# A-star算法

## 解决什么问题？

1、ROS机器人：全局路径规划，根据给定的目标位置和全局地图进行总体路径规划。

2、游戏中NPC移动计算，线上游戏的BOT移动计算

## 基本原理

将地图虚拟化，划分成小方块，以便使用二维数组表示。之后采用广度优先搜索思想不停寻找周围的点，选择出一个新的点作为起点再次循环查找，直至终点。

## 有关定义：

* 实际代价G：表示从起点出发移动到地图上当前单元格的移动耗费。例如，我们可以采用从起点开始，经过多少次上下左右移动才移动到指定点作为实际代价G。
* 预估代价H：表示从当前单元格移动到终点的预估耗费。在实际编程中，我们通常采用曼哈顿距离(Manhattan Distance)作为预估代价H，当然也可以采用欧几里得距离(Euclidean Distance)作为预估代价。
* 路径总代价F：F = G + H，表示该单元格点的总耗费。
* open列表：记录左右被用来考虑寻找最短路径的单元格，通常采用优先队列(priority queue)数据结构。
* close列表：记录已经被淘汰的单元格。一般为与地图对应的布尔二维数组(bool)
* 父亲点列表pre：记录open列表中元素间的逻辑联系；
* 单元格代价列表valueF：记录每一个单元格的最小总代价F。

## 具体搜索过程

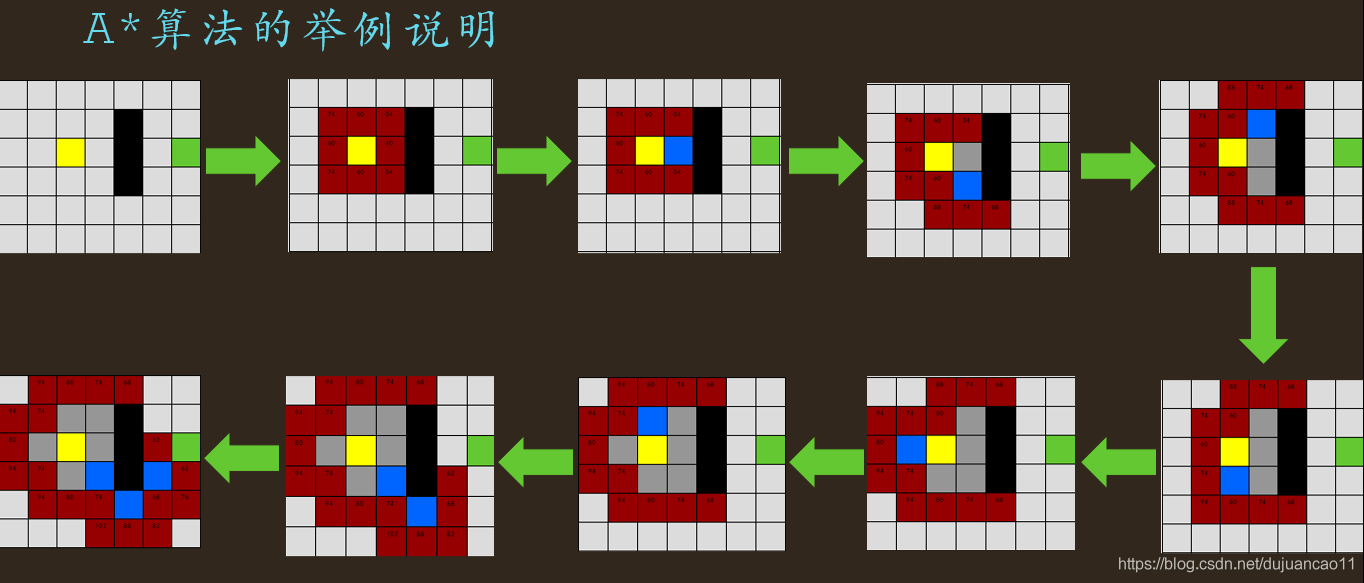


1、首先需要创建一张地图，可以有障碍物但是必须进行标记。

2、设置路径的起点和终点。

3、开始搜索路径：

1. 初始化open列表，close列表，pre列表和valueF列表；
2. 将起点加入open列表，然后将其周围四个点（或八个点，由需求决定）加入到open列表。将起点从open列表中移除并移动到close列表；
3. 依次判断周围这四个点是否在close列表中且是否越界，如果不在，以此计算周围点的G，H并更新F，如果对应单元格在valueF中的值为初始化值或较大，那么更新单元格对应valueF值。
4. 从周围的点中找出F最小的点，获得周围点的集合，然后将这个F最小的点从open列表中移除并移动到close集合中；
5. 跳转第（3）步。



## 结束条件

1、终点单元格被加入open列表并且被作为当前格查询时；

2、open列表被清空，表示不可能到达终点。