**《校外实训》**

**期末课程总结报告**

**学生姓名： 苏一涵**

**学 号： 36720232204041**

**专业班级： 软件工程**

**教 师： 卢伟清**

**二○二四 年 七 月 二十 日**

**目 录**

**1.题解………………………………………………………………P3**

**2.企业讲座收获……………………………………………………P30**

**3.本课程学习总结…………………………………………………P33**

**题解**

**1.N皇后问题**

**题目描述：**

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。

输入：皇后的个数n（n<=13）

输出：长度为n的正整数。输出结果里的每一行都代表一种摆法。行里的第i个数字如果是n，就代表第i行的皇后应该放在第n列。皇后的行、列编号都是从1开始算。

示例：

输入4；

输：2413

3142

思路：

要想让棋盘上摆n个皇后，就要让每个皇后的位置之间不冲突。

即所有皇后都不同列且都不在对角线上。

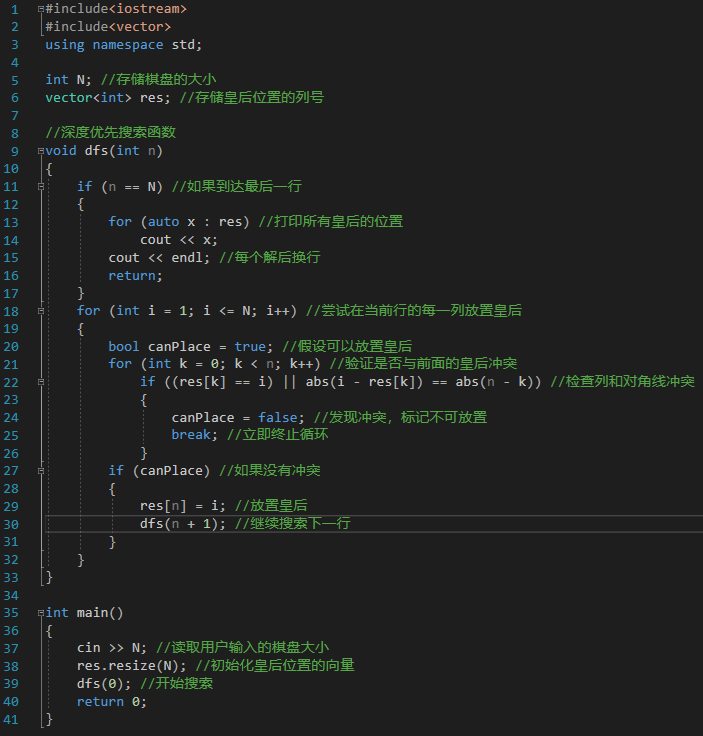
判断规则：

点(x1,y1)与点(x2,y2)在同一行，则有x1==x2：

点(x1,y1)与点(x2,y2)在同一对角线上，则有|x1-x2|==|y1-y2|。

所以我们可以尝试使用递归来一行一行搜索，当前k-1个皇后都满足条件时，再从第k-1皇后的下一行的每一列开始搜索，如果该行有解，用一个vector容器res记录该皇后的位置，直到记录了n个位置，输出该解法；

**代码实现：**



1. **熄灯问题**

**题目描述：**

有一个由按钮组成的矩阵，其中每行有6个按钮，共5行。每个按钮的位置上有一盏灯。当按下一个按钮后，该按钮以及周围位置(上边、下边、左边、右边)的灯都会改变一次。即，如果灯原来是点亮的，就会被熄灭；如果灯原来是熄灭的，则会被点亮。在矩阵角上的按钮改变3盏灯的状态；在矩阵边上的按钮改变4盏灯的状态；其他的按钮改变5盏灯的状态。

在上图中，左边矩阵中用X标记的按钮表示被按下，右边的矩阵表示灯状态的改变。对矩阵中的每盏灯设置一个初始状态。请你按按钮，直至每一盏等都熄灭。与一盏灯毗邻的多个按钮被按下时，一个操作会抵消另一次操作的结果。在下图中，第2行第3、5列的按钮都被按下，因此第2行、第4列的灯的状态就不改变。

请你写一个程序，确定需要按下哪些按钮，恰好使得所有的灯都熄灭。根据上面的规则，我们知道1）第2次按下同一个按钮时，将抵消第1次按下时所产生的结果。因此，每个按钮最多只需要按下一次；2）各个按钮被按下的顺序对最终的结果没有影响；3）对第1行中每盏点亮的灯，按下第2行对应的按钮，就可以熄灭第1行的全部灯。如此重复下去，可以熄灭第1、2、3、4行的全部灯。同样，按下第1、2、3、4、5列的按钮，可以熄灭前5列的灯。

输入：5行组成，每一行包括6个数字（0或1）。相邻两个数字之间用单个空格隔开。

0表示灯的初始状态是熄灭的，1表示灯的初始状态是点亮的。

输出：5行组成，每一行包括6个数字（0或1）。相邻两个数字之间用单个空格隔开。其中的1表示需要把对应的按钮按下，0则表示不需要按对应的按钮。

示例：

输入：2

0 1 1 0 1 0

1 0 0 1 1 1

0 0 1 0 0 1

1 0 0 1 0 1

0 1 1 1 0 0

0 0 1 0 1 0

1 0 1 0 1 1

0 0 1 0 1 1

1 0 1 1 0 0

0 1 0 1 0 0

输出：PUZZLE #1

1 0 1 0 0 1

1 1 0 1 0 1

0 0 1 0 1 1

1 0 0 1 0 0

0 1 0 0 0 0

PUZZLE #2

1 0 0 1 1 1

1 1 0 0 0 0

0 0 0 1 0 0

1 1 0 1 0 1

1 0 1 1 0 1

由题意可知，每个灯只有亮或者不亮两种状态；

对于按钮，当按下同一个按钮两次产生的效果等于没按，所以我们对每一个按钮只需要考虑按一次或者不按两种情况，同时按钮按下的顺序对结果没有影响。

所以，我们可以使用bitset<6>类型来表示每一行灯的状态，src表示原始灯状态，lights表示当前灯状态，res表示结果，即需要按下的按钮状态。

以及一个bitset<6>类型的line变量来表示每种按钮组合。

现在我们从灯阵列的第一行开始分析：假定第一行灯状态已经确定，要熄灭第1行某个亮着的灯(假设位于第n列), 那么唯一的办法就是按下第2行第n列的开关。为了使第1行的灯全部熄灭, 第2行的开关状态就是唯一的。同理，当第2行开关按过之后，为了使第二行的灯全部熄灭，第3行的开关状态也是唯一的。

所以我们可以通过枚举第一行灯的状态来确定整个灯阵列的状态，从而找到所有灯都熄灭的按键方式。

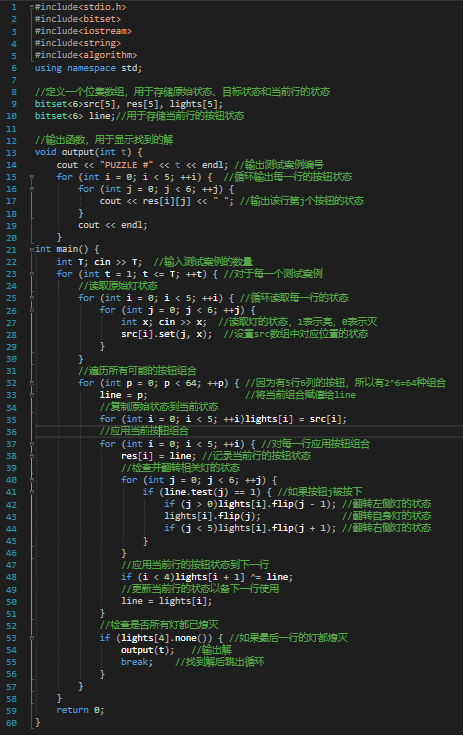
实现过程：遍历从0到63（即2^6-1），穷举所有可能的按钮组合。对于每一种按钮组合line，将其应用于原始状态src，产生新的状态lights。

应用过程包括：对当前行和上下行（如果存在）进行异或操作，以及对当前行的每个按钮，如果按下（line.test(j) == 1），则翻转自身及相邻按钮的状态。

如果在操作完所有行后，最后一行的灯都处于熄灭状态（lights[4].none()），则找到一个解。

输出解res。

**代码实现：**



1. **拨钟问题**



由题意可知，题目要求找到用最少的操作数使得所有时钟指向12点的方法，经过分析，我们发现：1.单个时钟每次旋转90度，每4次旋转状态归回初始状态，累计在其上的旋转结果与%4的余数相同。

2.一共有9种不同的move，每种move的累计周期也是4。一旦执行4次后，就让时钟回到初始状态。

3.9种move，一共需要枚举4^9种=**262,144**不同的情况。

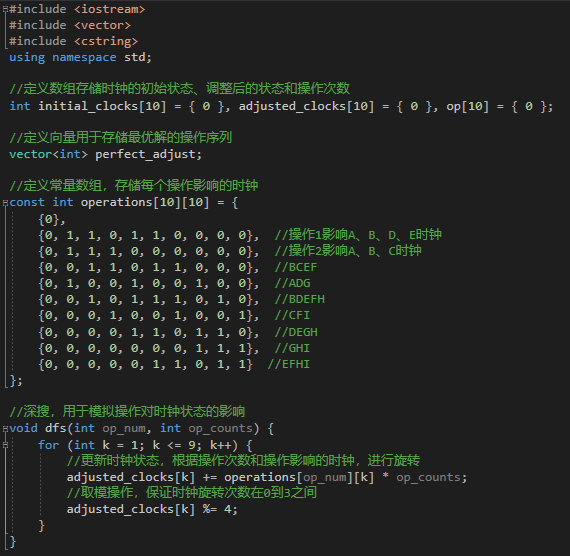
所以统计所有情况中move移动次序最少的那种就是最后的答案。

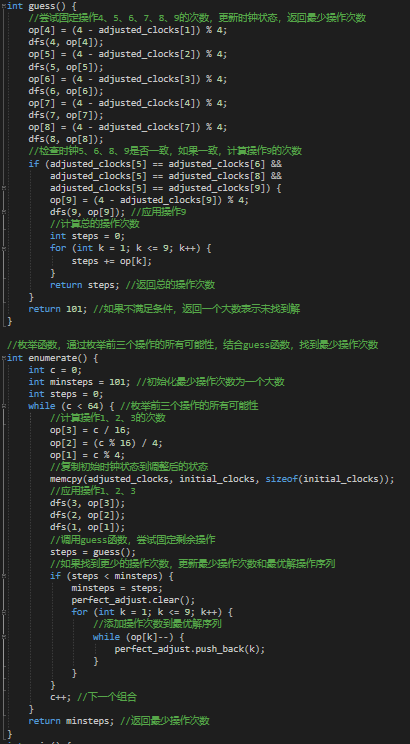
代码设计：1.状态设置：使用initial\_clocks数组存储时钟的初始状态。使用adjusted\_clocks数组存储调整后的时钟状态。使用operations二维数组存储每种操作会影响哪些时钟，以及影响的程度（旋转次数）。

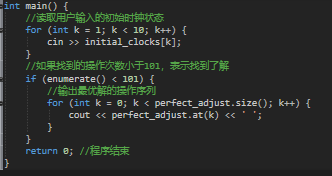
2.深搜：通过dfs函数模拟操作对时钟状态的影响。dfs函数接收操作编号和操作次数，根据operations数组更新adjusted\_clocks数组

3.优化：通过guess函数尝试固定一些操作，减少搜索空间。guess函数先假设操作4、5、6、7、8、9的次数，尝试将时钟状态调整目标状态。enumerate函数通过枚举前三个操作（操作1、2、3）的所有可能性，结合guess函数的预估操作，找到最少操作次数。使用minsteps跟踪并记录最少操作次数。使用perfect\_adjust向量存储达到目标状态的操作序列。

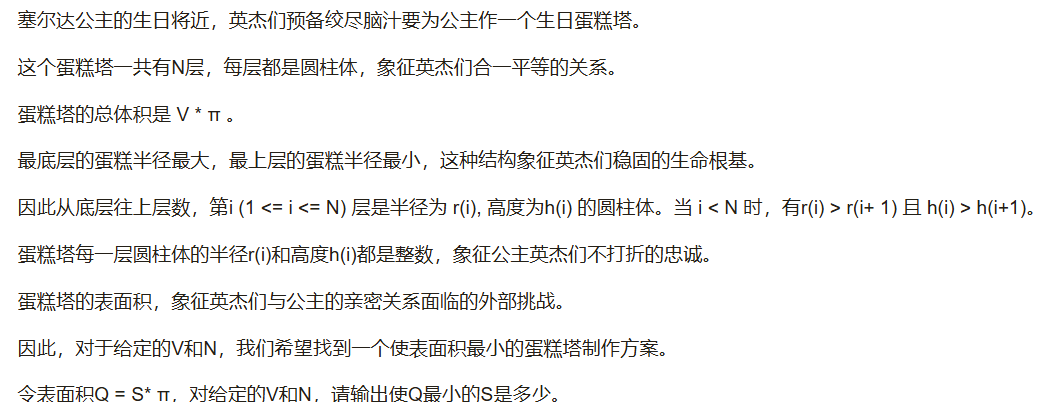
**代码实现：**

****

****

****

1. **英杰们的蛋糕塔**

**题目描述：**

****

**分析：**

1.显然，由题意的数据范围想到的第一件事儿就是搜索，毕竟N的范围只有20。

2.于是我们就要枚举每一层的R和H了，但显然只这样暴力的枚举肯定是会超时的，所以我们必须要想办法优化。

3.优化的思路就是剪枝：

三个剪枝条件：

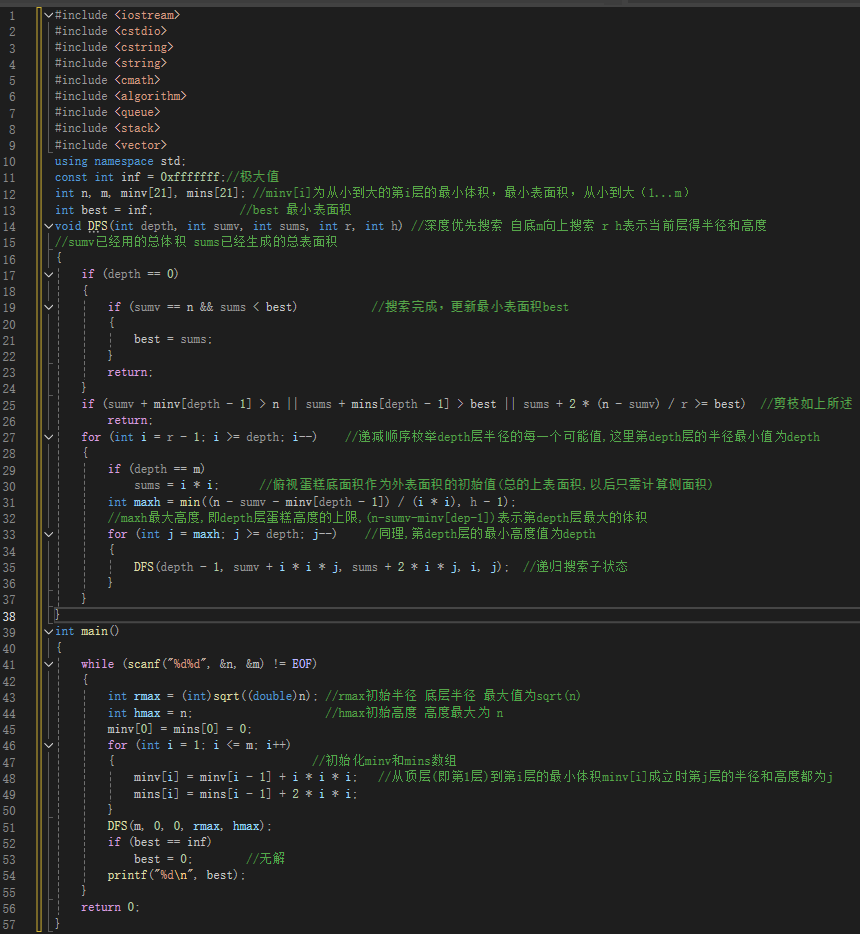
1、已经搜索过的体积加上还未搜索过的最小体积不能比总体积n 大

2、已经搜索过的表面积加上还未搜索过的最小表面积不能比之前的最小总表面积best 大

3、n-sumv既所剩体积记作dv 还需要的表面积为s

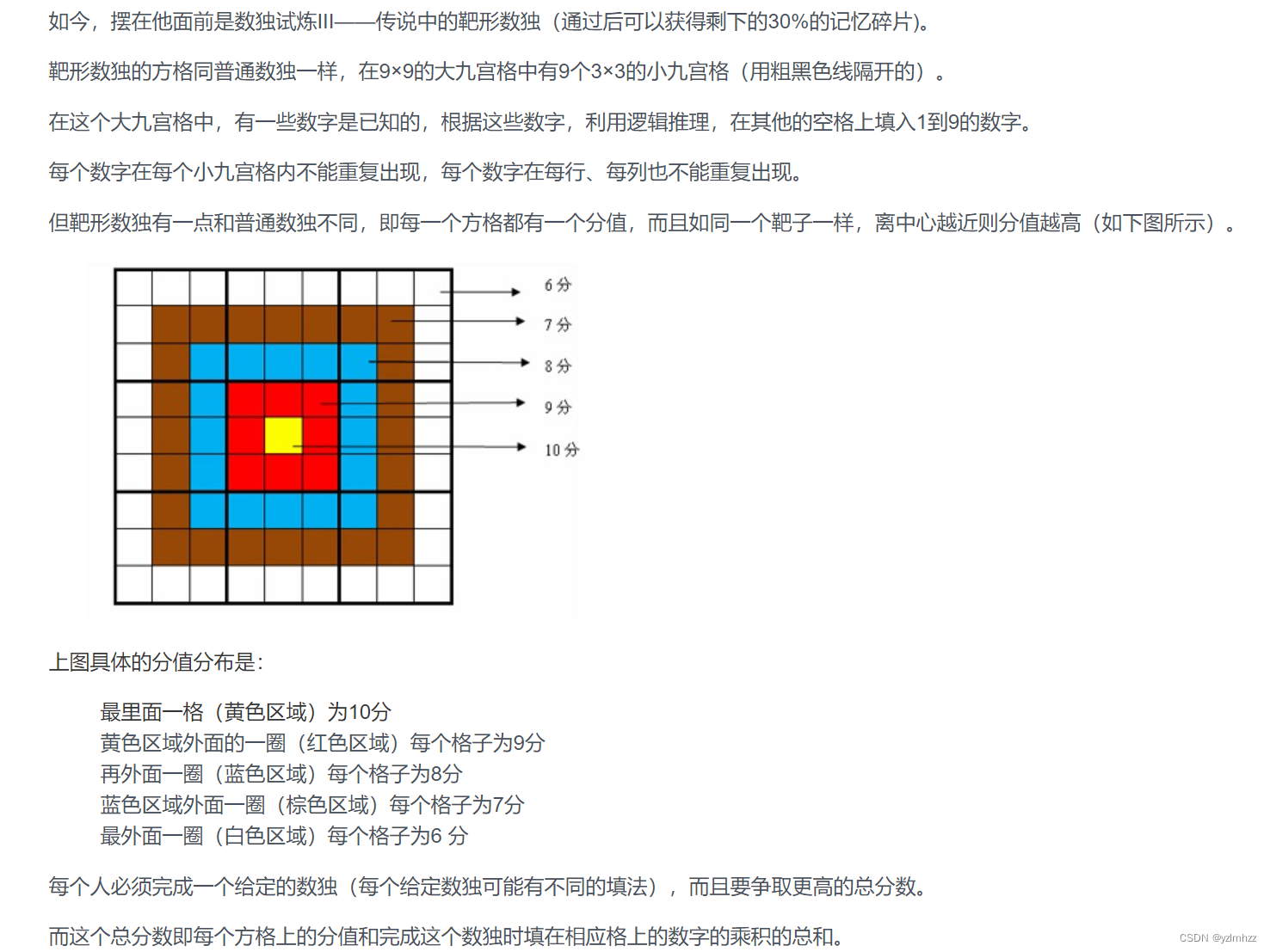
s=2\*ri\*hi+2\*r(i-1)\*h(i-1)+... >=2\*ri\*hi\*ri/r+2\*r(i-1)\*h(i-1)\*r(i-1)/r+...=2\*dv/r（i从depth-1取,r为当前半径 ri/r<1）所以得到还需要的最小表面积s=2\*(n-sumv)/r，如果最小的s和已经搜索过的表面积sums依然比best大 就不用继续搜索了

**代码实现：**

****

1. **寻找林克的回忆（3）**

题目描述：

****

**分析：**

很显然，这也是一道深搜题，需要搜索的就是每个空填写的答案数，如果只是普通的搜索，那么复杂度会来到O（9^n），这肯定会超时，所以这里采用位操作来进行优化。

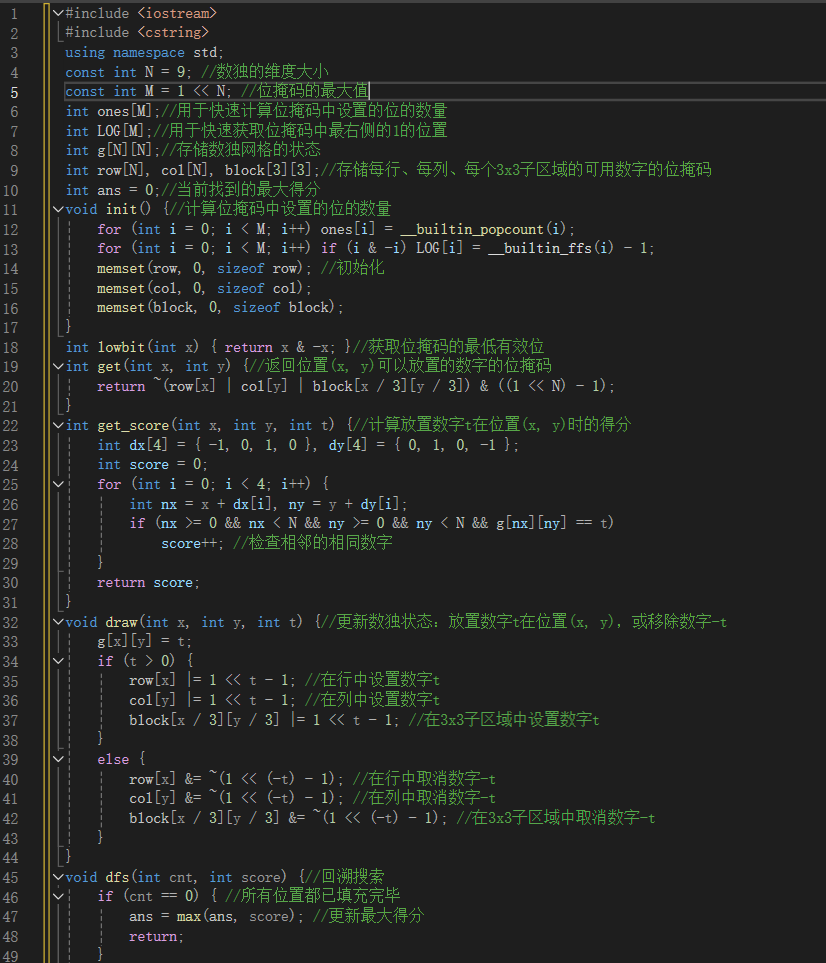
利用二进制，我们用二进制中的1表示还未被使用，0表示已被使用，而0、1所在位置便是代表的什么数，如最末位表示数1的状态，倒二位表示数2的状态...以此，我们将数组降维到了r[9],c[9],block[3][3]，用一个数便能表示1-9九个数的使用状态。再通过&运算我们可以轻松求得数独某一位置上行、列、小九宫格共同可使用的数。

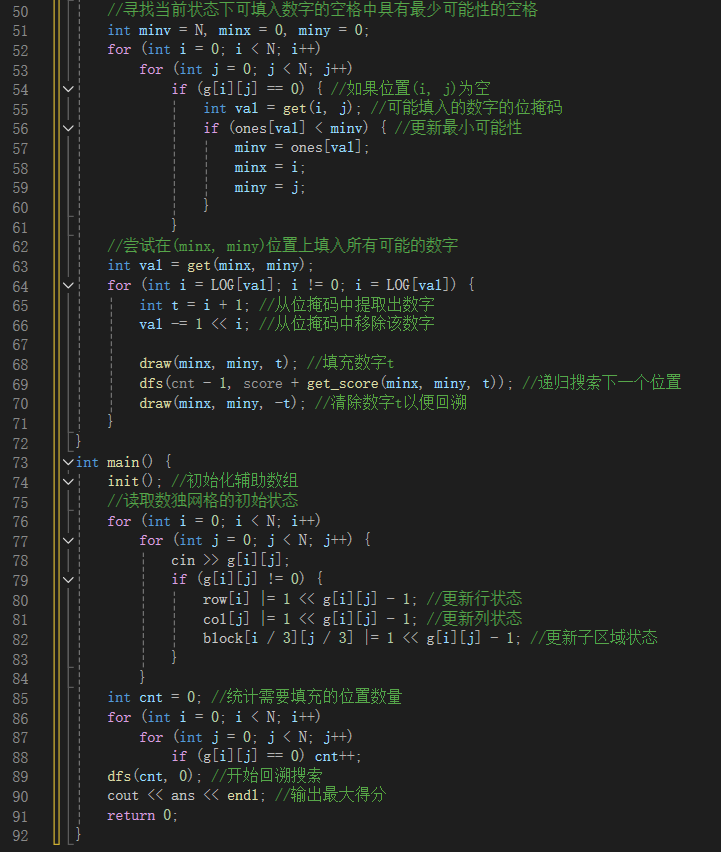
dfs(int cnt, int score) 函数是递归回溯函数，它遍历所有可能的数独填充方案，并计算每种方案的得分。cnt 表示剩余需要填充的位置数量，score 表示当前已有的得分。

如果 cnt 为 0，说明已经找到了一个完整的解决方案，此时比较并更新 ans 的值。

否则，寻找当前状态下可填入数字的空格中具有最少可能性的空格 (x, y)，然后对这个空格的所有可能性进行递归尝试。

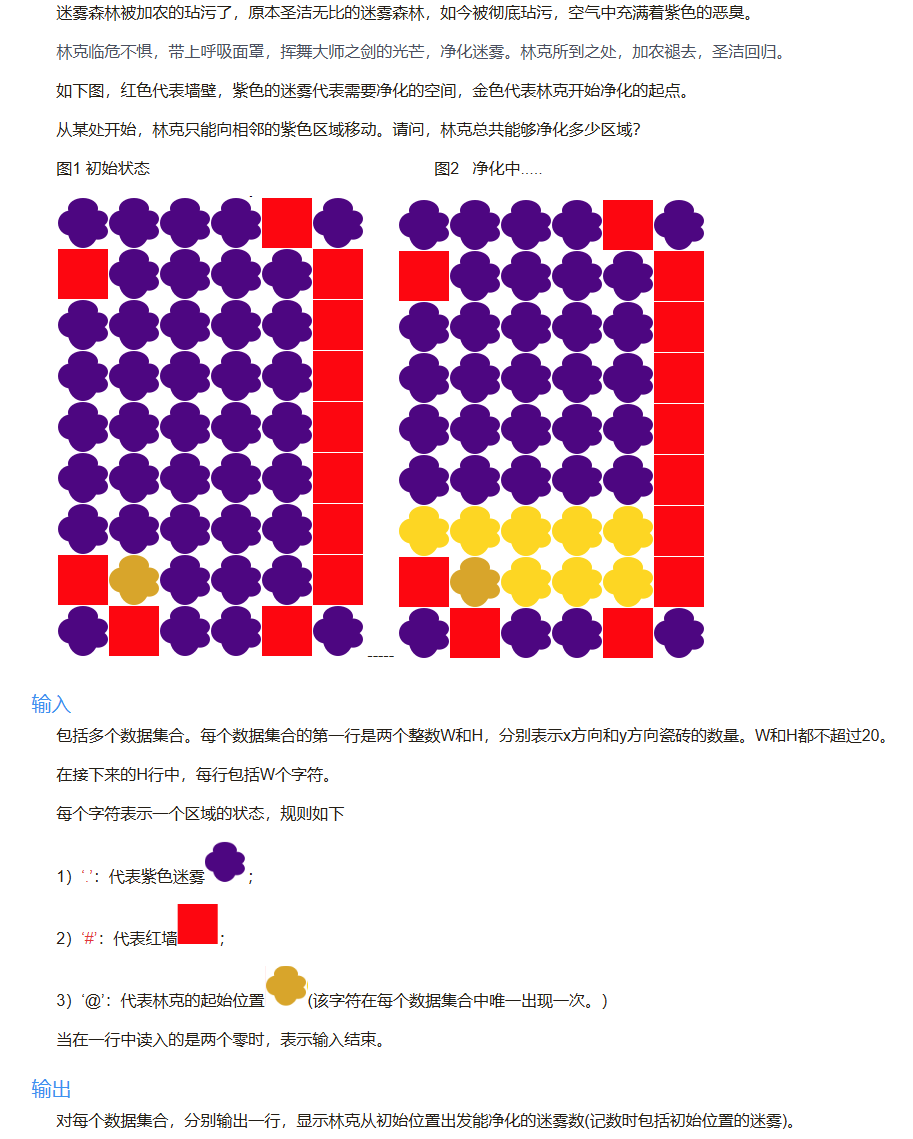
**代码实现：**

****



1. **净化迷雾森林**

题目描述：



分析：

由题意可得，这道题的数据范围比较友好，所以直接采取暴力广搜的做法。

**思路：**

状态表示：使用一个二维字符数组fogforest[27][27]来表示迷宫的状态，额外的空间是为了防止越界。

使用整数变量res来累计净化迷雾的数量。

DFS：使用递归函数dfs(int i, int j)从起始位置开始搜索迷宫。

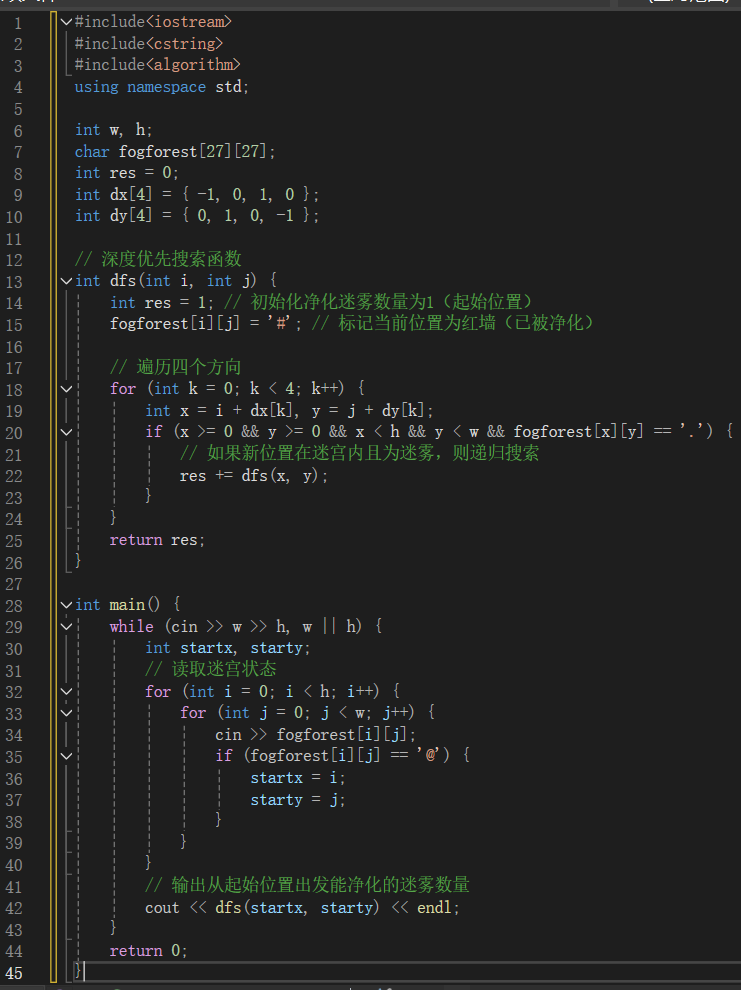
对于每一个可达的位置，如果该位置是迷雾（.），则递归地访问它的四个相邻位置。

当访问一个迷雾位置时，将其标记为红墙（#），表示已被净化。

边界条件：在dfs函数中，检查当前位置是否在迷宫内且为迷雾状态。

如果当前位置超出边界或为红墙，则不进行进一步的搜索。

**代码实现：**



1. **加农的入侵**

**题目描述：**

****

**分析：**

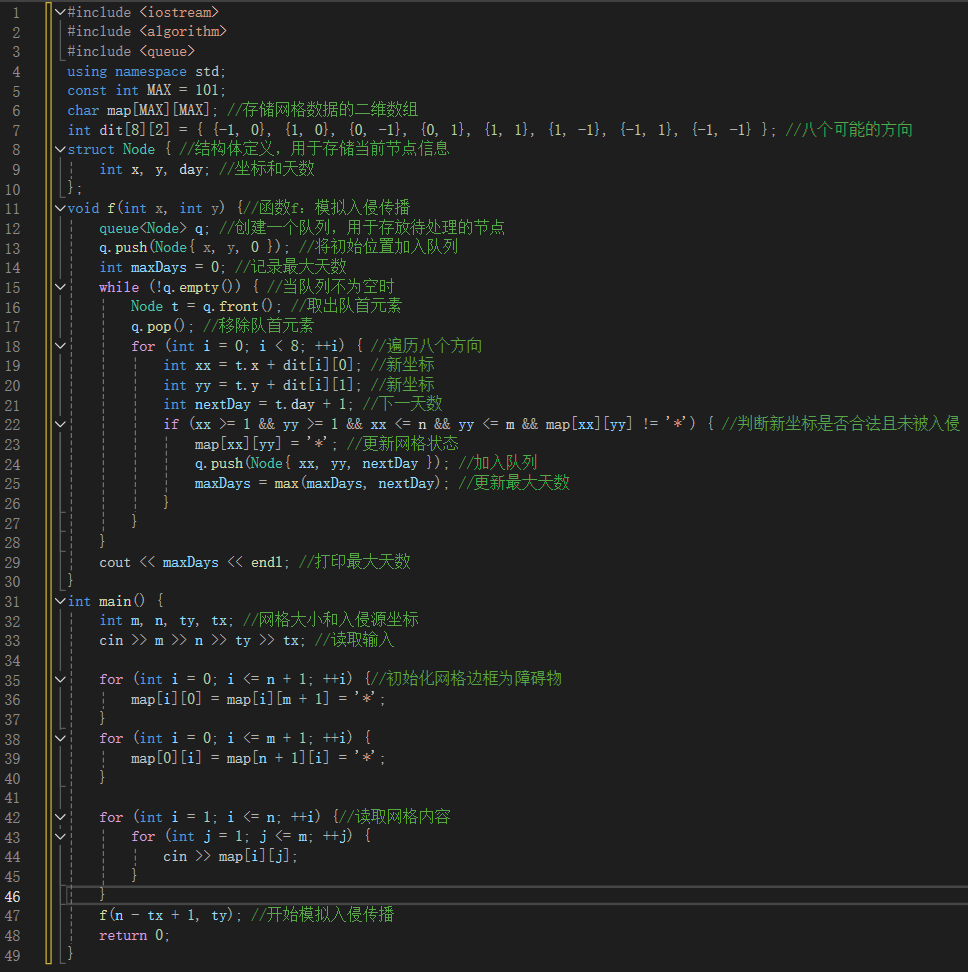
这题是一道标准的广搜，解题思路主要有三步

1.状态表示：使用二维数组表示迷宫状态，额外留出空间避免越界。

2.搜索：使用递归实现DFS，遍历迷宫中所有可达迷雾位置。

3.输出结果：输出从起始位置出发能净化的迷雾数量。

**代码实现：**



1. **Dijkstra求最短路（2）**

题目描述：



分析：这道题的常规思路就是暴力搜索，但是这样的复杂度太高了，所以我们考虑用小跟堆来优化算法。

小根堆是一种特殊的二叉树结构，它的性质是在任意时刻，父节点的值都比子节点的值要小。这种特性使得我们可以快速地找到堆顶元素（即最小值）。在 Dijkstra 算法中，我们需要不断找出当前距离起点最近的顶点进行扩展，因此使用小根堆能够有效地降低时间复杂度。

原理：首先将起点的初始距离设为 0，其余顶点的初始距离设为无穷大（通常用一个足够大的数值表示，比如 LLONG\_MAX 或者 INT\_MAX），并将起点放入小根堆中。

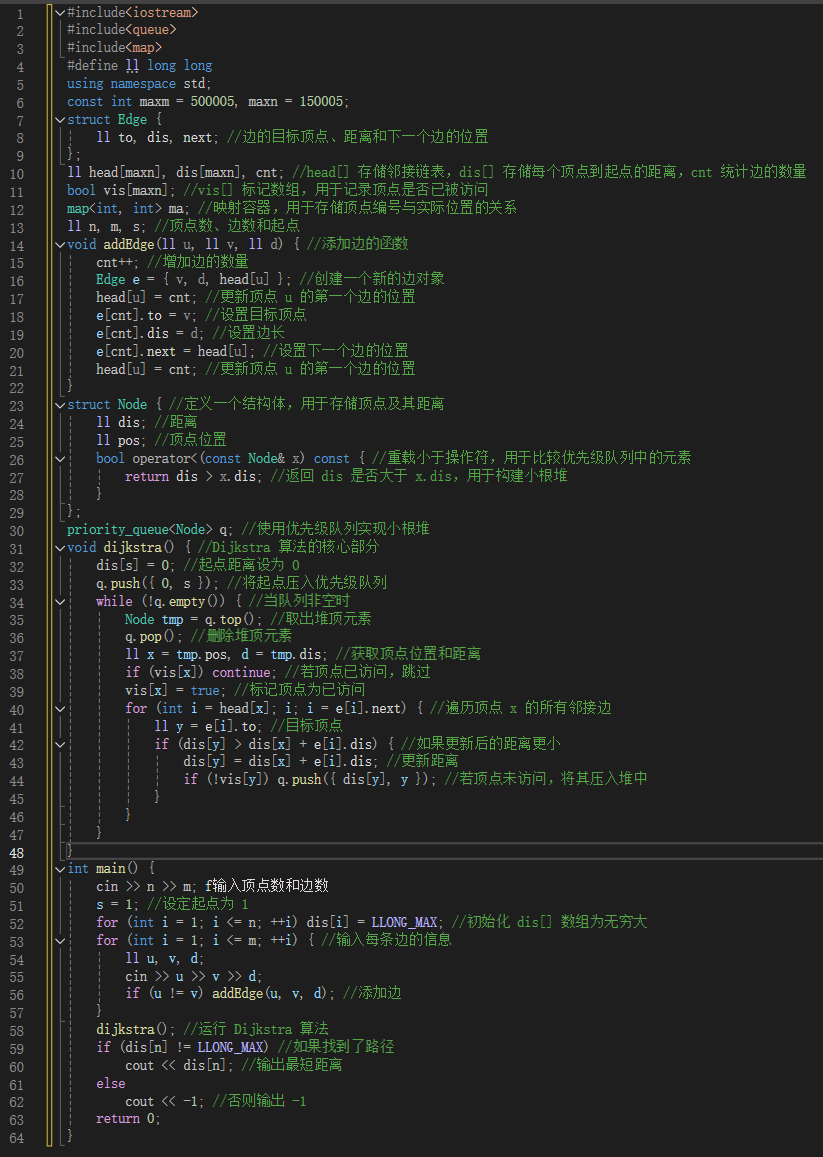
核心循环：只要小根堆不为空，就重复以下步骤：

弹出堆顶元素（即当前距离起点最近的顶点），并标记为已访问。

对于该顶点的所有邻居顶点，如果它们还没有被访问过，那么尝试更新它们的距离。如果更新后的距离更小，就把这些顶点重新放入小根堆中。

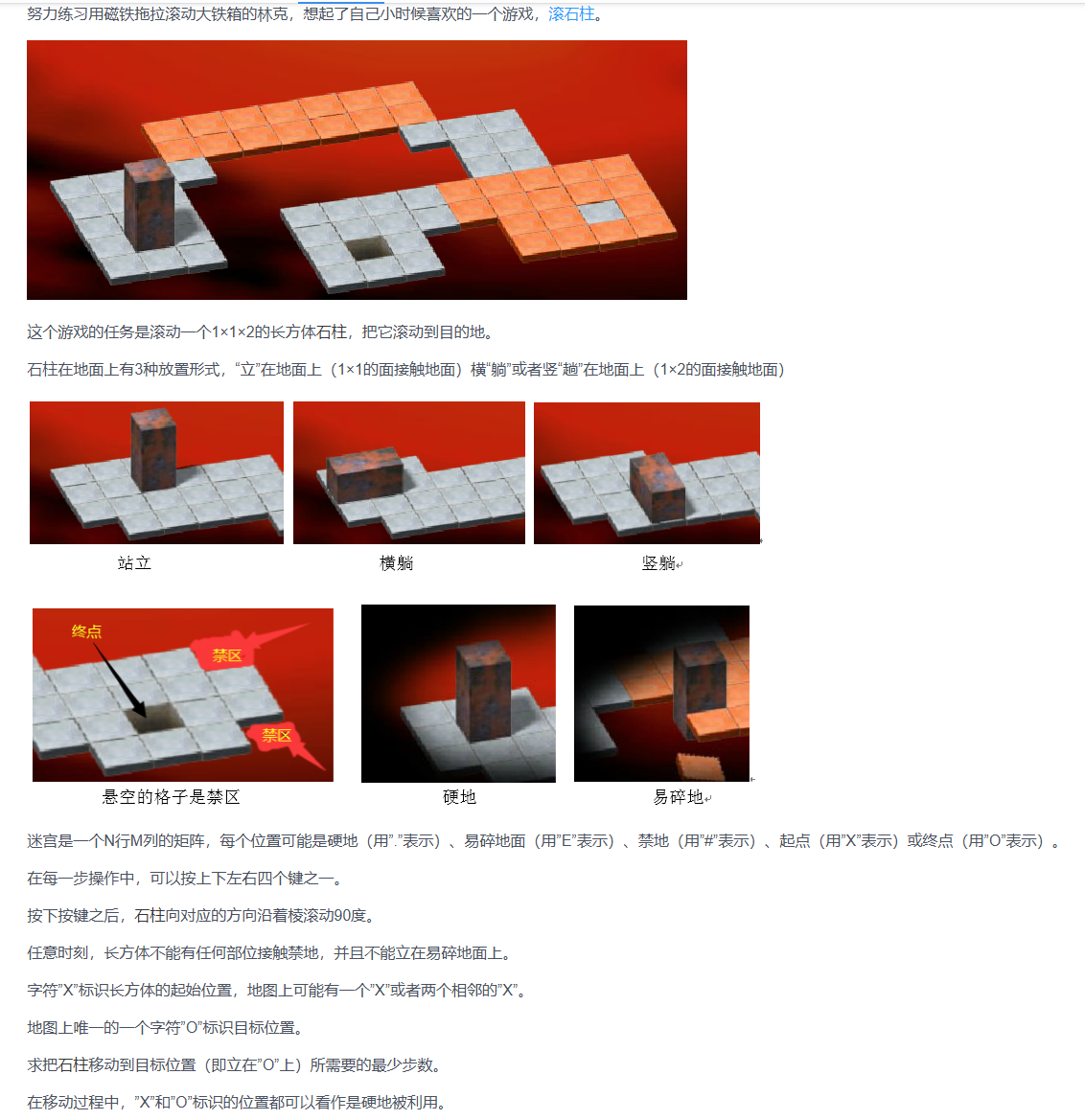
结束条件：当小根堆为空时，说明所有顶点都被访问过了，此时就可以得到从起点到各个顶点的最短距离。

**代码实现：**



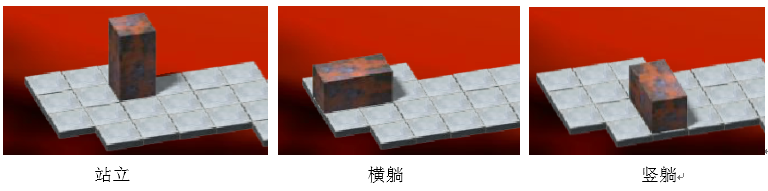
1. **滚石柱**

题目描述：



分析：

本题是迷宫问题的变形, 要找到"最少移动次数", 使用BFS算法, 遍历所有路径。注意到在本题中，石柱有三种状态, 立着放, 竖着放和横着放. 每种状态要移动的方式不同, 所以要区分开来。



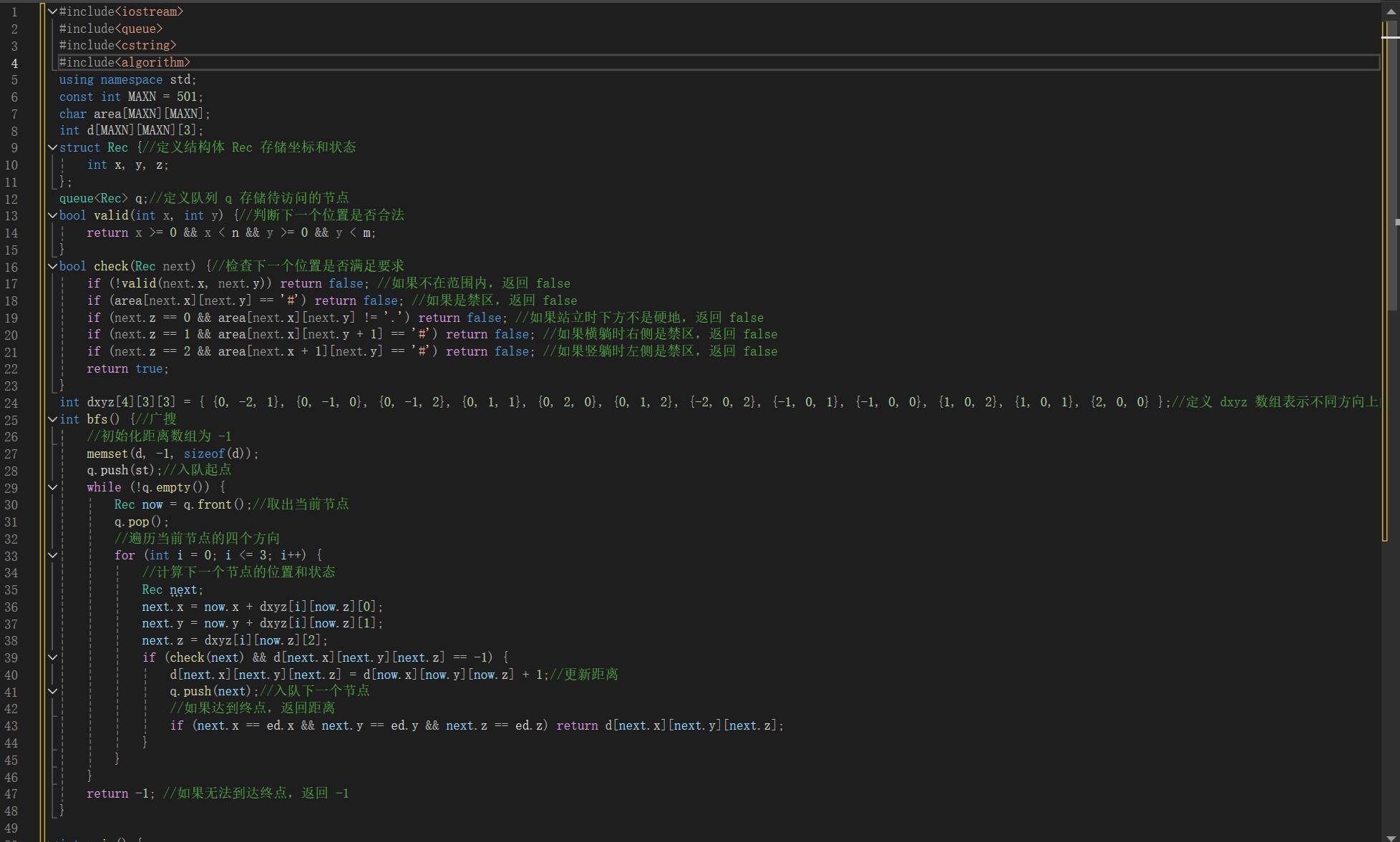
鉴于横着放和竖着放时, 有两个方格在地图上, 为了简化解题过程, 我们设置一个"核心", 配合石柱的"状态"我们就能确定方块的位置. 立着放时, 显然石柱"核心"标记在下面这个石柱; 竖着放时, 我们将"核心"标记在上面的方格上; 横着放时, 我们将"核心"标记在左边的方格上。这里我们按照上图从左到右的顺序, 用0, 1, 2三个数字来表示三种状态。

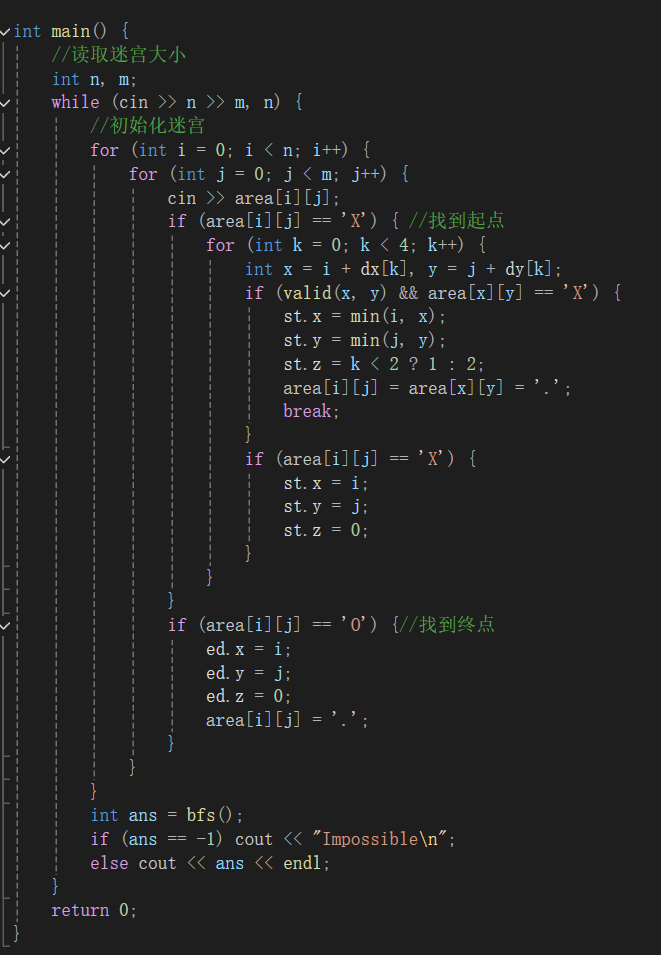
对应三种状态, 我们还需要定义三种状态下向四个方向移动的方式: 

和经典的迷宫问题相同, 我们需要记录一下某个位置是否已经"走过"了. 但是与迷宫问题不同的是, 我们定义一个位置已经"走过"了需要注意两个点, 即核心位置和状态都要相同。

最后检查一遍。

**代码实现：**





**10.突袭路线**

题目描述：



分析：

本题需要将问题转化为图论问题，即给定一个有向图，找到任意一个拓扑排序序列。问题的关键在于在BFS的框架上实现TopSort（拓扑序列）。

拓扑排序是一种特殊的排序方法，它对于有向无环图来说是可行的。在这个算法中，我们需要按照一定的顺序对图中的所有顶点进行排列，使得对于每一条边(u, v)，u都在v之前出现。具体来说，就是找到一种排列方式，使得对于每一条边(u, v)，u都排在v前面。

套用到这道题中，由于林克想要从背后没有人看守的敌人开始攻击，因此我们可以认为这些敌人是"入度"为0的顶点。

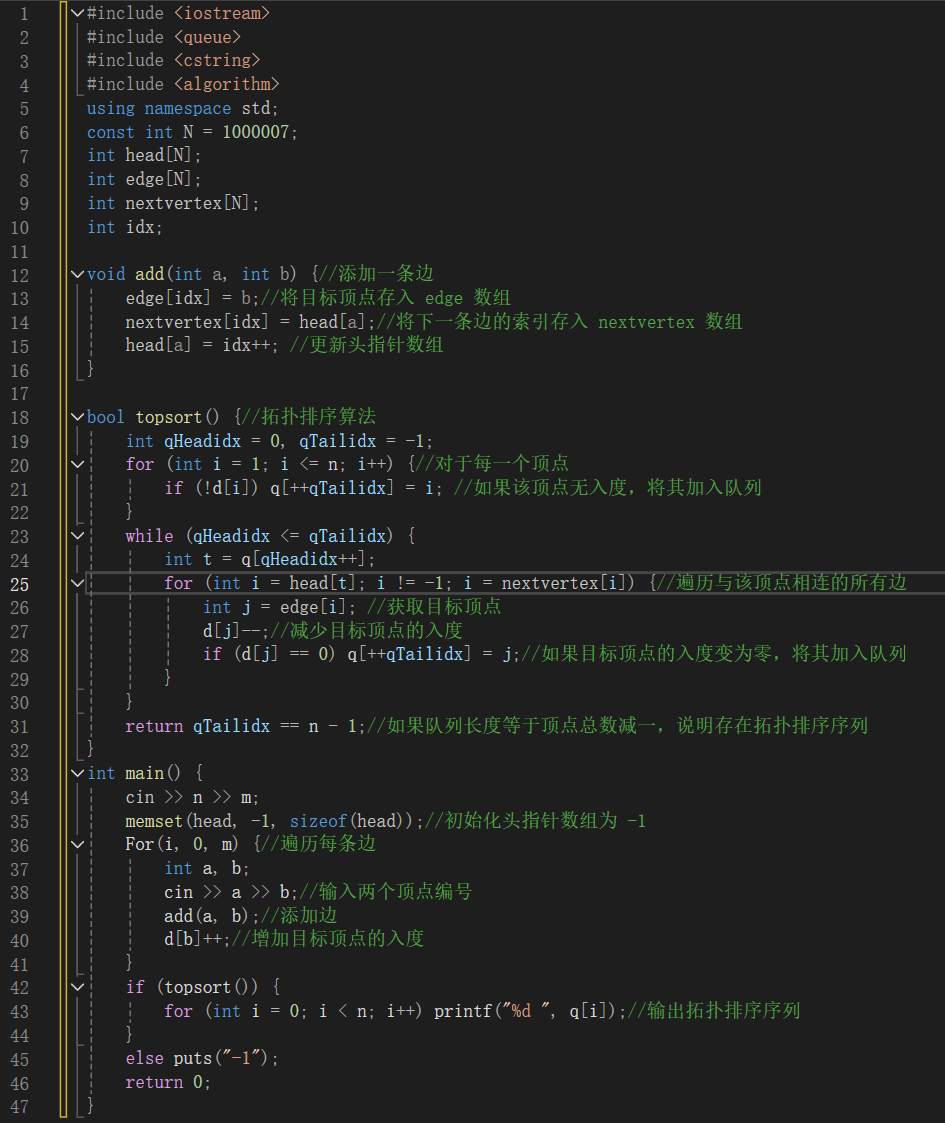
拓扑排序算法的基本思想是：首先选择所有入度为0的顶点，然后删除它们以及它们对应的边，接着再处理剩余的顶点，直到所有的顶点都被处理完毕为止。

因此我们需要完成三步：1.初始化一个队列，用于存储所有入度为0的顶点。

2.在每次循环中，取出队列中的第一个顶点t，然后遍历与其相邻的所有边，减少对应目标顶点的入度。如果某个目标顶点的入度变为0，将其加入队列。

3.循环结束时，如果队列中的顶点数目等于总顶点数减一，那么就找到了一个拓扑排序序列；否则，说明图中有环，不存在拓扑排序序列。

**代码实现：**



**企业讲座收获**

**讲座与发布会感悟报告**

作为一名学生，这些讲座和发布会不仅极大地扩展了我对科技发展的视野，也深深激发了我对未来的憧憬。每一个发布会和演讲，都让人充满了对科技无限可能的期待，这些前沿的科技突破不仅改变了我们与技术的交互方式，更塑造了未来社会的轮廓。

一、GPT-4o的发布：AI技术的新高峰

GPT-4o的发布标志着人工智能领域迈向了一个全新的高峰。这个强大的语言生成和理解模型，不仅具备了实时会话语音和视觉交互功能，还能够提供各种情感风格的声音，从而大大提高了人机交互的自然度和响应的灵活性。尤其是GPT-4o与现实世界交互的能力，可以实现视频通话、辅导作业和工作等多种应用场景。这些通过与现实世界产生的数据交互，进一步提升了模型对现实的理解能力。

二、COMPUTEX 2024：黄仁勋的前瞻视野

在COMPUTEX 2024的演讲中，英伟达CEO黄仁勋对可持续发展和绿色能源的探讨，让我深刻了解到科技创新与社会责任之间的紧密联系。他谈到，高算力NPU赋能PC市场的重生，算力成本的降低将数据中心从成本中心转变为营收中心。此外，互联技术日益开放，通过Ethernet、NVLink及InfiniBand的协同作用，提升了数据中心的效率。而液冷技术的普及则推动了绿色和可持续AI生态的发展。算力成本的降低使得训练更大模型成为可能，加上高速传输和液冷技术的普及，驱动了“AI工厂”的崛起。

三、微软Copilot+PC特别发布会：编程革命的前奏

微软的Copilot+PC特别发布会展示了编程和开发领域的巨大变革。Mehdi提到，Copilot将很快采用OpenAI新发布的GPT-4o。在预览Demo中，Copilot通过语音指导用户在《我的世界》游戏中挖矿。而微软发布的Recall功能，可以检索用户在电脑上做过的任何事情，例如电话会议、PPT里的具体一页，甚至是具体浏览过的网页。同时，Live Captions功能扩展支持44种语言，可以将多种语言实时转写为英语。这一系列功能让我看到了人工智能与人类创造力的完美结合，为编程领域带来了更加高效的解决方案。

四、Google I/O 2024：AI与开发者工具的融合

在Google I/O 2024开发者主题演讲中，Google展示了其在AI和开发者工具方面的最新进展。Google AI Studio中的JIMI API提供了强大的多模态应用开发能力，支持跨平台、跨设备的无缝体验。演讲重点介绍了如何利用AI提升开发效率，包括代码优化、翻译和灵感生成等，通过GEMINI API开发者竞赛鼓励创新应用的开发。此外，AI在Android系统中的运用，如GEMININANO模型的设备端推理，以及Android Studio和Chrome中的AI集成，大大提升了开发者生产力和用户界面的卓越体验。Google对数据隐私和可持续发展的重视，进一步增强了我对科技公司未来发展方向的信心。

五、Elon Musk：引领未来的科技探索

Elon Musk的一系列项目更新，如CyberTruck和Neuralink，让我看到了未来科技探索的无限可能。CyberTruck不仅是汽车工业的革新，更是可持续出行的一次尝试。而Neuralink项目则让我对脑机接口技术的未来充满期待。这项技术对医学、健康和人类认知都可能产生深远影响。马斯克在访谈中详细介绍了Neuralink的总体规划，包括治愈瘫痪、强化身体机能的潜力，以及设备的具体功能和改进策略。

Neuralink的目标是通过在大脑和计算机之间建立高带宽连接，帮助人们解锁数字独立。Neuralink设备能够读取和写入大脑中的电信号，使我们可以通过思想来控制电脑和手机。团队正在不断改进设备的设计和手术技术，争取让设备更加安全、高效，能够为更多人带来希望。Neuralink不仅可以解锁数字独立，还可能扩展到物理世界的独立性，如控制机械臂和轮椅，甚至与机器人进行物理交互。长期目标是实现人类智能和数字智能之间更紧密的共生关系，通过增强大脑功能来应对人工智能的风险。

总结与展望：

在这些讲座和发布会中，我不仅学到了许多前沿的科技知识，更深刻地体会到了科技对社会和个人生活的深远影响。从GPT-4o到Neuralink，从绿色能源到编程革命，这些技术突破和创新无一不在拓展我们对未来的认知。我深信，未来的科技发展将带来更多的便利和可能性，而我也渴望成为这个激动人心时代的一部分，为构建更加美好的未来贡献自己的力量。

**本课程学习总结**

在过去的四周里，我有幸参与了校外实训课程。这段经历不仅让我深入理解了编程和算法的核心概念，还让我接触到了业界最前沿的技术和理念，真是一段难忘的学习旅程。接下来，我在这里分享我在这次实训中的所见所闻和收获。

一、在线题目的训练

实训的第一部分是为期三周的在线题目训练。每天面对一系列具有挑战性的算法问题，起初我感到非常吃力，甚至有些沮丧。然而，随着时间的推移和不断的练习，我逐渐掌握了多种算法技巧，包括拓扑排序、深度优先搜索（DFS）、广度优先搜索（BFS）以及剪枝策略等。这些技巧的掌握充实了我的解题手段，也让我受益匪浅，接下来我来具体介绍一下其中的几个算法技巧。

拓扑排序

拓扑排序是一种用于处理有向无环图（DAG）的特殊排序方法。通过学习拓扑排序，我明白了如何根据任务之间的依赖关系来安排执行顺序。这不仅在理论上让我受益匪浅，实际操作时我也发现了它在任务调度和资源分配中的广泛应用。每当我成功实现一个拓扑排序算法时，心中的成就感无以言表。

深度优先搜索（DFS）

深度优先搜索（DFS）是我在实训中最喜欢的算法之一。它的灵活性和强大功能让我能够在树或图中探索路径。在学习DFS时，我尝试了递归和非递归两种实现方式，最终在解决迷宫和图的连通性问题时找到了乐趣。每次成功找到一条路径时，我都感受到了一种探索未知的快感。

广度优先搜索（BFS）

与DFS相比，广度优先搜索（BFS）更适合寻找最短路径。在学习BFS的过程中，我学会了如何利用队列来实现这一算法。特别是在解决最短路径问题时，BFS的高效性让我深刻体会到算法选择的重要性。每当我用BFS成功解决一个问题时，都会忍不住与同学分享我的喜悦。

剪枝

剪枝是一种减少不必要计算的重要策略。在解决旅行商问题（TSP）时，我运用了剪枝技术，避免了许多冗余的计算。这不仅提高了算法的效率，也让我意识到，合理的思考和策略设计是解决复杂问题的关键。

二、知名企业的讲座

实训的最后一周，我们有幸聆听了来自OpenAI、Google以及埃隆·马斯克等知名企业和人物的精彩演讲。这些讲座不仅拓宽了我的视野，也激发了我对未来技术发展的兴趣。

OpenAI

OpenAI的讲座让我对人工智能领域的最新进展有了更深入的了解，尤其是在自然语言处理方面的应用。他们分享了AI如何在翻译、文本生成等领域发挥作用，这让我对未来的应用场景充满期待。听完讲座后，我对AI的潜力有了更深刻的认识，甚至开始思考自己是否可以在这一领域有所作为。

Google

Google的演讲围绕大数据分析和机器学习技术展开。他们分享了如何利用这些技术优化搜索引擎和推荐系统，并强调了数据的重要性。这让我意识到，作为计算机专业的学生，掌握数据分析技能将是未来职业生涯中的一项重要资产。听完后，我迫不及待地想要在这方面进行深入学习。

埃隆·马斯克

埃隆·马斯克的讲座更是令人振奋。他讲述了自己对未来科技的看法，尤其是在人工智能和太空探索方面的设想。他的远见卓识和创新精神激励着我勇于追求梦想，并不断挑战自我。听完他的演讲，我感受到了一种强烈的使命感，想要为推动科技进步贡献自己的力量。

三、总结与展望

这次实训经历对我来说是一次宝贵的财富。通过大量的实践操作，我不仅掌握了重要的算法技巧，还对计算机科学的前沿领域有了更深刻的理解。更重要的是，这些知名企业和行业领袖的分享让我认识到，技术的进步需要不断的创新和探索精神。

在未来的学习和工作中，我计划继续深化自己的技术能力，并积极关注行业动态，以期能够为推动技术进步贡献自己的一份力量。我相信，只要保持好奇心和持续学习的态度，就能够在计算机科学这个充满机遇的领域中找到属于自己的位置。

总之，这次实训经历对我个人的成长意义重大。每一步的成长都是一次新的开始，我期待着未来的挑战与机遇。