**华东师范大学数据科学与工程学院实验报告**

|  |  |  |
| --- | --- | --- |
| **课程名称**：人工智能 | **年级**：2019 | **上机实践成绩**： |
| **指导教师**：李翔 | **姓名**：张雯怡 |  |
| **上机实践名称**：A\*算法 | **学号**：10195501425 | **上机实践日期**： |
| **上机实践编号**：实验2 | **组号**： | **上机实践时间**：2022.4.5 |

1. **实验目的**

编程完成两道算法题，理解并掌握A\*算法。

1. **实验任务**

**Q1：小明玩球**

题目描述：⼩明在⼀个九宫格中随机摆了⼋个球，每个球上标有1-8中的某⼀数字（球上数字不重复）。九宫格中留有⼀个空格，该空格⽤0表示。空格周围的球可以移动到空格中。现在，给出⼀种初始布局（即初始状态）和⽬标布局（本题的⽬标布局设为123804765），现在⼩明想找到⼀种最少步骤的移动⽅法，实现从初始布局到⽬标布局的转变，你能帮帮他吗？

**Q2：爱跑步的小明**

题目描述：众所周知，⼩明身材很好。但⾃从他博⼠毕业当⽼师后，他就⾃我感觉身体变差了，于是他就想锻炼了。为了不使⾃⼰太累，他提出⼀种从⼭顶跑步到⼭脚的锻炼⽅法。

千寻万觅，终于在郊区找到这样⼀座⼭。这座⼭有N个地标，有先⾏者在这些地标之间开辟了M条道路。并且这些地标按照海拔从低到⾼进⾏了编号，例如⼭脚是1，⼭顶是N。

⼩明这个⼈对跑步的⽅式很挑：

（1）只跑最短路径。但⼀条最短路径跑久了会烦，需要帮他设计K条最短路径。

（2）不想太累，每次选道路的时候只从(海拔的)⾼处到低处。

现在问题来了，给你⼀份这座⼭地标间道路的列表，每条道路⽤(Xi, Yi, Di)表示，表示地标Xi和地标Yi之间有⼀条⻓度为Di的下坡道路。你来计算下⼩明这K条路径的对应⻓度，看看⼩明的锻炼强度⼤不⼤？

**Q1和Q2均要求只能⽤A\*算法。**

1. **使用环境**

Q1:vs code python Q2:Dev C++ C++

1. **实验过程**

**A\*算法的理解：A\*算法区别于dfs、bfs等盲目搜索的区别在于A\*使用了启发式搜索，就是说给定了一个估价函数f(x) = g(x) + h(x)。其中，x为待评价的节点，假设初始状态或位置为s，目标状态或位置为t，g(x)为从s到x的最佳路径的cost；h(x)为从x到目标t的最佳路径cost的估计值，称为启发函数或是估价函数；而f(n)为从初始节点s经过节点n到达目标节点t的最佳路径cost的估计值。如果启发函数满足 h(x) <= h\*(x)，则可以证明当问题有解时，A\*算法一定可以得到一个cost最小的结果。**

然后用A\*算法来解决上面的两个问题。

**Q1：小明玩球**

**启发h(x)选择：**

对于这个问题能想到两种启发，第一个就是初始状态的九宫格和目标状态九宫格中位置不匹配的数字个数，第二个就是不在位的数字归位所需的最短距离和，由于这里数字块的移动只能上、下、左、右四个方向，距离的计算一定是采用曼哈顿距离。

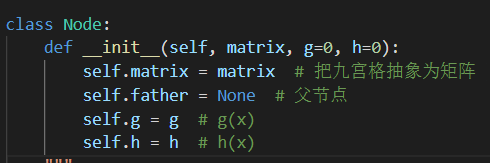
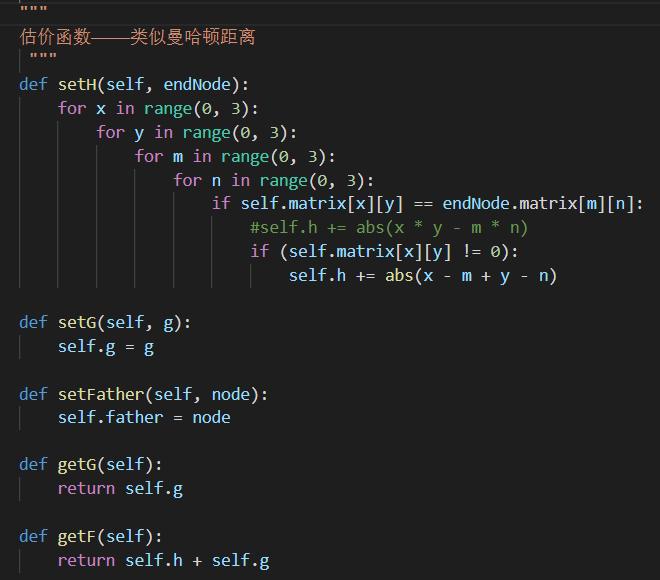
显然第一种启发过于简单，效果肯定没有第二种好。因此启发h选用参考曼哈顿距离。

但是这里h(x)不是直接使用曼哈顿距离 |x1-x2| + |y1-y2| ，而是 |x1-x2+y1-y2| ，这个地方后面再作详细说明。

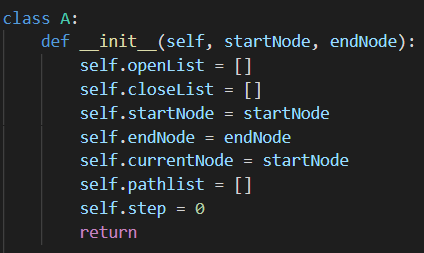
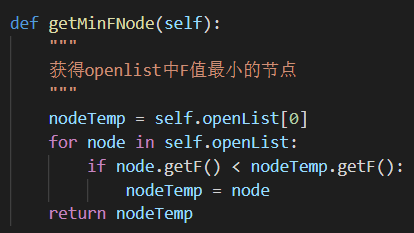
**程序设计过程：**

首先把九宫格及其状态抽象，每个状态节点有它的g、h和前继/父节点。

其中g值使用从初始到当前状态节点的steps，启发h使用过一个类似曼哈顿距离的函数。因为f = g + h ，就不在类内添加变量了直接写成函数。

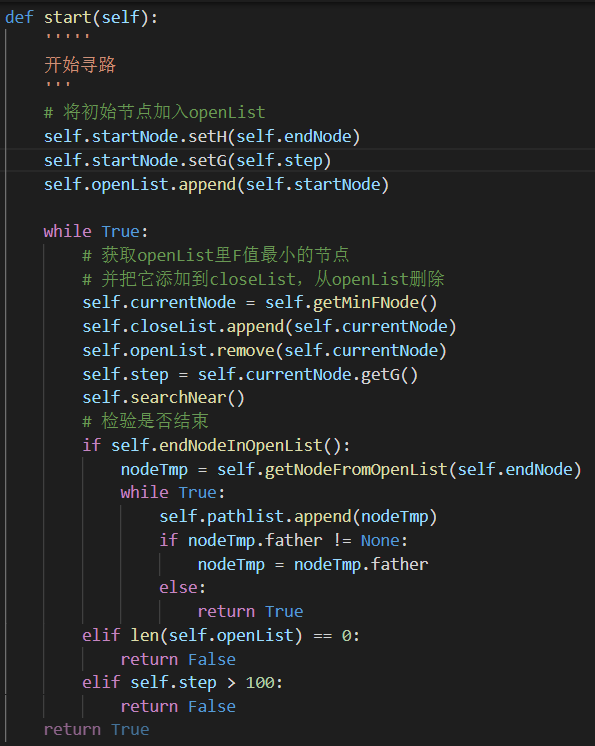
 

然后设置一个变量openList用于存放那些搜索图上的叶节点，就是已生成但为扩展的节点；变量closeList用于存放图中的非叶节点，也就是已经被生成和被扩展的节点；

取openList中f最小的节点node进行扩展，如果node是目标节点，则找到一个解，算法结束，否则扩展node；

对于node的子节点m，如果m既不在openList也不在closeList，则将m加入openList；如果 m 在closeList，说明从初始节点到m有两条路径，如果新路径的cost更小，则将m（新找到的）从closeList中删除并取出放入openList中；

重复前面所述步骤，直至找到一个解结束；或openList为空，即无解。

**最终运行结果：**

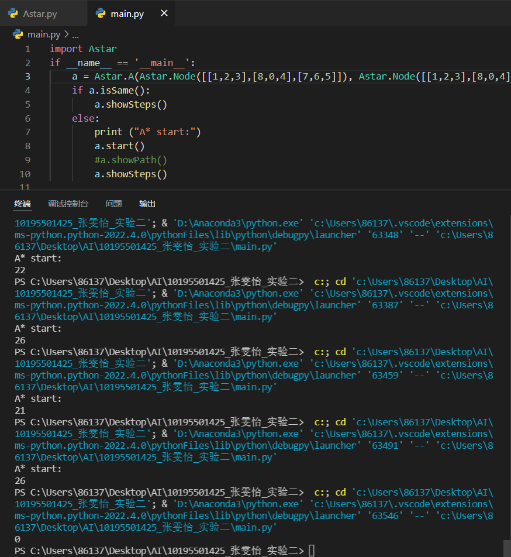
输入1：024657318 输出1：22

输入2：587346120 输出2：26（跑了比较久，大概1min）

输入3：375148206 输出3：21

输入4：512768340 输出4：26（大概30s）

输入5：123804765 输出5：0



**Q2：爱跑步的小明**

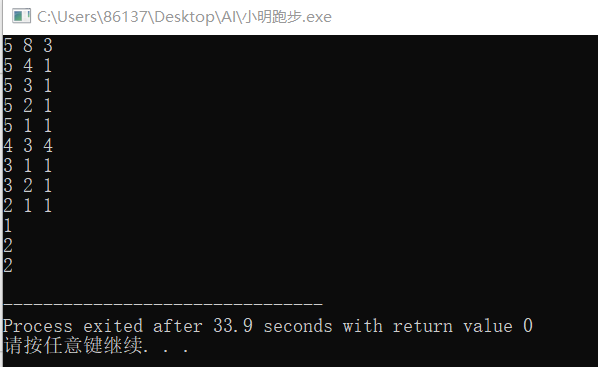
这道题就是使用A\*算法解决k条最短路径问题。

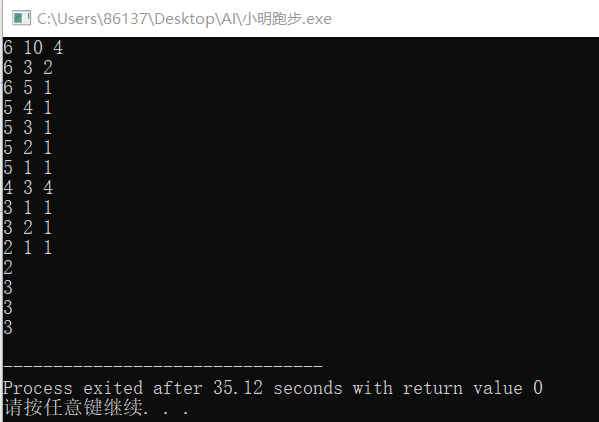
**启发h(x)选择：从n到当前点走过的距离 + 从当前点到终点的最短路径。**

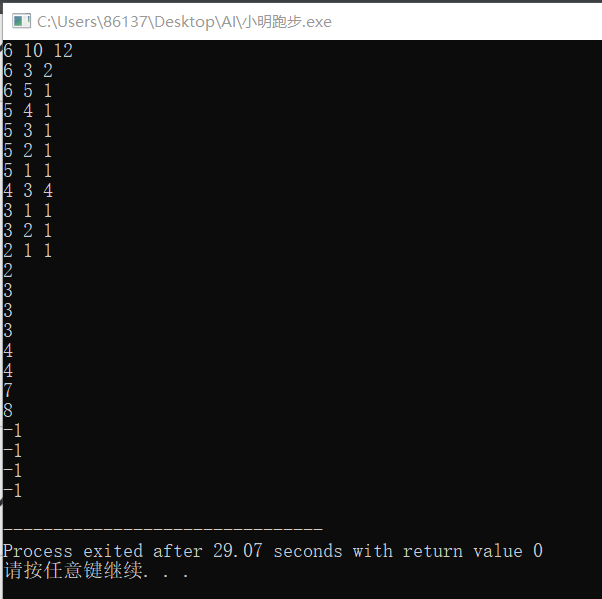
**程序设计过程：**

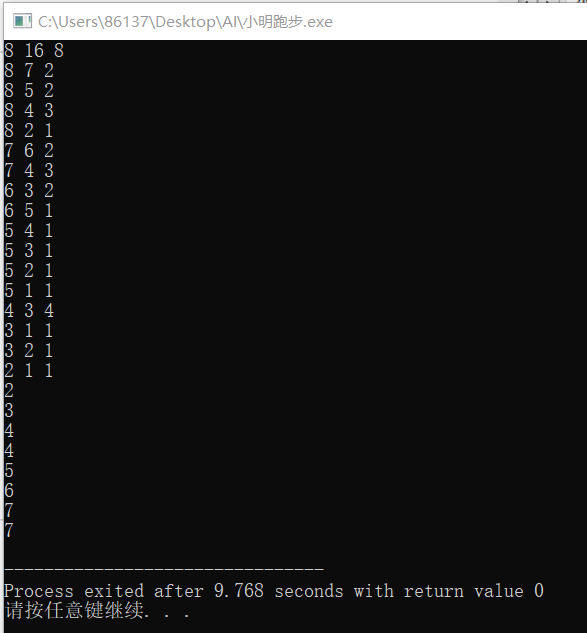
先建反向边求出每个点到山脚1的最短路径，对于每条路径的估价函数定义为这个点到n走过的距离加上这个点到终点的最短路，每次从堆中取出最小值，直至走到山脚1时给出路径答案。

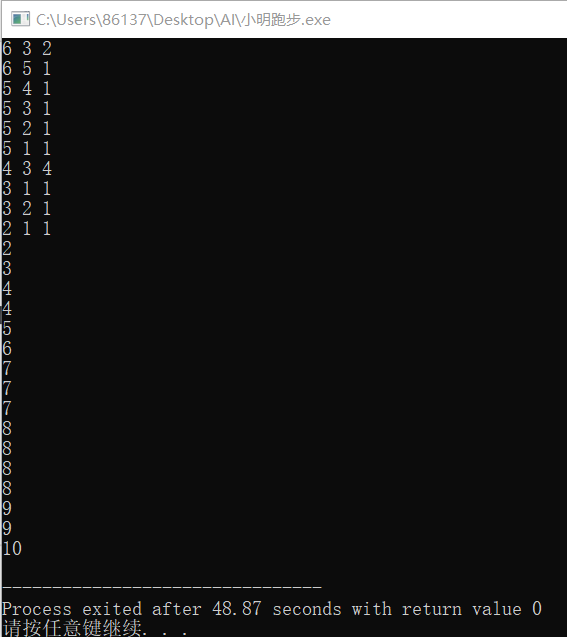
**程序运行结果：**

输出1：1 2 2 

输出2：2 3 3 3 

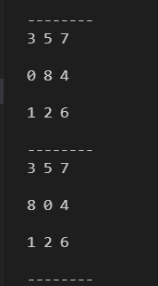
输出3：2 3 3 3 4 4 7 8 -1 -1 -1 -1 

输出4：2 3 4 45 6 7 7 

输出5：2 3 4 4 5 6 7 7 7 8 8 8 8 9 9 10 

1. **总结**

A\*算法以启发式搜索区别于与之有点类似的盲搜中的dfs，其中启发函数h(x)的选择对整个算法的效率有决定性的作用。

解决小明玩球的问题时我在启发函数的选择上踩了坑，一开始直接想当然地使用了曼哈顿距离作为启发，然而在测试用例中第二个输入跑出来的结果总是28，而别的同学几乎都是26，确定了后面代码逻辑没有问题之后嫌疑就到了启发函数h(x)上。后面在和同学讨论中我们发现了这样一个问题：输出用例2的路径，发现在这一步我的算法做出了不同的决策。是因为将矩阵拉伸成向量后，对于一个数字要移动到另一个位置上，使用曼哈顿距离其评分是一样的，但实际执行策略有顺时针和逆时针移动，两种决策之间正好相差2steps。一种正确的类似曼哈顿距离的启发函数是|x1-x2+y1-y2|，可以避免两个策略估值相同。