人工智能实验（九）

实验报告

姓名：钟怡宁

学号：17341217

班级：计科八班

专业：计算机科学与技术

**一：实验目的**

掌握理解几种常见的盲目搜索算法和启发式搜索算法，并实现利用一种启发式搜索算法，一种盲目搜索算法解决迷宫路线的寻找问题。

**二：算法原理**

1. **一致代价搜索**

一致代价搜索在每步的代价一致时，相当于宽度优先搜索，所以先介绍一下宽度 优先搜索。宽度优先搜索是一个针对图和树的搜索算法，以树为例来介绍，宽度 优先即按层搜索树中的节点，把当前要扩展的状态的后继状态放在边界的最后， 直到找到路径或者遍历完所有的节点为止。

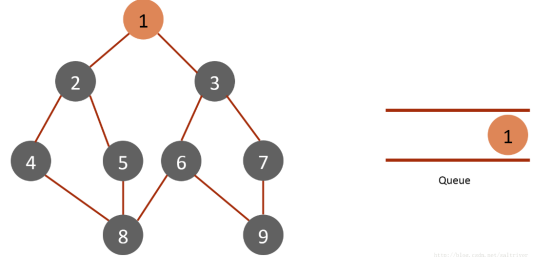
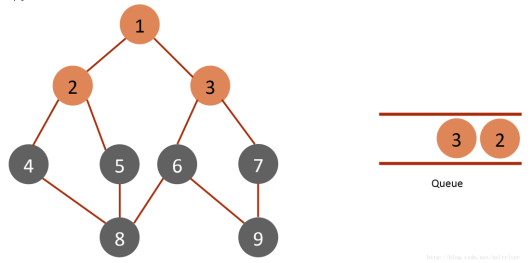
BFS的执行步骤可以归纳为：

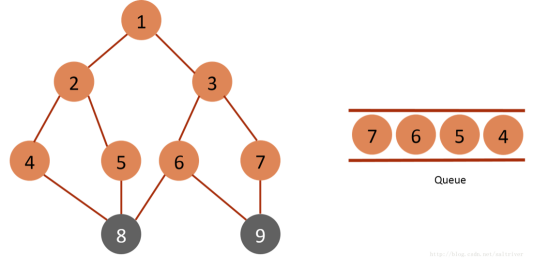
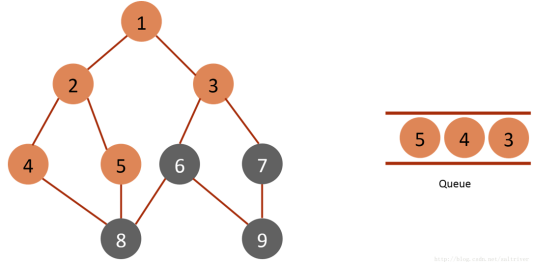
1、把起始父节点放入queue；

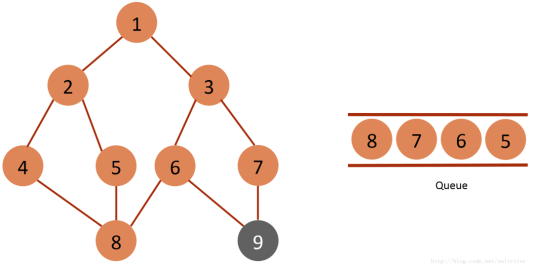
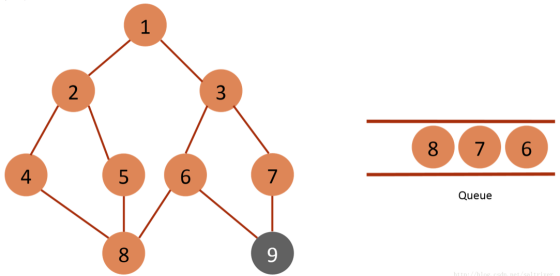
2、重复该步骤，直到queue为空为止：

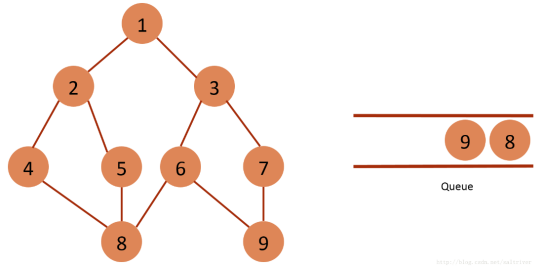
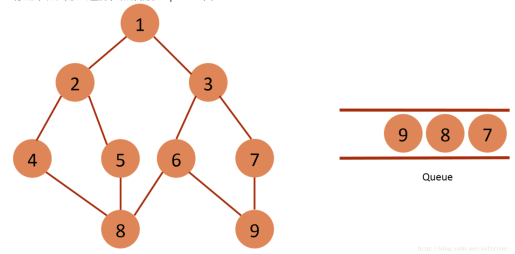
1) 从queue中取出队列头的点；

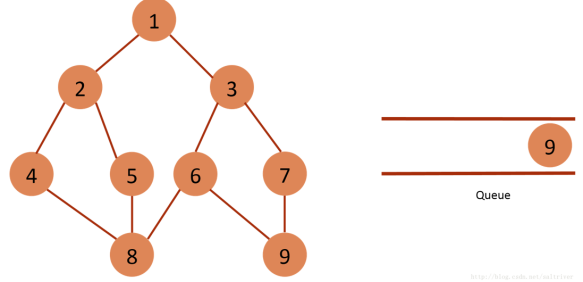
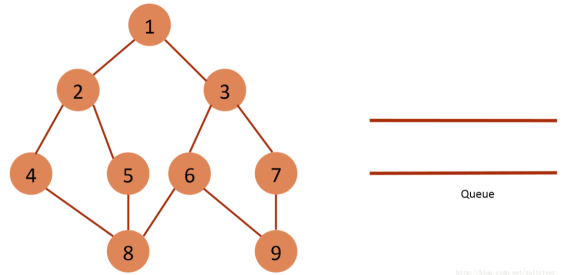
2) 找出与此点且尚未遍历的字节点，进行标记，然后全部放入queue中。

宽度优先搜索算法执行的基本原理如上，图片来源于 （https://blog.csdn.net/qq\_38442065/article/details/81634216）

**一致代价搜索**就是在宽度优先搜索的基础上加多了一些条件限制。即在队列中， 把压入的节点按照路径的成本升序排序，总是扩展成本最低的那条路径。所以在 动作成本一致时，一致代价搜索会变为宽度优先搜索。当动作成本不一致时，在 选择节点进行扩展之前，会计算队列中每个节点的路径成本进行比较，然后选择 路径成本最小的节点进行扩展搜索操作。其余的原理和宽度优先搜索一致。

**一致代价搜索的性能分析**：

假设每个动作的成本>=s>0,在一致代价搜索中，所有成本较低的路径都会在成本 较高的路径之前被扩展，给定成本，该成本的路径数量是有限的，成本小于最优 路径的路径数量也是有限的，最终可以找到最短的路径，所以一致代价搜索具备 完备性和最优性。

设b为问题中一个状态最大的后继状态个数，d为最优解的路径长度，如果扩展节 点在压入队列之后再判断当前扩展节点是否为目标节点，则可知宽度优先搜索的 时间复杂度为，空间复杂度为最后一层 搜的节点的个数的函数级即为，如果再压入队列之前就判 断当前节点是否为目标节点，则时间复杂度和空间复杂度为，原理上差别 不大，如果在压入队列之前就进行判断，原来情况下最后一层的节点扩张就是可 以忽略的，因而时间和空间复杂度会有变化。在此基础上分析一致代价搜索的时 间复杂度和空间复杂度，假设最优路径的成本为C，已知每个动作的成本>=s>0， 则搜索的路径长度最长为，即在一致代价搜索中，d=,则如果扩展节 点在压入队列之后再判断当前扩展节点是否为目标节点，一致代价搜索的时间和 空间复杂度为，如果再压入队列之前就判断当前节点是否为目标节点， 则一致代价搜索的时间复杂度和空间复杂度为。

1. **A\*搜索**

A\*搜索是启发式搜索，在理解的时候可以在一致代价搜索的基础之上进行理解， 会好理解很多。在一致代价搜索中，我们没有考虑边界上的节点哪一个更具有“前 景”，我们总是扩展从初始状态到达当前状态的成本最小的那条路径，没有考虑从 当前状态沿着当前路径到达目标路径的成本。因此引入了启发式搜索，对于一个 具体的问题，构造一个专用于该领域的启发式函数，该函数用于估计从节点 n到达目标节点的成本，对于所有满足目标条件的节点n，，在不同的问 题领域中，的估计有不同的方法。如果只利用启发式函数来对边界队列 中的节点进行排序，希望找到成本最低的解，但是这种做法忽略了从初始状态到 达节点n的成本，可能会选择了离初始状态很远，但根据看起来离目标很近 的节点，从而误入歧途，找不到最优解，具备完备性，但是不具备最优性。

对于A\*搜索，则是定义了评价函数，是从初始节点到达 目标节点的路径成本，是从节点n到达目标节点的成本的启发式估计值，利 用对边界队列中的节点进行排序，然后选择值最小的节点进行扩展， 并更新被访问过的节点的，直到找到目标节点。

**启发式函数设计：**

**可采纳性：**假设是从节点n到目标节点的最优路径成本，（当节点n到目标 节点是不连通的，），当对于所有的节点n，满足。

**一致性（单调性）：**假设，每个状态转移的成本是非负的。对 于任意节点，满足

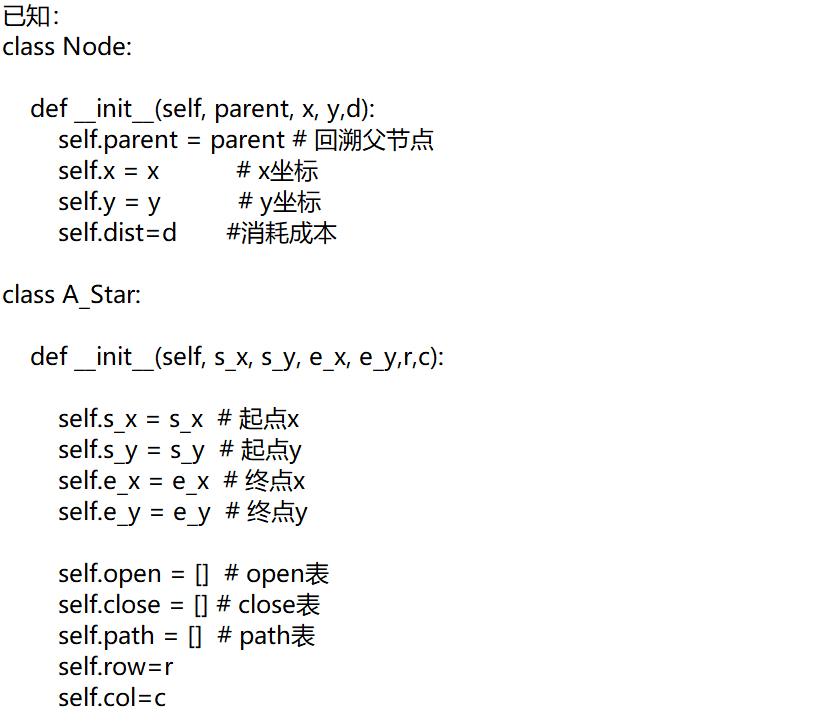
**A\*搜索性能分析：**

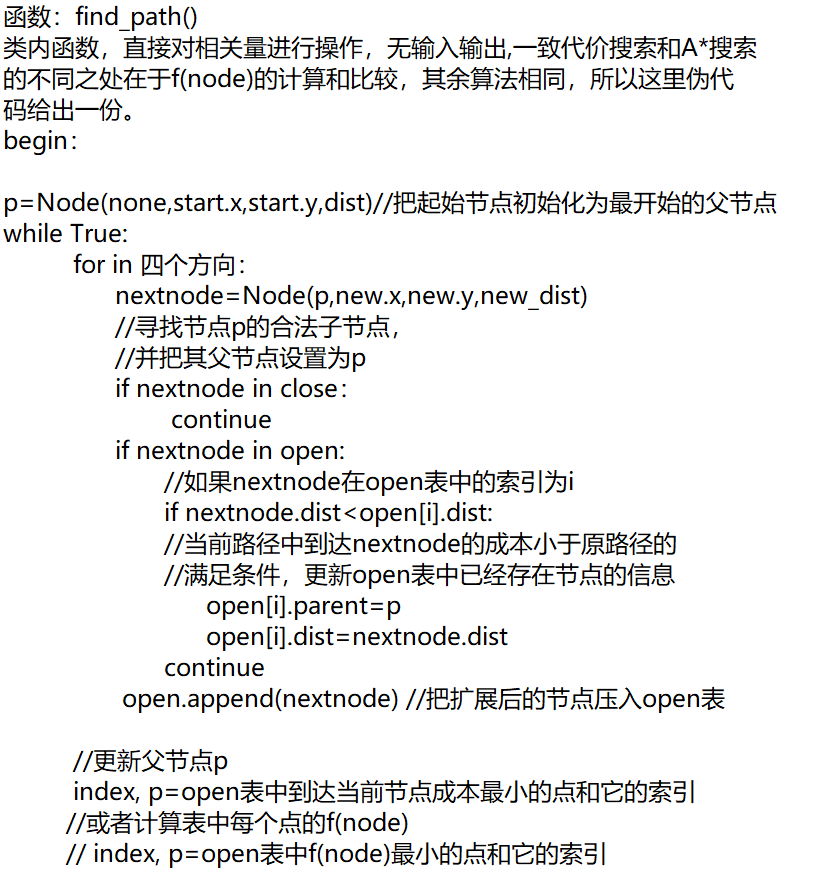
A\*搜索可以看作是宽度优先搜索加上了启发式信息，具备完备性，最优解一定会 在所有成本大于最优解路径成本的路径之前被扩展到，因此A\*搜索具备最优性， 可采纳性意味着最优性。当时，A\*搜索会变成一致代价搜索，因此一致 代价搜索的时间和空间复杂度下限也适用于A\*搜索，因此A\*搜索仍可能时指数 复杂度，除非找到好的函数。

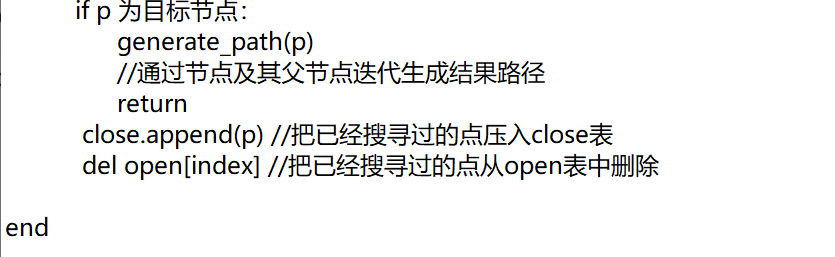
1. **算法实现注意点**

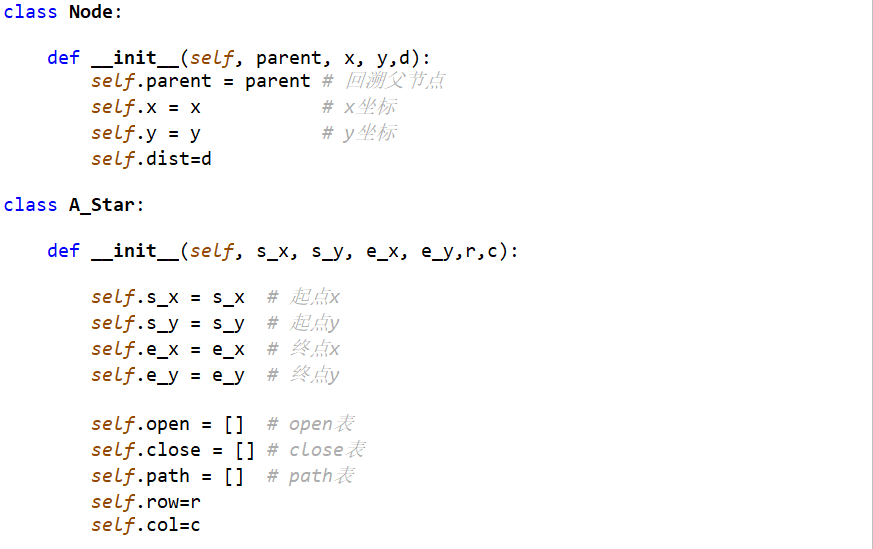
一致代价搜索和A\*搜索在实现时要维护两个表，open表和close表，open表内存 放每次扩展的点，close表内存放每次选中的用来扩展节点的点，表示该节点不需 要再次搜查，并把该节点从open表中删除。

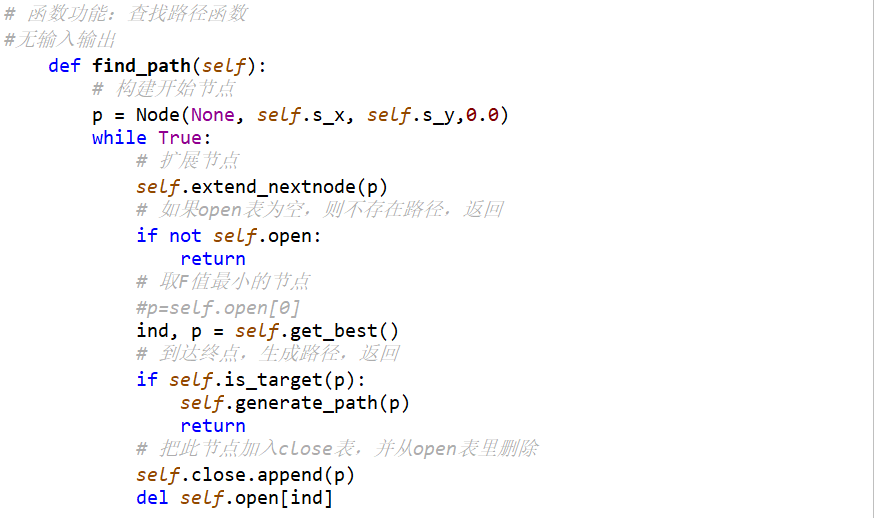
**三：伪代码展示**

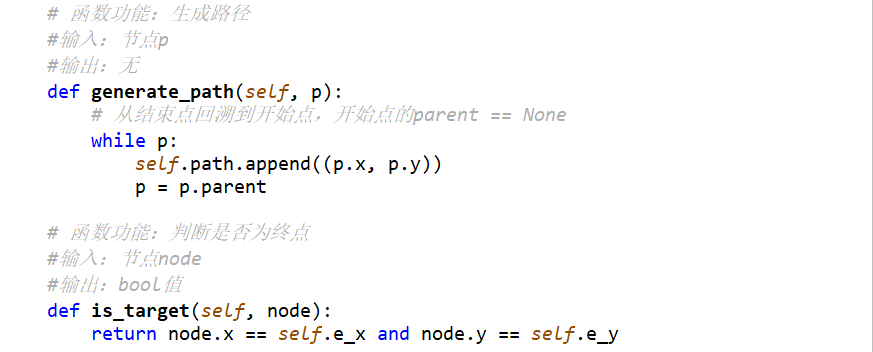


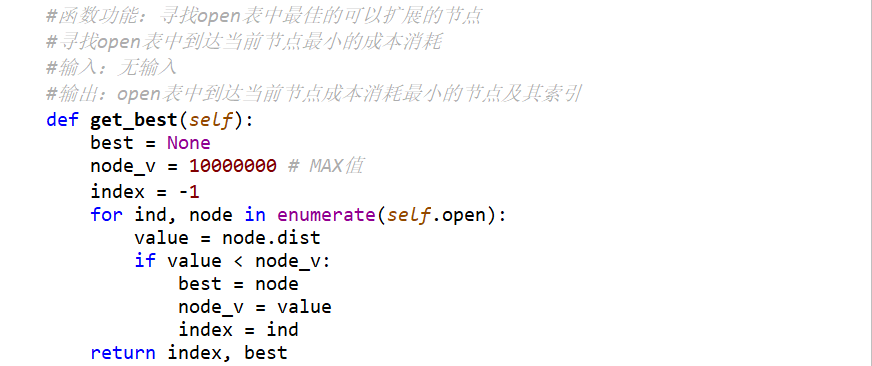


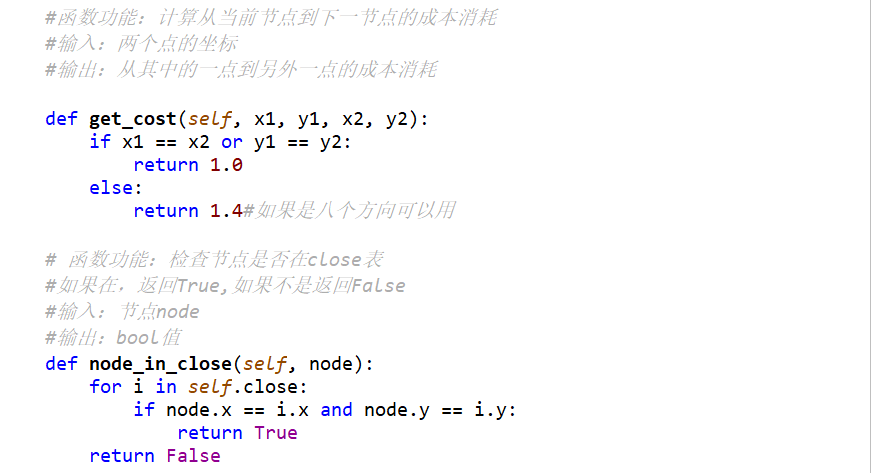


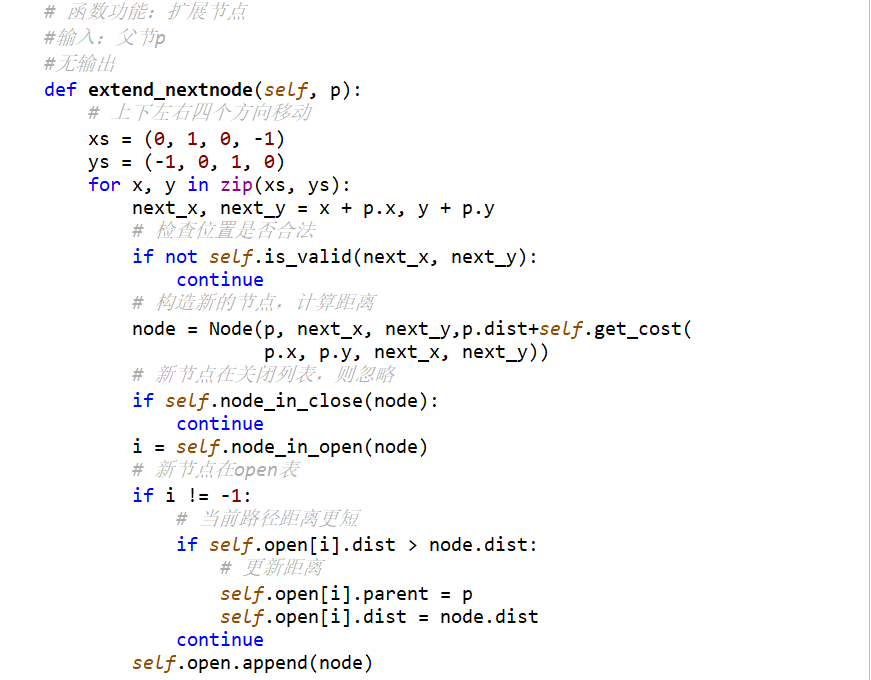
**四：代码展示：一致代价搜索** 





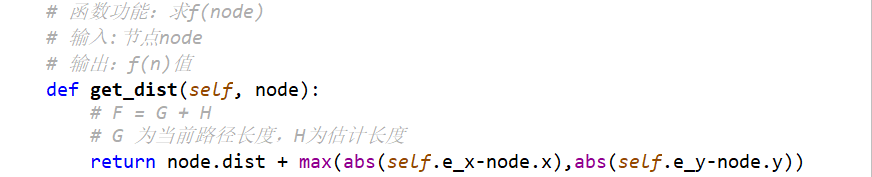


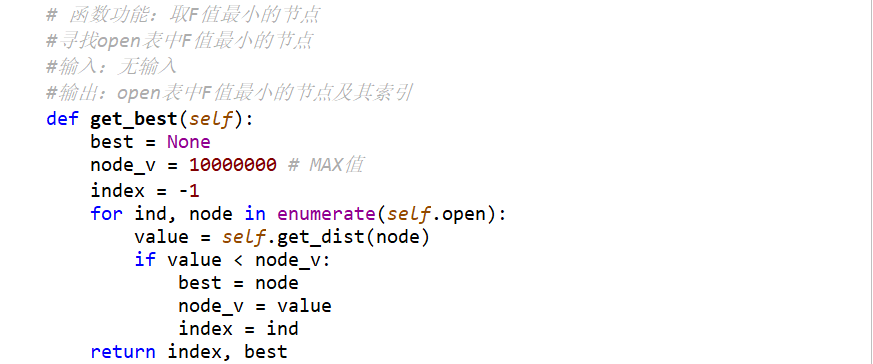


**代码展示：A\*搜索**

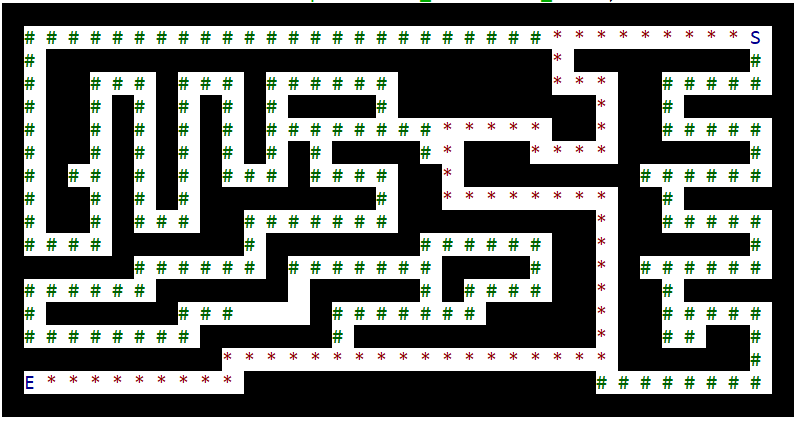
**这里只展示和一致代价搜索不一样的函数：**





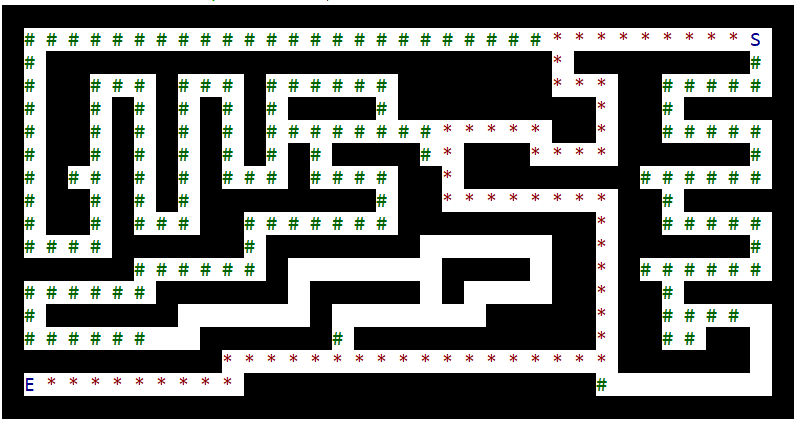
**五：结果分析**

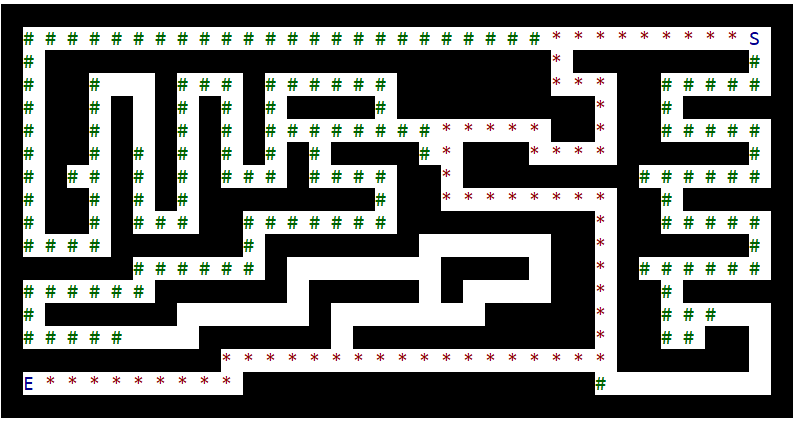
**一致代价搜索的结果图：其中绿色的#号表示搜过的点，红的的\*号表示最终的路径。白色部分表示可以走的路径，黑色部分表示墙壁不能通过。**



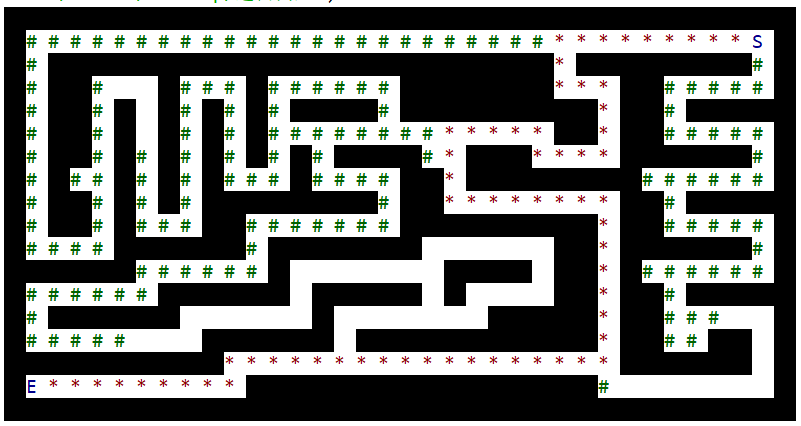
可以看出，一致代价搜索可以找出一条通往目标的节点，所以一致代价搜索具有完备性，一致代价搜索本质上是宽度优先搜索的变形，所以最优路径一定会在搜索其他路径之前被搜索到，一致代价搜索是最优的，根据图中所有可能存在的路径的比对，搜索出来的路径是最短的，具有最优性。

**A\*搜索的结果图如下：其中绿色的#号表示搜过的点，红的的\*号表示最终的路径。白色部分表示可以走的路径，黑色部分表示墙壁不能通过。**

1. 切比雪夫距离
2. 曼哈顿距离



1. 欧式距离



可以看出A\*搜索可以搜索出一条路径，所以A\*搜索具备完备性，搜索出来的路径也是所有可能路径中最短的，A\*搜索具有最优性。

**一致代价搜索和A\*搜索的时间复杂度和空间复杂度对比分析：**

从一致代价搜索的结果图可以看出，基本上所有的节点都被搜索过，因为在现在的条件下，一致代价搜索退化成了宽度优先搜索，所以时间复杂度和空间复杂度上都是，

理论上在找到目标节点之前，长度为d的路径应该都会被搜索到，所以基本上所有的点都会被找到。在A\*搜索中，会优先搜寻f(n)值最小的节点的后继节点，所以会有部分节点不会被搜寻到，会在一定程度上减小时间复杂度和空间复杂度，如上面的A\*搜索的结果图一样，会有部分节点不会被搜索，因为可以确定那条路径不会是最优的。但是A\*搜索的时间和空间复杂度仍然是指数级的。通过对比A\*搜索的三种h(n)的设计方式，可以看出不同的h(n)对结果是有影响的，以曼哈顿距离和欧式距离为参考的h(n)设计，需要遍历的点要比以切比雪夫距离为参考的h(n)设计需要遍历的点少，所以一个好的启发式函数设计可以降低时间和空间复杂度。虽然在本实验例子中，降低的效果并不是非常明显，但是还是可以看出是有效果的。

**六：思考题**

**答：盲目搜索的优缺点：**

**宽度优先算法**具有完备性和最优性，但由于空间复杂度是指数级的，内存需求会 限制算法的运行。

**深度优先算法**在状态空间优先而且无重复路径的情况下具备完备性，不具备最优 性，但是空间复杂度是线性的。

**一致代价搜索**具备完备性和最优性，但是时间和空间复杂度仍是线性的，而且没 有考虑当前节点的“前景”问题。

**深度受限搜索**，解决了可能会运行很长时间的问题，不具备完备性和最优性。

**迭代加深搜索**解决了宽度优先搜索和深度优先搜索的存在的问题，具备完备性和 最优性。

**双向搜索**，具备完备性和最优性，时间和空间复杂度为指数级。

**总的来说：**是一种无信息搜索，通常按照预定的搜索策略进行搜索，简单易于实 现，可以解决一些比较简单的问题，不会考虑问题本身的性质和当前状态的“前 景”问题。效率不高，不适用于复杂问题的求解。

**盲目搜索适用的场景：**

我觉得盲目搜索适用于当前状态的下一状态无法确定但有限，而且比较简单的问 题，像找最短路线，遍历查询问题等。

**启发式搜索的优缺点：**

可能具有完备性和最优性，空间和时间复杂度可能为指数级，如果启发式函数设 计的很好，会降低时间和空间复杂度。

有信息搜索，利用已有的启发式信息来引导搜索过程，可以减少搜索范围，降低 问题复杂度，缺点是：启发式函数的设计并不容易。一般说来，启发信息越强， 扩展的无用节点就越少。引入强的启发信息，有可能大大降低搜索工作量，但不 能保证找到最佳路径。因此，在实际应用中，最好能引入降低搜索工作量的启发 函数而又能找到最佳路径。

**启发式搜索适用的场景：**

我觉得启发式搜索适用于当前状态的下一状态有限且可估计，问题相对复杂一点 的场景，像八数码问题，迷宫问题，可以在一定程度上提高算法的性能问题。

**七：参考资料**

1. **实验课PPT，原理课PPT**
2. **https://blog.csdn.net/diamonjoy\_zone/article/details/65630144**

**https://blog.csdn.net/qq\_38442065/article/details/81634216**