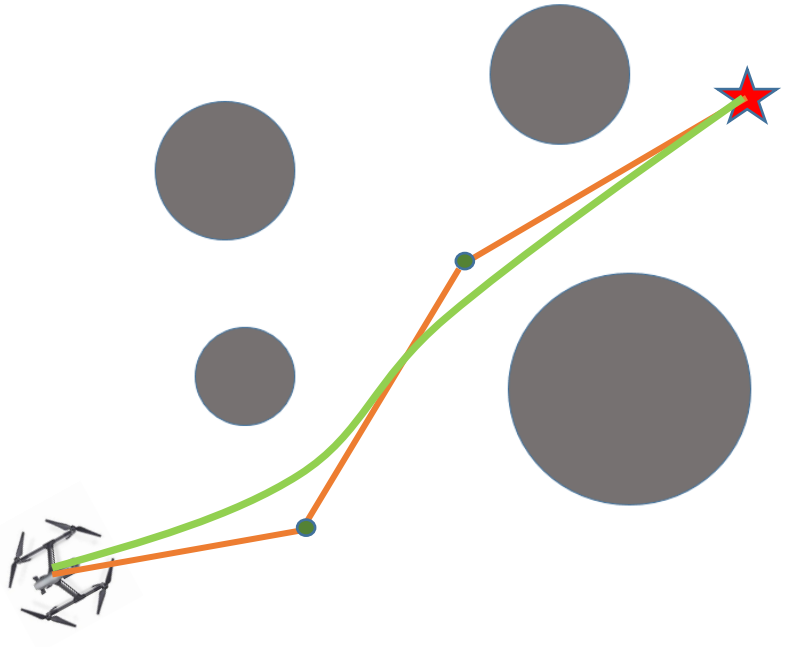
机器人轨迹规划

Joe　　艾若机器人　　joe_ir@163.com

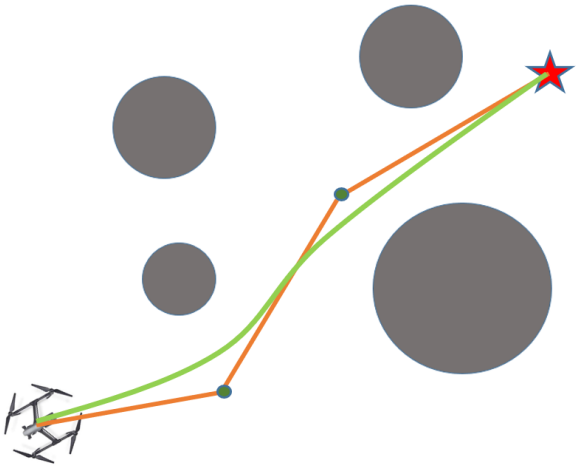
公众号：Joe学习笔记

轨迹规划

路径规划 + 轨迹优化



课程大纲



路径规划 ＋ 轨迹优化

- 路径规划
  - 基于搜索的路径规划
    - Dijkstra
    - A*
  - 基于采样的路径规划
    - RRT
    - RRT*
    - Informd RRT*
  - 基于智能算法的路径规划
    - 遗传算法
    - 蚁群算法
- 轨迹优化
  - 轨迹表示方法
    - 多项式曲线
    - 贝塞尔曲线
  - 轨迹优化目标
    - 最小化snap
    - 轨迹长度
  - 轨迹约束方法
    - 软约束
    - 硬约束

机器人实验： 在基于ROS的机器人平台上实现算法

联系方式

关注公众号：Joe学习笔记，获取PPT和代码

邮箱：joe_ir@163.com

基于搜索的路径规划

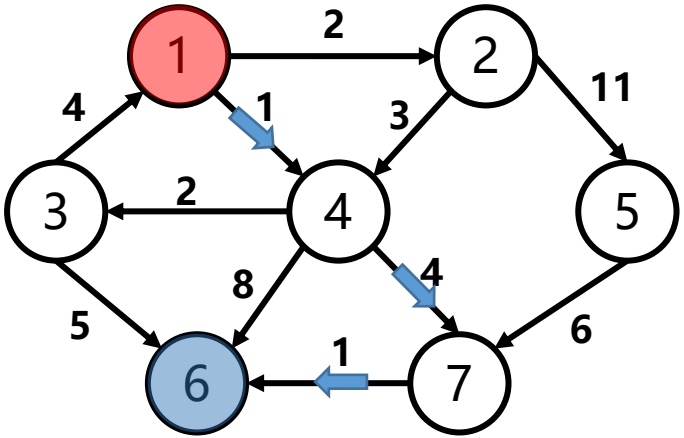
Joe　　艾若机器人　　joe_ir@163.com

公众号：Joe学习笔记

01
PART

Dijkstra算法讲解

Dijkstra算法



找到一条从v1到v6的最短路径

从起点开始逐步扩展，每一步为一个节点找到最短路径

While True:

1.从未访问的节点选择距离最小的节点收录（贪心思想）

2.收录节点后遍历该节点的邻接节点，更新距离

open list: 1(0)

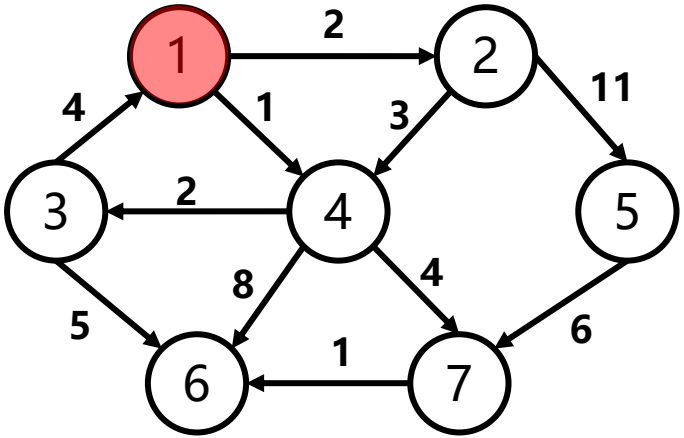
closed list:

为什么被收录的节点已经找到距离起点最短的路径？（反证法）

g(1->2->4) < g(1->4) ➔ g(1->2) < g(1->4)

矛盾

Dijkstra算法



找到一条从v1到v6的最短路径

从起点开始逐步扩展，每一步为一个节点找到最短路径

While True:

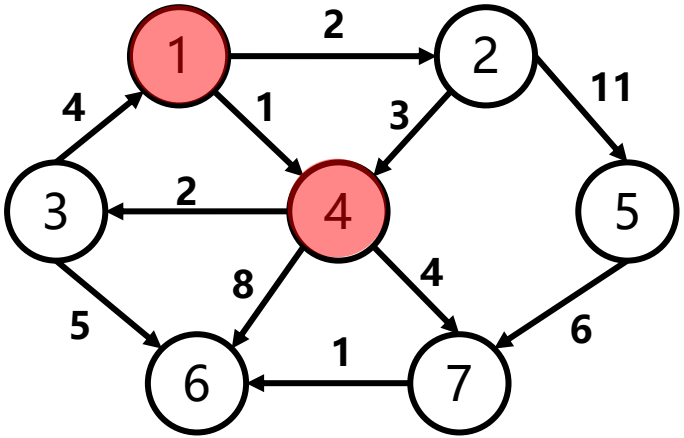
1.从未访问的节点选择距离最小的节点收录（贪心思想）

2.收录节点后遍历该节点的邻接节点，更新距离

open list: 2(2) 4(1)

closed list: 1(0)

Dijkstra算法



找到一条从v1到v6的最短路径

从起点开始逐步扩展，每一步为一个节点找到最短路径

While True:

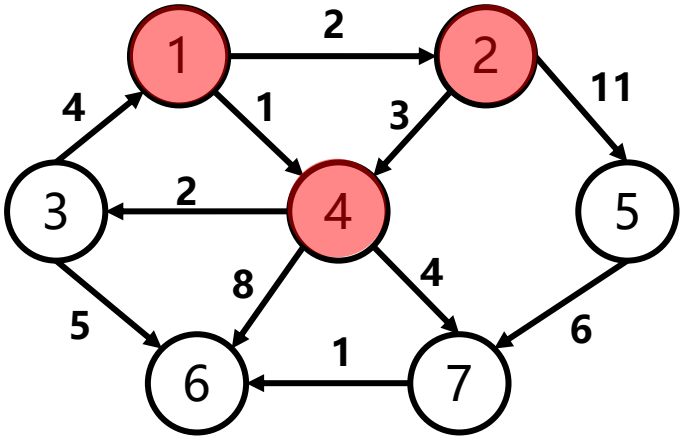
1.从未访问的节点选择距离最小的节点收录（贪心思想）

2.收录节点后遍历该节点的邻接节点，更新距离

open list: 2(2) 3(3) 6(9) 7(5)

closed list: 1(0) 4(1)

Dijkstra算法



找到一条从v1到v6的最短路径

从起点开始逐步扩展，每一步为一个节点找到最短路径

While True:

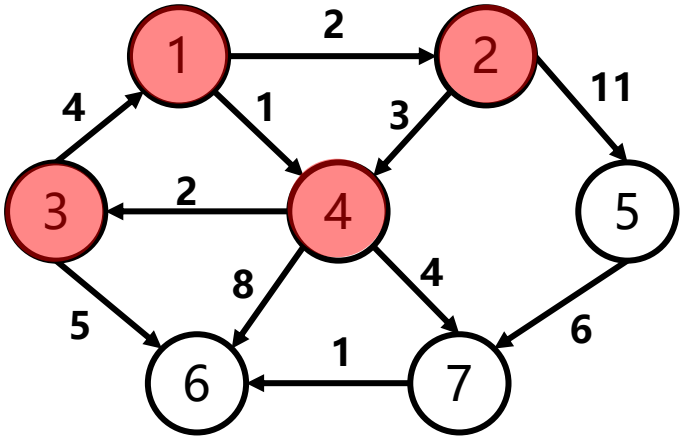
1.从未访问的节点选择距离最小的节点收录（贪心思想）

2.收录节点后遍历该节点的邻接节点，更新距离

open list: 3(3) 6(9) 7(5) 5(13)

closed list: 1(0) 4(1) 2(2)

Dijkstra算法



找到一条从v1到v6的最短路径

从起点开始逐步扩展，每一步为一个节点找到最短路径

While True:

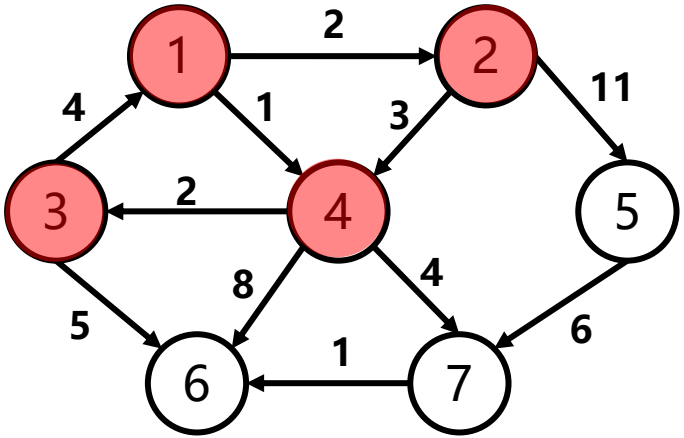
1.从未访问的节点选择距离最小的节点收录（贪心思想）

2.收录节点后遍历该节点的邻接节点，更新距离

open list: 6(9) 7(5) 5(13)

closed list: 1(0) 4(1) 2(2) 3(3)

Dijkstra算法



找到一条从v1到v6的最短路径

从起点开始逐步扩展，每一步为一个节点找到最短路径

While True:

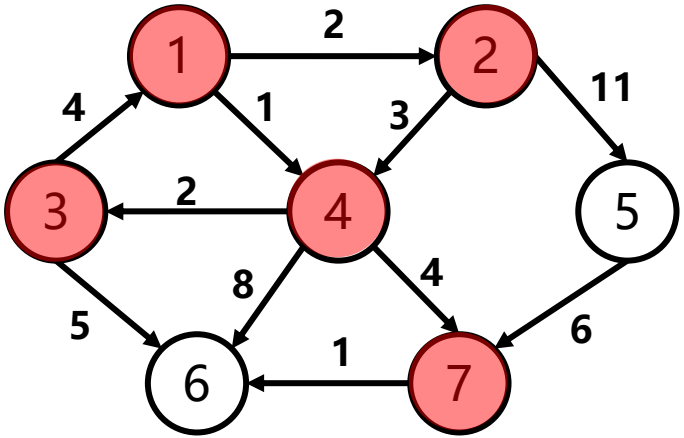
1.从未访问的节点选择距离最小的节点收录（贪心思想）

2.收录节点后遍历该节点的邻接节点，更新距离

open list: 6(8) 7(5) 5(13)

closed list: 1(0) 4(1) 2(2) 3(3)

Dijkstra算法



找到一条从v1到v6的最短路径

从起点开始逐步扩展，每一步为一个节点找到最短路径

While True:

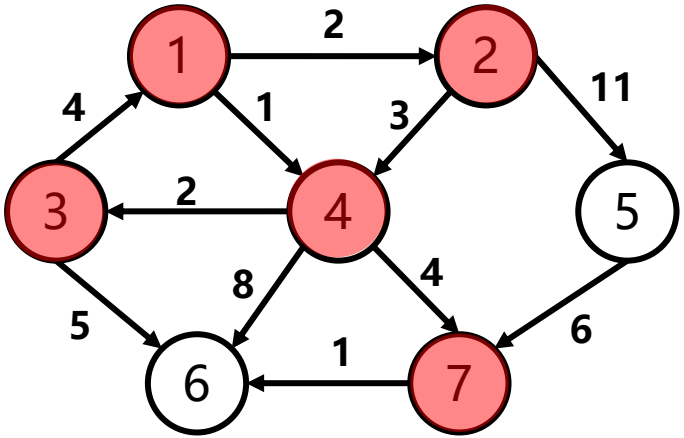
1.从未访问的节点选择距离最小的节点收录（贪心思想）

2.收录节点后遍历该节点的邻接节点，更新距离

open list: 6(8) 5(13)

closed list: 1(0) 4(1) 2(2) 3(3) 7(5)

Dijkstra算法



找到一条从v1到v6的最短路径

从起点开始逐步扩展，每一步为一个节点找到最短路径

While True:

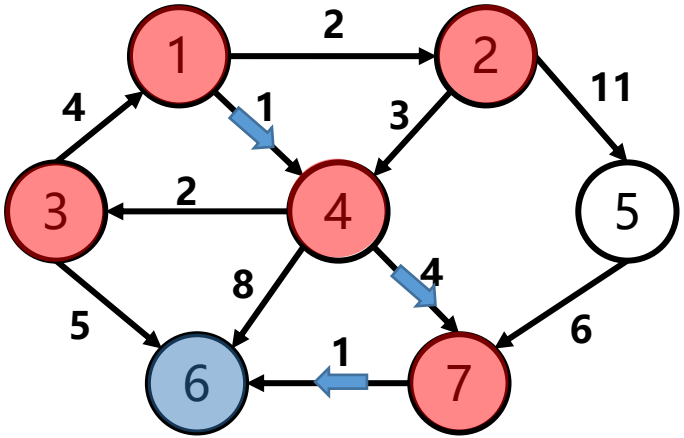
1.从未访问的节点选择距离最小的节点收录（贪心思想）

2.收录节点后遍历该节点的邻接节点，更新距离

open list: 6(6) 5(13)

closed list: 1(0) 4(1) 2(2) 3(3) 7(5)

Dijkstra算法



找到一条从v1到v6的最短路径

从起点开始逐步扩展，每一步为一个节点找到最短路径

While True:

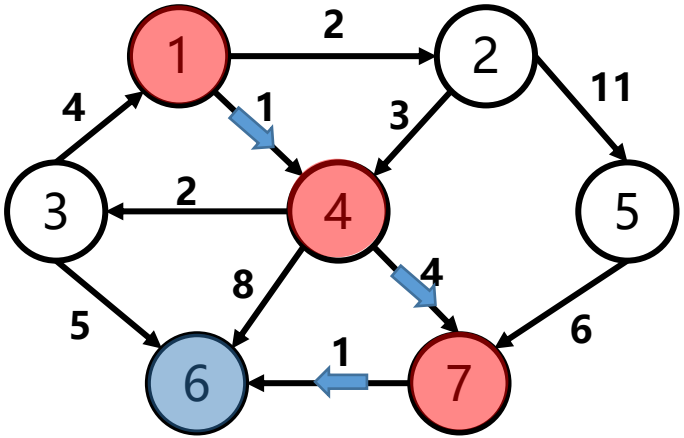
1.从未访问的节点选择距离最小的节点收录（贪心思想）

2.收录节点后遍历该节点的邻接节点，更新距离

open list: 6(6) 5(13)

closed list: 1(0) 4(1) 2(2) 3(3) 7(5)

Dijkstra算法



找到一条从v1到v6的最短路径

开始

将起点放入 open list 中

While True

if open list 为空

搜索失败，结束

取 open list 中$g(n)$最小的节点

将节点加入 closed list 中

if 节点为终点

找到路径，结束

遍历当前节点的未在 closed list 中的邻接节点

if 节点在 open list 中

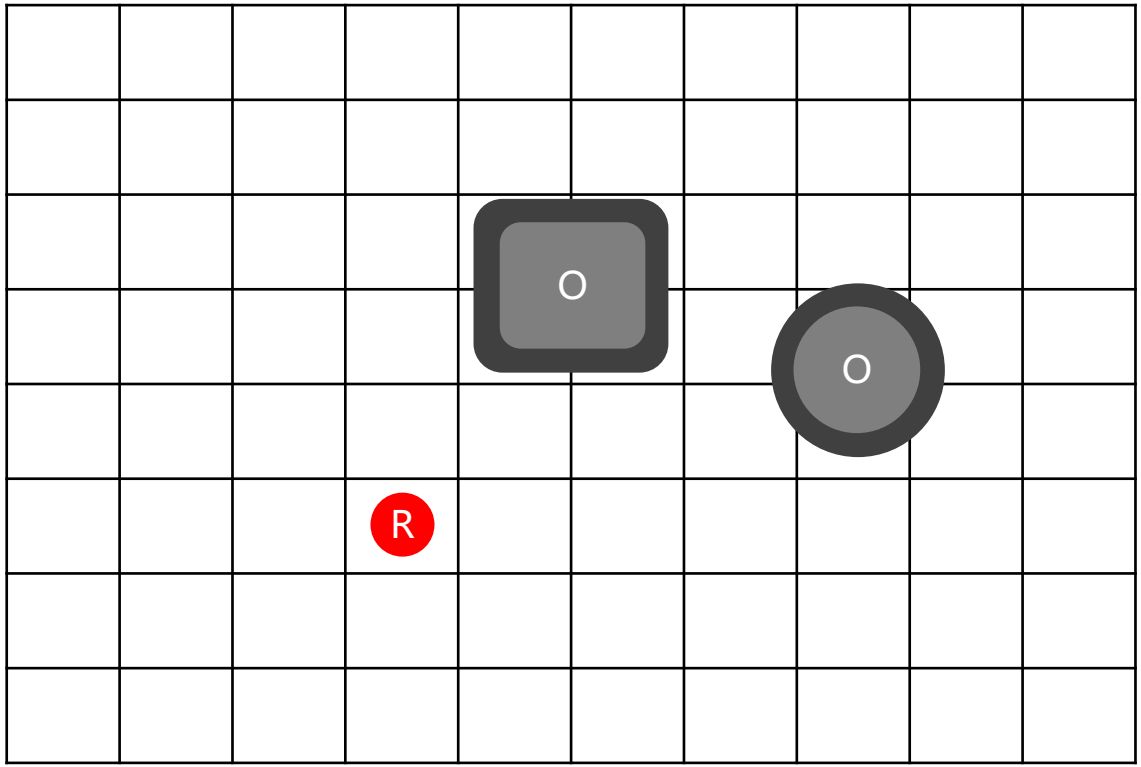
更新节点$g(n)$值

else

计算节点$g(n)$值，加入 open list

结束

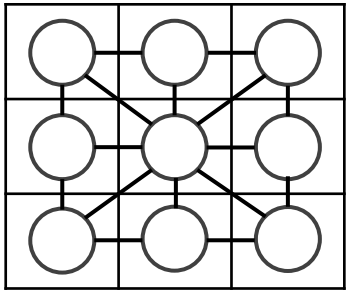
栅格地图

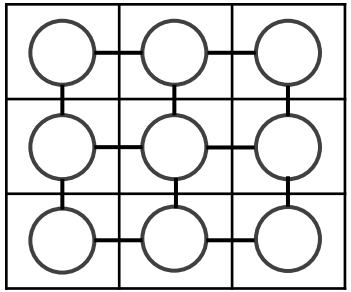


构型空间
(Configuration Space)

栅格地图

栅格地图转换为有权图

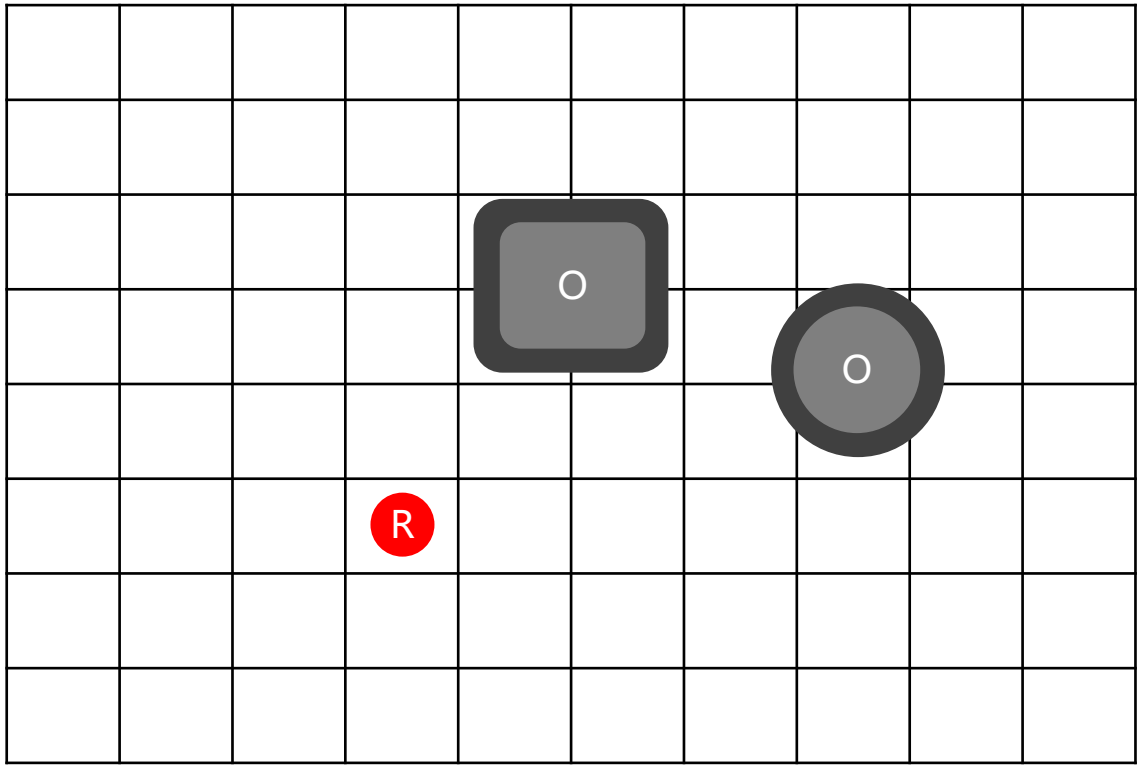




02
PART

Dijkstra算法代码讲解

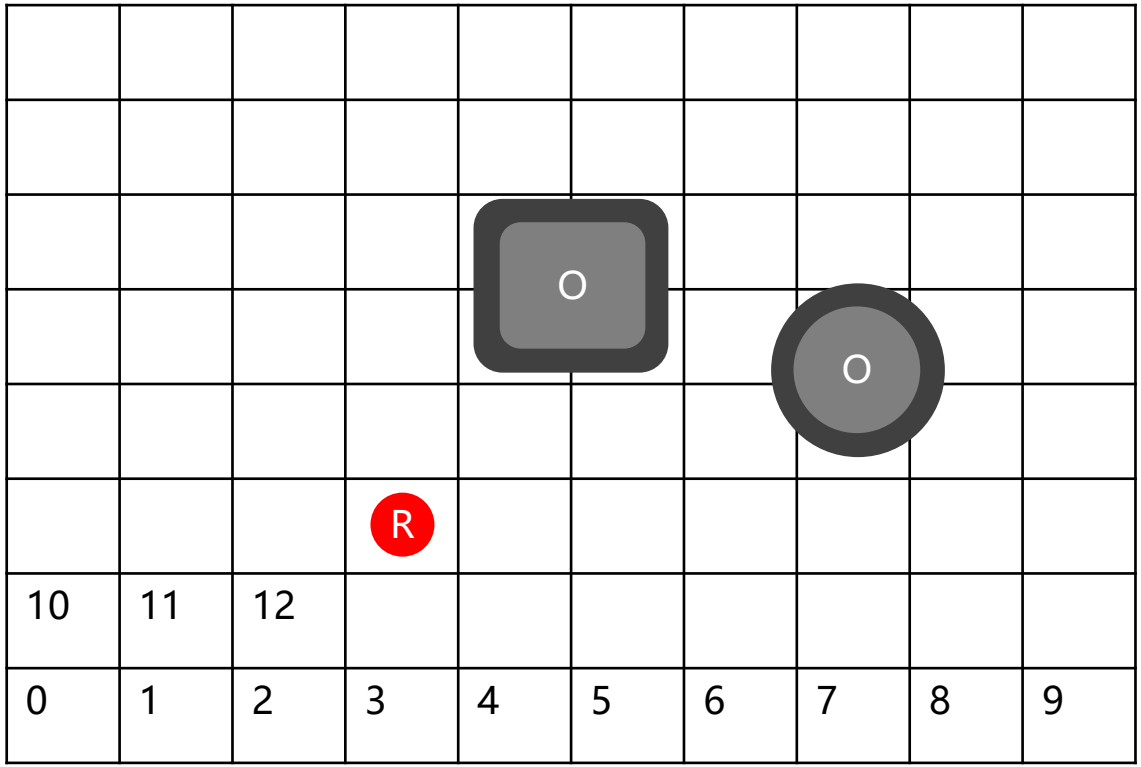
栅格地图



y_width

x_width

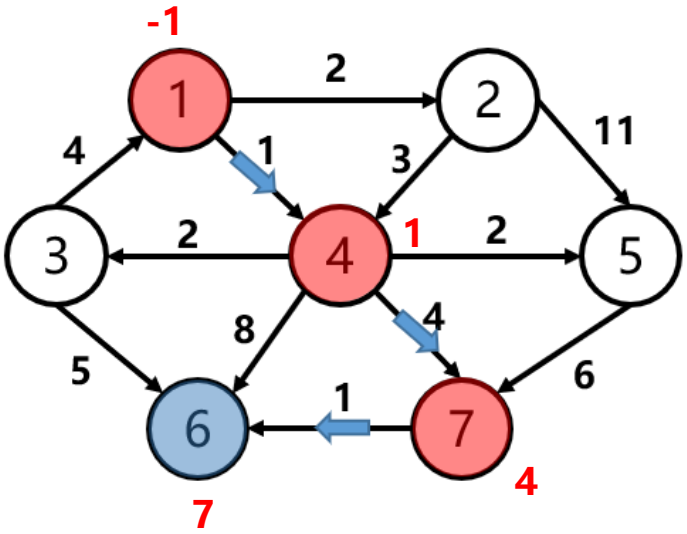
栅格地图



代码框架

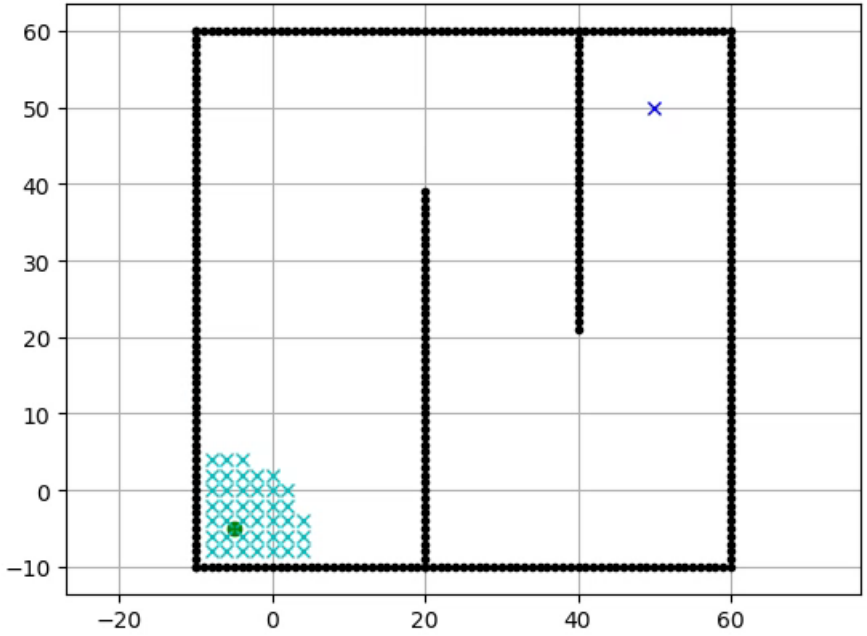
```
开始
    将起点放入 open list 中
    While True
        if  open list 为空
            搜索失败，结束
        取 open list 中g(n)最小的节点
        将节点加入 closed list 中
        if  节点为终点
            找到路径，结束
        遍历当前节点的未在 closed list 中的邻接节点
            if 节点在 open list 中
                更新节点g(n)值
            else
                计算节点g(n)值，加入 open list
结束
```

获得路径



找到一条从v1到v6的最短路径

实验结果



Dijkstra算法

算法复杂度分析（Optional）

哈希表: O(V^2 + V) = O(V^2)

开始

将起点放入 open list 中

While True **O(V)**

if open list 为空

搜索失败，结束

取 open list 中$g(n)$最小的节点 **O(V)**

将节点加入 closed list 中

if 节点为终点

找到路径，结束

遍历当前节点的未在 closed list 中的邻接节点 **O(1)**

if 节点在 open list 中

更新节点$g(n)$值 **O(1)**

else

计算节点$g(n)$值，加入 open list

结束

Dijkstra算法

算法复杂度分析（Optional）

哈希表: O(V^2 + V) = O(V^2)

优先级队列: O(V * logV + V * logV)
= O(V * logV)

开始

将起点放入 open list 中

While True **O(V)**

if open list 为空

搜索失败，结束

取 open list 中$g(n)$最小的节点 **O(logV)**

将节点加入 closed list 中

if 节点为终点

找到路径，结束

遍历当前节点的未在 closed list 中的邻接节点 **O(1)**

if 节点在 open list 中

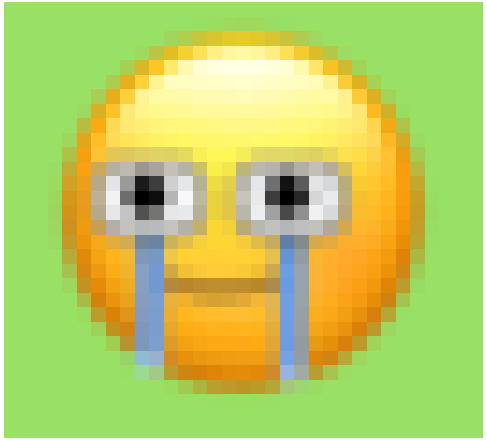
更新节点$g(n)$值 **O(logV)**

else

计算节点$g(n)$值，加入 open list

结束

彩蛋



发现好难讲，NG无数次。水平有限，有错误的话希望大家多多见谅。

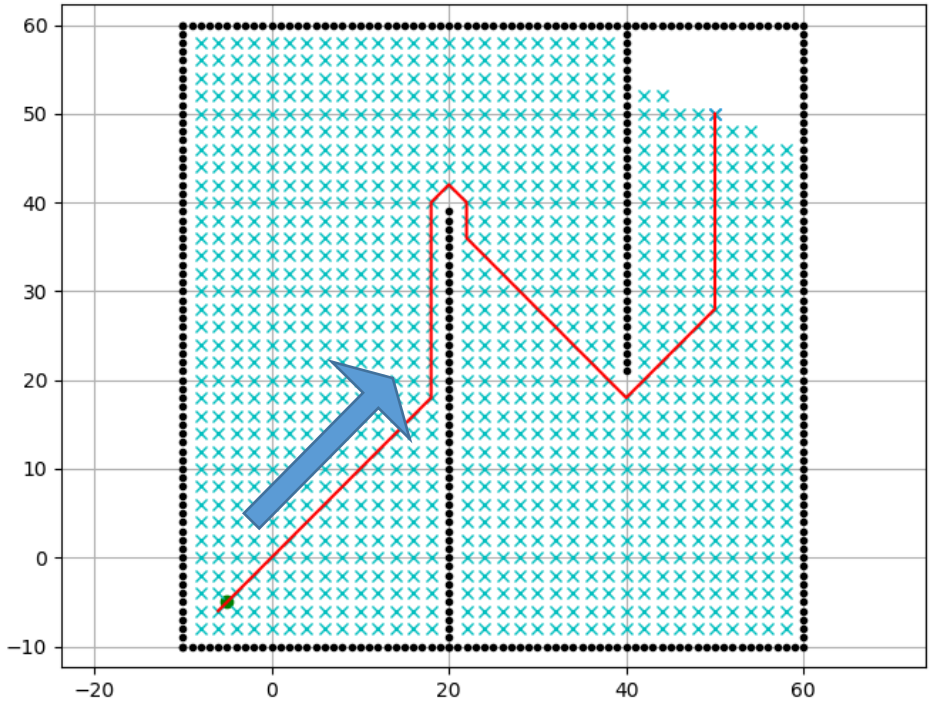
感觉Dijkstra算法是路径规划算法里面最难理解的，大家可以多看一些其他资料，特别是去学一学数据结构中介绍的Dijkstra算法，会比较系统。

理解了Dijkstra算法接下来的A*算法就非常容易了，加油💪！

感谢大家观看！

A*算法讲解

A*算法



Dijkstra算法结果

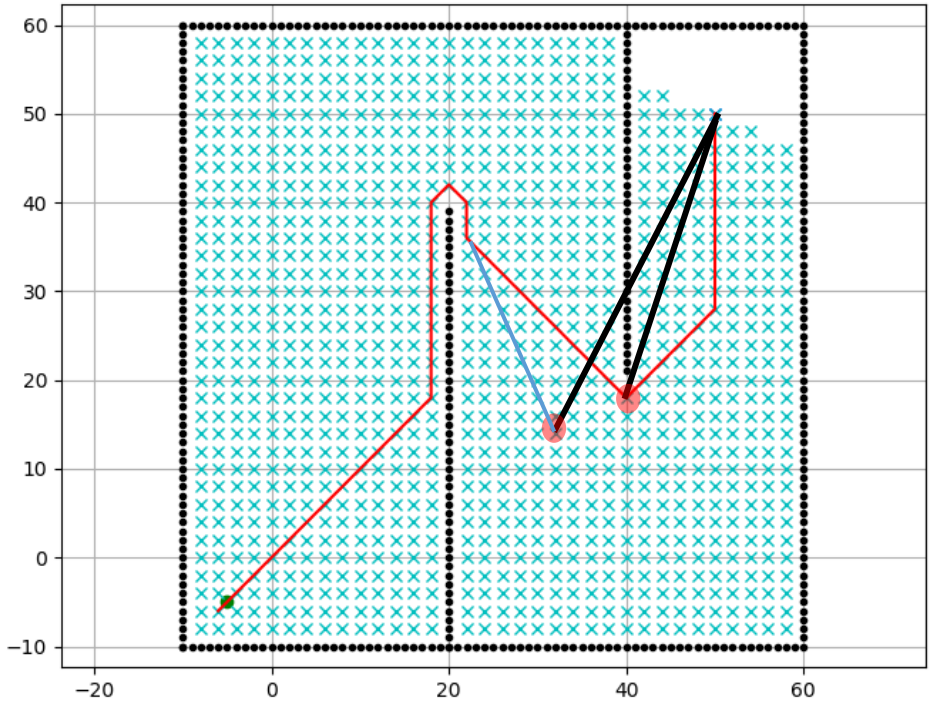
A*算法提出的动机(motivation)：

减少收录的栅格数目，增加搜索速度

F(n)=g(n) +h(n)

增加启发式函数(Heuristics)

A*算法



Dijkstra算法结果

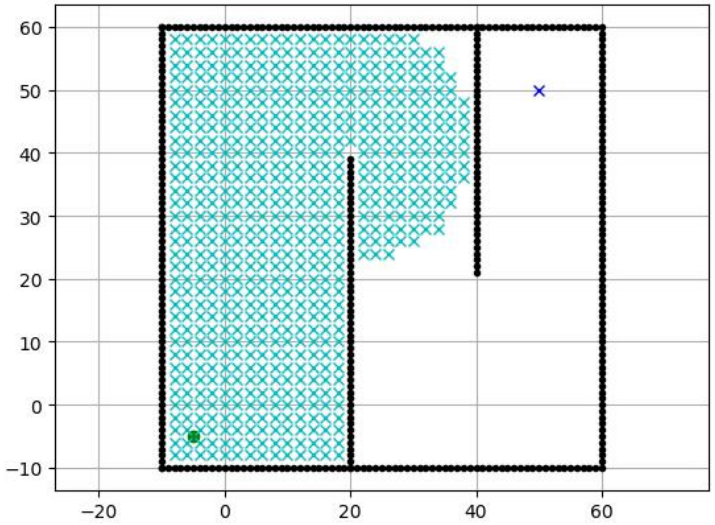
A*算法提出的动机(motivation)：

减少收录的栅格数目，增加搜索速度

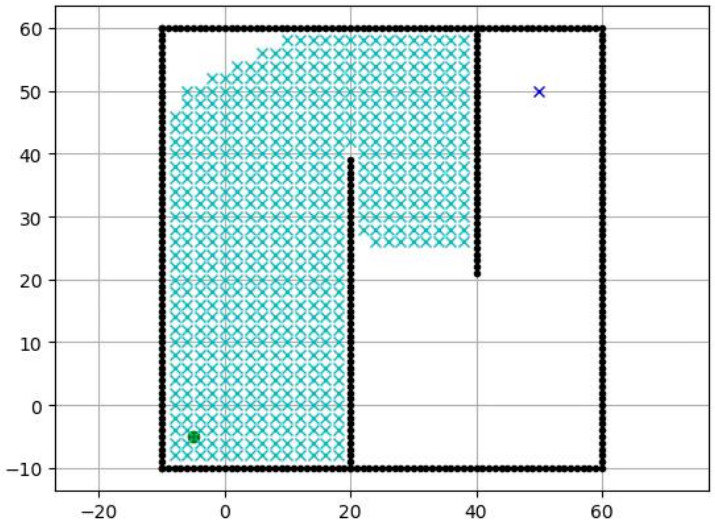
F(n)=g(n) +h(n)

增加启发式函数(Heuristics)

实验结果



Dijkstra算法扩展



A*算法扩展

A*算法

开始

将起点放入 open list 中

While True

if open list 为空

搜索失败，结束

取 open list 中$g(n)$最小的节点

将节点加入 closed list 中

if 节点为终点

找到路径，结束

遍历当前节点的未在 closed list 中的邻接节点

if 节点在 open list 中

更新节点$g(n)$值

else

计算节点$g(n)$值，加入 open list

结束

（a）Dijkstra 算法

开始

将起点放入 open list 中

while True

if open list 为空

搜索失败，结束

取 open list 中$\boldsymbol{g(n)+h(n)}$最小的节点

将节点加入 closed list 中

if 节点为终点

找到路径，结束

遍历当前节点的未在 closed list 中的邻接节点

if 节点在 open list 中

更新节点$g(n)$值

else

计算节点$g(n)$值，加入 open list

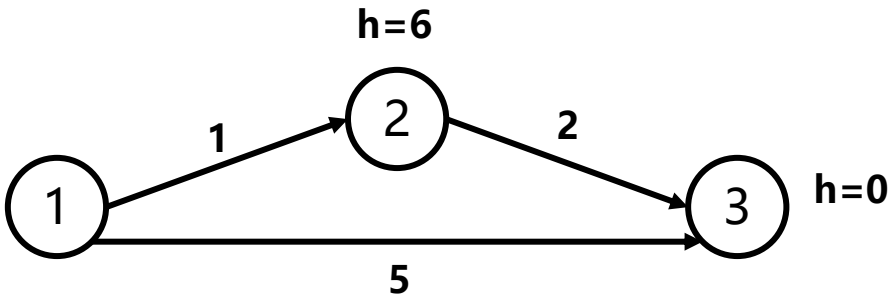
结束

(b) A*算法

A*算法

保证最优性要求: h(n) <= *h(n)

在实际应用中权衡速度和最优性



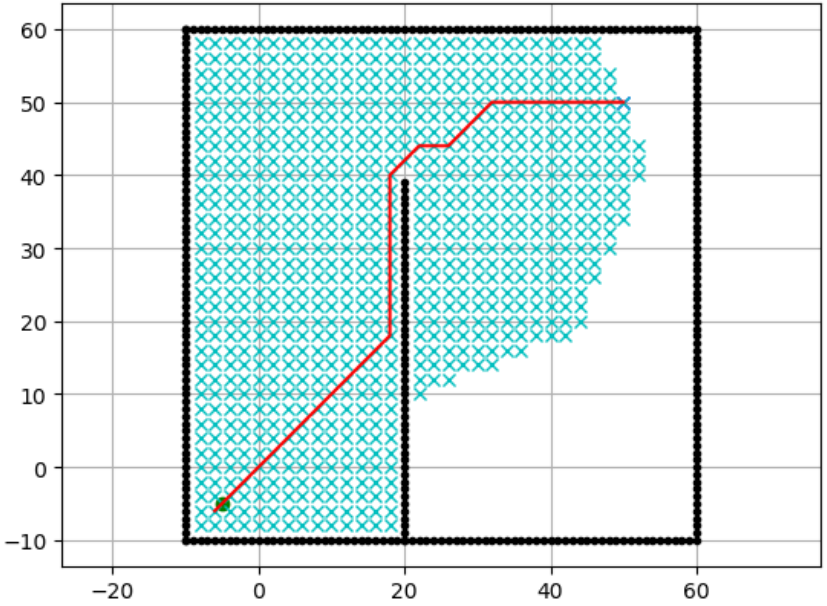
open list: 2(1+6) 3(5+0)

closed list: 1(0)

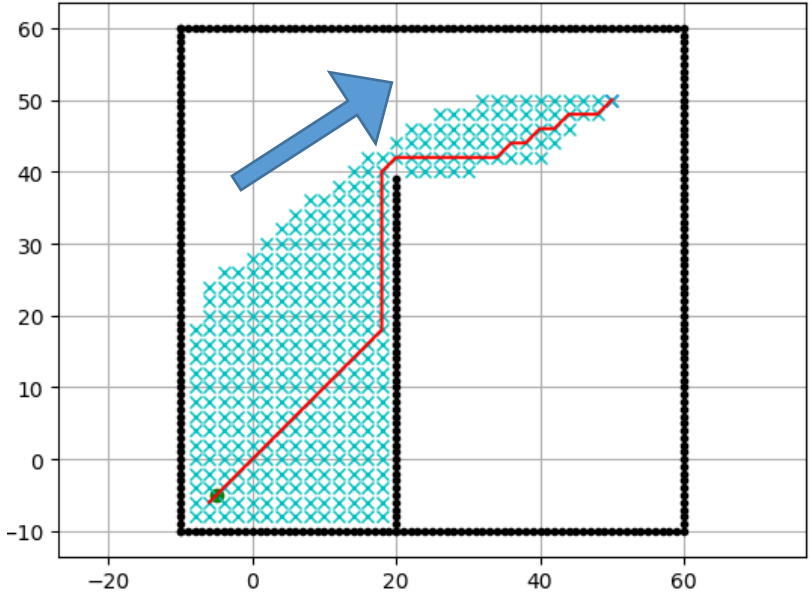
03
PART

A*算法代码讲解

实验结果比较

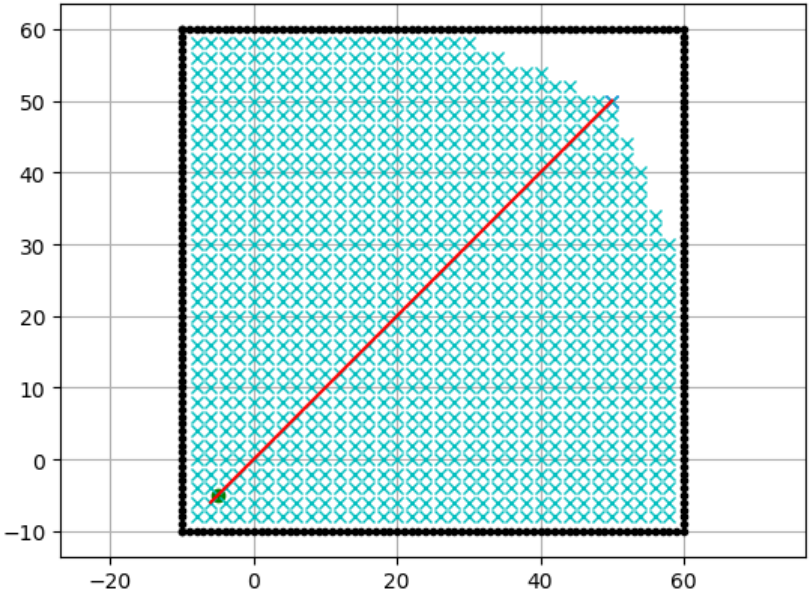


Dijkstra算法结果

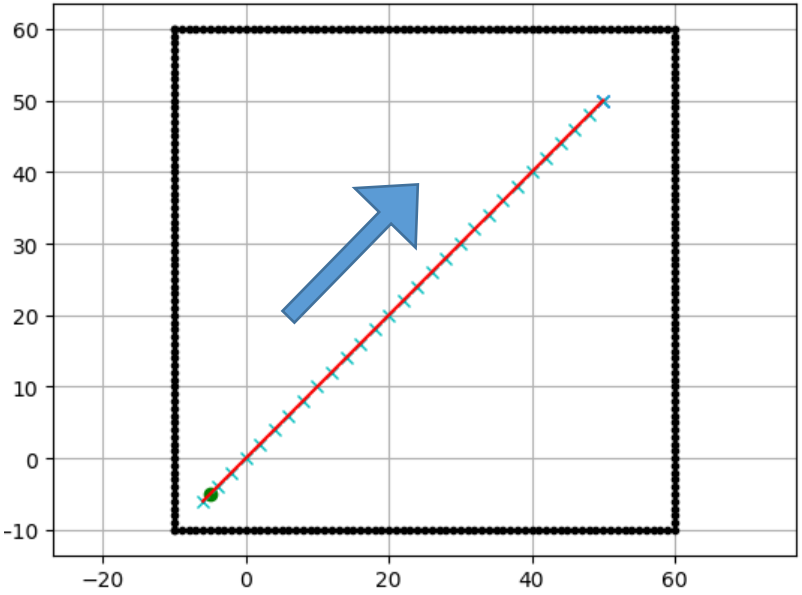


A*算法结果

实验结果比较



Dijkstra算法结果



A*算法结果

THANKS