人工智能

沙瀛

信息学院
2020.3

ARTIFICIAL INTELLIGENCE

启发式搜索

- 启发式信息与估价函数
- A算法之局部择优搜索与全局择优搜索
- A*算法

启发式信息与估价函数

- **启发信息**

  用于指导搜索过程且与具体问题求解有关的控制信息称为启发信息。

- **启发信息分类**
  - 有效地帮助确定扩展节点的信息。
  - 在扩展节点时，用于决定要生成哪一个或哪几个后继节点。
  - 用于确定某些应该从搜索树中抛弃或修剪的节点。

估价函数

在扩展节点时，用来描述节点重要程度的函数称为估价函数。其一般形式为：

$$f(x)=g(x)+h(x)$$

其中，g(x)为初始节点S0到节点x已实际付出的代价，h(x)是从节点x到目标节点Sg的最优路径的估计代价，启发信息主要由h(x)来体现，故把它称为启发函数。

A算法

- 在状态空间搜索中，如果每一步都利用估价函数f(n)=g(n)+h(n)对Open表中的节点进行排序，则称A算法。它是一种为启发式搜索算法。

A算法

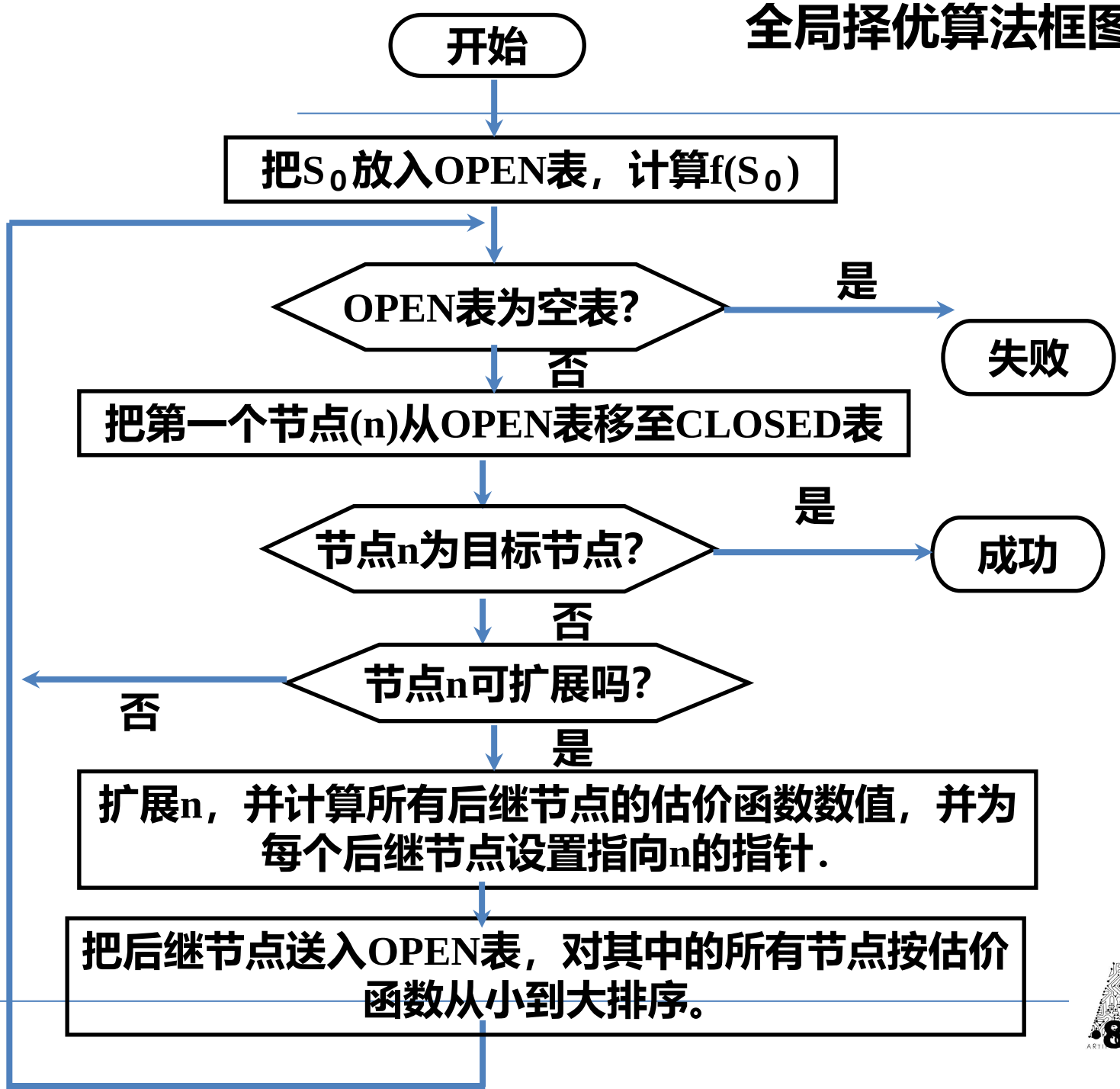
- 类型：
  - 全局择优：从Open表的所有节点中选择一个估价函数值最小的进行扩展。
  - 局部择优：仅从刚生成的子节点中选择一个估价函数值最小的进行扩展。

启发式搜索方法

全局择优

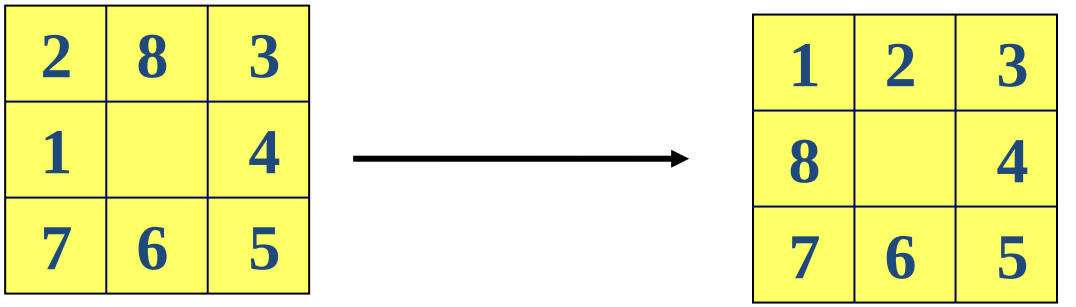
- (1)把初始节点S_0放入Open表中，$f(S_0)=g(S_0)+h(S_0)$；
- (2)如果Open表为空，则问题无解，失败退出；
- (3)把Open表的第一个节点取出放入Closed表，并记该节点为n；
- (4)考察节点n是否为目标节点。若是，则找到了问题的解，成功退出；
- (5)若节点n不可扩展，则转第(2)步；
- (6)扩展节点n，生成其子节点$n_i(i=1,2,\cdots)$，计算每一个子节点的估价值$f(n_i)(i=1,2,\cdots)$，并为每一个子节点设置指向父节点的指针，然后将这些子节点放入Open表中；
- (7)根据各节点的估价函数值，对Open表中的全部节点按从小到大的顺序重新进行排序；
- (8)转第(2)步。

全局择优算法框图


开始
把S_0放入OPEN表，计算$f(S_0)$
OPEN表为空表?
是
失败
否
把第一个节点(n)从OPEN表移至CLOSED表
节点n为目标节点?
是
成功
否
节点n可扩展吗?
否
是
扩展n，并计算所有后继节点的估价函数数值，并为每个后继节点设置指向n的指针.
把后继节点送入OPEN表，对其中的所有节点按估价函数从小到大排序。

•例子

八数码难题



(初始状态)

(目标状态)

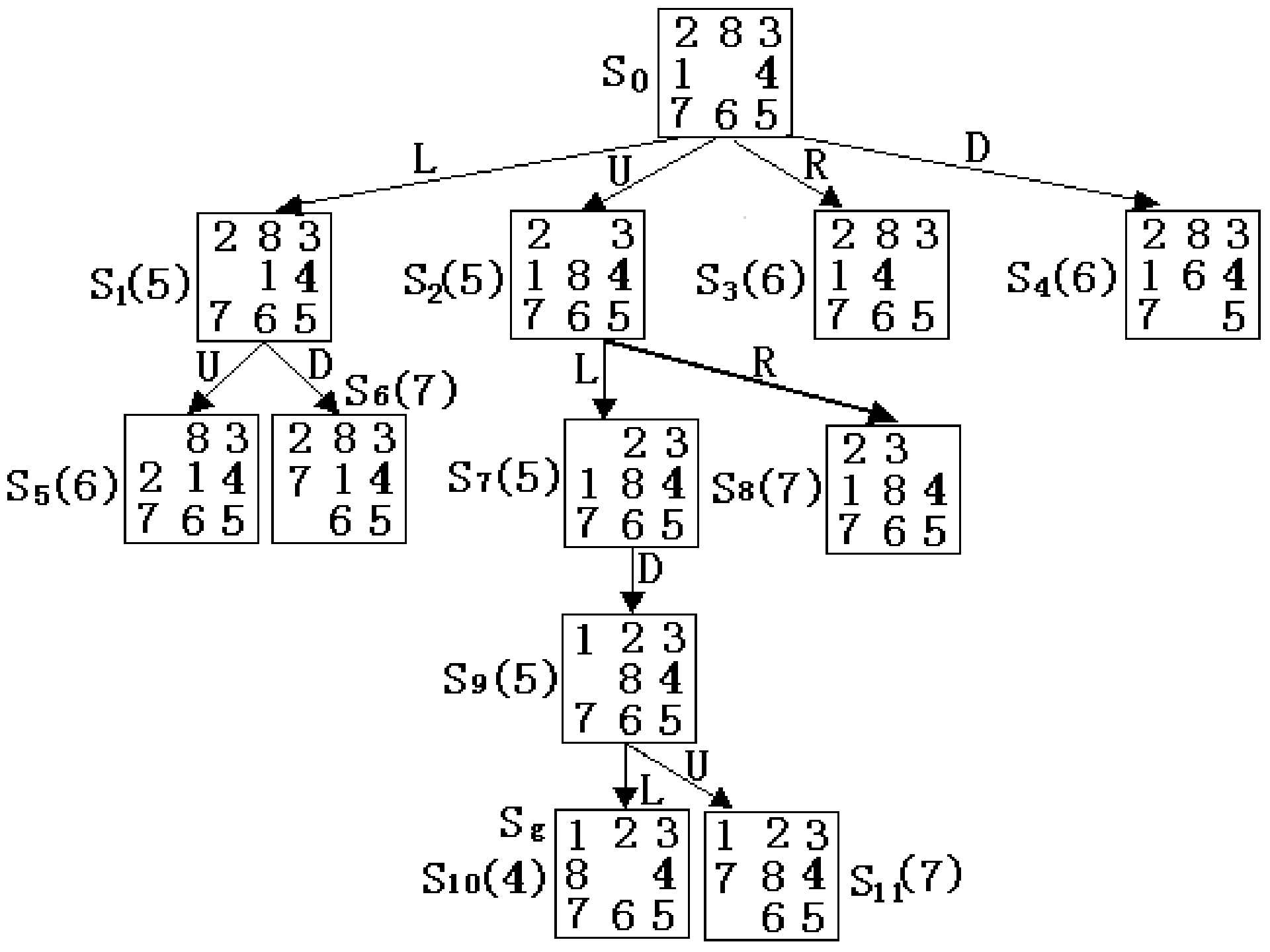
•例子

八数码难题

令：f(x)=d(x)+h(x)

其中，d(x)为节点x的深度，h(x)为节点x与目标节点之间不相同的数字个数,即“不在位”的数码个数。

搜索过程如下：


S0
2 8 3
1 4
7 6 5
L
U
R
D
S1(5)
2 8 3
1 4
7 6 5
S2(5)
2 3
1 8 4
7 6 5
S3(6)
2 8 3
1 4
7 6 5
S4(6)
2 8 3
1 6 4
7 5
U
D
S6(7)
S5(6)
8 3
2 1 4
7 6 5
2 8 3
7 1 4
6 5
L
R
S7(5)
2 3
1 8 4
7 6 5
S8(7)
2 3
1 8 4
7 6 5
D
S9(5)
1 2 3
8 4
7 6 5
L
U
Sg
S10(4)
1 2 3
8 4
7 6 5
1 2 3
7 8 4
6 5
S11(7)

•例子

问题的解为：

$S_0 \rightarrow S_2 \rightarrow S_7 \rightarrow S_9 \rightarrow S_{10}$

OPEN表：

S_0

S_1, S_2, S_3, S_4

S_2, S_3, S_4, S_5, S_6

$S_7, S_3, S_4, S_5, S_6, S_8$

$S_9, S_3, S_4, S_5, S_6, S_8$

$S_{10}, S_3, S_4, S_5, S_6, S_8, S_{11}$

$S_3, S_4, S_5, S_6, S_8, S_{11}$

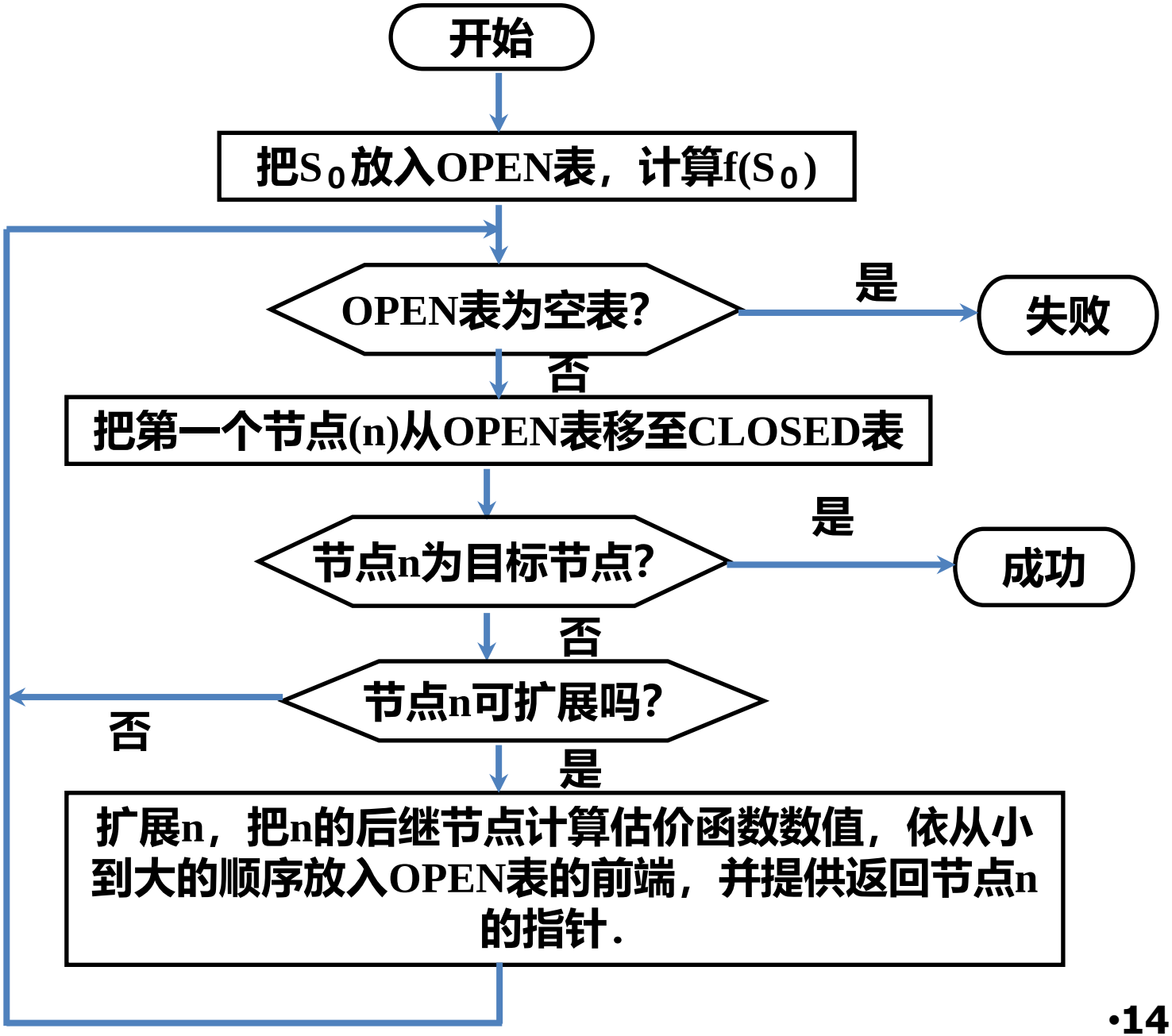
CLOSED表：

$S_0, S_1, S_2, S_7, S_9, S_{10}$

局部择优

- 将初始节点S0放入OPEN表中，计算估价函数f(s0)。
- 若OPEN表为空，则问题无解，退出。
- 将OPEN表中第1个节点N放入CLOSED表中，并按顺序编号为n。
- 若$N=S_g$，则已求得问题的解，退出。
- 生成N的所有子节点集合SUB，并且删除已在OPEN及CLOSED中存在的节点。
- 计算SUB中每个节点i的估价函数值f(i)，并对SUB中的所有节点按照估价值进行升序排序
- 将SUB中的所有节点放入OPEN的首部，并填入其父节点编号n，转（2）。

局部择优算法框图


开始
把S_0放入OPEN表，计算$f(S_0)$
OPEN表为空表?
是
失败
否
把第一个节点(n)从OPEN表移至CLOSED表
节点n为目标节点?
是
成功
否
节点n可扩展吗?
否
是
扩展n，把n的后继节点计算估价函数数值，依从小到大的顺序放入OPEN表的前端，并提供返回节点n的指针.

A*算法

- A*算法对扩展节点选择方法做了一些限制，选用了一个比较特殊的估价函数。
- 估价函数的定义：
  对节点n定义$f^*(n)=g^*(n)+h^*(n)$，表示从S开始约束通过节点n的一条最佳路径的代价。
  希望估价函数f定义为：$f(n)=g(n)+h(n)$
  ——g是g^*的估计，h是h^*的估计

A*算法

- 重排OPEN表是依据f(x)=g(x)+h(x)
- g(n)是对最小代价g*(n)的估计，且g(n)>0,g(x)>=g*(n)。
- h(x)为h*(x)的下界，即对所有的x存在h(x)≤h*(x)。

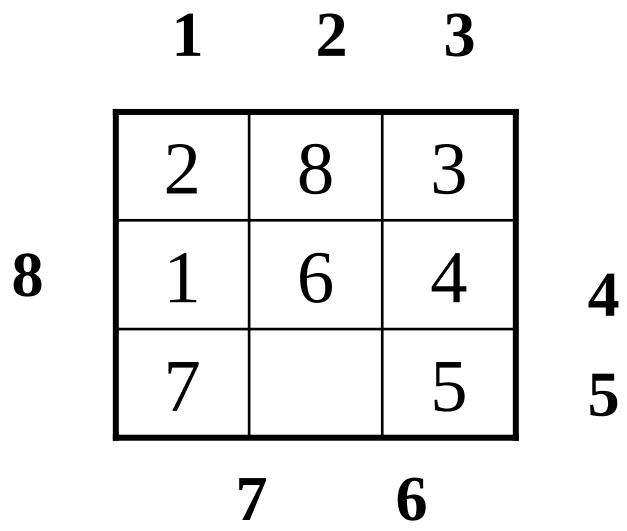
A*算法的最优性

- A*算法的搜索效率很大程度上取决于启发函数h(n)。一般来说，在满足h(n) ≤h*(n)的前提下，h(n)的值越大越好。h(n)的值越大，说明它携带的启发性信息越多，A*算法搜索时扩展的节点就越少，搜索效率就越高。A*算法的这一特性被称为最优性。

•例子

A*条件举例

- 八数码问题
- h1(n) = “不在位”的将牌数
- h2(n) =将牌“不在位”的距离和



1：1
2：1
6：1
8：2

•例子

A*条件举例

- h1(n) <=h2(n)<=h*(n)
- 所以它们做为h(n)时是A*算法，且h2优于h1。

作业

- 对于八数码问题重新定义估价函数：f(x)=d(x)+h(x)，其中，d(x)为节点x的深度，h(x)是所有棋子偏离目标位置的距离之总和，试分别用局部择优搜索及全局择优搜索画出搜索树

•应用

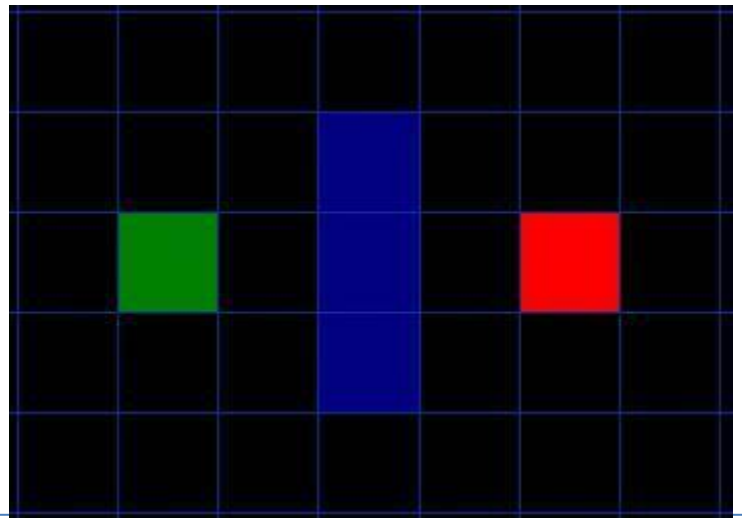
路径规划之A*算法

- 在游戏以及无人机、无人驾驶上航线规划上最常见
- 星际争霸等游戏
- SLAM应用

•应用

搜索区域

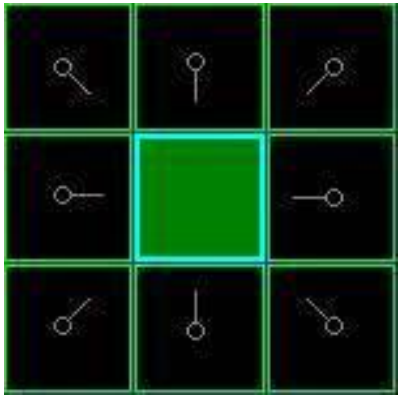
- 我们假设某人要从A点移动到B点，但是这两点之间被一堵墙隔开。如图1，绿色是A，红色是B，中间蓝色是墙



启发式搜索方法

开始搜索

- 从起点 A 开始，并把它就加入到一个由方格组成的 open list(开放列表) 中
- 把 A 从 open list 中移除，加入到 close list(封闭列表) 中， close list 中的每个方格都是现在不需要再关注的



•应用

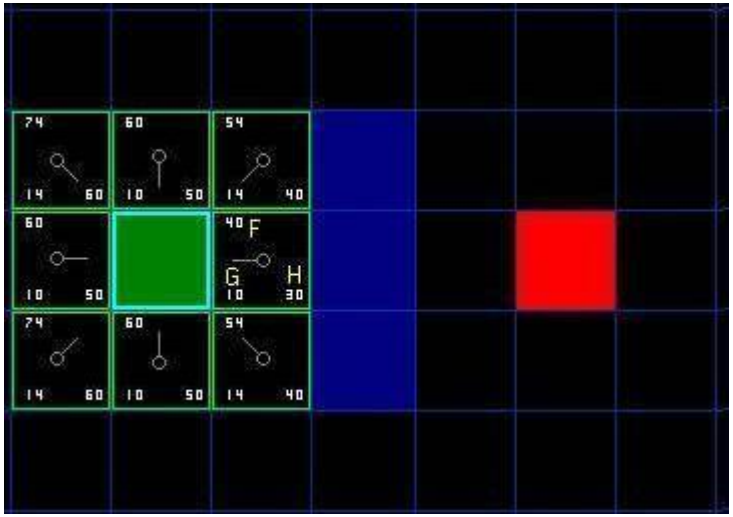
路径排序

- F = G + H
- G 是从起点 A 移动到指定方格的移动代价
- 有很多方法可以估算 H 值。这里我们使用 Manhattan 方法，计算从当前方格横向或纵向移动到达目标所经过的方格数，忽略对角移动，然后把总数乘以 10 。

•应用

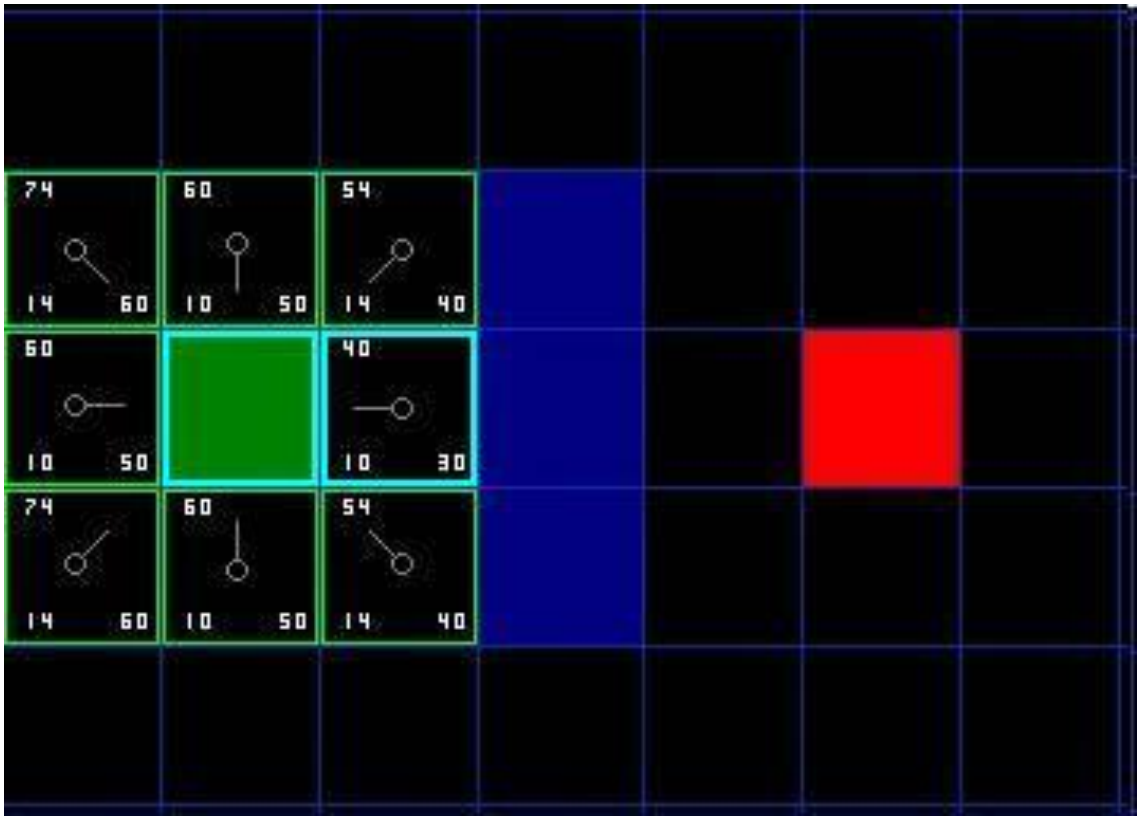
路径排序

- 把G和H相加便得到F。我们第一步的结果如下图所示。每个方格都标上了F，G，H的值，就像起点右边的方格那样，左上角是F，左下角是G，右下角是H



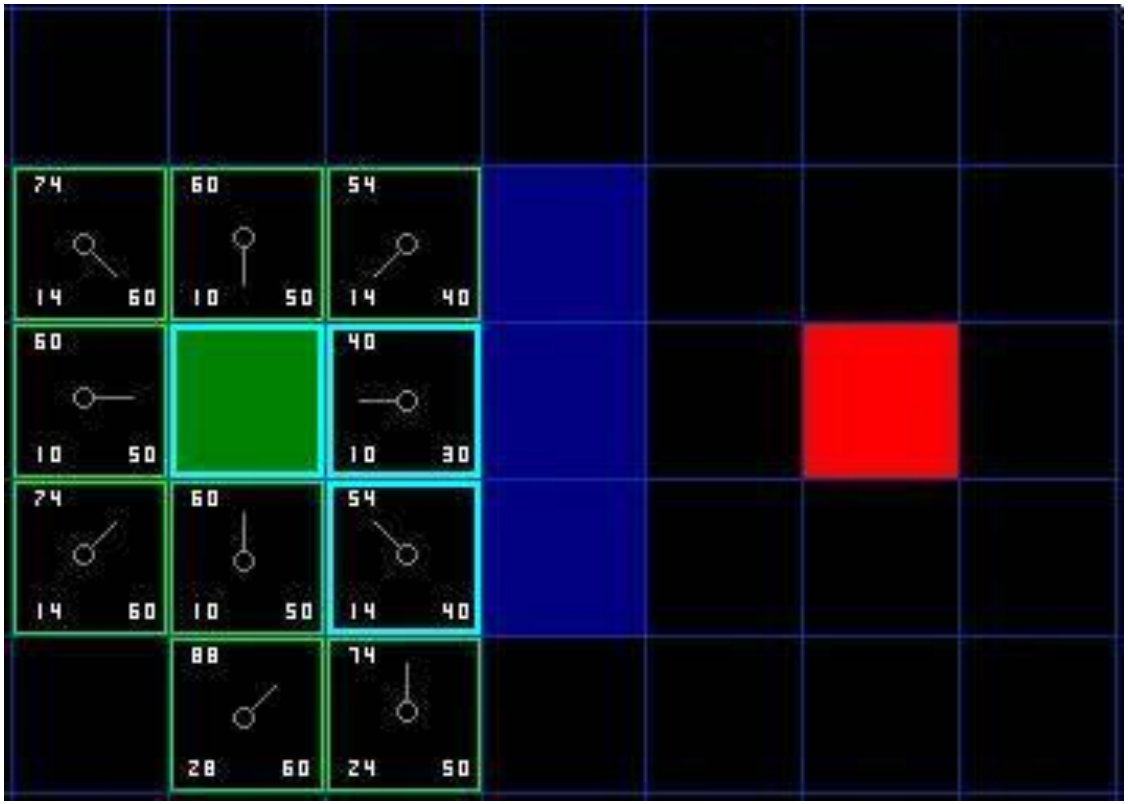
•应用

继续搜索



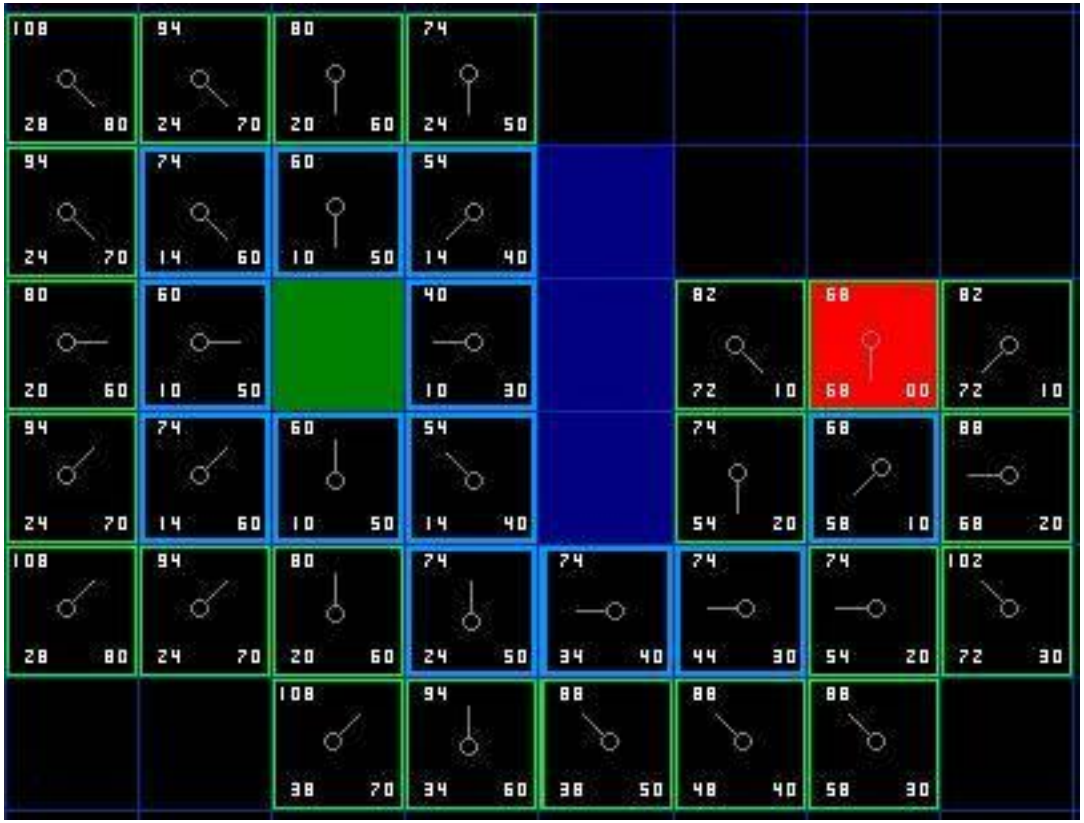
•应用

继续搜索


74
14 60
60
10 50
54
14 40
60
10 50
40
10 30
74
14 60
60
10 50
54
14 40
88
28 60
74
24 50

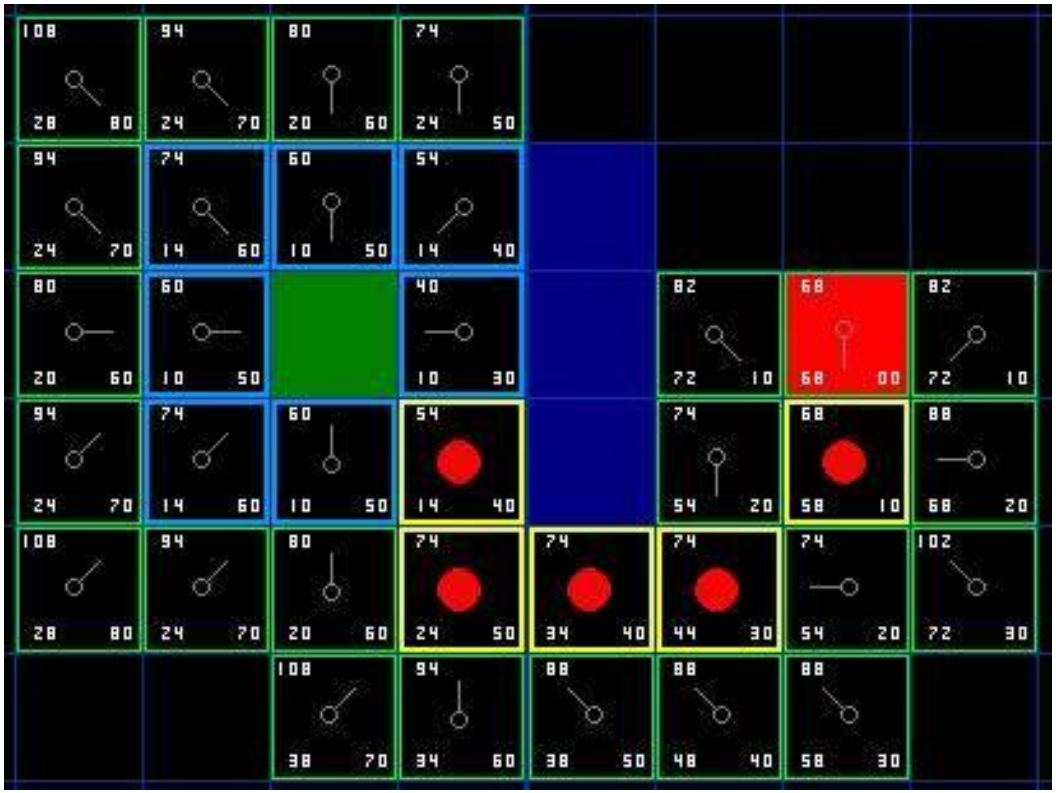
•应用

继续搜索


108 28 80
94 24 70
80 20 60
74 24 50
94 24 70
74 14 60
60 10 50
54 14 40
80 20 60
60 10 50
40 10 30
82 72 10
68 68 00
82 72 10
94 24 70
74 14 60
60 10 50
54 14 40
74 54 20
68 58 10
88 68 20
108 28 80
94 24 70
80 20 60
74 24 50
74 34 40
74 44 30
74 54 20
102 72 30
108 38 70
94 34 60
88 38 50
88 48 40
88 58 30

•应用

继续搜索


108 94 80 74
28 80 24 70 20 60 24 50
94 74 60 54
24 70 14 60 10 50 14 40
80 60 40 82 68 82
20 60 10 50 10 30 72 10 68 00 72 10
94 74 60 54 74 68 88
24 70 14 60 10 50 14 40 54 20 58 10 68 20
108 94 80 74 74 74 74 102
28 80 24 70 20 60 24 50 34 40 44 30 54 20 72 30
108 94 88 88 88
38 70 34 60 38 50 48 40 58 30

启发式搜索方法

算法总结

- 1. 把起点加入 open list 。
- 2. 重复如下过程：
- a. 遍历 open list ，查找 F 值最小的节点，把它作为当前要处理的节点。
- b. 把这个节点移到 close list 。
- c. 对当前方格的 8 个相邻方格的每一个方格？
- ◆ 如果它是不可抵达的或者它在 close list 中，忽略它。否则，做如下操作。
- ◆ 如果它不在 open list 中，把它加入 open list ，并且把当前方格设置为它的父亲，记录该方格的 F ， G 和 H 值。
- ◆ 如果它已经在 open list 中，检查这条路径（即经由当前方格到达它那里）是否更好，用 G 值作参考。更小的 G 值表示这是更好的路径。如果是这样，把它的父亲设置为当前方格，并重新计算它的 G 和 F 值。如果你的 open list 是按 F 值排序的话，改变后你可能需要重新排序。
- d. 停止，当你
- ◆ 把终点加入到了 open list 中，此时路径已经找到了，或者
- ◆ 查找终点失败，并且 open list 是空的，此时没有路径。
- 3. 保存路径。从终点开始，每个方格沿着父节点移动直至起点。

本章结束！