校外实训课程材料

1. AI辅助编程经历
2. 问题描述

随着编程水平的进阶，小华知道是全面扩充小鲁知识面，帮助他快速升级的时候到了，他再给小鲁布置了道经典题：

输入一个正整数N，请写一个程序，输出N皇后问题的全部摆法。

（选自xmuoj平台2024年校外实训（WEEK1）B05）

1. AI工具描述

Baidu Comate，中文名为“文心快码”，是由百度开发的AI编程助手，它基于文心大模型，结合百度积累多年的编码现场大数据和外部优秀开源数据为开发者提供智能code生成和辅助功能。

作为个人使用其的主要功能有：

* 智能推荐：它能通过code上下行提供code推荐，便利写code的过程。
* 智能生成：它通过注释生成相应功能的code，并对已完成的函数具有函数注释、行间注释、生成单测、code解释、函数拆分、调优建议等功能。
* 智能问答：它提供多轮对话智能问答功能，解决在写code过程中遇到的问题。

1. 提问过程

会在这个题卡住的原因在于通过问题描述根本不理解题目的意思，如果不是在先前就了解过N皇后的问题的话。所以第一个问题：N皇后问题题目解释。

在解答之后整个问题就十分明了了。所以之后问了算法相关问题：

对使用的算法进行讲解，使用类似算法记得经典例题。

1. 反馈结果及验证

#include <iostream>

#include <vector>

using namespace std;

bool isSafe(vector<string>& board, int row, int col, int n) {

// 检查列上是否有皇后互相冲突

for (int i = 0; i < row; ++i) {

if (board[i][col] == 'Q')

return false;

}

// 检查左上对角线是否有皇后互相冲突

for (int i = row - 1, j = col - 1; i >= 0 && j >= 0; --i, --j) {

if (board[i][j] == 'Q')

return false;

}

// 检查右上对角线是否有皇后互相冲突

for (int i = row - 1, j = col + 1; i >= 0 && j < n; --i, ++j) {

if (board[i][j] == 'Q')

return false;

}

return true;

}

bool solveNQUtil(vector<string>& board, int row, int n) {

// 如果所有行都成功放置了皇后

if (row == n) {

// 输出解

for (int i = 0; i < n; ++i)

cout << board[i] << endl;

cout << endl;

return true;

}

// 考虑当前行的每一列

for (int col = 0; col < n; ++col) {

// 检查当前位置是否安全

if (isSafe(board, row, col, n)) {

// 放置皇后

board[row][col] = 'Q';

// 递归放置下一行的皇后

if (solveNQUtil(board, row + 1, n))

return true;

// 如果下一行放置失败，则回溯，移除当前皇后

board[row][col] = '.';

}

}

// 如果没有找到解，返回false

return false;

}

bool solveNQ(int n) {

// 创建一个N×N的棋盘，初始化为'.'

vector<string> board(n, string(n, '.'));

// 调用递归函数放置皇后

return solveNQUtil(board, 0, n);

}

int main() {

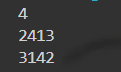
int n = 4; // 示例：求解4皇后问题

solveNQ(n);

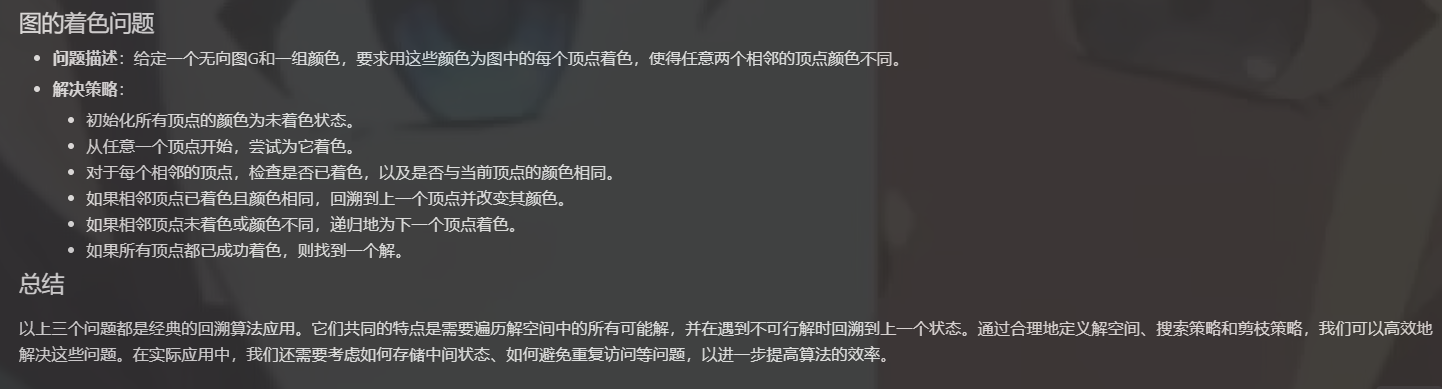
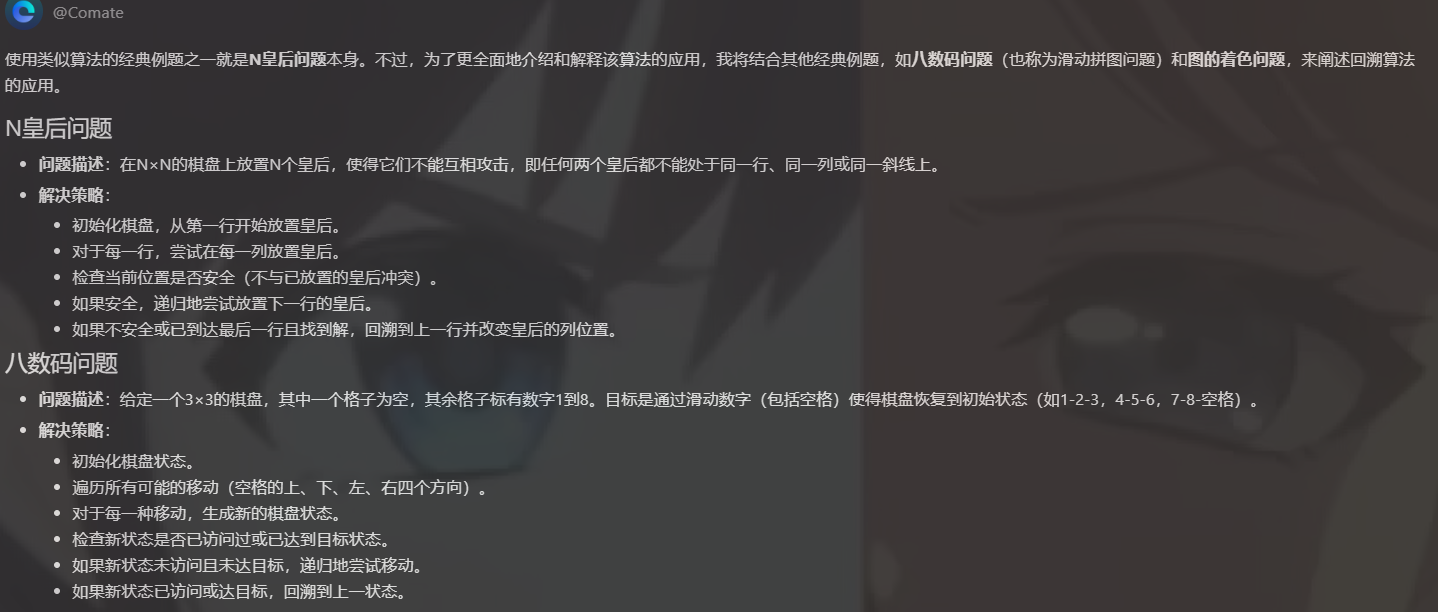
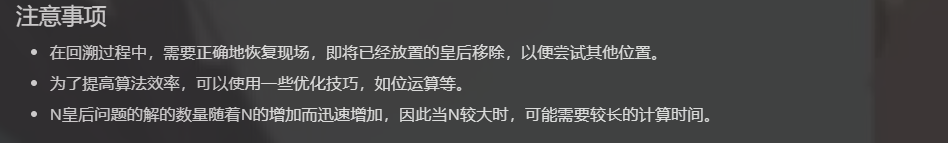
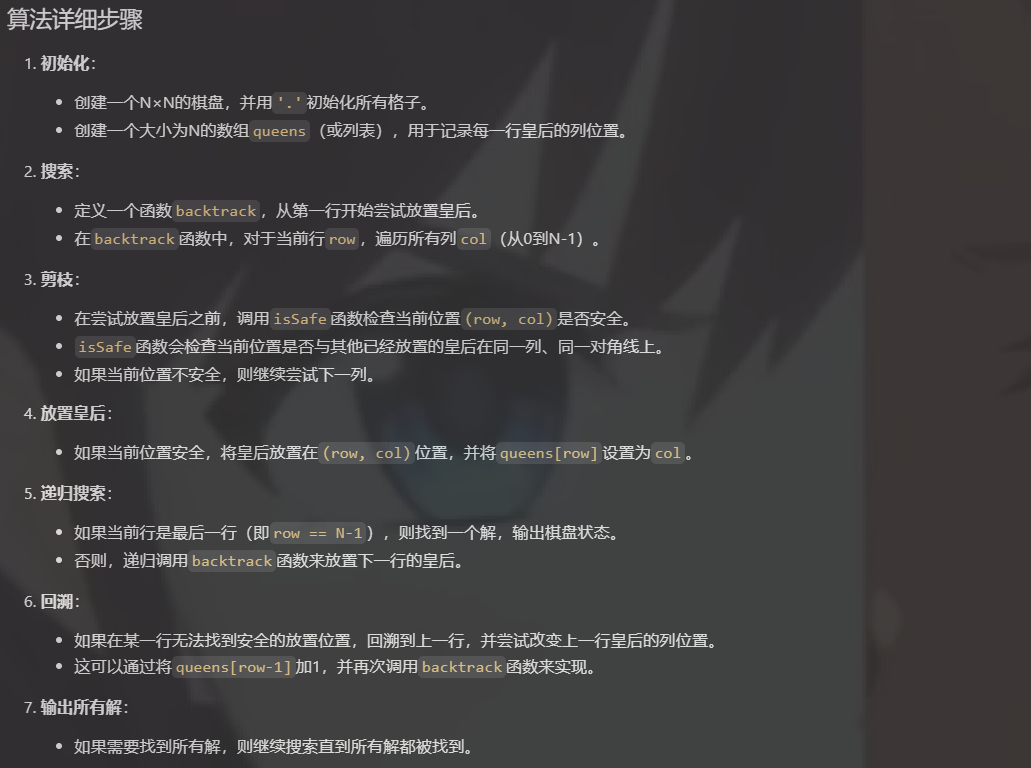
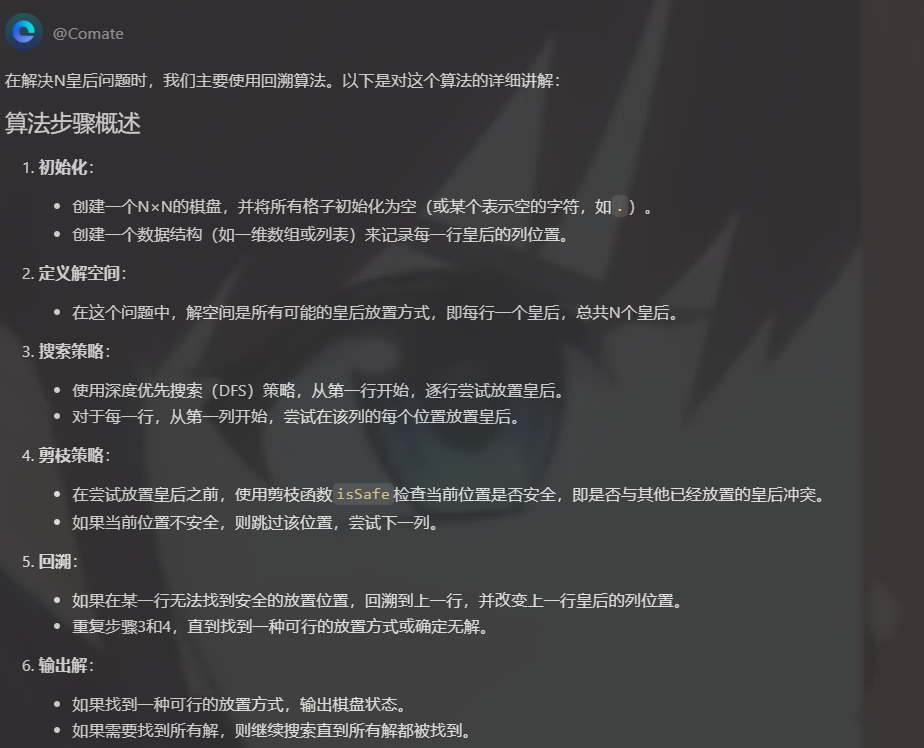
return 0;

}

运行结果：



对code稍作改动，即可达成原题要求：



1. 心得小结

AI辅助编程使得写code变得更加轻松，合理使用也能学到很多。在不能理解题意或某某处de不出来的bug的时候，AI能够提出清晰的思路，帮助解答疑惑。但高性能的AI也有可能让人产生惰性，可能会让人倾向于一个tab键写出一整串code的爽感，而忽视了自身写code能力的练习。

1. OJ题解
2. 问题描述

给定一个n个点m条边的有向图，点的编号是1到n，图中可能存在重边和自环。

请输出任意一个该有向图的拓扑序列，如果拓扑序列不存在，则输出-1。

若一个由图中所有点构成的序列A满足：对于图中的每条边(x, y)，x在A中都出现在y之前，则称A是该图的一个拓扑序列。

数据范围：1≤n,m≤10^5

（选自xmuoj平台2024年校外实训一（WEEK3）C021）

1. 建模（IPO分析）

Input：输入从1到n构成的有向图。

Process：从遍历没有被指向的节点，每遍历完一个节点，就删除该节点指向下一个节点的边，从而产生下一个没有被指向的节点。

Output：如果每个节点都被遍历，则输出遍历的节点顺序，否则输出-1；

1. 查资料（算法要点解析）

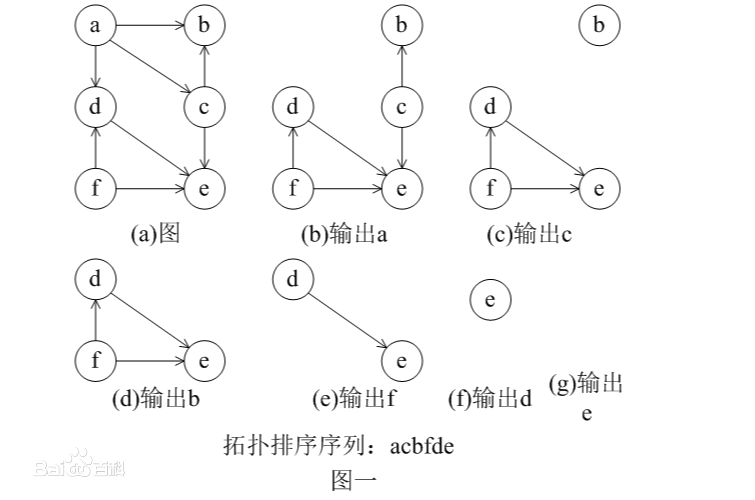
* 对于一个有 n 个顶点的 DAG，可能存在多个不同的拓扑序列。
* 如果图中存在环，则不能得到拓扑序列。如果图中不存在环，则至少存在一个拓扑序列。
* 可以使用广度优先搜索、深度优先搜索等图遍历算法求解拓扑序列。
* 拓扑序列在很多场景下有应用，比如为工程项目排序建立先后关系、计算程序中任务的执行顺序等。
* 拓扑序列可以用于检测图中是否存在环。如果可以成功生成拓扑序列，说明图中没有环，反之则说明有环。
* 对于一个DAG，可以根据拓扑序列将其转化为等价的序列图。

出度(Out-degree)：对于一个图中的某个顶点,出度表示从该顶点指向其他顶点的边的数量。它描述了从该顶点发散出去的连接数

入度(In-degree)：对于一个图中的某个顶点,入度表示指向该顶点的边的数量。它描述了进入该顶点的连接数。

度数(Degree)：对于一个图中的某个顶点,度数等于其出度与入度的和。它描述了与该顶点相关的边的总数。

①从图中选择一个入度为0的顶点，并输出该顶点；
②从图中删除该顶点及其相关联的有向边，调整被删除有向边的终点的入度（入度减1）；
③重复①和②；
④直到所有顶点均被输出，拓扑序列完成；否则，无拓扑序列。



选自https://blog.csdn.net/qq947467490/article/details/131743221

拿本题例子举例

第一步，遍历节点1，放入队列，删除1->2,1->3的边。

第二步，遍历节点2，放入队列，删除2->3的边。

第三步，遍历节点3，放入队列。

第四步，队列长度等于n，输出队列。

1. 编码及调试

#include<iostream>

#include<algorithm>

#include<queue>

#include<vector>

using namespace std;

int n, m;

vector<int> in;

vector<int> res;

vector<vector<int>> edges;

int main()

{

    cin >> n >> m;

    edges.resize(n);

    in.resize(n);

    for (int i = 0; i < n; i++)

        in[i] = 0;

    for (int i = 0; i < m; i++)

    {

        int f,t;

        cin >> f >> t;

        f--;

        t--;

        edges[f].push\_back(t);

        in[t]++;

    }

    queue<int> q;

    for (int i = 0; i < n; i++)

    {

        if (in[i] == 0)

            q.push(i);

    }

    while (!q.empty())

    {

        int cur = q.front();

        q.pop();

        res.push\_back(cur);

        for (int next : edges[cur])

        {

            in[next]--;

            if (in[next] == 0)

                q.push(next);

        }

    }

    if (res.size() < n)

        cout << -1 << endl;

    else

    {

        for (int ans : res)

            cout << ans + 1 << " ";

        cout << endl;

    }

    return 0;

}

运行结果：

1. 解题心得

拓扑排序需要对图的算法，如深度优先遍历等具有一定基础，属于对有向无环图的一种进阶遍历，其本身code与算法也不复杂，容易上手，能较好训练演算能力。而且上一题是非常抽象的滚石柱问题，考验立体想象与编程能力，尽管之前有多到同样算法的深搜题，该题也是比较难啃的。而本题作为最后一题，则较为轻松，宛如柳暗花明。

1. OJ在线编程课程总结
2. 掌握知识点

枚举算法，递归算法，分治算法，排序算法，深搜算法。其中，深搜算法和排序算法掌握较好，递归算法较差。

1. AC的情况

虽然oj上的题全部AC了，但没有把握能说完全掌握了算法知识，其中很多题是借助了网络进行解答的。

1. 本课程学习情况总结

本课程的主要收获主要来源于对oj上题目的练习，为了解题而迫使学习了相关算法。在没有领悟到相关算法的解法时，解题往往很困难，通过学习，看题目讲解，在掌握之后解题也变得容易起来。解题时间主要以课堂居多，碰到过被某道题卡住的情况，会用部分时间在课下解答。所以总的来说本课程学习情况自我评价良好。

1. 讲座心得

本课程选取讲座主要围绕AI展开，主要有新型产品的发布会和对主要负责人的访谈。由于本人并不是人工智能专业，也不打算从事AI相关研究和工作，故有关技术层面的话题将选择抛弃，心得主要以当前AI的用途和未来愿景为主。

首先是GPT-4O，展示了新一代GPT的智能。GPT已经能够解读人类的情感，并进行模仿，可以识别图片，并进行实时对话交流。在日常生活中GPT随时都有用处，将来也可能会推出各种新功能。

然后跟随着NAVIDIA的创始人黄仁勋，展示了NAVIDIA在科技领域领先于世界，也展现了作为华人对于大国科技的自豪。现在还比较有印象的是有领先于世界运行速率的GPU，以及未来可期的机器人，都是AI领域非常卓越的成就，让身为后人的我们对未来充满期待。顺带也适当了解了当今世界的伟人，在公众面前他们是人类的骄傲，但是在私下他们也和普通人一样有自己的爱好。

接着是微软的Copilot+PC，在AI的助力下，PC也能够进一步升级，各种各样先进的功能接踵而至。Google也在AI的辅助下推出RECALL和GEMINI系统，使搜索功能也变得智能。对于个人而言，要说最吸引我的地方，那就是可以在Adobe的软件上搭载AI，辅助进行绘画、剪辑、调音等，如果真的有一天能够使用这些功能进行创作的话，可想而知，技术力的门槛将会大大降低。再者，“陪玩”功能也比较吸引我，这将给出各种游戏不一样的游戏体验，游戏研发者是否也能接AI开发出新型的游戏？而广泛地说，PC和Google都是大部分人生产和生活中的必需品，丰富的功能将会带来极大便利。

最后是马斯克Elon Musk的两个项目，这两个项目展示了AI在其他领域的应用。一个在工厂上应用于生产，一个通过对人脑进行研究将有望用于医学或生活等领域。

总的来说，AI方面的科技领域正在逐渐进步的阶段，对未来的各种高科技的出现一片向好。但是，AI产生的问题也是不可忽视的，比如诸多影视作品中出现的AI取代人类导致人类危机的问题，这需要要求研发者们对此予以重视，绝对避免AI违抗人类的现象出现。再者，由于AI的技术进步，艺术的创作对能力的要求降低，而对审美和想象力的要求就会进一步上升，可能这对部分创作者不会是一个好事，但却是必须要面临的难题。更深者可能会涉及政治层面的问题。

然而，与其抱着悲观的想法去阻止AI的进步，不若以乐观的心态期待未来的新科技。

1. 课程心得及建议

上完本课程，对本课程留下的印象已经成了坐在机房里当码农。尽管刷算法题能够提升写code的能力，但有些题目对于基础较差的人来说确实有难度，希望能合理安排题目难度梯度。

随着课程的推进，后面来上课的人愈来愈少，一方面考勤比较宽松签完到就能走，另一方面一个人用自己电脑写code确实比一群人在机房写code轻松许多。希望能够在课堂内容上进行改进，进行算法讲解等内容，而不是一味地刷题。