://www.quantable.io/questions/2612

- https://www.quantable.io/questions/2613
- https://www.quantable.io/questions/3297
- https://www.quantable.io/questions/3350
- https://www.quantable.io/questions/2881
- https://www.quantable.io/questions/3470

2.5 Stochastic Process and Stochastic Calculus

2.5.1 Markov Chain

注记

这一小节除了 Coin triplets Part C, Color balls 那两题太复杂可以跳过以外, 其余应该都是 Markov Chain 中的例题, 具体可以查看这个网页https://mpaldridge.github.io/math2750/S08-hitting-times.html

2.5.2 Martingale and Random walk

Drunk man

注记

这又是属于教材中的经典例题, 关于 random walk 和 gambler's ruin, 这个笔记里面涵盖了所有比较重要的性质, https://www2.math.uu.se/~sveralm/kurser/stokprocmn1/slumpvandring_eng.pdf

2.5.3 Dynamic Programming

Dice Game

拓展题目

- **Heard on the street**- Question 4.65
- https://www.quantable.io/questions/2454

2.5.4 Brownian Motion and Stochastic Calculus

Brownian motion

拓展题目

- **150**- Chapter 2.7 Probability. Stochastic calculus- Question 7
- https://www.quantable.io/questions/2555
- https://www.quantable.io/questions/2504
- https://www.quantable.io/questions/2684

Ito's lemma

拓展题目

- **150**- Chapter 2.7 Probability. Stochastic calculus- Question 13, 16, 18, 19, 20

2.5.5 Additional Related Questions

以下这些是我觉得和绿皮书原题关联不大, 但是仍然有启发性的题目. 大家可以自行选择.

拓展题目

- **DanProb**- Chapter 1 Discrete Probability- Question 33
- https://www.quantable.io/questions/3781

2.6 Finance

绿皮书里面关于 Finance 的覆盖并不是很全面, 在这里我强烈推荐**150** Chapter 2.3 Financial instruments: options, bonds, swaps, forwards, futures. 看完基本上就够用了. 除此之外还可能面试还可能会涉及推导 swaption 的 pricing formula 的问题, 可以参考以下两个网页 https://oaji.net/articles/2017/6309-1529923544.pdf,

https://www.applied-financial-mathematics.de/sites/default/files/Teaching/InterestRateMdellingWS19/IRModellingLecture_Part_03.pdf

2.6.1 Option Pricing

American v.s. European options

拓展题目

- **150**- Chapter 1 First Look: Fifteen Questions- Question 1

Black-Scholes-Merton differential equation

注记

如果直接回答出和绿皮书一模一样的推导, 有时候面试官可能并不会满意. 因为绿皮书里其实直接未经推导告诉了你, 你应该 short $\frac{\partial V}{\partial S}$ unit of underlying stock. 这个时候就会有人问这个 $\frac{\partial V}{\partial S}$ 是怎么来的. 一个相对更加完善的推导是基于 self-financing portfolio 的, 可以参考https://www.columbia.edu/~mh2078/FoundationsFE/BlackScholes.pdf.

拓展题目

- **Joshi**- Question 2.4, 2.29

Black-Scholes formula

拓展题目

- **Joshi**- Question 2.6, 2.14, 2.16, 2.30

2.6.2 The Greeks

这一节有很多 greeks 关于 underlying 的图像, 但是书中能印刷出来展示的是十分有限的, 我觉得如果能自己写一个 jupyter notebook 复现这些图像对理解记忆会更有帮助. 此外除了这些常见的 greeks 外, 我觉得 vanna $\frac{\partial^2 V}{\partial S \partial \sigma}$ 和 volga $\frac{\partial^2 V}{\partial \sigma^2}$ 也是非常频繁出现的.

Delta

注记

复用绿皮书里面的 notation, 当 $\tau \to 0$ 或者 $\sigma \to \infty$, 我自己的记忆方法是, delta 的图像会逐渐趋向于一个 step function. 另外 Step function 的导数是 Dirac delta function, 所以当 $\tau \to 0$ 或者 $\sigma \to \infty$, gamma 的极限是 Dirac delta function. 我觉得这样的描述更加简洁精确.

拓展题目

- **Joshi**- Question 2.21

Vega

拓展题目

- **150**- Chapter 1 First Look: Fifteen Questions- Question 12
- **Joshi**- Question 2.55, 2.56

2.6.3 Option Portfolios and Exotic Options

Binary Options

拓展题目

- **Heard on the street**- Question 2.10, 2.11, 2.12
- **Joshi**- Question 2.20, 2.47

2.6.4 Other Finance Questions

Interest rate models

拓展题目

- **Joshi**- Question 4.4, 4.6

2.6.5 Additional Related Questions

以下这些是我觉得和绿皮书原题关联不大, 但是仍然有启发性的题目. 大家可以自行选择.

拓展题目

- **Joshi**- Question 2.58, 2.59

2.7 Algorithms and Numerical Methods

我觉得这一部分大多数内容算得上是准备 Leetcode 的子集, 和 Leetcode 有重合的地方就在后面讲 leetcode 的内容再说.

2.7.1 Algorithms

Naximum contiguous subarray

注记

题解中提到了 max drawdown 的代码实现是一个拓展, 这个的确会有人问, https://stackoverflow.com/questions/22607324/ 里面有 $O(n)$ 的算法实现.

2.7.2 The Power of Two

2.7.3 Numerical Methods

Monte Carlo simulation

注记

对于 Standard Normal Random Variable Generation, 可以了解一下 Box-Muller Method：https://en.wikipedia.org/wiki/Box%E2%80%93Muller_transform

拓展题目

- **150**- Chapter 2.6 Monte Carlo simulations. Numerical methods- Question 3, 5

3 补充话题

在这一部分, 我将补充介绍一些量化面试书里面没有涵盖的但是面试中可能会出现的话题. 例如 Leetcode, Linear Regression, Machine Learning, C++, Python 八股文. 这些话题的出现与否取决于 job description, 面试官出题偏好, 以及自己的简历里是否有相关经历.

3.1 Leetcode

这一部分的目的, 是希望读者能够在 quant 非 developer 岗位的 live coding 面试中, 遇到 medium 难度的题目时也能游刃有余. 至于互联网公司那些更加困难的面试, 我也没有经历过, 也不认为有那个水平, 所以不在这里班门弄斧了.

我觉得 leetcode 的准备分为三个阶段.

第一阶段是熟悉基本的数据结构与算法, 即 stack, queue, binary search 的基本概念. 它们之于 leetcode 题目就像微积分之于随机过程一样. 这一部分大概率是在本科学习中已经接触过的. 对应到具体编程语言上, 还需要掌握这些数据结构的常用操作. 例如 Python 里面的 list.append(), 遍历字典, dic.get(), 创建二维列表等等, 我觉得这部分其实是 LeetCode 对工作最有帮助的地方.

第二个阶段是跟着题单系统刷题. 我自己用过的题单有代码随想录以及 labuladong, 我感觉都是非常类似的. 这一阶段是在原有的基础上, 对于常见的解题套路建立自己的印象, 例如滑动窗口, Dynamic Programming. 每一个套路对应见过几道题. 但在这个时候由于见过的题目还不够多, 对于这些套路理解还不够深, 很多题还不能融会贯通, 还是得靠记忆模板来解决.

第三个阶段是在熟悉各个套路基础上接着做更多的题, 以求融会贯通. 刷到后来你会发现, 真正常见的思路其实就那么几类, 重要的是看到题目条件能联想到相应解法, 而非死记硬背答案. 这方面令我收益最多的是左程云的视频. Github 主页为https://github.com/algorithmzuo. B 站的视频主页为https://space.bilibili.com/8888480. 前面的题解中我也很多次提到了他的视频, 他的视频最大的优点是他会提及为什么这题可以这么做. 此外, 他的讲解非常细致, 有非常详细的例子和图解, 对于单调栈, 单调队列这类相对小众的题材都有专门的专题. 我觉得唯一的缺点, 就是不太适合时间特别紧的人学习.

但话说回来, 学习本来就很难有捷径, 很多时候还是得靠量变到质变. 如果你有比较充足的时间, 比如刚拿到硕士项目的 offer, 一般在 2 月到 4 月之间, 在正式入学前就可以先花时间准备这些需要长期投入的内容.

最后还有一点补充的是, 准备 leetcode 需要做好笔记加时常复习. 记忆规律告诉我们, 高频率的复习比所谓的一次彻底学懂更有助于形成长期记忆. 哪怕我们完全独立想出了最优解, 过了一个月都很容易忘掉的. 这个时候就需要记笔记了.

我的做法是, 每做完一次题, 可以直接把题目和自己的代码丢给 chatgpt, 让 gpt 用

几句话说清楚这一题的思路, 然后我将这段话作为注释贴在我的答案上面, 然后点提交, 这样这些注释就可以被保存. 这些思路的目的是让你复习的时候扫一眼就能想起这一题的代码该怎么写.

我觉得有两类内容尤其值得记下来: 第一种是类似于解题公式一样的需要频繁使用的固定步骤. 以下有几个例子, 第一个例子是 quick sort 里面的 three way partition, (two way partition 会超时) 这里的 threeway partition 的做法可以当标准流程一样记下来, 所以我写在了注释里面.

Listing 1: Leetcode 912

```
class Solution:
    def threeWayPartition(self, arr, left, right, pivot_val):
    # first(不含) 往左是<x, last(不含) 往右是>x, i是当前的考察的数字
    # 若arr[i] < x, 交换first,i,first++, i++
    # 若arr[i] > x, 交换last,i,last--, i不变
    # 若arr[i] == x, i++

        i = left
        first = left        # 小于 pivot 的区域右边界
        last = right    # 大于 pivot 的区域左边界

        while i <= last:
            if arr[i] < pivot_val:
                arr[first], arr[i] = arr[i], arr[first]
                first += 1
                i += 1
            elif arr[i] > pivot_val:
                arr[last], arr[i] = arr[i], arr[last]
                last -= 1
            else:
                i += 1

        return first, last

    def quickSort(self, nums, low, high):
        if low >= high:
            return

        # Choose random pivot and get its value
        pivot_idx = random.randint(low, high)
        pivot_val = nums[pivot_idx]
```

```
        # Get partition points
        left, right = self.threeWayPartition(nums, low, high, pivot_val
    )

        # Sort left and right portions
        self.quickSort(nums, low, left - 1)
        self.quickSort(nums, right + 1, high)

    def sortArray(self, nums: List[int]) -> List[int]:
        if not nums:
            return nums
        self.quickSort(nums, 0, len(nums) - 1)
        return nums
```

第二个例子是最长递增子序列的 $O(N \ln N)$ 解法, 这里的注释也是相当于口述了一遍这个最优解的流程, 当然 $O(N^2)$ 解法也放在了注释里面, 以供回看时参考.

Listing 2: Leetcode 300

```
class Solution:
    def lengthOfLIS(self, nums: List[int]) -> int:
    '''
    dp = [1 for _ in range(len(nums))]

    for i in range(len(nums)):
    for j in range(i):
    if nums[j]<nums[i]:
    dp[i] = max(dp[i],dp[j]+1)

    return max(dp[:])
    '''
    # 核心思想:
    #   使用一个数组 ends[], ends[i] 表示长度为 i+1 的递增子序列中末尾
    最小的那个数
    #   ends 数组是严格递增的, 用二分查找 ends数组中 >=num最左位置
    #   如果 找到, 就替换
    #   否则就扩展 ends 长度, 表示发现了更长的子序列
```

```
    res = 0

    ends = []
    for num in nums:

        loc = self.bs(ends,num)

        if loc!=-1:
            ends[loc] = num
        else:

            ends.append(num)
            res+=1

    return res

    def bs(self, ends, num):
        #找ends 中 >=num的最左位置
        left = 0
        right = len(ends)-1
        ans = -1
        while(left<=right):
            mid = (left+ right)//2

            if ends[mid]< num:
                left=mid+1
            else:
                ans = mid
                right = mid-1

        return ans
```

第三个例子就是用来检测 linked list 是否有环的 Floyd's Tortoise and Hare Algorithm.

Listing 3: Leetcode 142

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, x):
#         self.val = x
```

```
#         self.next = None
"""
核心思路（Floyd 判圈法）:
步骤:
- 定义两个指针 tortoise 和 hare 都指向链表头
- 慢指针每次走 1 步，快指针每次走 2 步
- 若两者相遇，则说明链表中存在环
- 若快指针到达 None，则说明链表无环,直接返回 None
- 若存在环,将 hare 重置到链表头
- 让 hare 与 tortoise 每次都前进一步
- 当它们再次相遇时，相遇点即为环的起始节点
"""
class Solution:
    def detectCycle(self, head: Optional[ListNode]) -> Optional[
    ListNode]:
        # Initialize tortoise and hare pointers
        tortoise = head
        hare = head

        # Move tortoise one step and hare two steps
        while hare and hare.next:
            tortoise = tortoise.next
            hare = hare.next.next

            # Check if the hare meets the tortoise
            if tortoise == hare:
                break

        # Check if there is no cycle
        if not hare or not hare.next:
            return None

        # Reset either tortoise or hare pointer to the head
        hare = head

        # Move both pointers one step until they meet again
        while tortoise != hare:
            tortoise = tortoise.next
            hare = hare.next
```

```
        # Return the node where the cycle begins
        return tortoise
```

除了这种流程式的模板, 第二种就是最优解中最关键的想法本身, 最常见的是 dp 题里面 dp 数组下标含义, 递推方程式每一个可能性分别代表什么意思. 同样的我们可以举几个例子说明: 第 1 个例子是一道很常见的 dp:leetcode 115, 这里其实光看代码很难看明白 $dp[i][j] = dp[i-1][j-1] + dp[i-1][j]$ 是什么意思, 我们所有的注释其实就是让你能快速会想起这行代码的思路的手段.

Listing 4: Leetcode 115

```
class Solution:
    def numDistinct(self, s: str, t: str) -> int:
    # - 定义 dp[i][j] 表示: s 的前 i 个字符中有多少个子序列等于 t 的前
     j 个字符。
    # - 初始化:
    #   - dp[i][0] = 1, 任何字符串都包含一个空子序列。
    #   - dp[0][j] = 0, 空字符串无法匹配非空目标串。
    # - 状态转移:
    #   - 如果 s[i-1] == t[j-1]:
    #       - 可以使用 s[i-1]: dp[i-1][j-1]
    #       - 也可以不使用它: dp[i-1][j]
    #       => dp[i][j] = dp[i-1][j-1] + dp[i-1][j]
    #   - 如果 s[i-1] != t[j-1]:
    #       - 只能跳过当前字符: dp[i][j] = dp[i-1][j]
    # - 最终答案是 dp[len(s)][len(t)]

        dp = [[0] * (len(t)+1) for _ in range(len(s)+1)]
        for i in range(len(s)):
            dp[i][0] = 1
        for j in range(1, len(t)):
            dp[0][j] = 0
        for i in range(1, len(s)+1):
            for j in range(1, len(t)+1):
                if s[i-1] == t[j-1]:
                    dp[i][j] = dp[i-1][j-1] + dp[i-1][j]
                else:
                    dp[i][j] = dp[i-1][j]
        return dp[-1][-1]

        '''
```

```
        dp = [0] * (len(t)+1)
        dp[0] = 1
        for i in range(1, len(s)+1):
            for j in range(len(t), 0,-1):
                if s[i-1] == t[j-1]:
                    dp[j] = dp[j-1] + dp[j]
                #else:
                #    dp[j] = dp[j]
        return dp[-1]
        '''
```

还有一类题目是非常巧妙地利用了数据结构来达成最优解, 我们的笔记就可以记录这些数据结构是在做什么. 例如 leetcode 295 题.

Listing 5: Leetcode 295

```
class MedianFinder:
    # 用两个堆动态维护数据流的中位数:

    # maxheap（大顶堆）: 存储前半部分较小的数（注意使用负数实现大顶堆）
    # minheap（小顶堆）: 存储后半部分较大的数

    # 保持以下不变量:
    # 1. 两个堆的元素个数差不超过 1;
    # 2. maxheap 中的最大值（即负数最小） minheap 中的最小值;
    # 3. 若总元素个数为奇数, 则多出的那一个堆顶就是中位数
    def __init__(self):
        self.minheap = []
        self.maxheap = []

    def addNum(self, num: int) -> None:
        if not self.maxheap or num<=-self.maxheap[0]:
            heapq.heappush(self.maxheap,-num)
        else:
            heapq.heappush(self.minheap,num)

        self.balance()

    def findMedian(self) -> float:
```

```
        if len(self.minheap) == len(self.maxheap):
            return (self.minheap[0] - self.maxheap[0])/2
        elif len(self.minheap) > len(self.maxheap):
            return self.minheap[0]
        else:
            return -self.maxheap[0]


    def balance(self):
        if abs(len(self.minheap) - len(self.maxheap)) ==2:
            if len(self.minheap) > len(self.maxheap):
                heapq.heappush(self.maxheap,-1* heapq.heappop(self.
    minheap))
            else:
                heapq.heappush(self.minheap,-1* heapq.heappop(self.
    maxheap))

# Your MedianFinder object will be instantiated and called as such:
# obj = MedianFinder()
# obj.addNum(num)
# param_2 = obj.findMedian()
```

Leetcode 这一部分是我学习起来最为困难的, 因为牵涉的知识过于庞杂, 最优解又过于巧妙, 常让人觉得气馁, 感觉自己并不是学习这些知识的料. 如果你也有相似的烦恼, 我想分享一些在如下视频里面所学到的 https://www.bilibili.com/video/BV1L14y1B7ef, 这个还是左程云的视频. 我从中所理解到的就是, 这些最优解的原型是几代人前赴后继, 反复锤炼形成的, 自己想不到是非常正常的事情. 例如 merge sort 这么巧妙的算法是出自冯诺依曼的. 学习算法其实就是在学习这些精妙而美丽的思想传统, 请不要以为短暂的困难而放弃, 当你积累了足够多的思想传统, 你自然就会明白很多问题是怎么想到的.

以上这些例子就是我关于学习 Leetcode 的心得体会.

3.2 C++ 八股文

这一部分的目的, 是希望读者回答出 C++ 的一些通过口头问答考察的题目. 笔者并没有遇到过强制要求我用 C++ 写的 OA 以及面试, 日常工作也不涉及 C++, 故也只能以一个求职者的视角来阐述自己的准备过程.

首先, 一个高效的刷题清单是 **150** Chapter 2.4 C++ Data structure, 这一节由浅入深, 并不会一上来就摆上一堆非常精深的问题难倒读者, 很适合作为一个起点.

如果想要额外提高这一部分知识水平, 这一部分有两个非常著名的资源我十分推荐. 第一个是 Youtube 上 Cherno https://www.youtube.com/@thecherno 的 C++ 视频, 相应的笔记可以在如下这个网站找到 https://www.nagi.fun/Cherno-CPP-Notes/. 第二个是侯捷的视频课程: C++ 面向对象高级编程. 这两个教程的例子选用都非常精妙, 适合在对于基本语法有一定了解后进一步加深理解.

3.3 Python 八股文

面试有的时候会涉及一些 python 专有的结构, 例如 iterator, generator, decorator. Youtube 上 Corey Schafer 的 Python 教程是我见过的最好的资源, https://www.youtube.com/playlist?list=PL-osiE80TeTt2d9bfVyTiXJA-UTHn6WwU, 针对以上概念都有专门的讲解, 比 gpt 生成的质量要高很多. 大家只需在这个列表里面搜索自己想要深究的概念就行了, 每一个概念都会有一个视频对应专门讲解.

3.4 Statistical Inference

这一部分指的是概率论之后第一门统计课中属于经典统计学中的内容, 包括参数估计, 假设检验等等. 例如解释一下 p-value 是什么.

这部分内容更多是复习性质, 一个适合复习的参考资料是 Larry Wasserman 所著 All of Statistics [11]. 这本书的亮点在于对于统计学中的一些基本概念 (confidence interval, p-value) 的解释, 是我目前见过讲得最到位的.

此外关于假设检验, 我在面试中多次被问到诸如" 如果要检验数据是否符合某种性质, 要用什么假设检验? " 这样的问题. 我在这里列出了我被问到过的假设检验的名字, 大家可以查漏补缺.

- **Normality Test**
  - Kolmogorov - Smirnov Test
  - Shapiro - Wilk Test
  - Jarque - Bera Test
- **Unit Root Test**
  - Augmented Dickey - Fuller (ADF) Test
  - Phillips - Perron (PP) Test
  - Kwiatkowski - Phillips - Schmidt - Shin (KPSS) Test
- **Autocorrelation Test**

  - Durbin – Watson Test
  - Ljung – Box Test

- **Cointegration Test**
  - Engle – Granger Test

3.5 Time Series

面试中我遇到的 Time Series 的内容一般包括解释什么是 stationarity, unit root test, ARIMA, 还有 cointegration. 主要都是集中于基本概念.

如果想速成相关概念, 可以参考 David Ruppert, David S. Matteson 所著 Statistics and Data Analysis for Financial Engineering [9] 的相关章节. 此外, Vladas Pipiras 的 Notes 是一份高屋建瓴的材料, 只看基础的几章都大有裨益. 链接为: https://pipiras.web.unc.edu/topics-in-time-series/.

3.6 Linear Regression

在2.3.6中我们补充了两个相关的 Linear regression 的小知识. 现在我们从一个更为整体的角度再来看一下 Linear regression.

Linear regression 可以从 assumption-diagnostic-remedy 的角度入手, 即 linear regression 有不同的假设, 不同的假设各自可以通过一些假设检验或者是画图来进行判断是否成立, 如果不成立就会有一些办法补救. 一个比较快速的 cheat sheet 可以参考如下的这个网站 https://people.duke.edu/~rnau/testing.htm.

此外, 我自己学习 linear regression 中所用的教材 Kutner 等人所著 Applied Linear Statistical Models [5], 是一部 linear regression 的百科全书. 书里对于这些内容的介绍更加详细. 这本书着墨较少的只有 Gauss-Markov Theorem, 以及利用矩阵代数的方法来讲述 linear regression 里面的结果. 这两部分的内容都可以在 Peng Ding 的 notes https://arxiv.org/abs/2401.00649 [2] 找到. 这份 notes 的 1-5, 9-12 章用来准备面试可谓绰绰有余.

3.7 Machine Learning

Machine Learning 的标准教材是 ISLR[3]. ISLR 中对于 Classification 和 Unsupervised Learning 介绍相对更为全面.

对于 Classification, 除去数学推导以外, 一个常见的问题是不同的分类器有什么区别, 例如 LDA, QDA, decision tree 的 decision boundary 有什么区别. 以下的 sklearn 网页涵盖了很多常见分类器的 decision boundaries.

https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html#sphx-glr-auto-examples-classification-plot-classifier-comparison-py

希望可以帮助大家更直观地理解各个分类器的区别.

对于 Unsupervised Learning. 我遇到的问题大多数是完整讲述一个算法的流程, 包括 PCA, K-means, Hierarchical clustering.

除了 ISLR, 在这里我也想推荐两本我特别喜欢的 Machine Learning 的参考书. 一本是李航所著《机器学习方法》(原《统计学习方法》)[13], 在 2025 年作者又出了新版. 这本书的优点在于其叙述和 notation 非常清晰易懂, 对于很多算法作者还提供了一些数值例子来帮助理解. 还有一本是 Kevin Murphy 所著 Probabilistic Machine Learning: An Introduction [7]. 这本书可以在作者主页找到 http://probml.github.io/book1. 这本书包罗万象, 即包含了对很多传统模型的深入介绍, 又介绍了很多最近发表的新模型, 非常适合用来进阶.

最后补充一点, xgboost 这一块我本人也确实在面试中被问到过. 如果想快速了解 xgboost, 观其大略的话, 可以看 StatQuest 的视频 https://www.youtube.com/watch?v=OtD8wVaFm6E, 非常清晰易懂.

4 补充题目汇总

这一节将正文中出现的补充题目按来源汇总, 用作索引.

4.1 150

- Chapter 1: Question 1, 7, 12
- Chapter 2.3
- Chapter 2.4
- Chapter 2.5: Question 5, 6, 8, 10
- Chapter 2.6: Question 3, 5
- Chapter 2.7: Question 7, 13, 16, 18, 19, 20
- Chapter 2.8: Question 8, 11

4.2 Joshi

- Chapter 2: Question 2.4, 2.6, 2.14, 2.16, 2.20, 2.21, 2.29, 2.30, 2.47, 2.55, 2.56, 2.58, 2.59
- Chapter 4: Question 4.4, 4.6
- Chapter 8: Question 8.6, 8.24

4.3 DanProb

- Chapter 1: Question 11, 13, 14, 15, 22, 33

4.4 Heard on the Street

- Question 1.6, 1.21
- Question 2.10, 2.11, 2.12
- Question 3.1
- Question 4.25, 4.35, 4.58, 4.65

4.5 Brainteaser

- 1.1 'Aha' Questions – Logical Conundrums – Question 4, 16, 17

4.6 Quantable

- https://www.quantable.io/questions/3712
- https://www.quantable.io/questions/2517
- https://www.quantable.io/questions/2746
- https://www.quantable.io/questions/2726
- https://www.quantable.io/questions/3016
- https://www.quantable.io/questions/2745
- https://www.quantable.io/questions/2454
- https://www.quantable.io/questions/2555
- https://www.quantable.io/questions/3669
- https://www.quantable.io/questions/3693
- https://www.quantable.io/questions/2545
- https://www.quantable.io/questions/3075
- https://www.quantable.io/questions/3076
- https://www.quantable.io/questions/3077
- https://www.quantable.io/questions/2882
- https://www.quantable.io/questions/2563
- https://www.quantable.io/questions/3375
- https://www.quantable.io/questions/2527
- https://www.quantable.io/questions/2528
- https://www.quantable.io/questions/3903
- https://www.quantable.io/questions/3781
- https://www.quantable.io/questions/3297
- https://www.quantable.io/questions/3350
- https://www.quantable.io/questions/2493

- https://www.quantable.io/questions/2618
- https://www.quantable.io/questions/2577
- https://www.quantable.io/questions/2619
- https://www.quantable.io/questions/2480
- https://www.quantable.io/questions/2482
- https://www.quantable.io/questions/2610
- https://www.quantable.io/questions/2612
- https://www.quantable.io/questions/2613
- https://www.quantable.io/questions/2881
- https://www.quantable.io/questions/3470
- https://www.quantable.io/questions/2504
- https://www.quantable.io/questions/2684
- https://www.quantable.io/questions/3850

4.7 其它

- https://xinweizhang.github.io/Duplicating-data-in-linear-regression
- https://quantinterview.github.io/Random-Point-Circle

参考文献

[1] Timothy Falcon Crack. *Heard on the street: Quantitative questions from wall street job interviews*. Timothy Crack, 26th edition, 2025.

[2] Peng Ding. *Linear Model and Extensions*. 2025.

[3] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: with Applications in R*. Springer US, second edition, 2021.

[4] Mark Joshi, Nicholas Denson, and Andrew Downes. *Quant Job Interview Questions and Answers*. Pilot Whale Press, second edition, 2013.

[5] Michael H. Kutner, Christopher J. Nachtsheim, John Neter, and William Li. *Applied Linear Statistical Models*. McGraw-Hill Education, 5th edition, 2005.

[6] Ivan Matić, Dan Stefanica, and Radiša Radojičić. *Probability and stochastic calculus quant interview questions*. FE Press, 2021.

[7] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022.

[8] Radoičić Radoš, Matić Ivan, and Dan Stefanica. *Challenging Brainteasers for Interviews*. FE Press, LLC, 2023.

[9] D. Ruppert and D.S. Matteson. *Statistics and Data Analysis for Financial Engineering: with R examples*. Springer New York, second edition, 2015.

[10] Dan Stefanica, Radoičić Radoš, and Tai-Ho Wang. *150 Most Frequently Asked Questions on Quant Interviews*. FE Press, LLC, third edition, 2024.

[11] L. Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer, 2004.

[12] Xinfeng Zhou. *A Practical Guide To Quantitative Finance Interviews*. Xinfeng Zhou, second edition, 2020.

[13] 李航. *机器学习方法 (第 2 版)*. 清华大学出版社, 2025.