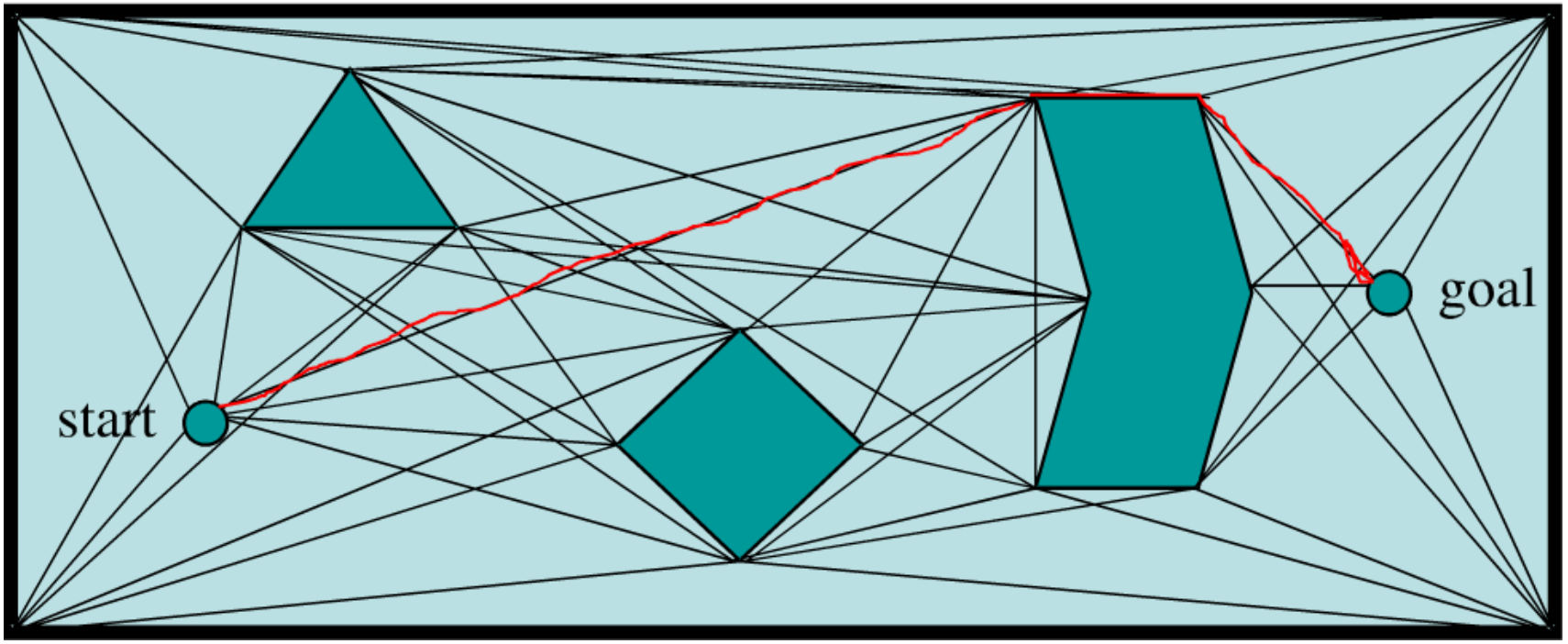
第2次大作业:截止日期:12月26日,可组队(1-4人)

1. 社会力模型的实现：给出一个典型的场景，比如单房间疏散或对流(过人行横道), 实现图形化仿真，撰写研究报告；

2. **选做——**神经网络模型的实现：根据所给数据集，实现单房间疏散或对流(过人行横道)的行人仿真,实现图形化仿真，撰写研究报告。

一、路径规划问题



路径规划

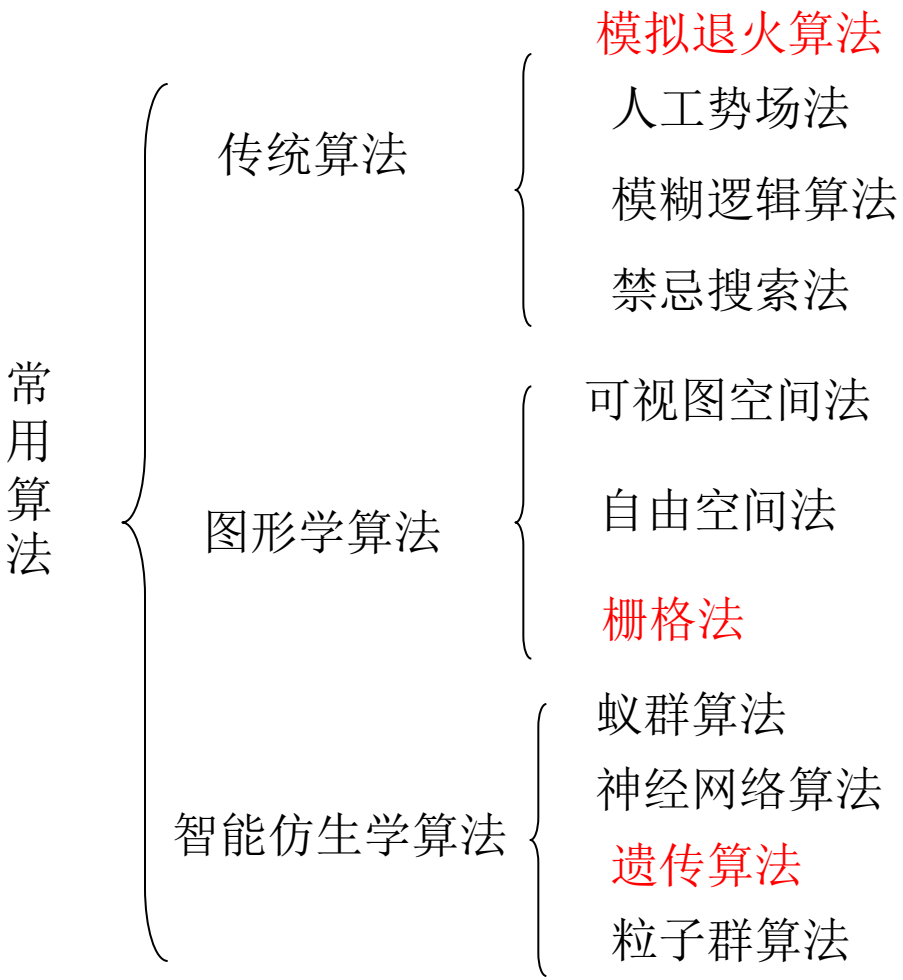
路径规划：是指在具有障碍物的环境中，按照一定的评价标准，寻找一条从起始状态到目标状态的无碰撞路径。

应用：路径规划技术在很多领域都具有广泛的应用，在高新科技领域的应用有：

- 机器人的自主无碰运动；
- 无人机的壁障突防飞行；
- 巡航导弹躲避雷达搜索；
- GPS导航，城市路网规划导航等.

路径规划的核心

- 路径规划的核心就是算法的设计



路径算法问题的分类

基于先验完全信息全局路径规划 —— 静态，离线规划

基于传感器信息的局部路径规划 —— 动态，在线规划

路径算法问题的分类

- 路径搜索是应用相应算法寻找一条行走路径，使预定的性能函数获得最优值。

- 本问题属于静态路径规划，即全局路径规划。我们采用图形法，栅格法来研究。其性能函数定义为路径最短。

二、算法设计思路

- 首先利用栅格法，即栅格分解图将移动机器人所在环境分割成规则且均匀的栅格，每个栅格只可能有两种状态：占据或自由，对应着占据栅格和自由栅格。栅格的尺寸通常同移动机器人的尺寸和步长一致。

- 采用A*算法作为搜索策略。在这里就要用到启发式搜索启发中的估价是用估价函数表示的,f(n) = g(n) + h(n)。
- g(n)是在状态空间中从初始节点到n节点的实际代价，
- h(n)是从n到目标节点最佳路径的估计代价。我们这里将其设为n到目标节点的直线距离。

二、算法设计思路

- **A* (A-Star)算法是一种静态路网中求解最短路最有效的方法。**
- **公式表示为 *f(n) = g(n) + h(n)***
- **其中 ：**
- **f(n) 是从初始点经由节点n到目标点的估价函数，**
- **g(n) 是在状态空间中从初始节点到n节点的实际代价，**
- **h(n) 是从n到目标节点最佳路径的估计代价。**
- **保证找到最短路径（最优解的）条件，关键在于估价函数h(n)的选取 .**

常见估价函数

（1）曼哈顿距离：

就是在欧几里德空间的固定直角坐标系上两点所形成的线段对轴产生的投影的距离总和，例如在平面上，坐标（x1,y1）的点P1与坐标（x2, y2）的点P2的曼哈顿距离为：|x1 - x2| + |y1 - y2|。

（2）欧氏距离：

它是在m维空间中两个点之间的真实距离。在二维和三维空间中的欧氏距离的就是两点之间的距离。例如在平面上，坐标（x1,y1）的点P1与坐标（x2, y2）的点P2的欧氏距离为：sqrt((x1-x2)^2+(y1-y2)^2）。

（3）切比雪夫距离：

是两个向量之间各分量差值的最大值。例如在平面上，坐标（x1, y1）的点P1与坐标（x2, y2）的点P2的切比雪夫距离为：max(|x1 - x2| , |y1 - y2|)

算法实现思路

- **Begin**

  读入初始状态和目标状态;

  把初始状态加入open表中

  while（未找到解&&状态表非空）

① 在open表中找到评价值最小的节点，作为当前结点，并加入到closed表中;

② 判断当前结点状态和目标状态是否一致，若一致，跳出循环；否则跳转到③;

③ 对当前结点，分别按照上、下、左、右，左上、左下，右上，右下8个方向移动空格位置来扩展新的状态结点，并计算新扩展结点的评价值f并记录其父节点;

④ 对于新扩展的状态结点，判断其是否与open表和close表中的数据有重复，若不重复，把其加入到open表中;if 重复，说明此路不通，需要换一个起点;

  End while

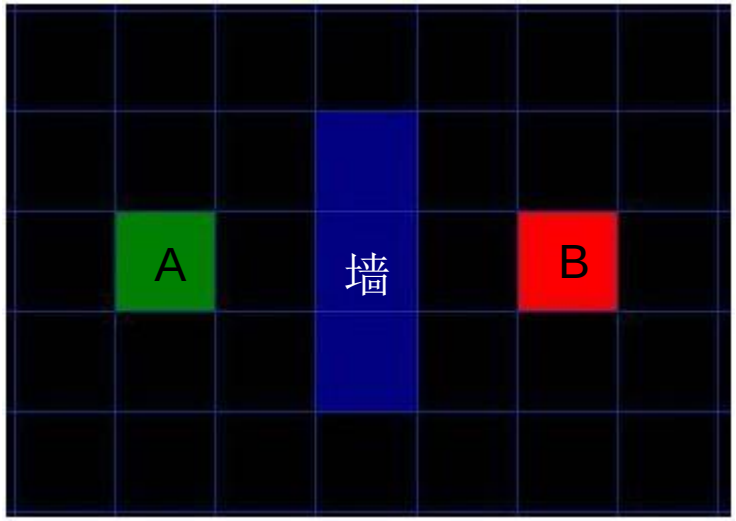
  输出结果;

- **End**

示例——搜索区域（1/5）

假设有人想从A点移动到一墙之隔的B点，如下图，绿色的是起点A，红色是终点B，蓝色方块是中间的墙.搜索区域被我们划分成了方形网格。

简化搜索区域，是寻路的第一步。这一方法把搜索区域简化成了一个二维数组。数组的每一个元素是网格的一个方块，方块被标记为可通过的和不可通过的。路径被描述为从A到B我们经过的方块的集合。一旦路径被找到，我们的人就从一个方格的中心走向另一个，直到到达目的地。



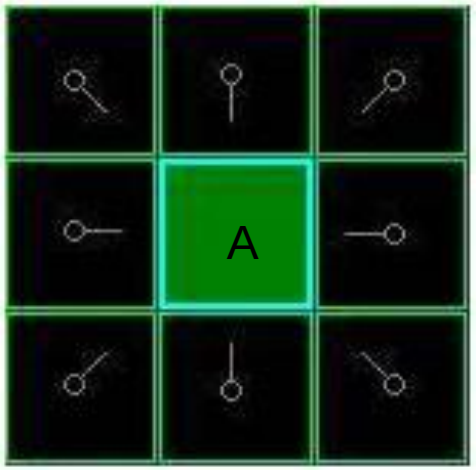
示例——开始搜索（2/5）

1，从点A开始，并且把它作为待处理点存入一个“开启列表”。你的路径可能会通过它包含的方格，也可能不会。基本上，这是一个待检查方格的列表。

2，寻找起点周围所有可到达或者可通过的方格，跳过有墙，水，或其他无法通过地形的方格。也把他们加入开启列表。为所有这些方格保存点A作为“父方格”。当我们想描述路径的时候，父方格的资料是十分重要的。

3，从开启列表中删除点A，把它加入到一个“关闭列表”，列表中保存所有不需要再次检查的方格。在这一点，你应该形成如图的结构。在图中，暗绿色方格是你起始方格的中心。它被用浅蓝色描边，以表示它被加入到关闭列表中了。所有的相邻格现在都在开启列表中，它们被用浅绿色描边。每个方格都有一个灰色指针反指他们的父方格，也就是开始的方格。

接着，我们选择开启列表中那个F值最低的的临近方格，大致重复前面的过程。



示例——路径评分1（3/5）

选择路径中经过哪个方格的关键是下面这个等式：F = G + H，这里：

G = 从起点A，沿着产生的路径，移动到网格上指定方格的移动耗费。

H = 从网格上那个方格移动到终点B的预估移动耗费。这经常被称为启发式的。

我们的路径是通过反复遍历开启列表并且选择具有最低F值的方格来生成的。

G表示沿路径从起点到当前点的移动耗费。此例，我们令水平或者垂直移动的耗费为10，对角线方向耗费为14。我们取这些值是因为沿对角线的距离是沿水平或垂直移动耗费的的根号2，或者约1.414倍。

H值此处使用曼哈顿方法，它计算从当前格到目的格之间水平和垂直的方格的数量总和，忽略对角线方向，然后把结果乘以10。这是对剩余距离的一个估算，而非实际值，这也是这一方法被称为启发式的原因。

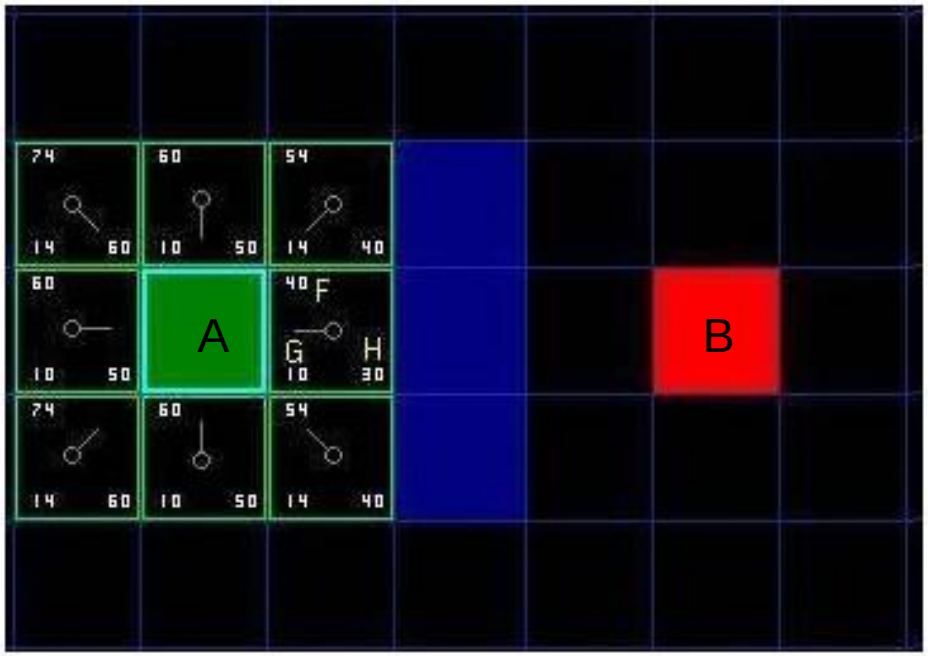
F的值是G和H的和。F,G和H的评分被写在每个方格里。F在左上角，G在左下角，H则在右下角。

示例——路径评分2（3/5）

第一步搜索的结果可以在左图表中看到：

写字母的方格里，G = 10。这是因为它只在水平方向偏离起始格一个格距。紧邻起始格的上方，下方和左边的方格的G值都等于10。对角线方向的G值是14。

H值通过求解到红色目标格的曼哈顿距离得到，其中只在水平和垂直方向移动，并且忽略中间的墙。用这种方法，起点右侧紧邻的方格离红色方格有3格距离，H值就是30。这块方格上方的方格有4格距离(只能在水平和垂直方向移动)，H值是40。每个格子的F值由G和H相加得到



示例——继续搜索1（4/5）

为了继续搜索，从开启列表中选择F值最低的方格。然后，对选中的方格做如下处理：

4，把它从开启列表中删除，然后添加到关闭列表中。

5，检查所有相邻格子。跳过那些已经在关闭列表中的或者不可通过的，把他们添加进开启列表(如果该格子不在里面的话)。把选中的方格作为新的方格的父节点。

6，如果某个相邻格已经在开启列表里了，检查现在的这条路径是否更好。换句话说，检查如果我们用新的路径到达它的话，G值是否会更低一些。如果不是，那就什么都不做。另一方面，如果新的G值更低，那就把相邻方格的父节点改为目前选中的方格。最后，重新计算F和G的值。

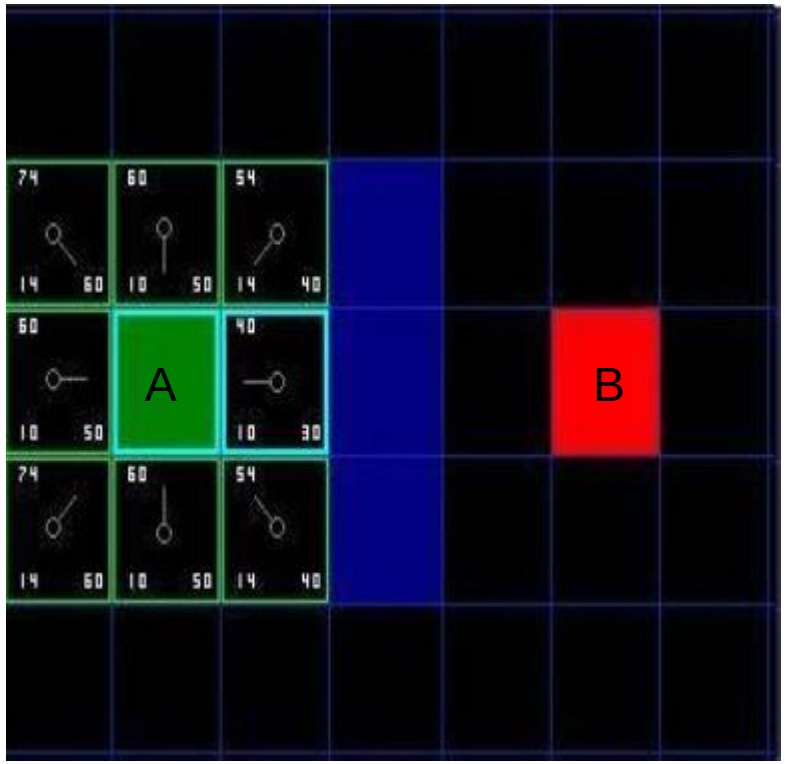
示例——继续搜索2（4/5）

图示：

最初的格方格中，在起点被切换到关闭列表中后，还剩8格留在开启列表中。这里面，F值最低的那个是起始格右侧紧邻的格子，它的F值是40。因此我们选择这一格作为下一个要处理的方格。它被用蓝色突出显示。

首先，我们把它从开启列表中取出，放入关闭列表。然后我们检查相邻的格子。右侧的格子是墙，略过。左侧的格子是起始格。它在关闭列表里，跳过它。其他4格已经在开启列表里了，于是我们检查G值来判定，如果通过这一格到达那里，路径是否更好。我们来看选中格子下面的方格。它的G值是14。如果我们从当前格移动到那里，G值就会等于20。因为G值大于14，所以这不是更好的路径。

当我们对已经存在于开启列表中的每个临近格重复这一过程的时候，我们发现没有一条路径可以通过使用当前格子得到改善，所以我们不做任何改变。检查过了所有邻近格后就可以移动到下一格了。



示例——继续搜索3（4/5）

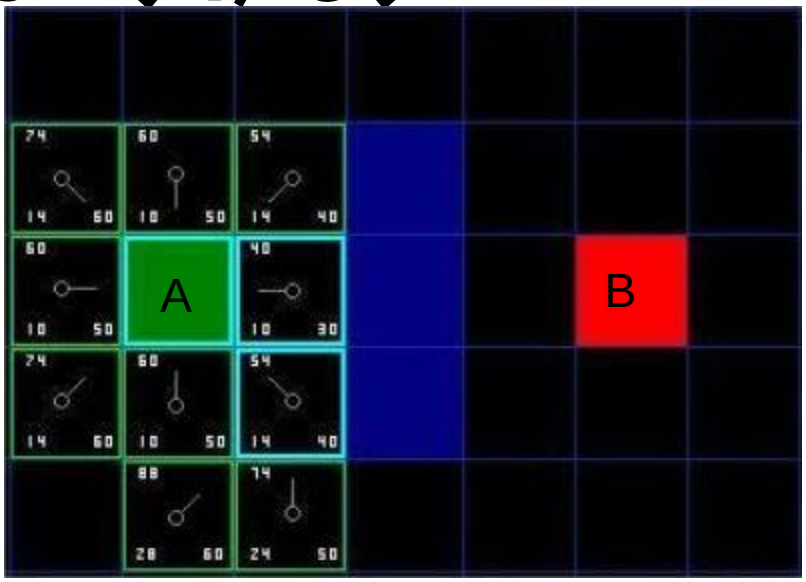
现在里面只有7格了，我们仍然选择其中F值最低的。这次，有两个格子的数值都是54。我们就选择起始格右下方的格子，如图：

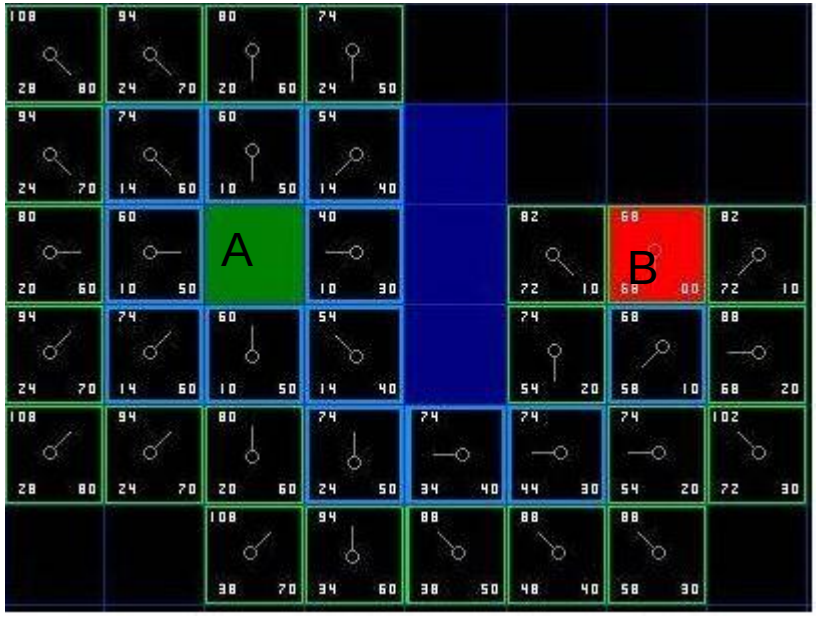
这次，当检查相邻格的时候，发现右侧是墙，略过。上面一格也被略过。也略过了墙下面的格子。

这样一来，就剩下了其他5格。当前格下面的另外两个格子目前不在开启列表中，于是我们添加他们，并且把当前格指定为他们的父节点。其余3格，两个已经在关闭列表中（起始格，和当前格上方的格子),于是我们略过它们。最后一格，在当前格的左侧，将被检查通过这条路径，G值是否更低。

检查完后，准备好检查开启列表中的下一格了。

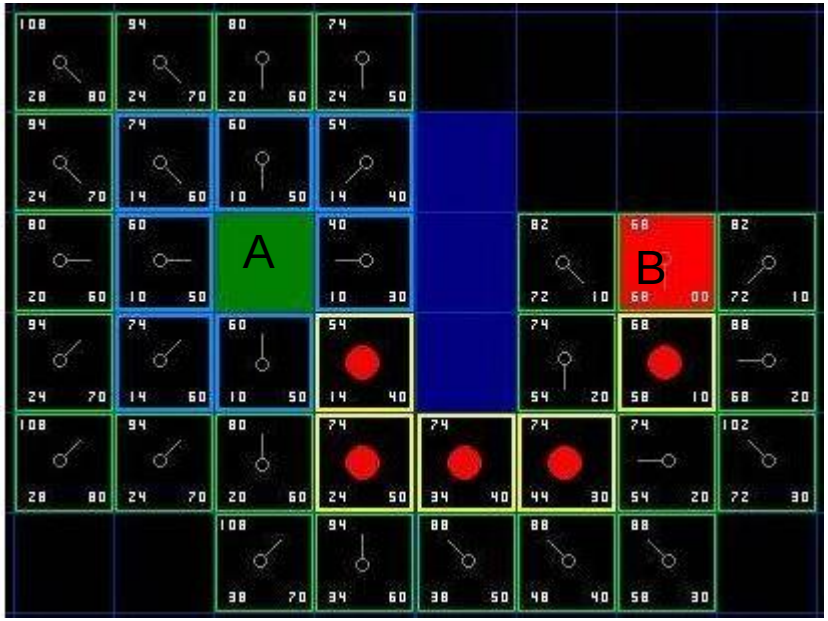
我们重复这个过程，直到目标格被添加进关闭列表，就如在图中所看到的





示例——确定路径（5/5）

从红色的目标格开始，按箭头的方向朝父节点移动。这最终会引导你回到起始格，这就是路径！看起来应该像图中那样。从起始格A移动到目标格B只是简单的从每个格子（节点）的中点沿路径移动到下一个，直到你到达目标点。



A*算法思想

1，把起始格添加到开启列表。

2，重复如下的工作：

a) 寻找开启列表中F值最低的格子。我们称它为当前格。

b) 把它切换到关闭列表。

c) 对相邻的格中的每一个？

* 如果它不可通过或者已经在关闭列表中，略过它。反之如下。

* 如果它不在开启列表中，把它添加进去。把当前格作为这一格的父节点。记录这一格的F,G,和H值。

* 如果它已经在开启列表中，用G值为参考检查新的路径是否更好。更低的G值意味着更好的路径。如果是这样，就把这一格的父节点改成当前格，并且重新计算这一格的G和F值。如果你保持你的开启列表按F值排序，改变之后你可能需要重新对开启列表排序。

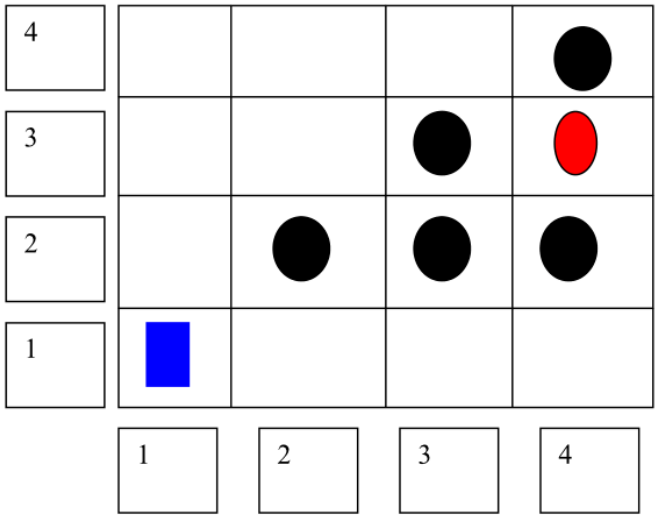
d) 停止，当你

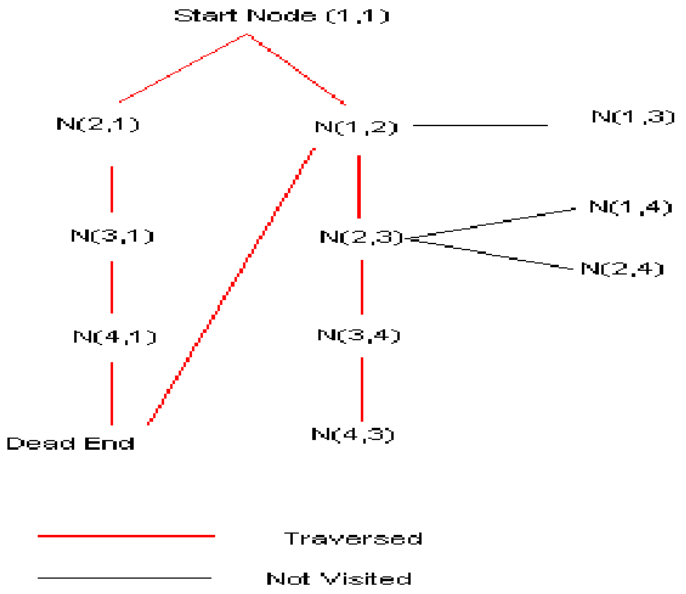
* 把目标格添加进了关闭列表(注解)，这时候路径被找到，或者

* 没有找到目标格，开启列表已经空了。这时候，路径不存在。

3.保存路径。从目标格开始，沿着每一格的父节点移动直到回到起始格。这就是路径。

Example 2

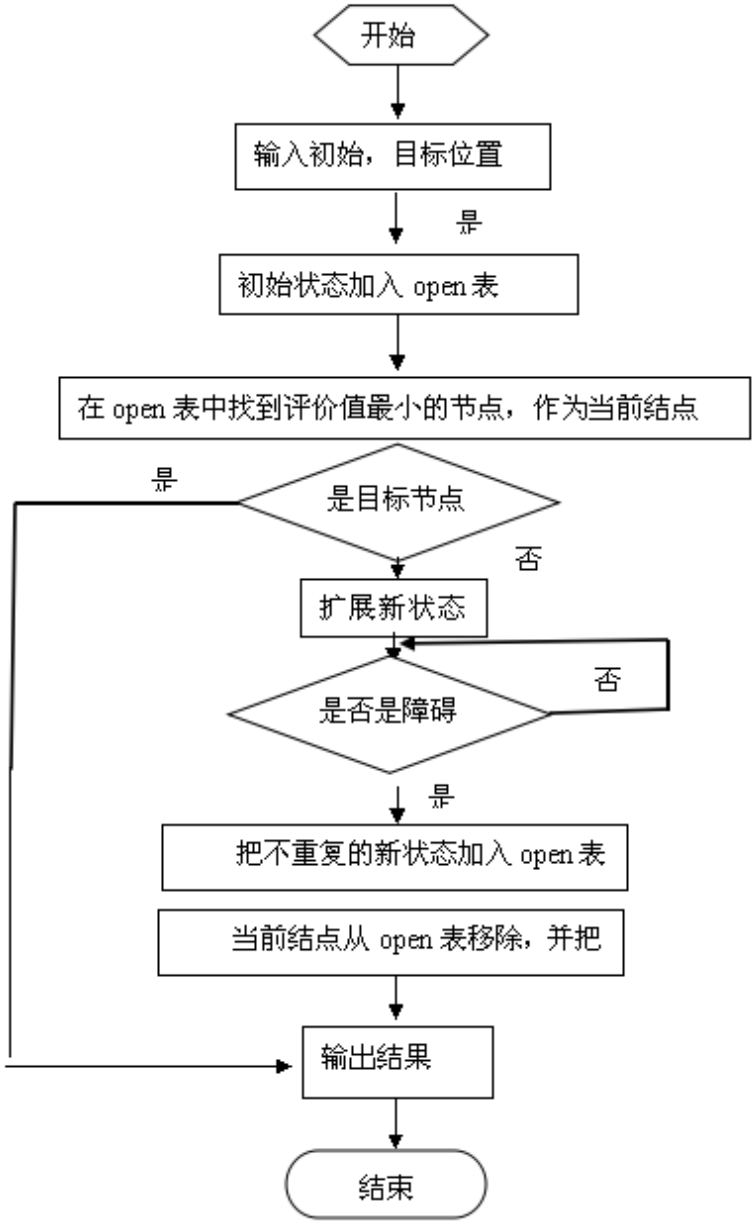




Open table and closed table

Questions?

算法流程图


开始
输入初始，目标位置
是
初始状态加入 open 表
在 open 表中找到评价值最小的节点，作为当前结点
是
是目标节点
否
扩展新状态
否
是否是障碍
是
把不重复的新状态加入 open 表
当前结点从 open 表移除，并把
输出结果
结束

The A* Algorithm

1) **Put the start node on the list OPEN** and calculate the **cost function f (n).** {h (n) = 0; g(n) = distance between the goal and the start position, f(n) = g(n).}

2) **Remove from the List OPEN the node with the smallest cost function and put it on CLOSED**. This is the node n. (In case two or more nodes have the cost function, arbitrarily resolve ties. If one of the nodes is the goal node, then select the goal node)

3) If n is the goal node then terminate the algorithm and use the pointers to obtain the solution path. Otherwise, continue

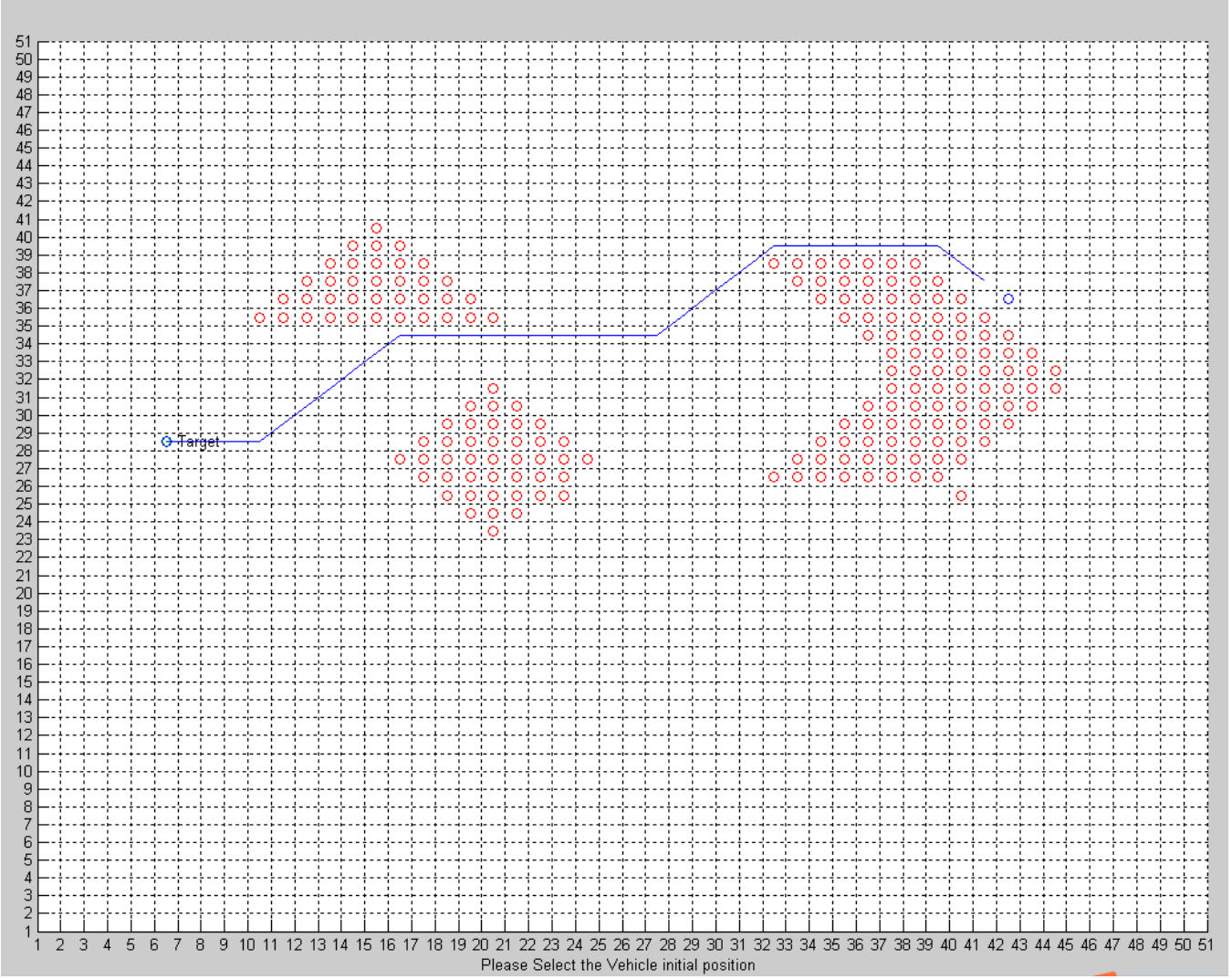
4) Determine all the successor nodes of n and compute the cost function for each successor not on list CLOSED.

5) Associate with each successor not on list OPEN or CLOSED the cost calculated and put these on the list OPEN, placing pointers to n (n is the parent node).

6) Associate with any successors already on OPEN the smaller of the cost values just calculated and the previous cost value. (min(new f(n), old f(n)))

7) Goto step 2. ;

演示结果



二、算法设计思路

- 采用A*算法作为搜索策略。在这里就要用到启发式搜索启发中的估价是用估价函数表示的,
- ***f(n) = g(n) + h(n)***
- *g(n)*是在状态空间中从初始节点到n节点的实际代价,
- *h(n)*是从n到目标节点最佳路径的估计代价。我们这里将其设为n到目标节点的直线距离。
- 保证找到最短路径的条件，关键在于估价函数*h(n)*的选取