第 3 次大作业

索引

数据结构

迷宫信息使用二维数组（char** maze）存放。另外根据 A*算法需要，使用二维数组存放了迷宫点状态信息（Status** status）；使用优先队列（class PriorQueue）存放了开放点列表。优先队列使用链表实现，以值域指向的结点估值函数作为权值，权值小的优先出队。

算法描述

起点等于终点时直接输出该点坐标，结束。以下的算法实现无法找到这种解，故单独列出，虽然正常的迷宫一般不会出现这种情况。

本次大作业使用了据说很快很高端的 A*启发式算法。该算法维护一个开放结点队列，每个结点除坐标外还包含已开销信息 G=该点到出发点的实际曼哈顿距离、估计将开销信息 H=该点到目的点的直线曼哈顿距离、估值函数 F=G+H、前驱结点指针 prev。

搜路时先将起点记为开放，压入开放结点队列。之后每次从队列中弹出一个结点，关闭该节点，依次判断该点相邻的四个点：

是否已关闭（为墙、越界、或被标记为关闭）；

是否为未访问结点；

是否为开放结点。

若该点已关闭，则跳过不予考虑。

若该点尚未被访问，则判断其是否为目标结点。若是，则调用输出函数回溯结点输出结果，结束。否则将其记为开放，压入开放列表待查。

若该点为开放结点，则判断其新 G 值与原 G 值得大小关系。若原 G 值不大于新 G 值，说明新路并不比老路近，则跳过进入下个结点；若原 G 值大于新 G 值，则将该节点的开放结点指针置为新结点地址，进入下个结点。

重复以上过程直到到达终点或队列清空。如果队列清空了仍未到达终点，说明迷宫无解，这时主函数将返回 1。

复杂度分析

该算法的时间复杂度与迷宫具体形状和起点终点位置有关。最坏情况是几乎每个点都被遍历过，这时时间复杂度为 O(mn)。

相比时间复杂度而言，该算法空间复杂度比较固定，由于需要存储点状态信息，需要固定 mn 规模的内存空间，另外还需要与路径长度相关的空间存储路径信息，这部分内存不会超过 O(mn)，因此空间复杂度固定为 O(mn)。