搜索技术
--启发式搜索算法

万永权

目录

CONTENTS

启发式搜索

01

盲目搜索策略会导致所需扩展的节点数很多，产生很多无用节点，搜索效率较低。

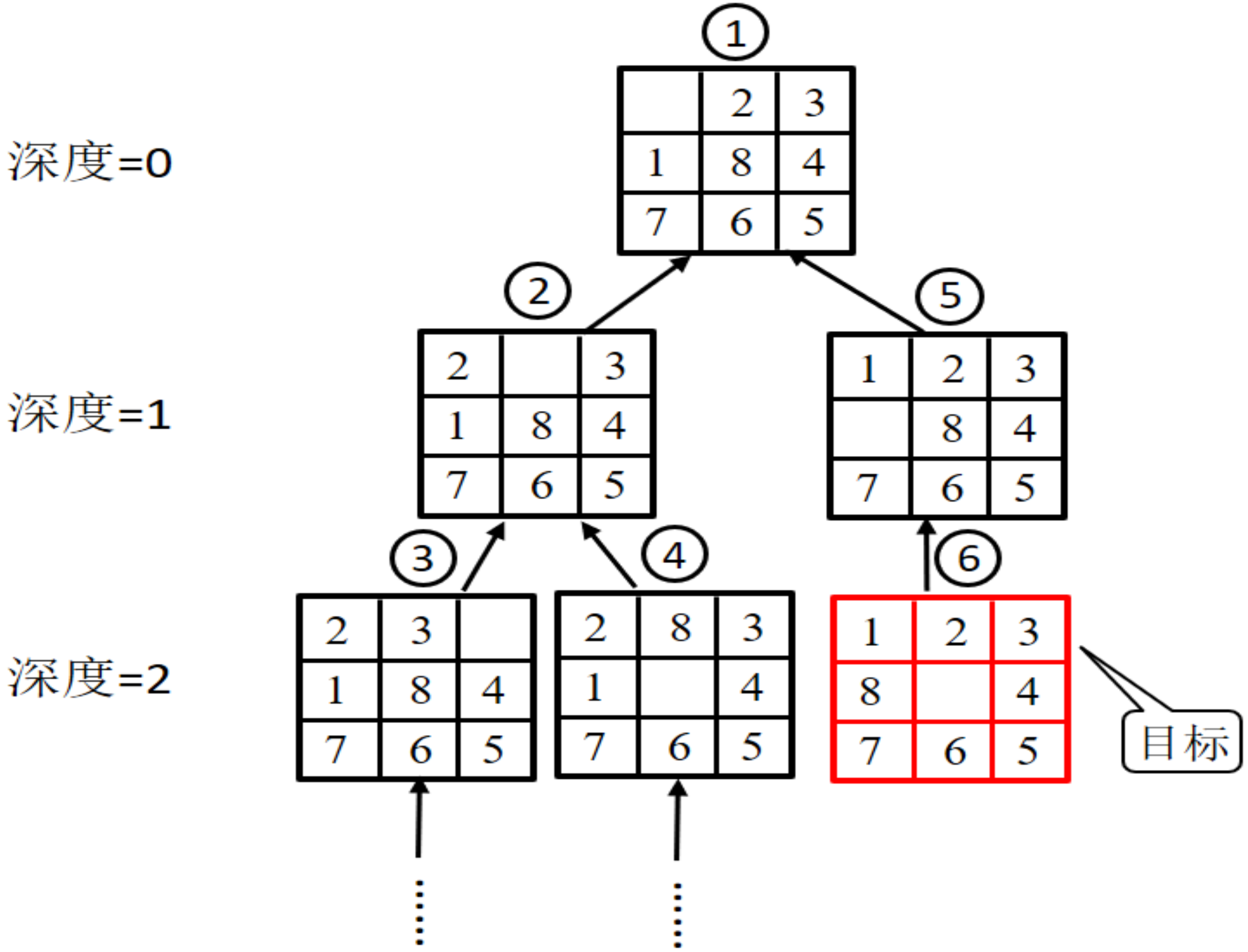


图3.4八数码问题中节点⑤比节点①的状态好

启发式搜索

◆启发式搜索（Heuristically Search）又称有信息搜索（Informed Search），利用问题拥有的启发信息来引导搜索，达到缩小搜索范围、降低问题复杂度的目的。

利用问题的某些控制信息（如解的特征）来引导搜索。

这种控制信息称为**搜索的启发信息。**

利用启发式信息定义节点的**启发函数** $h(n)$

特点

① 深度优先，效率高，无回溯。

② 但**不能保证得到最优解**。

使用的数据结构：**OPEN表**、**CLOSED表**

启发信息与估价函数

- 启发信息：
  - 在搜索过程中，用于决定要扩展的下一个节点的信息，即用于指导搜索过程且与具体问题求解过程有关的控制信息称为启发信息。
- 估价函数：
  - 决定下一步要控制的节点信息称作“最有希望”的节点，其“希望”的程度通常通过构造一个函数来表示，这种函数被称为估价函数。

  - 不能保证找到最优解
  - 强： 降低搜索工作量，但可能导致找不到最优解
  - 弱：一般导致工作量加大，极限情况下变为盲目搜索

估价函数（evaluation function）

如何评价一个节点在最佳路径上的可能性呢? 我们采用**估价函数**来进行估计:

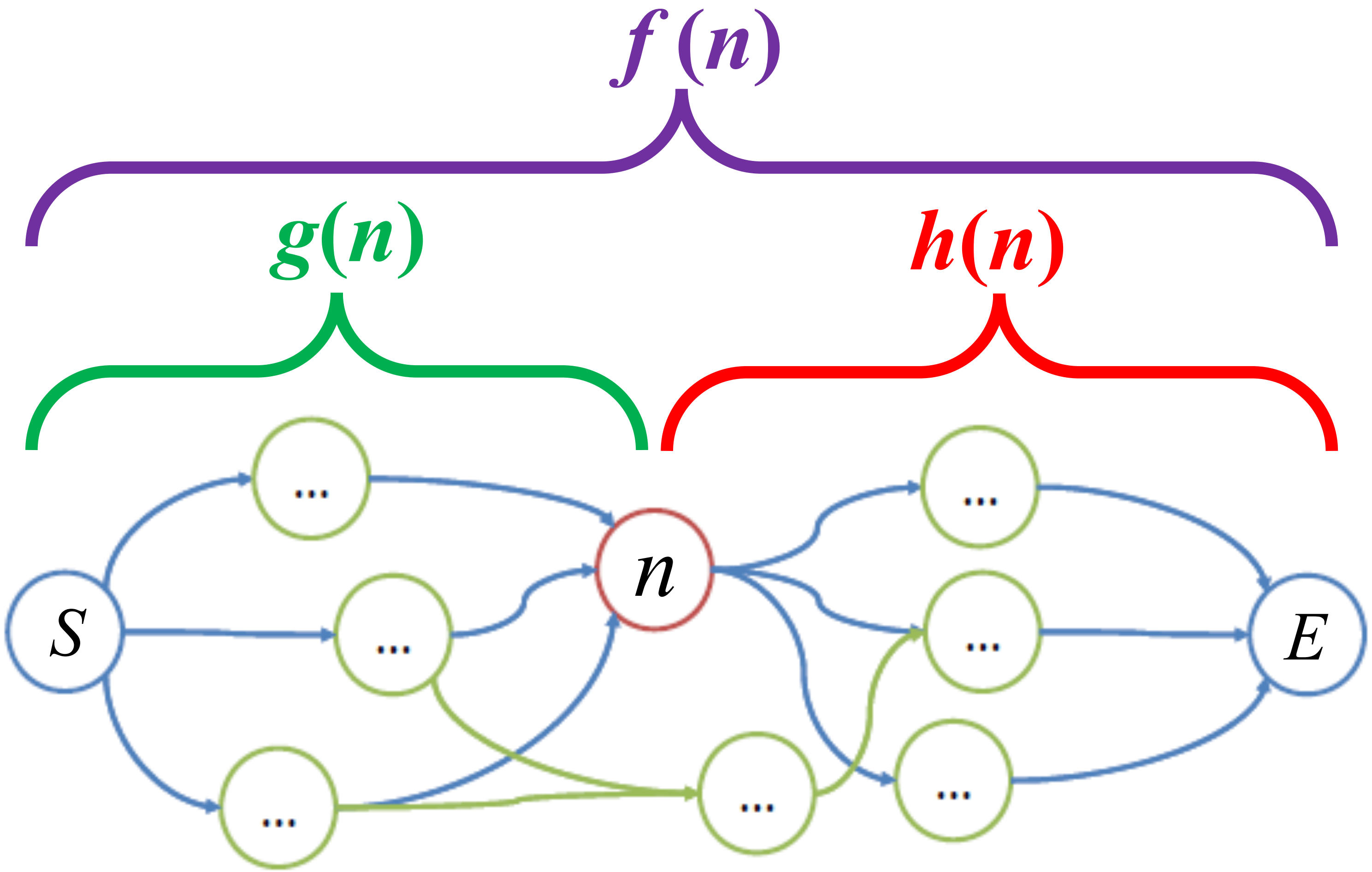
***f*(n)** 定义为从初始节点 s 出发、经过节点 n 、到达目标节点的**最佳路径**代价值的**估计值.**

一般形式：$\boldsymbol{f(n) = g(n) + h(n)}$.　其中，$n$为当前节点，即待估价节点。

(1) $\boldsymbol{g(n)}$ 为从初始节点到节点n 的路径**实际代价**；

(2) $\boldsymbol{h(n)}$ 为从节点 n 到目标节点的**最佳路径**的**估计代价**

估价函数 $f(n) = g(n) + h(n)$



- $g(n)$： 为从初始状态S到达节点n 的最佳路径上代价的**实际值**
- $h(n)$： 从节点 n 到目标状态E的最佳路径上代价的**估计值**，称为**启发函数**。
- $f(n)$ 为从初始状态S经过节点n到达目标状态E的最佳路径上代价的**估计值**，称为**评价函数**。

启发函数的设计

启发函数

如何定义一个启发函数呢？

启发函数并没有固定的模式，需要具体问题具体分析。

通常可以参考的思路有

① 一个节点到目标节点的距离或差异的度量；

② 一个节点处在最佳路径上的概率；

③ 根据经验的主观打分。

启发函数

八数码问题

- $f_2(S)$ = 没有处于目标状态中正确位置数字的数量
  （相当于给出了当前状态与目标状态的距离）

1	2	3
8		4
7	6	5

目标状态

f_2

1	3	2
8		4
5	6	7

= 4

当前状态

启发式搜索策略

每次移动的时候，**错位数码牌**的个数要小于交换前**错位数码牌**个数。

错位数码牌的个数是指每个数码的位置与最终状态的对比，如果位置不相同，则说明此数码不在目标位置。

图中红色字体标识的数码为正确位置数码，由此我们可以发现下图中左边初始图案不在目标位置的数码个数为4个。

2	8	3
1	6	4
7		5

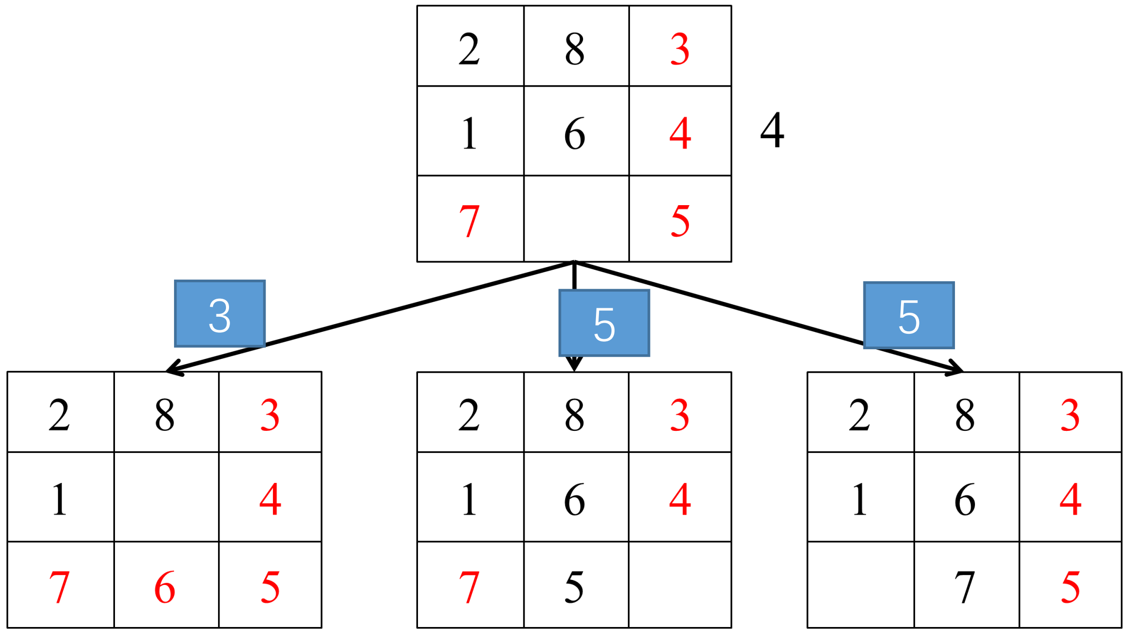
→

1	2	3
8		4
7	6	5

八数码游戏寻找正确位置数码个数

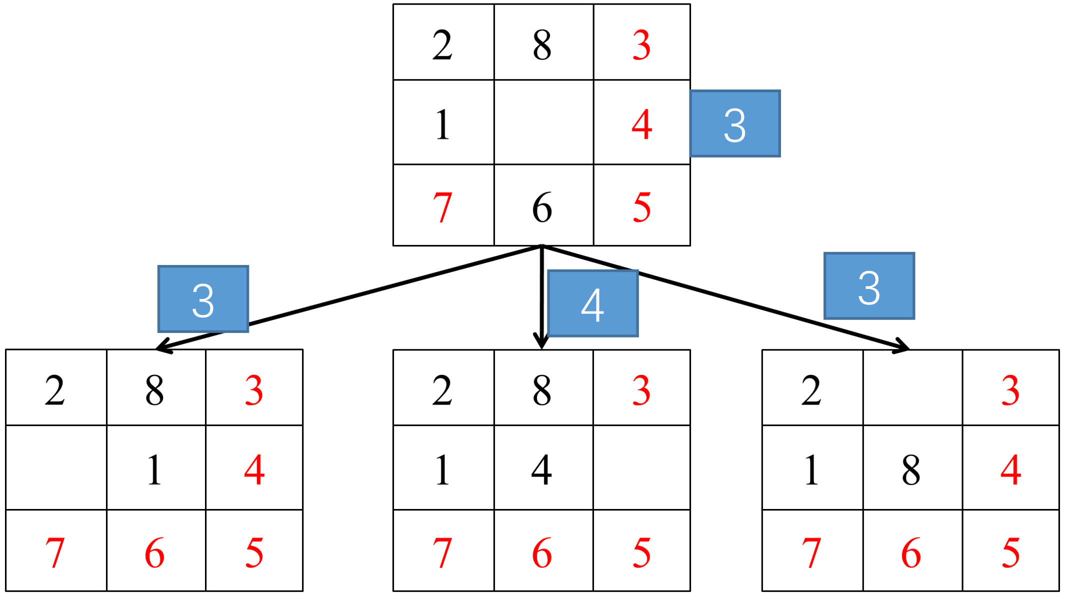
启发式搜索策略

- 由下图所示可得，不在正确位置数码个数小于等于4的只有左下方的格局，那么下一步选择的就是扩展左下方的状态。



八数码游戏

再次调用此算法如下图所示：



八数码游戏

这样一步一步地进行，最终即可得到最终格局。

启发函数

八数码问题

- $f_3(S)$ =不在目标位置的数字距离目标位置水平距离和垂直距离之和

该函数给出了一个更好的距离评估

1	2	3
8		4
7	6	5

目标状态

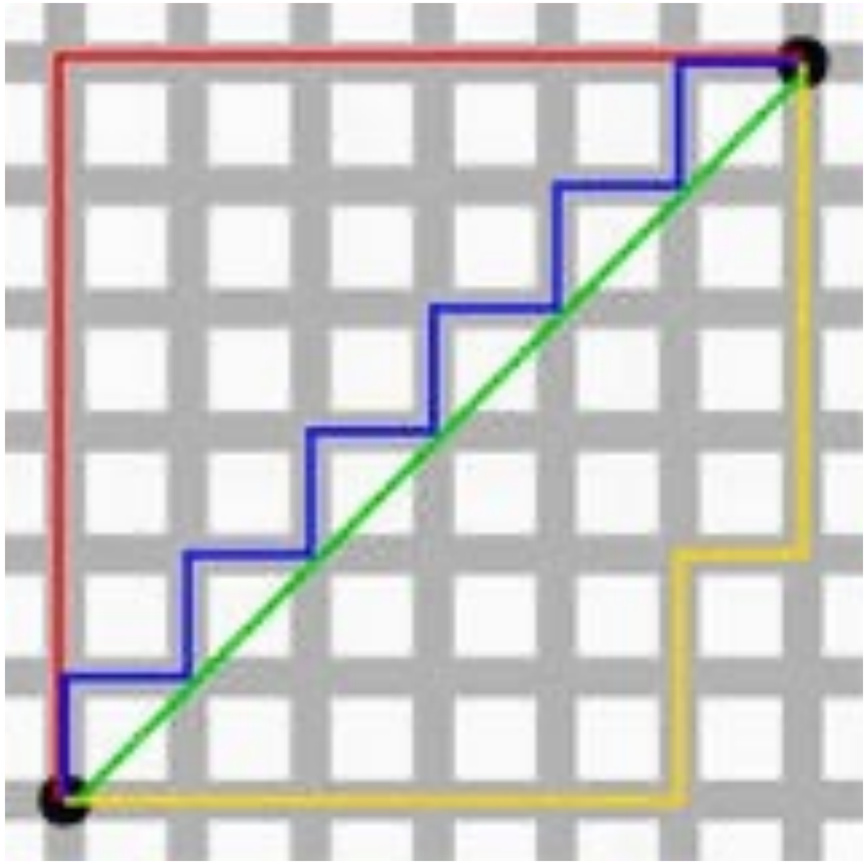
$$f_3\left(\begin{array}{|c|c|c|}\hline 1 & \boxed{3} & \boxed{2} \\ \hline 8 & & 4 \\ \hline \boxed{5} & 6 & \boxed{7} \\ \hline\end{array}\right) = 1+1+2+2=6$$

当前状态

曼哈顿距离，教材p84

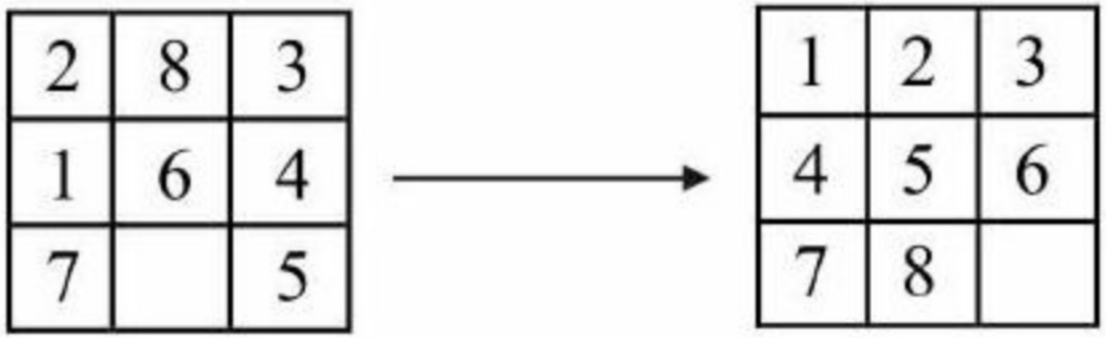
曼哈顿距离

- 两点在南北方向上的距离加上在东西方向上的距离，
- 即$d(i,j)=|x_i-x_j|+|y_i-y_j|$。



曼哈顿距离计算

- 计算左图中各数字的曼哈顿距离
  - 1.当前距离；
  - 2.空格左移后的距离；
  - 3.空格上移后的距离；

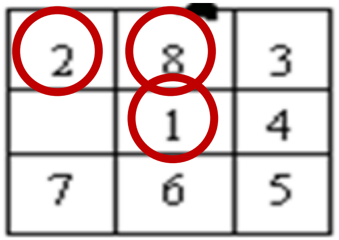


（a）初始状态　　（b）目标状态

(a)为目标状态，分别计算图（b）中的 *h1* and *h2*

1	2	3
8		4
7	6	5

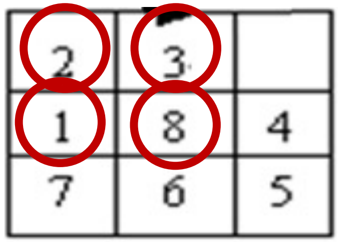
(a) goal state



(b)

1	2	3
8		4
7	6	5

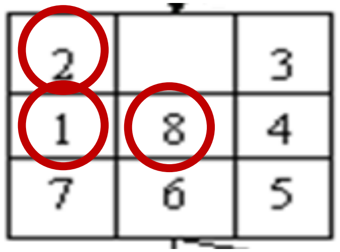
(a) goal state



(b)

1	2	3
8		4
7	6	5

(a) goal state



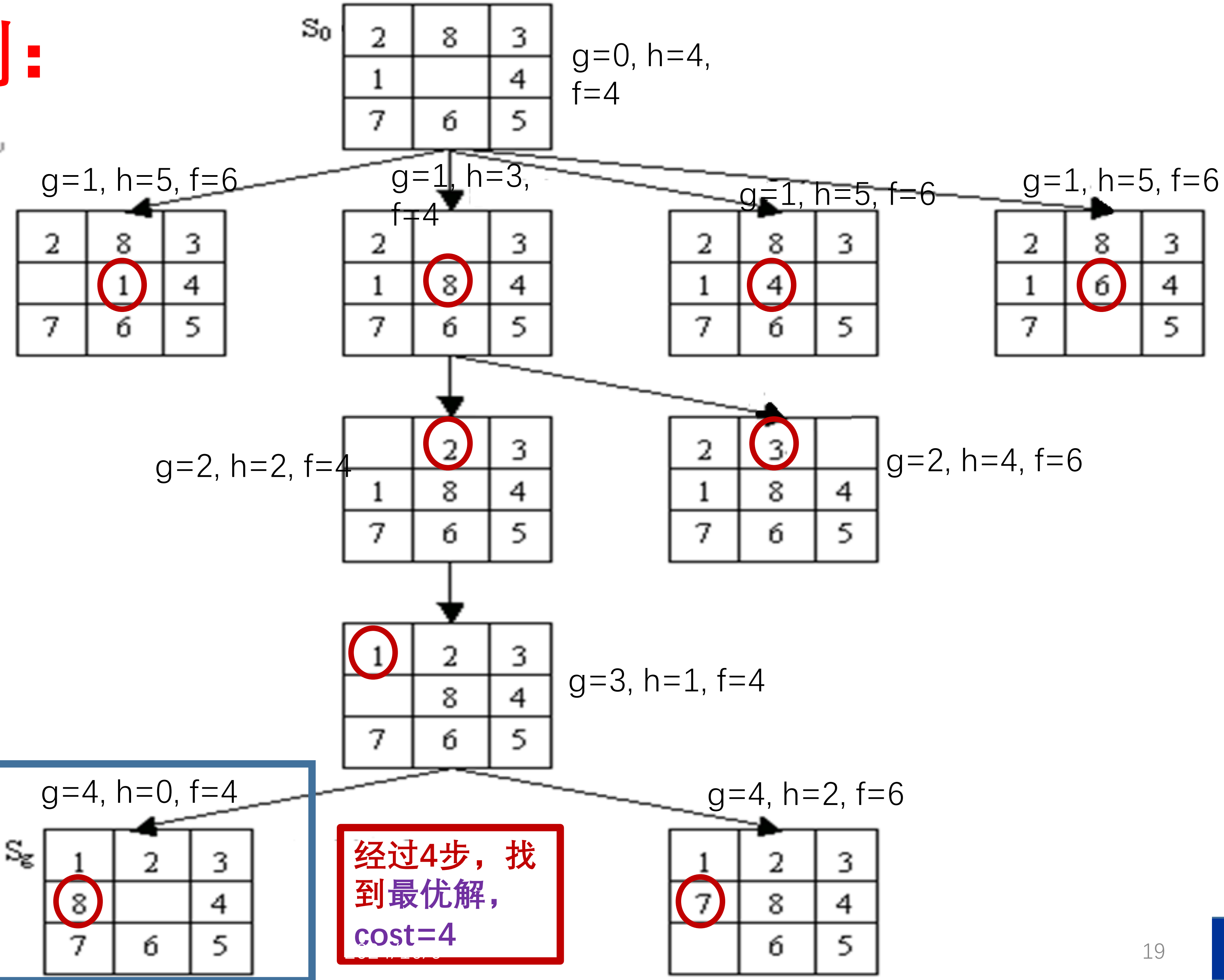
(b)

- h1(n) = “错位数码牌”的个数
- h2(n) = 所有数码牌与其目标位置之间的曼哈顿距离之和。

1	2	3
8		4
7	6	5

goal state

例：



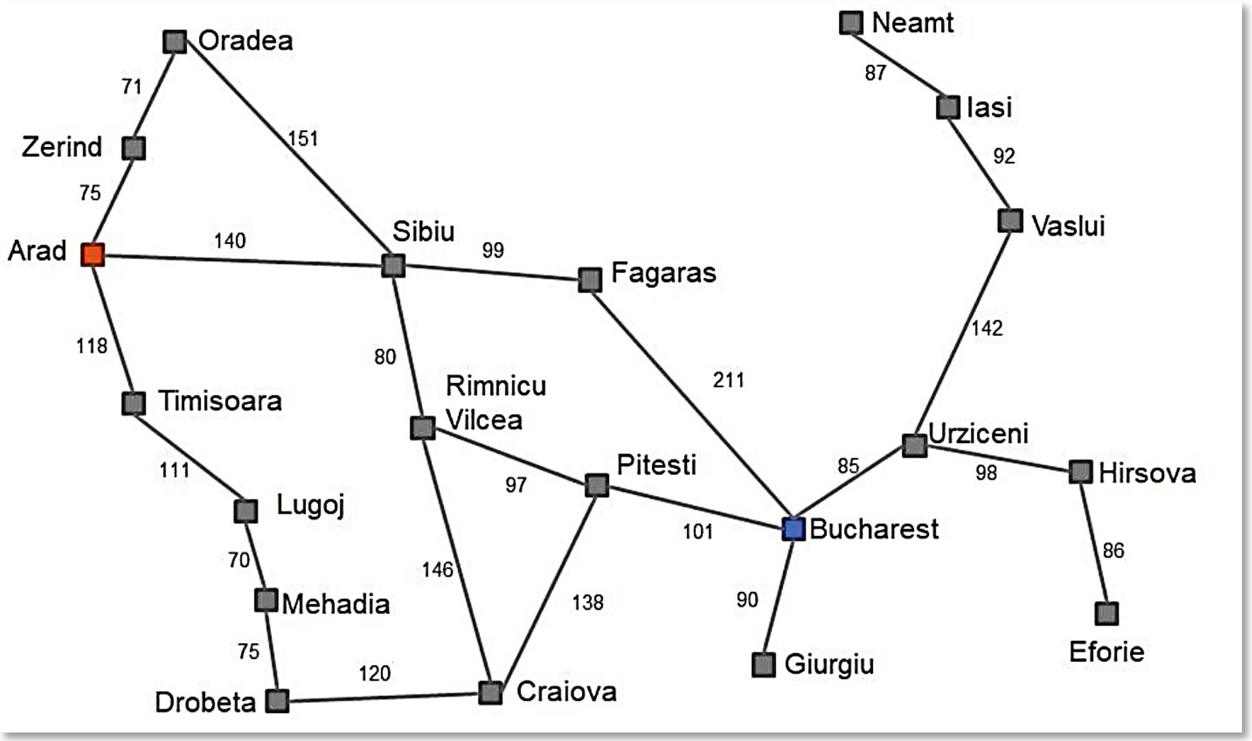
h(n)=所有数码牌与其目标位置之间的曼哈顿距离之和

经过4步，找到最优解，cost=4

A*算法应用

罗马尼亚度假问题

启发信息



启发信息： 到Bucharest的直线距离

Arad	366
Bucharest	0
Craiova	160
Drobeta	242
Eforie	161
Fagaras	176
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	100
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A*算法应用

罗马尼亚度假问题

初始状态

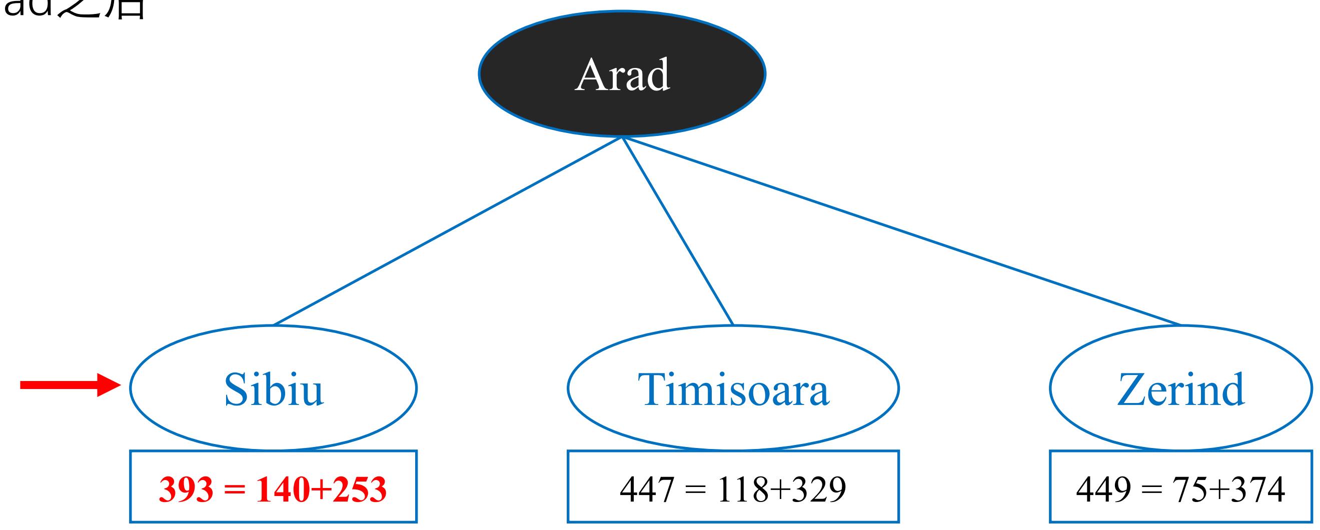
Arad

366 = 0+366

A*算法应用

罗马尼亚度假问题

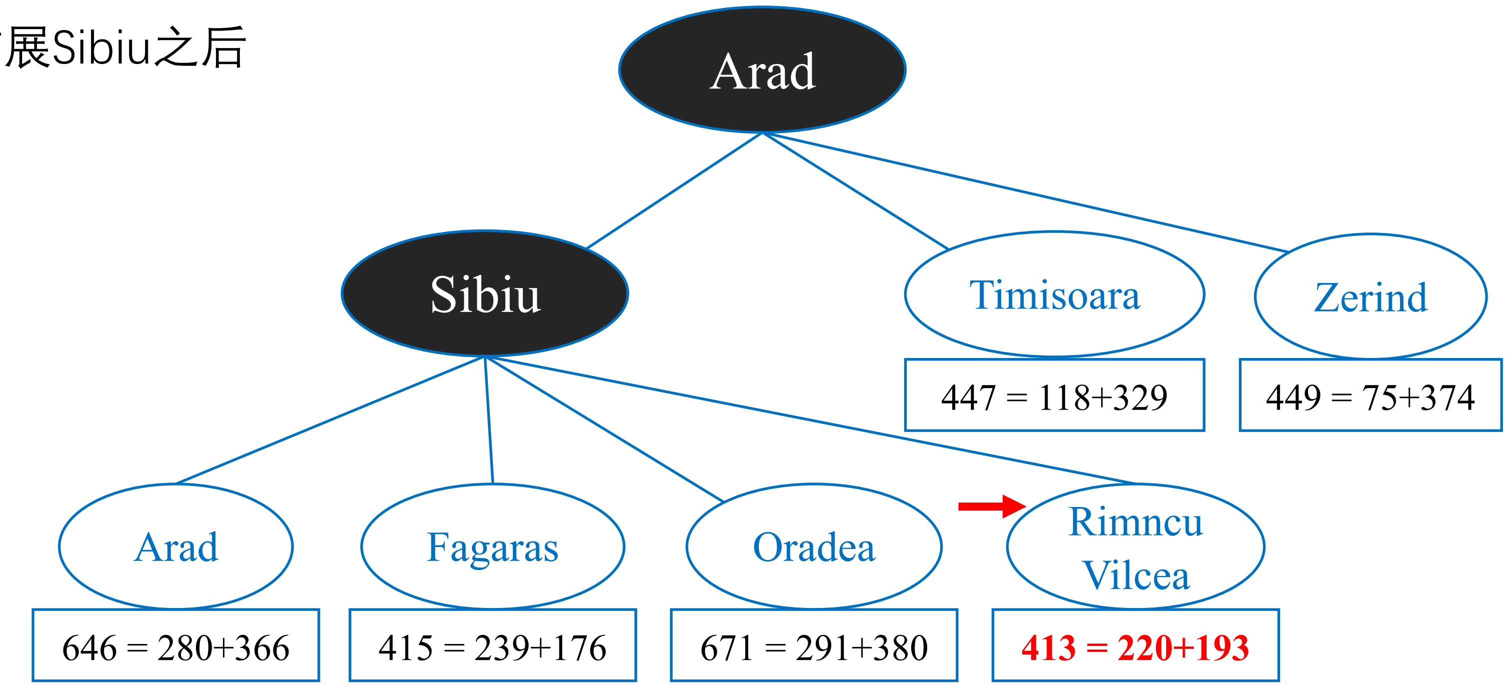
扩展Arad之后



A*算法应用

罗马尼亚度假问题

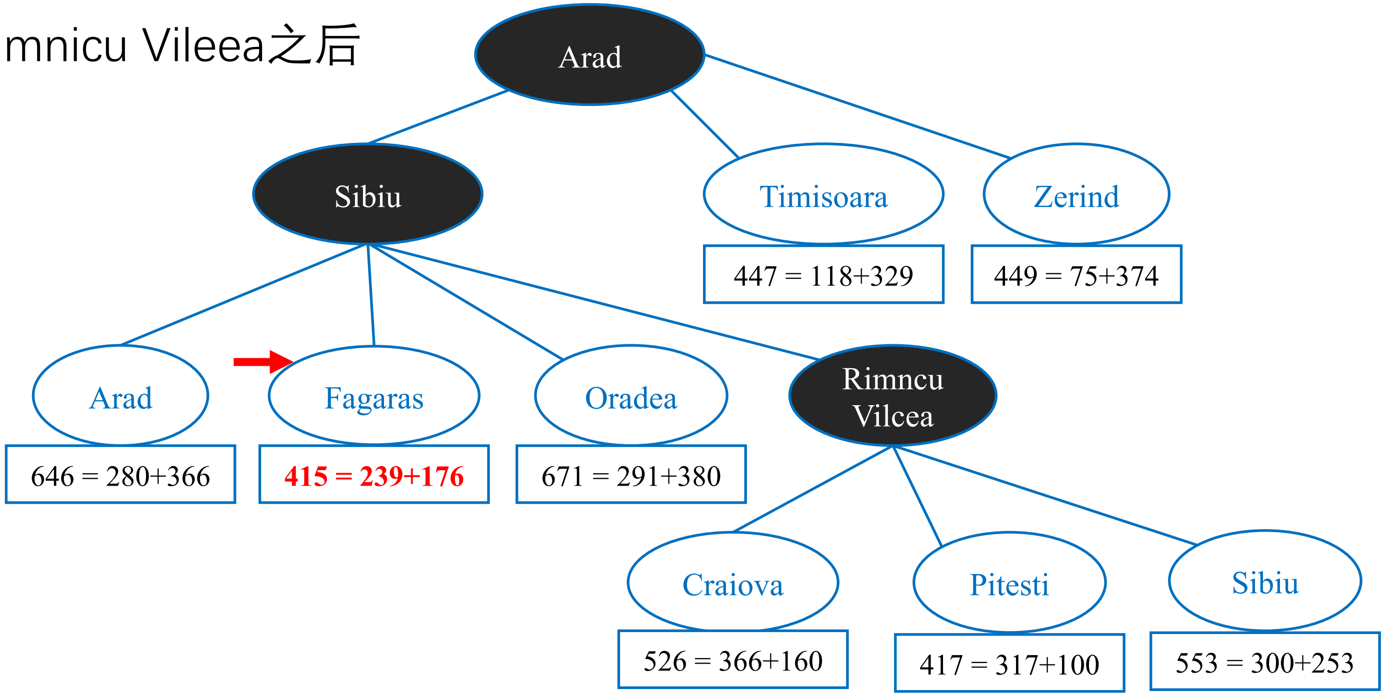
扩展Sibiu之后



A*算法应用

罗马尼亚度假问题

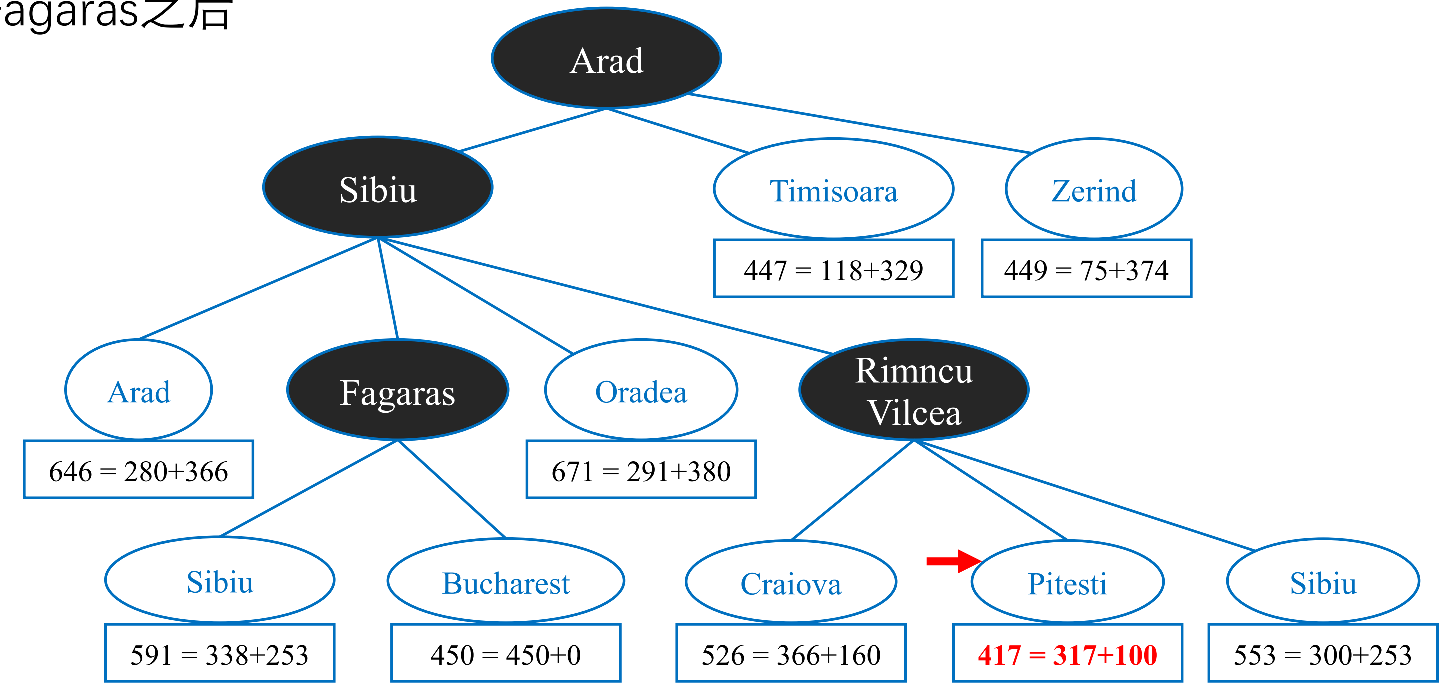
扩展Rimnicu Vileea之后



A*算法应用

罗马尼亚度假问题

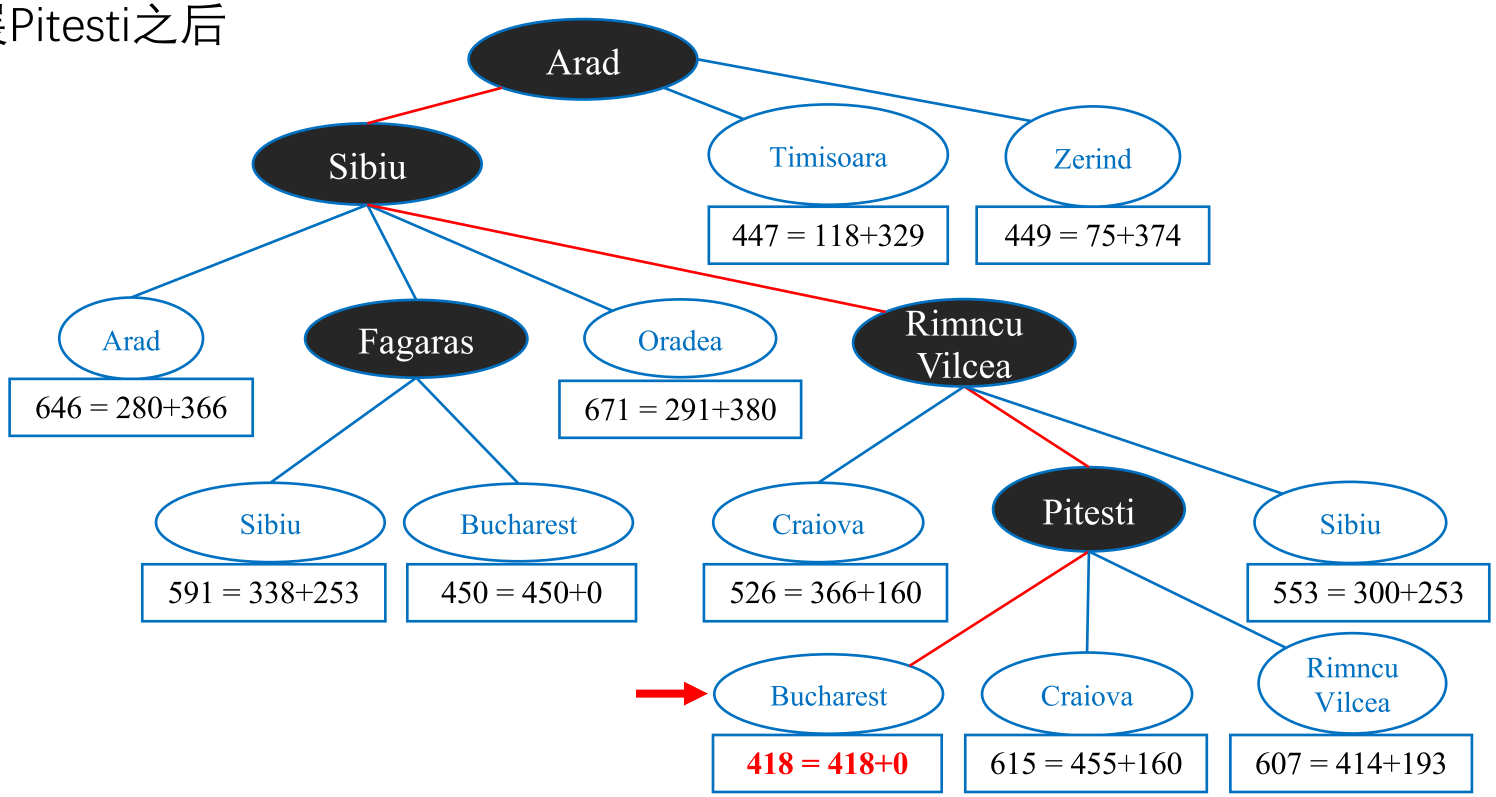
扩展Fagaras之后



A*算法应用

罗马尼亚度假问题

扩展Pitesti之后



用A* 算法解决“修道士与野人渡河”问题

在河的左岸有K个传教士、K个野人和1条船，传教士们想用这条船将所有成员都从河左岸运到河右岸去，但有下面条件和限制：

(1) 所有传教士和野人都会划船。

(2) 船的容量为r，即一次最多运送r个人过河。

(3) 任何时刻，在河的两岸以及船上的野人数目不能超过传教士的数目，否则野人将吃掉传教士。

(4) 允许在河的某一岸或船上只有野人而没有传教士。

(5) 野人会服从传教士的任何过河安排。

请采用A*算法搜索出一个确保全部成员安全过河的合理方案。

A* 算法解决“修道士与野人渡河”问题

Step1 设计状态空间表示，采用三元组形式

表示一个状态，令S=(m, c, b)，其中：

- m为未过河的传教士人数，m∈[0, K]，已过河的传教士人数为K－m。
- c为未过河的野人数，c∈[0, K]，已过河的野人数为K－c。
- b为未过河的船数，b∈[0, 1]，已过河的船数为1－b。
- 初始状态为S_0 =（K，K，1），表示全部成员及船都在河的左岸，目标状态为Sg =（0,0,0），表示全部成员及船都已到达了河右岸。

用A* 算法解决“修道士与野人渡河”问题

Step2 设计**操作集合**，即**过河操作**。

- 设计两类操作算子：
  - L_{ij}操作表示将船从左岸划向右岸，第一下标i表示船载的传教士人数，第二下标j表示船载的野人数；
  - Rij操作表示将船从右岸划回左岸，下标的定义同前。
  - 这两类操作需满足如下限制： (1) $1\leq i+j\leq r$;　　(2) $i\neq 0$时，$i\geq j$。
- **假设K=5，r=3**，则合理的操作共有16种，其中
  - 船从左岸到右岸的操作有：L_{01}、L_{02}、L_{03}、L_{10}、L_{11}、L_{20}、L_{21}、L_{30};
  - 船从右岸到左岸的操作有：R_{01}、R_{02}、R_{03}、R_{10}、R_{11}、R_{20}、R_{21}、R_{30}。

用A* 算法解决“修道士与野人”问题

Step3 设计满足A* 算法的启发函数

- 在初始状态(5,5,1)下，若不考虑限制(3)--野人吃人，则至少要操作9次

  （初始时，**船与人在河同侧**，每次运过3人过去，然后1人回来，重复4.5个来回）

- 相当于：1个人固定作为船夫,每摆渡一次,只运2个人过河（即往返一趟，运送2人过河）。

初始状态：	10人	河	0人
摆渡1：	7人	3人 →	0人
摆渡2：	7人	← 1人	2人
摆渡3：	5人	3人 →	2人
摆渡4：	5人	← 1人	4人
摆渡5：	3人	3人 →	4人
摆渡6：	3人	← 1人	6人
摆渡7：	1人	3人 →	6人
摆渡8：	1人	← 1人	8人
摆渡9：	0人	2人 →	8人

用A* 算法解决“修道士与野人”问题

Step3 设计满足A* 算法的启发函数

- ◆**是否可以令h(x)=m+c？**
- ◆稍分析就可以发现，**不可以**，因为 h(x)=m+c 不满足 h(x)≤h*(x) 。

  如: 对状态x=(1,1,1)，h(x)=m+c=2，而此时最短路径上的代价h*(x) =1，即只需1步就可完成。

- ➢按要求，**应该： h(n) ≤ h*(n)**
- ➢但此刻，**h*(x)** =1**< h(x)** =2，不满足A*算法的条件，
- ➢实际上：h*(x)应该有一个上限：h*(x) ≤ m+c，即过河次数不高于2K次，因为一共才2K个人，摆渡次数不可能超过2K次，取h*(n) = m+c。

用A* 算法解决“修道士与野人”问题

Step3 设计满足A* 算法的启发函数

分情况讨论（r=3）：

- 假设**船与人同在左岸**，b=1（船未过河），状态为(m,c,1)。
- 不考虑“野人会吃人”的约束条件，当最后一次恰好3人同船过河时，效率最高，**单独算1次摆渡**。
- 剩下m+c-3个人运过河，需要运送 $\frac{(m+c-3)}{2} * 2 = \mathrm{m} + \mathrm{c} - 3$次，

故：**一共需要运送/摆渡 m+c-3+1 = m+c-2 次（单向摆渡次数）**。

用A* 算法解决“修道士与野人”问题

Step3 设计满足A* 算法的启发函数

分情况讨论：

- 假设船在右岸，即船与人在河的不同侧，b=0，**初始状态为(m,c,0)**。
- 则首先需要额外有1个人把船划从右岸划回左岸，消耗1次，
- 同时左岸人数增多1，即总人数变为： m+c+1
- 转变成第一种情况，即：**(m+c,0)** ↔ **(m+c+1, 1)**
- **第一种情况**的初始状态为（ **m+c,1**），一共需要运送 **m+c-2 次**
- 现在，用m+c+1代替上式中的 m+c，则一共需要运送 m+c-1次
- 再加上最开始“消耗1次”，则**第二种情况**共需要运送(m+c-1) **+1** = **m+c 次**。

用A* 算法解决“修道士与野人”问题

Step3 设计满足A* 算法的启发函数

两种情况结合，得到：

$$h(n) = \begin{cases} m + c - 2, & b = 1 \\ m + c, & b = 0 \end{cases}$$

可写作：

$$\boldsymbol{h(n) = m + c - 2b}$$

此时，$h(n)=m+c-2b \le h^*(n)=m+c$，满足A*算法的条件。

图3.7 采用A*算法解决传教士与野人渡河问题示例