Curso使用介绍：以解TSP问题为例

工具：

- cursor（pro有14天试用期）
- Python 3.9以上，推荐3.11

教学目标：

- 理解Chat 模式的作用
- 理解如何用LLM解决复杂的编程问题；并理解【测试】在AI辅助代码生成中的重要作用
- 理解如何用LLM进行复杂问题的探究和代码学习

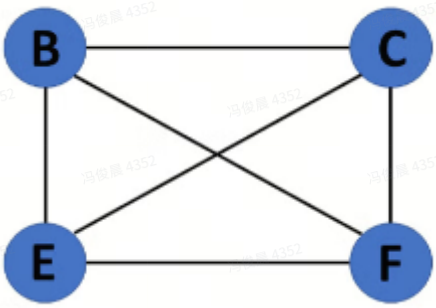
背景介绍

图的基本知识

图（graph）数据结构有两个实体，顶点（vertex）和边（edge）。边可以分为有向边（连接顶点有顺序）和无向边（连接顶点无顺序）。

根据边的密度，一个图结构可以计算“连通度”。

我们在这个课程里研究的是全连通图（fully connected graph），即任何两个顶点之间都有边，且为无向边。



旅行商问题（Traveling Salesman Problem）

问题描述：

想象一下，你是一位需要拜访多个客户的销售人员。你的任务是：

- **从出发点出发**：比如你的公司所在的城市。

- **访问每一个客户所在的城市一次且仅一次**：不能遗漏任何一个客户，也不能重复拜访。
- **最终回到出发点**：完成所有拜访后，回到公司。

你的目标是**规划一条总路程最短的路线**。换句话说，你想节省时间和旅行费用。

为什么这个问题重要？

- **实用性强**：不仅适用于销售路线规划，还广泛应用于物流配送、印刷电路板布线、机械手路径规划等领域。
- **优化资源**：解决这个问题可以最大化地利用时间和资源，降低成本，提高效率。

问题的重点和难点：

1. **组合爆炸（计算复杂度高）**：
   - 随着城市数量的增加，可能的路线组合数量会以阶乘速度增长。
   - **举例**：如果有 5 个城市，可能的不同路线有 (5－1)!=24种；如果有 10 个城市，可能的路线就增加到 (10－1)!=362,880 种。
   - **影响**：对于城市数量较少的情况，我们可以通过枚举所有可能的路线来找到最短的。但当城市数量增加，这种方法在计算上变得不可行。
2. **缺乏高效的精确算法**：
   - 旅行商问题被认为是一个 **NP 完全问题**。这意味着目前没有已知的算法可以在多项式时间内找到大规模问题的精确解。
   - **挑战**：如何在合理的计算时间内找到最优或接近最优的解决方案？
3. **需要权衡精度和计算时间**：
   - **精确算法**：比如动态规划算法，可以找到最优解，但计算时间随着城市数量增加迅速变长。
   - **近似算法和启发式算法**：如贪心算法、遗传算法，可以在较短时间内找到“足够好”的解，但不一定是最优解。

总结：

旅行商问题看似简单，即找到一条最短的路线访问所有城市并回到起点，但其背后隐藏着复杂的计算挑战。理解和解决这个问题对于优化现实世界中的诸多应用具有重要意义。对于计算机背景的你们，这也是深入了解算法设计、优化和计算复杂性的绝佳案例。

VSCode配置Tips

安装必要的扩展

打开左侧工具栏

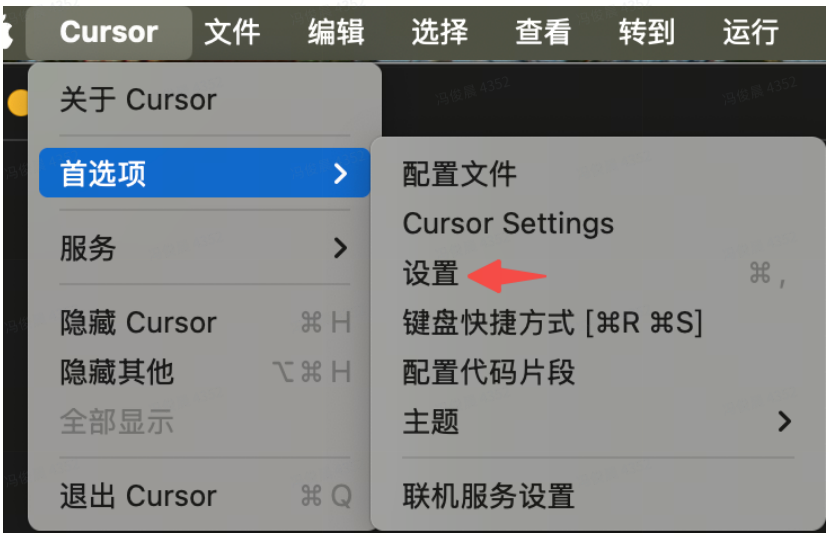
- 安装python和python debuger这两个扩展。这两个扩展均来自ms-python
- 安装Simplied Chinese

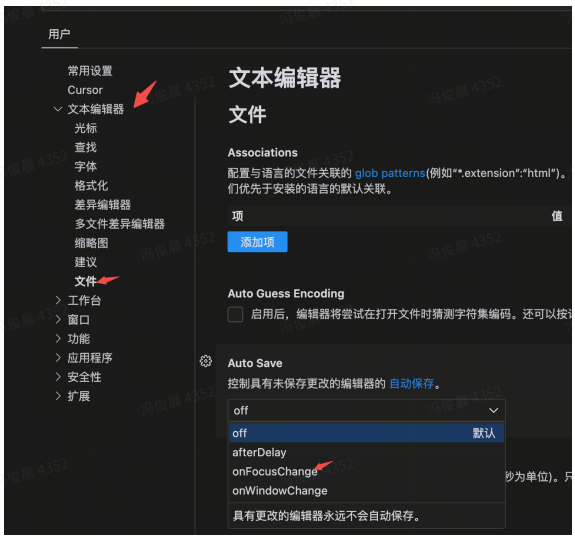
- 如果是Mac用户，安装makefile-tools，这个扩展来自ms-vscode

设置自动保存

Cursor默认是没有自动保存的，这可能导致调试的bug。

请按照下面的方法设置自动保存。记得最后点击把“设置”关掉才能保存配置





Cursor使用Tips

搬运一个边建宇老师的tutorial：AI编程利器-Cursor

CMD+K

Cursor最常用的命令，在inline环境下生成代码，可以接受简单的指令和反馈。

CMD+L

当问题比较复杂、代码自身的上下文不够、需要额外的文件作为引用时，可能inline编辑就不够了，需要使用进阶的聊天模式。

还有一种更加重要的应用，就是在学习新知识和进行设计讨论时，你会非常需要聊天模式。AI辅助编程不仅仅是完成代码这么简单。

Debbug With AI

运行失败时，将命令行终端结果直接放到context运行。如果你是windows系统，可能没有debug with AI，可以使用add to chat，把报错信息输出到chat context，然后提问。

@Context

在chat 模式下，"@Codebase"可以实现代码库的RAG问答。这个对于熟悉一个新的代码仓库非常有帮助

context还可以提供git commit（不确定是否可以援引git issue），web信息（低配版perplexity）和具体的代码/类

图片

对于前端而言，UI原型图是一个必备的背景信息。也可以通过image上传。

更多内容参考Cursor的官方介绍

https://www.cursor.com/features
www.cursor.com

Tab

当你在同一个上下文里工作时间足够长，Cursor会自动预测你的下一步工作，然后提供整段代码的edit，你可以通过tab->cmd+Y来快速完成类似工作。例如，将下面练习的所有函数注释逐个变成英文。

Usage Based Pricing

当你用默认Claude sonet或者4o进入debug死循环后，你可能需要o1-preview的帮助。但是要使用o1-preview必须打开usage based pricing

基础任务

代码在https://git.bg.huohua.cn/fengjunchen/code_with_ai

任务是补全graph.py中的GraphAdvanced类的四个方法

解决最短路径问题

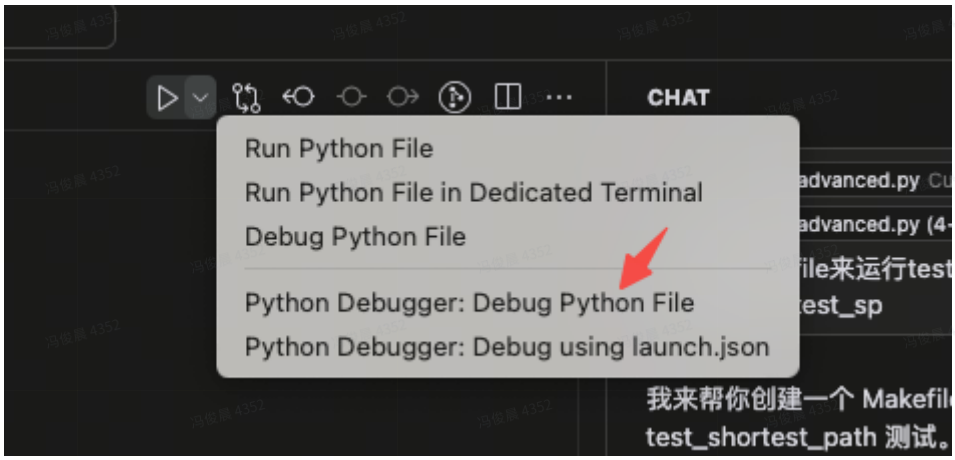
请使用Cursor的cmd+K（Edit）功能完成最短路径算法。

- prompt：“写/完成任务/go”

💡 如果报错了，使用debug with AI进行自动修正。

💡 如果debug with AI依然失败了。

1. 检查你的文件是否保存了（文件名右侧是不是小圆点）
2. 需要进行人工step调试，请在将窗口焦点移到test文件下，然后按照下图点击运行->Python Debugger:Debug Python File。前置条件是安装了Python Debugger插件



🏆 Mac：make test_sp

Windows：python3 -m unittest test_graph_advanced.TestGraphAdvanced.test_shortest_path

🎖️ 请使用CMD+L，让cursor向你解释什么是dijkstra算法。并以当前算法为例，解释时间复杂度的计算方法。

解决小规模图的TSP问题

在命令行中运行

> make test_tsp_small

然后尝试用debug with AI的功能解决当前代码的问题

🏆 Mac：make test_tsp_small

Windows：python3 -m unittest test_graph_advanced.TestGraphAdvanced.test_tsp_small

🎖️ 请使用CMD+L，让cursor向你解释什么是Held-Karp算法，以及这里面的bit operator（>>）在干什么

解决大规模图的TSP问题

在Chat模式下询问大模型技术选型怎么做，并合作完成一个解法。

🏆 Mac：make test_tsp_large

Windows： python3 -m unittest test_graph_advanced.TestGraphAdvanced.test_tsp_medium

🥇 不准用贪心算法，怎么实现？

如果有10w个节点，当前硬件性能有什么限制？怎么突破？

进阶任务

兼顾效率和效果的TSP解法

在Chat模式下询问大模型技术选型怎么做，并合作完成一个复杂的近似算法。

🏆 Mac：make test_tsp_medium

Windows：python3 -m unittest test_graph_advanced.TestGraphAdvanced.test_tsp_large

🥇 比赛一下，谁的方案用时少，路径短

课外阅读：Coursera，Generative AI for Software Development

https://www.coursera.org/learn/introduction-to-generative-ai-for-software-···

Coursera | Online Courses & Credentials From Top Educators. Join for Free | Coursera

Learn online and earn valuable credentials from top universities like Yale,···