CISDI 中冶赛迪

测试题总结

主　　研　　人：陈双仪

参　　研　　人：

审　　核　　人：

智能算法研究所

中冶赛迪技术研究中心有限公司

二零一八年五月

目　录

1. 概述

在工业生产中，进行重载运输时需要规避有人区域和重要设备曲线，因此需要对吊车运行路径进行规划。

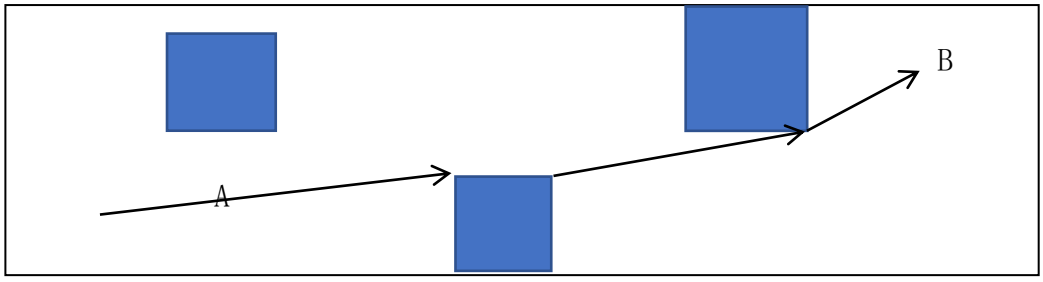

图 1 效果示意图

如图 1 所示，从 A 点运行至 B 点，在避开正方形障碍物的同时，找寻最短路径。

1.1 需求达成确认

```
1 0 3 3 0 3 0 0 0 0
0 0 3 0 0 3 0 3 0 0
0 0 3 0 0 3 3 3 0 0
0 0 3 0 0 0 0 0 0 3
3 0 0 0 0 3 0 0 0 0
3 3 0 3 0 0 3 3 0 3
3 0 0 0 0 3 3 0 0 0
0 3 0 0 0 0 0 0 0 0
3 3 3 0 0 3 0 3 2 3
3 0 0 0 0 3 3 3 0 3
(0,0)-->(1,0)-->(2,0)-->(3,0)-->(4,1)-->(5,2)-->(6,3)-->(6,4)-->(7,5)-->(7,6)-->(7,7)-->(8,8)
```

图 2 运行结果示意图

如图 2 所示，图片上方结点数组代表地图，其中 1 代表起点，2 代表终点，3 代表障碍物，0 代表可以通过。下方输出的结点序列(x_i,y_i)代表由 A*算法得出的最短路径（其中$x_i,y_i \in [0,9]$）。

2. 实现具体思路

文件中的函数

void swap(**int** idx1, **int** idx2) 、**void** adjust(**int** nIndex) – 堆排序函数，用于选出 open 表中 F 值（F=G+H）最小的邻居点

void if_insert_to_opentable(**int** x, **int** y, APoint curr_node, APoint end_node, **int** w) – 判断邻居点是否可以进入 open 表

void get_neighbors(APoint cur_node, APoint end_node) – 对某点周围的八个邻居点进行查找

int main() – 使用 A*算法得到最短路径

2.1 初始化结点数组

根据自定义的地图矩阵，遵循“1 代表起点，2 代表终点，3 代表障碍物，0 代表可通过”的规则初始化结点数组地图。针对某结点，初始化内容包括

- 坐标
- 起点到此点的距离 G 值
- 启发函数预测的此点到终点的 Manhanttan 距离 H 值
- 结点类型（1 代表起点，2 代表终点，3 代表障碍物，0 代表可通过）
- 父结点
- 状态标志：是否在 close 表中
- 状态标志：是否在 open 表中

2.2 A*算法

在初始化地图结点数组后，主函数的下一部分将执行 A*算法得到最短路径。

首先从起点开始搜索，加入 open 表。然后利用 get_neighbors 函数查询起点周围的共八个邻居点（这些点认当前点为父结点），再通过 if_insert_to_opentable 函数判断这些点是否符合加入 open 表的条件。条件分别是 1、不是障碍点 2、不在 close 表中 3、不在 open 表中。若符合所有条件，则将该点加入 open 表中。利用堆排序函数选出 open 表中具有最小 F 值（F = G+H）的点，作为下一步到达的点。将当前点从 open 表删除，加入 close 表。

若某当前点的邻居点仅符合加入 open 表的条件 1、2，但已在 open 表中，则判断该邻居点是否是相对该当前点更优的路径点。若该邻居点的 G 值小于该点的 G 值加上该当前点到该邻居点的距离，则该邻居点是相对该当前点更优的路径点。将上一步移动取消，将当前点从 open 表删除加入 close 表，调整该邻居点为当前点。

若当前点为终点，则搜索结束。

程序流程图如图 3 所示：

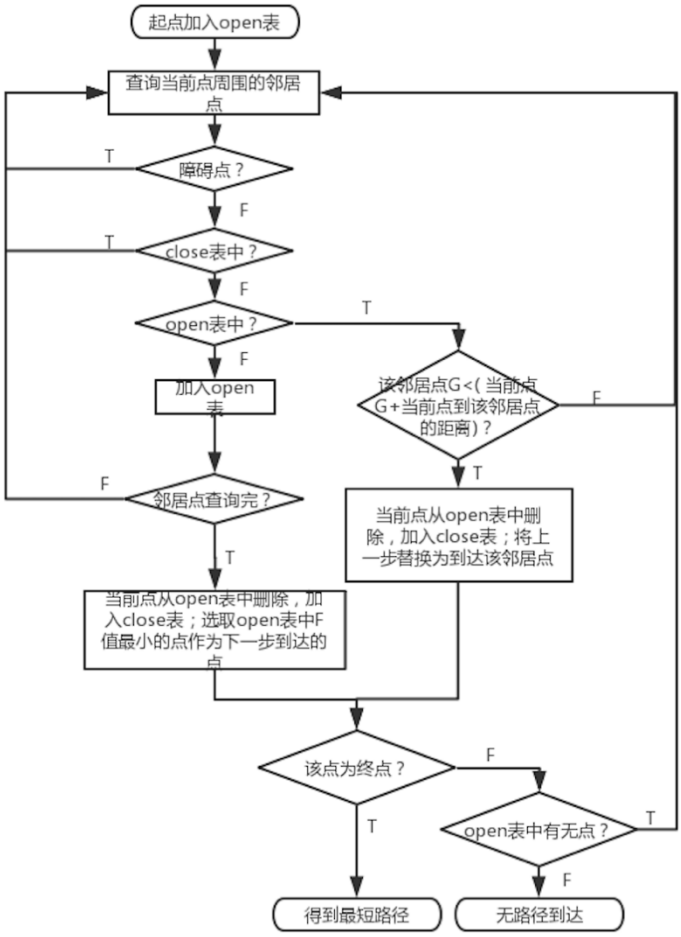

图 3 程序流程图

2.3 输出最短路径

主函数的下一部分将输出最短路径。首先由当前点（终点）向前依次查找父结点，得到路径结点序列，储存在 path_stack 数组中。再从起点依次输出，得到最短路径。

3. 小结

以上给出了 A*算法的实现思路。此次用 C 语言仅验证了 A*算法的原理，在解决实际具体问题时还存在以下问题：

1. 地图为手动输入数组，若地图较大，手动输入实现不易
2. 未考虑行驶物大小，地图度量单位

针对第二个问题，我想需要综合考虑行驶物大小形状、障碍物大小形状，才能决策出合理的地图度量单位，再进一步设计地图数组。当第二个问题解决了，第一个问题只需要设计区域转化为地图数组的函数即可解决。