同济大学

实习日记本

学　　院	计算机科学与技术学院
专　　业	计算机科学与技术专业
学　　号	2251745
学生姓名	张宇
接受单位	百度
地　　址	北京市海淀区上地十街百度大厦
指导教师姓名	李文根

大学生校外实习守则

一、高校校外实习是高等教育同社会主义建设相结合的重要环节，是培养合格人才的重要保证，每个学生必须按照教学计划的要求，认真参加各种形式的校外实习活动，并自觉地遵守本守则。

二、严格遵守实习单位的有关规章制度。如有违反，学校应会同实习单位按有关规定严肃处理。

三、自觉遵守劳动纪律、尊重实习带教人员，服从安排，严格执行安全操作规程。

四、妥善保管因实习需要而借阅的图纸、资料，严格遵守保密制度，放置泄密。

五、认真写好实习日记，客观记录实习的内容、问题及心得。

六、独立完成实习报告，系统总结、综合分析实习中的问题以及解决问题的方法和建议。

七、实习结束时，必须如数归还所借用的劳防用品、工具、仪器、资料及生活用品，如有损坏或遗失应按规定赔偿。

八、学生在实习期间造成伤残和死亡的，可按“大学生团体平安保险”有关规定给予赔偿。

实习起迄日期
自 2024 年 10 月 1 日起至 2025 年 7 月 1 日止
实 习 内 容
我的实习内容主要围绕百度松果菁英班的高水平编程训练项目展开。该项目的核心目标是帮助学员提升编程能力，并培养解决复杂问题的能力，尤其是在人工智能、数据科学等前沿技术领域的应用。项目采用理论与实践相结合的方式，通过每周的编程比赛逐步提升学员的算法水平和解题能力。 实习过程中的课程形式以每周一次的在线编程比赛为核心。比赛题目涵盖了从基础的算法应用到较为复杂的数学建模与图算法等多个方面，随着比赛的进行，题目的难度逐步递增。你在比赛中需要独立解决各种问题，并在规定时间内提交解决方案。这些题目不仅考验学员的算法和数据结构的应用，还要求学员具备较强的代码优化能力，特别是在处理复杂计算和优化问题时。 比赛安排上，课程的总体时长为一学年，每周的比赛时间固定，通常持续 3 小时。比赛后，平台会自动评估并给出详细的反馈，学员可以根据评测报告中的时间消耗、空间消耗等指标来分析和改进自己的解题过程。你通过这些反馈不断调整思路，优化代码，以应对更高难度的挑战。每周比赛的内容逐步过渡，从基础的排序和查找，到涉及更复杂的动态规划、图论等内容，确保学员在比赛中不断突破自我。 除了比赛本身，课程还安排了相关的技术讲座和学习材料，帮助学员深入了解一些高阶算法和编程技巧。这些讲座与比赛内容紧密结合，帮助你在实际编程过程中解决难题，提高自己的解题能力和技术水平。你在实习过程中，逐步提升了对复杂算法的理解和应用，尤其在图论、动态规划、贪心算法等方面，学会了如何快速分析问题并选择合适的算法进行解决。
指导教师批阅意见：

指导教师签名　　月　日

2024 年 10 月 13 日

实习日志 1 - 编程训练与算法挑战

今天是我在百度松果菁英班的第一天，参加了一个在线编程比赛，题目主要涉及排序和查找算法。尽管这些问题看起来比较基础，但在实现过程中，我遇到了一些挑战，特别是在处理不同数据类型和边界条件时。首先，题目要求我们使用快速排序和二分查找来完成任务，但当数据规模较大时，算法的执行效率成为了一个问题。

在处理输入时，我发现边界条件的判断非常重要。例如，输入数据的最小值和最大值需要特别注意，避免程序出现数组越界或非法操作。经过多次调试和修改，我终于解决了这些问题，确保代码能够正确处理不同类型的输入，并且没有出现任何错误。通过这次调试，我深刻体会到，编写高质量的代码不仅要确保正确性，还要考虑到程序的健壮性。

此外，我也意识到，在面对大数据时，仅仅依靠正确的算法是不够的。性能优化变得尤为重要，尤其是时间复杂度和空间复杂度的控制。在比赛中，我注意到，尽管我使用了经典的排序和查找算法，但由于没有对数据规模进行充分优化，程序的运行时间仍然偏长。通过系统提供的反馈，我了解到自己的代码可以进一步优化，特别是在执行效率上，我还需要更多关注内存和时间的消耗。

通过今天的比赛，我不仅提高了对排序与查找算法的理解，也加强了对算法优化的认识。接下来，我会更加注重在编写代码时，如何选择合适的算法来处理大规模数据，提高程序的执行效率，减少不必要的计算。我相信通过不断练习和总结，我能在之后的比赛中更好地应用这些技巧，提升自己的编程能力。

2024 年 10 月 27 日

实习日志 2 - 动态规划与优化

在今天的比赛中，我挑战了一个动态规划的题目，题目要求解决一个变种的背包问题。起初，面对复杂的题目，我有些迷茫，因为题目中涉及的背包容量和物品数量非常庞大，数据规模巨大，这让我对如何设计高效的解法产生了疑问。然而，通过反复思考问题的结构，我逐渐找到了合适的解题思路。首先，我通过将问题转化为子问题，并利用动态规划的思想，成功地设计了状态转移方程。

动态规划是一种有效的求解最优化问题的技术，它通过将大问题拆解为小问题，逐步求解得到最终的解。在本题中，我通过建立二维数组来存储每个子问题的解，从而有效解决了背包问题。然而，在实现过程中，我发现二维数组的空间消耗非常大，尤其是在处理物品数量和背包容量较大的情况下，内存使用十分紧张，程序的效率也受到了影响。

面对这个问题，我思考了一下优化的方法，最终决定采用滚动数组的技巧来减少空间复杂度。滚动数组是一种常用的优化方法，它通过只保留当前状态和上一状态，避免了不必要的空间占用。在应用了滚动数组后，空间复杂度从二维数组的 O(n * m)优化为一维数组的 O(m)，显著降低了内存使用，并且提高了程序的运行效率。

通过这次比赛，我不仅解决了复杂的背包问题，还深刻理解了动态规划的精髓，尤其是在处理大规模数据时，如何优化空间复杂度以避免不必要的内存占用。这次经验让我更加清晰地认识到，在编写高效代码时，优化空间和时间的消耗同样重要。

2024 年 11 月 10 日

实习日志 3 - 图论与最短路径

今天的比赛中，我处理了一个图的最短路径问题，题目要求我们在一个无向图中，求出从某个节点到其他所有节点的最短路径。刚开始，我尝试使用广度优先搜索（BFS）来解决这个问题。BFS 在图的遍历上很有效，能够保证最短路径的正确性，特别是在图的边权相等时。然而，由于图的规模非常庞大，节点数和边数都很多，直接使用 BFS 时，其时间复杂度为 O(n^2)，在处理大规模图时，程序运行超时，无法通过测试用例。

面对这个问题，我停下来思考了一下，考虑到图中可能有不同的边权，我意识到使用 Dijkstra 算法可能更为合适。Dijkstra 算法能够高效地解决带权图的最短路径问题，特别适用于图中边权不相等的情况。我决定实现 Dijkstra 算法，并使用优先队列来优化算法，优先队列可以帮助我高效地选择当前最短的节点，从而减少不必要的计算，避免了每次都扫描整个图的操作。

在应用了 Dijkstra 算法后，我的程序成功通过了测试，并且大大提高了运行效率。通过使用优先队列，我将算法的时间复杂度从 O(n^2)降低到了 O(n log n)，使得程序能够在规定时间内完成计算。这次比赛让我更加熟悉了图算法的应用，特别是在处理大规模图时，如何选择合适的算法进行优化。通过这次经验，我也更加理解了算法优化的重要性，尤其是在面对复杂的数据结构时，如何选择适合的工具来提高程序的效率。

2024 年 11 月 24 日

实习日志 4 - 回溯算法与递归

今天，我挑战了一个回溯算法的问题，题目要求求解一个排列问题。这个问题的核心是递归和状态回溯，要求生成给定集合的所有排列组合。刚开始，我直接采用了递归的方式进行解题，但由于没有考虑如何优化递归过程，导致程序运行时间过长，尤其是在输入规模较大的时候，程序无法在规定时间内完成计算。

在继续思考问题时，我意识到回溯算法的一个重要技巧就是剪枝，也就是在递归过程中，尽早排除那些无效的搜索路径，从而减少不必要的递归调用。通过对问题的搜索树进行分析，我发现一些不必要的排列可以直接排除，例如，某些已经访问过的元素不应该再次被选择，这样的重复计算显著影响了程序效率。

为了实现剪枝，我加入了一个哈希集合来记录已经选择过的元素，这样在递归调用时，我可以直接判断当前元素是否已经在路径中，避免重复计算。通过这种方式，我大大减少了递归的深度，并且减少了冗余的计算，显著提高了程序的执行效率。

通过这次比赛，我深刻理解了回溯算法的精髓，尤其是在搜索过程中如何通过合理的剪枝策略提升效率。回溯算法虽然看起来比较直观，但在实际应用中，尤其是在处理大数据量时，需要精心设计和优化。只有通过合理的剪枝和状态管理，才能在复杂的组合问题中保持高效，避免过多的冗余计算。这次比赛不仅让我掌握了回溯算法的基本思路，也让我意识到在算法设计中，效率和优化是同样重要的部分。

2024 年 12 月 8 日

实习日志 5 - 贪心算法的应用

今天我解决了一个关于贪心算法的问题，题目要求我们在给定的多个区间中，选择最大数量的不重叠区间。这个问题表面看起来简单，但实际上在实现过程中需要精确地处理区间的排序和选择，特别是如何合理地安排选择顺序，以保证能够选择到最大数量的不重叠区间。

在解决这个问题时，我首先仔细分析了题目的要求，并考虑到贪心算法的特性——即每次选择当前最优的解。我发现，解决这一类问题的关键在于如何选择下一个区间。经过分析，我认为最优的策略是选择结束时间最早的区间，因为它能留出更多的空间给后续的区间，从而增加选择的数量。因此，我将所有区间按照结束时间进行升序排序，并且从排序后的第一个区间开始，逐一选择不与当前区间重叠的区间。

实现过程中，我首先将所有的区间按结束时间排序，然后用一个变量记录当前已选择区间的结束时间。接着，从第一个区间开始，若当前区间的起始时间大于上一个选择区间的结束时间，就将其选中，并更新已选择区间的结束时间。这种方法通过每次选择最早结束的区间，确保在有限的时间内选择到最多的不重叠区间。

最终，我通过贪心算法在规定的时间内顺利完成了题目，且代码实现简洁高效。这次比赛让我更加深入地理解了贪心算法的应用，特别是在解决区间类问题时，如何通过贪心选择确保最优解。通过这次经验，我进一步认识到，贪心算法虽然简单直观，但在应用时需要细心地分析问题的结构，才能确保选择最优解，解决问题时也要关注算法的实现效率。

2024 年 12 月 22 日

实习日志 6 - 代码优化与调试

今天的比赛中，我遇到了一个需要优化代码执行效率的问题。题目要求处理大量数据计算，初始的实现方法是采用暴力解法，直接遍历每个数据点进行计算。这种方法虽然简单直观，但由于没有考虑到时间复杂度的问题，当数据量较大时，程序的执行时间大幅度增加，最终导致超时，无法通过测试用例。

面对这个问题，我开始思考如何优化算法以提高性能。经过分析，我发现数据计算中大量的查找操作是性能瓶颈的根源。在这种情况下，暴力查找每个元素的时间复杂度为 O(n)，导致了程序的执行速度过慢。为了解决这个问题，我决定使用哈希表来替代原本的线性查找。哈希表能够以常数时间 O(1)进行查找，从而大大提高了效率。

在调整了算法并引入哈希表后，我再次运行程序，发现程序的执行速度得到了显著提升，能够在规定时间内顺利通过评测。通过这种优化，时间复杂度从 O(n^2)降低到了 O(n)，极大地提升了程序的运行效率。

比赛结束后，我进行了代码的复盘，总结了优化思路。优化不仅仅是解题的一个步骤，更是提高代码质量和程序执行效率的重要环节。通过这次比赛，我更加理解了合理选择数据结构和算法的重要性，如何根据问题的特点来选择最合适的方案。优化代码不单单是让程序能通过测试，更是要关注如何在大数据量下保持高效。我也意识到，作为一名开发者，编写高效的代码是技术水平的重要体现，性能的提升是每个算法优化过程中的核心目标。

2025 年 3 月 9 日

实习日志 7 - 数学建模与算法设计

今天的比赛让我深刻体会到了数学建模在解决实际问题中的重要性。题目要求我们用数学模型描述某一场景，并通过算法计算出最优解。刚开始，我对如何将实际问题转化为数学模型并没有太多思路，面对问题时，我感到有些迷茫。然而，经过一番思考，我意识到将问题转化为图论问题是一个有效的解决方案。于是，我决定从图论的角度来分析问题，通过最短路径算法来求解。

问题的核心是要在一个复杂的场景中找到最优的路径，最短路径算法恰好能够帮助我在多个可能的路径中找到一条最优解。于是，我将场景中的各个元素抽象为图中的节点和边，并根据题目的要求赋予边不同的权重。通过这种方式，我能够将原本复杂的问题简单化，最终选择了 Dijkstra 算法来计算最短路径。

在算法的实现过程中，我还注意到输入数据的规模较大，因此我在实现时特别关注了算法的时间复杂度，确保在合理的时间内能够计算出最优解。经过不断调试和优化，最终我顺利解决了问题，并得到了正确的结果。

比赛结束后，我回顾了整个过程，意识到数学建模是解题的第一步。只有通过合理的建模，才能为后续的算法设计提供明确的方向，避免在解决过程中走弯路。这次比赛不仅提升了我的建模能力，也让我更加深刻地理解了数学和算法的结合。通过建模，我能更清晰地理解问题的本质，从而设计出更加高效的算法，解决复杂的问题。

2025 年 3 月 23 日

实习日志 8 - 递归与记忆化搜索

今天的比赛中，我再次遇到了递归问题，题目要求使用递归方法解决斐波那契数列问题。虽然这个问题看起来非常简单，是经典的递归问题之一，但在实际计算过程中，我发现存在大量的重复计算，导致时间复杂度显著增加，程序运行速度较慢，特别是当输入数值较大时，递归方法的执行时间变得不可接受。

在没有优化的情况下，递归会不断地计算已经计算过的子问题，造成了大量不必要的重复计算。比如，当我们计算第 n 个斐波那契数时，函数会多次计算第 n-1 个、第 n-2 个数，而这些值在之后的递归调用中还会再次被计算。这使得递归的时间复杂度从 O(2^n)急剧升高，无法在规定的时间内完成计算。

为了解决这个问题，我决定使用记忆化搜索（Memoization）技术来优化递归过程。记忆化搜索是一种通过缓存已经计算过的结果来避免重复计算的技术，能够显著减少冗余计算。我通过引入一个哈希表（字典）来存储已经计算的斐波那契数值，在递归调用时，首先检查该值是否已经计算过，如果计算过就直接返回缓存中的值，否则进行递归计算并将结果存入缓存。

经过优化后，程序的运行速度得到了显著提升，时间复杂度从 O(2^n)降低到 O(n)，使得程序能够高效地处理较大的输入值，并顺利通过了评测。这次比赛让我更加熟悉了递归的优化技巧，特别是在避免重复计算方面，记忆化搜索的优势非常明显。通过这次经验，我深刻理解了如何在递归算法中合理利用缓存技术，提升程序性能，减少不必要的计算。

2025 年 4 月 6 日

实习日志 9 - 压缩算法与空间优化

今天，我挑战了一个关于数据压缩的题目，要求在空间有限的情况下对数据进行压缩并能够顺利解压。题目中的数据量非常庞大，初期我采用了常规的数组存储方式来存储压缩数据。然而，随着数据规模的增大，内存的消耗迅速增加，这导致程序的内存使用过高，甚至出现了内存溢出的问题。因此，我必须寻找一种更为高效的压缩方法来解决这个问题。

经过思考和研究，我决定尝试使用哈夫曼编码算法来对数据进行压缩。哈夫曼编码是一种常用的无损数据压缩算法，通过根据字符出现的频率构建一棵哈夫曼树，使用较短的编码表示频率较高的字符，较长的编码表示频率较低的字符，从而实现数据的压缩。通过哈夫曼编码的优化，我能够有效地减少数据存储所需的空间，同时保证数据的完整性和可解压性。

在应用哈夫曼编码后，我不仅减少了数据的空间占用，还提高了数据处理的效率。通过构建哈夫曼树并为每个字符分配相应的二进制编码，我成功压缩了原始数据，并能够在不损失数据质量的前提下，将数据存储需求大幅度降低。解压时，利用哈夫曼树的结构，可以高效地恢复原始数据。

通过这次比赛，我深刻认识到，在面对实际问题时，如何进行合理的空间优化是一个非常重要的技能。哈夫曼编码不仅帮助我解决了内存消耗过大的问题，还让我理解了数据压缩算法在优化空间和提高效率方面的重要性。未来，我将在更多的项目中应用此类优化技术，以提升程序的性能和内存管理能力。

2025 年 4 月 20 日

实习日志 10 - 时间复杂度与大数据处理

今天的比赛让我深刻体会到时间复杂度在处理大数据时的重要性。题目要求我们处理一个包含百万级数据的排序问题，起初，我尝试使用暴力解法来解决，直接通过两层嵌套循环进行排序。然而，由于数据量庞大，暴力解法的时间复杂度为 O(n^2)，在面对大规模数据时，程序的运行时间非常长，无法在规定的时间内完成计算，甚至超时了。

在遇到性能瓶颈后，我决定重新审视问题，寻找更高效的解决方案。通过对题目的分析，我意识到对于大数据的排序，归并排序是一种更合适的选择。归并排序的时间复杂度为 O(n log n)，比暴力解法显著优化了性能，尤其适合处理大规模数据。在理解了归并排序的原理后，我开始实现这一算法，并对代码进行了细节优化，确保程序能够高效运行。

归并排序采用了分治策略，它将大数组分解为较小的子数组，然后递归地对每个子数组进行排序，最后将这些已排序的子数组合并成一个有序的数组。通过不断地分割和合并，归并排序能够有效地减少计算量，并确保时间复杂度保持在 O(n log n)级别。

经过改进后，程序顺利通过了评测，并且在处理百万级数据时，运行效率非常高。通过这次比赛，我更加深入理解了时间复杂度的概念，并且学会了如何根据数据的规模和问题的特性选择合适的算法，以保证程序的运行效率。这次经验让我意识到，在实际编程中，算法的选择和优化对于处理大数据至关重要，合理的算法设计不仅能提高效率，还能有效避免时间超限的问题。