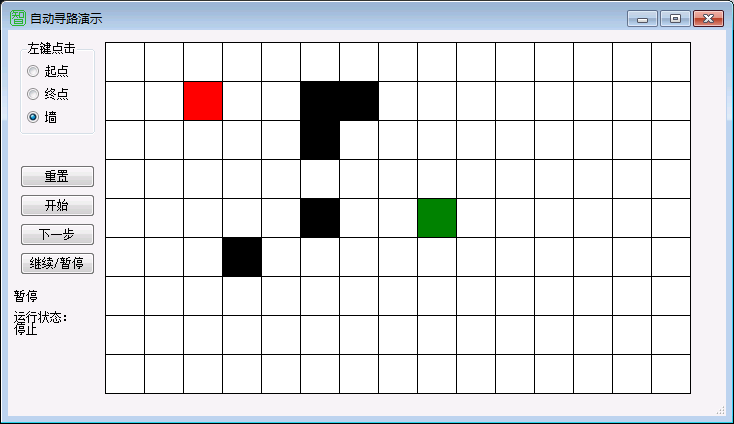
A\*算法仿真实验

请下载并安装附件（智能搜索算法教学软件.rar）里的智能搜索算法教学实验系统，然后点击A\*算法进行仿真实验。

实验要求如下：

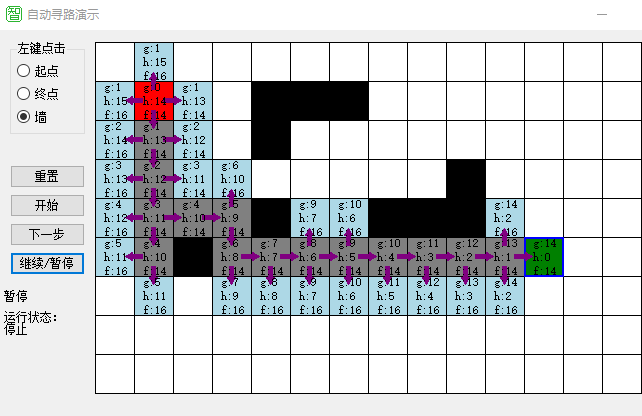
1. **单击"A\*算法介绍"，回顾A\*算法的基本原理。**
2. **在"A\*算法演示程序"中，选择"自动寻路问题演示"进行仿真实验**：

2.1设置起点、终点和墙：选中“起点”并单击某一方格可设置起点，选中“终点”并单击某一方格可设置终点，选中“墙”并单击若干个方格可设置墙，若单击“重置”则清除所有的设置。



2.2 运行A\*算法：单击“开始”，可以看到起点的实际代价g（搜索深度，即搜索步数）、估计代价h（起点到终点的哈密尔顿距离，即起点到终点的横向和纵向的方格数之和）和估价函数值f（f=g+h），然后依次单击若干次“下一步”后，可以看到有深蓝色边框的方格为当前正扩展的状态节点，天蓝色的方格为open表中待扩展的状态节点，灰色的方格为放入closed表的已扩展的状态节点，一直到搜索到终点为止，或者单击“继续”直接搜索到终点。如果单击“重置”则重新按照2.1和2.2重新进行实验。

2.3 结果记录：请拍照上传搜索到终点的实验结果图，并记录A\*算法扩展的节点数，分析起点和终点的实际代价g或估计代价h有什么特点，以及搜索过程中估价函数值f有什么特点？



A\*算法扩展的节点数：14

起点和终点的实际代价g或估计代价h的特点：起点的g与终点的h相等，起点的h与终点的g相等

搜索过程中估价函数值f的特点：f=g+h

3**.在"A\*算法演示程序"中，选择"8数码问题演示"进行实验**：

3.1观察8数码问题的起始状态和目标状态。以起始状态为例，在3\*3的方格里有0到8共9个数字，其中0表示空格，规则是0周围的数字可移到0所在的位置，这样相应的操作可以定义为空格的移动，即空格不在第1行，就可以上移，不在第3行就可以下移，不在第1列就可以左移，不在第3列就可以右移。

3.2运行A\*算法：单击“开始”，观察加入open表中的第一个状态为起始状态，其中启发函数h为该状态和目标状态不相同的位数，即不在位数。单击“下一步”，可看到open表的第一个状态从open表弹出，加入closed表，并扩展生成一些新状态，再单击“下一步”，新状态加入open表。重复上述步骤可以观察A\*算法求解8数码问题的搜索过程，或者单击“自动”，直至搜索到目标状态为主。



3.3结果记录：请拍照上传搜索到目标状态后的实验结果图，并记录A\*算法扩展的节点数和生成的节点数，分析起始状态和目标状态的实际代价g或估计代价h的特点，以及搜索过程中估价函数值f的特点。



A\*算法扩展的节点数和生成的节点数：6 11

起始状态和目标状态的实际代价g或估计代价h的特点：起始状态的g与目标状态的h相等，起始状态的h与目标状态的g相等

搜索过程中估价函数值f的特点：f=g+h

1. **在"A\*算法验证程序"中，选择"8数码问题"进行实验：**
   1. 判断问题是否可解：单击“随机产生”起始状态和“随机产生”目标状态，然后单击“计算”，如果无解，重复上述步骤。



* 1. 结果记录:请拍照上传实验结果图，并比较两种盲目搜索（深度优先搜索和宽度优先搜索）和不同启发策略的A\*算法求解结果，分析为什么A\*算法和宽度优先搜索都能找到最优解，即搜索步数都是最少，但它的性能，包括访问结点数和耗时都比宽度优先搜索要少？为什么A\*算法采用哈密尔顿距离作为启发函数比不在位数为启发函数的性能要好?



1. 因为A\*搜索算法中采用单调性启发函数，可以减少比较代价和调整路径的工作量，从而减少搜索代价
2. 因为采用哈密尔顿距离作为启发函数搜索的状态较少

实验心得：

启发式搜索算法使得搜索的效率成倍提高。而不同的启发式搜索算法差异也较大。总之启发式搜索算法是由h(n)决定的，好的估价函数将决定算法性能的好坏。

通过这次实验，使我对启发式搜索算法有了更进一步的理解，特别是估计函数h(n)所起到的巨大重用。一个好的估计函数对于启发式搜索算法来说是十分关键的。