****

课程设计报告书

**题目：迷宫游戏**

**学 院 计算机科学与工程学院**

**专 业 计算机类**

**学生姓名 王思哲**

**学生学号 202330451741**

**指导教师 徐红云**

**课程编号：045101571**

**课程学分 2**

**起始日期 2024年1月11日**

|  |  |
| --- | --- |
| 教  师  评  语 | 教师签名：  日期： |
| 成  绩  评  定 |  |
| 备  注 |  |

**迷宫游戏**

**一、选题背景**

设计一个随机生成迷宫的程序，包括随机的入口和出口，并模拟走迷宫的过程，不论有或没有出去的路径，程序都应显示行走的过程。

**二、方案论证(设计理念)**

迷宫的存储：

使用char类型的二维数组来存储迷宫，其中’.’代表道路，’#’代表墙壁，无法通行，’\*’代表终点，’@’代表已经走过的区域。

迷宫生成算法：

我设计了两种迷宫生成的算法，一种是基于prim算法生成的迷宫地图，一种是利用random库中的rand函数生成的随机数来生成迷宫地图。

其中第一种是基于prim算法生成的迷宫。该算法将每一个迷宫的单元格视为9个独立的格子，并将中间的格子用’.’表示的道路空出来，周围的8个格子用’#’充当墙壁，当随机处理某一格时随机选择上下左右四个方向中没有遍历过的那一格，并打通两格之间的墙壁，即将两格之间的’#’转化成’.’，不断重复该过程，直到没有可以选取的格子。由于这种算法本质上是生成了一棵树，因此根据树上两结点必有通路，只要保证出口和入口在任意一个道路旁就能保证一定存在出去的道路。

第二种是利用random库中的rand函数生成的随机数生成的迷宫。根据传入的道路的权重以及地图尺寸大小，随机生成代表x坐标与y坐标的数字，将数字代表的块用’.’表示空地。多次进行随机，直到达到道路的权重。这种算法生成的迷宫随机性较强，不能保证一定存在出去的道路。

迷宫遍历算法：

我设计了5种迷宫遍历算法，分别是DFS、BFS、A\*、递归、搜索与回溯。此外，还有一种手扶墙的模拟算法，即将右手放在右边的墙上并开始前进，手不离墙，最终总能走到出口或返回入口。但这种算法需要出口与入口都在边界且除出入口外边界都为墙体，且效率较低，不能适配随机数迷宫，因此放弃这种方法。

第一种是DFS，即深度优先搜索，程序会沿着路径的某一条走到不能再走为止，再选择另一条道路直到找到终点。

第二种是BFS，即广度优先搜索，程序会遍历所有可能的路径，在每一条路径上地毯式地向四周扩散直到找到终点。

第三种是A\*，即启发式搜索，程序会寻找距离终点最近的可达的路径点，在此基础上再次寻找，重复此过程直到找到终点。

第四种是递归，原理和DFS相同，不过写法变成了递归函数。

第五种是搜索与回溯，原理与递归相同，但加入了回退这一步。

显示路径算法：

由于找到终点后路径的显示不够醒目，我做了两个处理：一是让’@’符号和’\*’符号呈现不同颜色，二是让程序在完成寻找终点后自动显示出通往终点的道路。也即从终点开始用’\*’替换去往起点道路上的’@’。

**三、过程论述**

迷宫的存储：

直接用char类型的二维数组即可，由于prim算法要求将每个格子都视为3x3的格子，因此迷宫的长度必须是奇数，因此迷宫的长度被设定为了尺寸×2+1，即maze[MAXN\*2+1][MAXN\*2+1]。

迷宫生成算法：

prim算法：因为要从所有可达的路径点中随机选择一个，因此需要对储存的可选节点实现随机访问和数据删除，我选用了STL中的vector，它的erase函数可以删除元素并保持O(n)的时间复杂度并支持随机访问。利用rand函数得到的随机数决定选择节点以及尝试遍历的方向即可。

随机数生成：只需生成指定次数的随机数，将对应坐标设置成道路即可。

迷宫遍历算法：

DFS：使用数组模拟栈，遍历到节点将上下左右没遍历过的节点压入栈，再将栈顶元素弹出，并寻找其附近的节点。

BFS：使用数组模拟队列，遍历到节点将上下左右没遍历过的节点压入队列，再将队首元素弹出，并寻找其附近的节点。

A\*：使用数组模拟小根堆，利用上浮与下沉维护堆。创建一个结构体存储节点的x、y坐标以及与终点的欧几里得距离的平方（公式为(x-x1)^2+(y-y1)^2，其中x1、y1是终点坐标），重载小于运算符，比较欧几里得距离，利用堆的性质选取距离终点最近的节点并将上下左右没有遍历过的点插入堆中，可保证较低的时间复杂度。

递归：利用递归函数，首先判断是否到达终点，若没有则向上下左右四个方向中没有遍历到的启动递归函数，直到所有格都被遍历到或找到终点。

搜索与回溯：与递归原理相同，但是在启动递归函数之后加入一步回溯。由于直接使用搜索与回溯算法效率极低，因此引入二维数组access[MAXN\*2+1][MAXN\*2+1]，用于记录是否已经到达过从而剪枝优化效率。

显示路径算法：

新建一个pair<int,int>类型的二维数组fa[MAXN\*2+1][MAXN\*2+1]，用于存储遍历到该节点的上一个节点坐标，在遍历结束后如果找到终点则新建一个pair<int,int>类型的it变量等于终点的坐标，不断使it=fa[it.first][it.second]，并将it所指向的坐标变为’\*’，直到it等于迷宫的起点。

此外，使用宏定义定义了迷宫最大尺寸、墙壁字符、遍历字符、终点字符、冷却时间等数据使程序可以方便地修改。

**四、结果分析**

通过对不同尺寸的迷宫使用不同遍历方式得到的遍历次数数据，可以发现DFS的效率较为中规中矩，但是会出现距离终点只差一步却拐弯的问题，而BFS效率较低但稳定性较好，A\*取二者所长，效率较高且稳定性强，尤其是在道路权重较高的随机数迷宫中，几乎是走直线到达终点，递归方法与DFS一样，而搜索与回溯效率极低，只适合尺寸较小的迷宫。

**五、课程设计总结**

总结可以包括:课程设计过程的收获、遇到的问题，遇到问题解决问题过程的思考、程序调试能力的思考，课程设计实现过程中的收获和体会等。

在我解决这个题目的途中，遇到的第一个问题就是迷宫生成方式的问题。一开始我的设想是使用随机数迷宫，但考虑到随机数迷宫生成的稳定性较差，不能保证有出口，我对随机数进行了改良，使得道路的权重可调。后来，我通过查阅网上资料等方法学习了基于prim算法生成迷宫的方式，加以思考选用了vector容器来保证随机性与效率。

在解决迷宫遍历方式这一复杂问题的过程中，我充分利用了自我学习积累的数据结构相关知识，并将这些理论知识成功地应用到了实际问题的求解之中。首先，我针对不同的搜索策略，分别采用了堆、栈以及队列这三种基础数据结构来实现深度优先搜索（DFS）、广度优先搜索（BFS）和A\*搜索等三种不同的迷宫遍历方法。

在实现深度优先搜索时，我巧妙地运用了栈这一后进先出（LIFO）的数据结构，模拟了递归过程中的调用栈，确保搜索路径能够沿着迷宫的深度不断延伸；而在实现广度优先搜索时，则转而采用队列这一先进先出（FIFO）的数据结构，保证了搜索过程中始终优先探索与起始点距离最近的节点。

此外，在A\*搜索算法的设计中，我结合了堆这一具有优先级排序功能的数据结构，用于实时更新并存储待搜索节点及其估计代价，以期寻找到从起点到终点的最优路径。同时，为了更好地管理和操作迷宫中的各个节点，我还定义了相应的结构体，涵盖了节点的位置信息及预估代价（欧几里得距离）等多个关键属性。

为了提高代码的可读性和执行效率，我引入了重载运算符，使得在处理结构体对象时能够像操作基本数据类型一样直观便捷。例如，通过重载比较运算符，实现了对节点优先级的高效排序，极大地优化了A\*搜索算法的整体性能。

在这次的课程设计过程中，我深深体验到了自主学习与实践探索的重要性。首先，为了完成课程设计的任务，我广泛地运用了互联网资源进行深度挖掘和细致搜集所需资料，这一过程极大地锻炼了我独立检索信息、筛选有效知识的能力，使我在浩如烟海的信息海洋中找准方向，高效获取所需内容。同时，完成这次课程设计也促使我对数据结构等相关理论知识进行了系统的复习和巩固。在实际操作中，我不断回顾并应用这些理论，将抽象的概念转化为具体的设计实现，这不仅加深了我对数据结构内在逻辑的理解，也提高了我灵活运用知识解决实际问题的能力。更为重要的是，通过全程参与从需求分析、方案设计到编码实现的各个环节，我学会了如何独立构思程序框架，优化算法性能，并针对不同问题选择合适的数据结构来支撑整体设计。这种实战式的训练使我能够更加游刃有余地应对未来可能出现的各种编程挑战。

**参考文献**

[1]兔四．三套简单的迷宫地图生成方案［EB/OL］．https://zhuanlan.zhihu.com/p/27381213．2017-06-15．