人工智能 一种现代方法

——第三章读书报告

读书笔记的内容及组织形式

本次阅读报告我选择的是研讨材料中的英文教材：人工智能——一种现代方法，为了便于交流，一些名词的翻译参考了中文版教材。写这篇读书笔记的目的是为了自己以后学习 AI 知识时方便查阅，因此整理了书中主要的概念及知识点，并尽可能地使用方便自己理解地语言。对于一些细节问题没有深究，以后会继续完善。**读书报告的组织形式是知识点一——相关习题——知识点二——相关习题——知识点三——相关习题**。Formulation 部分的习题方法大致相似，因此挑选了一些具有代表性的。

知识点一：求解问题的形式化(formulation)

问题的形式化一般需要五个部分：**状态、初始状态、行动、转移模型、目标测试**，若问题属于寻找最短路径的类型，则再加上一项**路径耗散**。其中转移模型一般用函数 RESULT(s,a)描述：在状态 s 下执行行动 a 后达到的状态，从某一状态通过单步行动可以到达的状态集合称为**后继状态**。

eg：真空吸尘器世界问题（教材中一个简单的例子，摘抄下来方便理解）

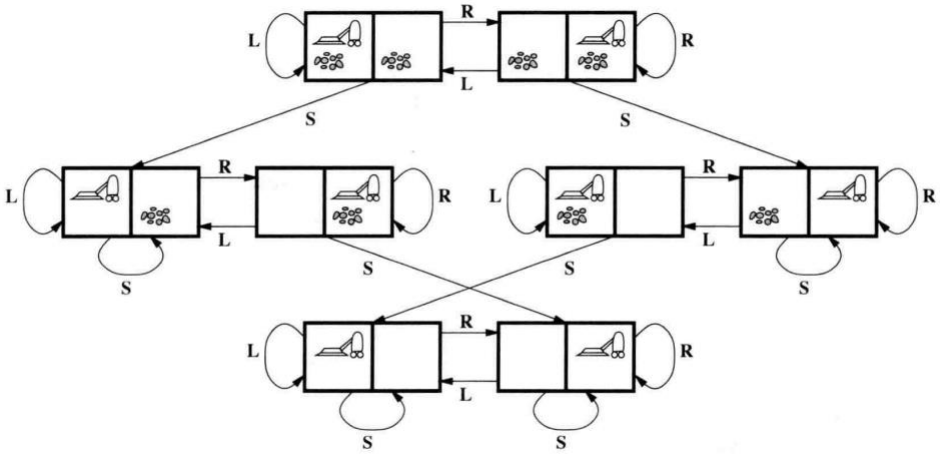
问题描述：有两个格子，可能有灰尘也可能没有，有一个吸尘器可以在两个格子间移动，现在需要吸尘器将这两个格子都打扫干净。则可将这个问题形式化如下：

状态(state)：该问题的状态有两个变量决定，分别是 Agent（吸尘器）的位置和灰尘的位置。因此该问题可能的状态有$2 \times 2^2 = 8$个。

初始状态(initial state)：任何状态都可以作为初始状态。

行动：Agent 左移、右移和吸尘。

转移模型(translation model)：如图：



目标测试(goal test)：本问题中，目标测试即为检测两个格子是否被清理干净。
路径耗散(step cost)：如果需要吸尘器移动次数最少，则需要计算路径耗散。本问题中，路径耗散即为 Agent 移动的次数。

1. 1 相关题解答

3.1 Explain why problem formulation must follow goal formulation.

答：因为我们在实际问题形式化的时候需要根据我们的目标忽略掉一些我们不关心的因素，如果我们不先进行目标的形式化，我们就不知道哪些因素是可以忽略的。（例如我们在寻找到达目的地的路径时，如果目标时路程最短，则我们可以忽略该条路径上的交通拥堵情况；但如果我们的目标是时间最短，该因素就不能忽略，也成为影响状态的一个变量）

3.2 Your goal is to navigate a robot out of a maze. The robot starts in the center of the maze

ses 113

facing north. You can turn the robot to face north, east, south, or west. You can direct the robot to move forward a certain distance, although it will stop before hitting a wall.

a. Formulate this problem. How large is the state space?

b. In navigating a maze, the only place we need to turn is at the intersection of two or more corridors. Reformulate this problem using this observation. How large is the state space now?

c. From each point in the maze, we can move in any of the four directions until we reach a turning point, and this is the only action we need to do. Reformulate the problem using these actions. Do we need to keep track of the robot's orientation now?

d. In our initial description of the problem we already abstracted from the real world, restricting actions and removing details. List three such simplifications we made.

答：a. 可以将该迷宫定义为一个中心在(0,0)，从(-1,1)到(1,1)的正方形。则该问题的状态为机器人所在的坐标(x,y)

初始状态：(0,0)，面朝北

行动：前进距离 d

转移模型：当前状态为面朝北，则$y+d$
当前状态为面朝南，则$y-d$
当前状态为面朝东，则$x+d$
当前状态为面朝西，则$y-d$

目标检测：坐标(x,y)，$|x|>1$或$|y|>1$

路径耗散：所走 d 之和

由于移动的距离 d 是一个连续量，因此每次可能到达的坐标(x,y)也是连续的，因此状态空间无限大。

b. 由于机器人只会停留在路口处，因此该问题的状态应记录机器人所在的路口，以

及它所面向的方向。将迷宫出口标记为出口节点

初始状态：迷宫中心，面朝北

转移模型：如果面前有另一个路口，则前往下一个路口；否则转向下一个方向

目标检测：检测是否处于出口节点

路径耗散：总移动距离

迷宫中的路口数是离散的，因此状态空间也是有限的。如果有 n 个路口，则状态空间为 4n。

c. 不需要再记录机器人面对的方向了，因为机器人的 action 是随机的，不需要根据当前状态判断可用的 action。

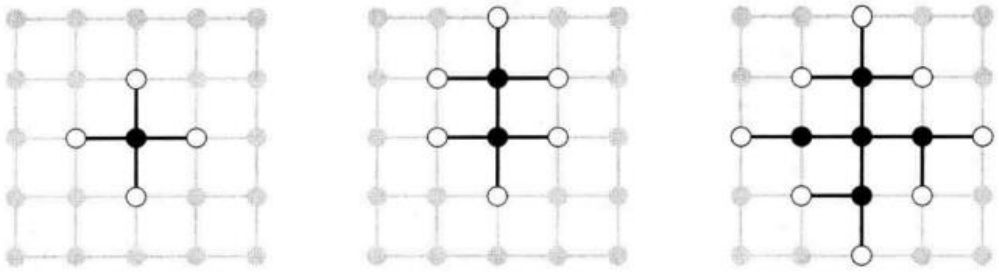
d. i）机器人只能朝东南西北四个方向前进

ii）不考虑机器人故障

iii）不考虑迷宫中是否有其他机器人

知识点二：搜索算法基础

首先通过例子看一些概念：



(a) (b) (c)

如上图所示，图(a)中只有**根节点(root node)**被探索过，图(b)中一个**叶节点(leaf node)**被探索，图(c)中根节点的**后继(successor)**以顺时针序被探索。途中黑色的节点为已探索的节点，灰色的结点是未探索的结点，白色结点将已探索区域和未被探索区域分隔开，称为**边缘(frontier/open list)**。

搜索算法需要一个数据结构来记录搜索树的构造过程。对树中的每个结点，我们定义数据结构包含四个元素：

STRUCT{

STATE:对应状态空间中的状态

PARENT:该节点的父节点（即产生该结点的结点）

ACTION:父节点生成该节点时所采取的行动

PATH-COST:代价，指从初始状态到达该节点的路径消耗

}

有了结点，就需要空间来存放。搜索算法希望可以根据喜欢的策略很容易地选择出下一个要扩展的结点，由此需要引入边缘存储。边缘存储最合适的数据结构是队列。队列的一些操作如下：

EMPTY?(queue)队列空时返回值为真

POP(queue)返回队列中的第一个元素并将它从队列中移除（返回的结点将作为下一个探索目标被探索）

INSERT(queue)在队列中插入一个元素并返回队列

知识点三：无信息搜索(uninformed search)

3.1 宽度优先(Breadth-first)

宽度优先搜索的规则是：先扩展根节点，接着扩展根节点的所有后继。一般搜索树同一层（深度相同）的结点都被扩展完之后才会扩展下一层。该方法首先是完备的，但不一定是最优的，因为该算法找到最浅的目标结点就停止，然而最浅的目标结点不一定是最优的。

由于每一层的每个结点都要扩展到，因此该算法的空间复杂度随深度呈指数级增长，一般的计算机内存根本无法支持 10 层以上的搜索。

3.2 一致代价搜索(Uniform-cost search)

一致代价搜索扩展的是路径消耗 g(n)最小的结点 n。可以通过将边缘结点集组织成按 g 值排序的队列来实现。一致代价搜索一定是最优的，但可能不完备：如果存在零代价行动就会陷入死循环。如果每一步的代价都大于等于某个小的正值ϵ，则可以使该算法完备。

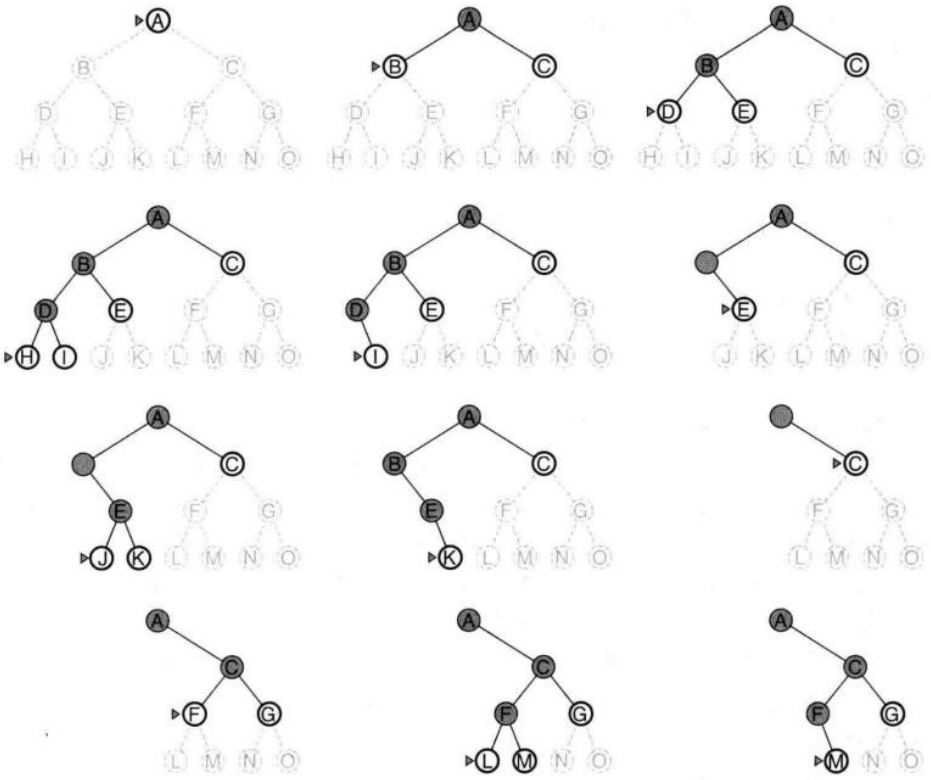
但是又引入了新的问题：如果ϵ的值非常小，那么即使可以使算法摆脱死循环（因为每次循环路径代价 g 都会加ϵ，总有一个时刻 g 会大于边缘上的其他结点的 g 值，跳出循环）但是这一过程有可能非常缓慢，因为每次加ϵ对 g 的改变都不多。

3.3 深度优先搜索(Depth-first search)

深度优先搜索总是扩展搜索树的当前边缘结点集中最深的结点。将最深层的结点扩展完以后，就从边缘结点中去掉，然后算法回溯到下一个还有未扩展后继的深度稍浅的结点，这样该算法就有了优于宽度优先算法的空间复杂度，深度优先算法只需要存储一条从根结点到叶结点的路径，以及该路径上每个结点的所有未被扩展的兄弟结点即可。D-16 时可节省 7000 亿倍的存储空间。

该算法不是最优的。且该算法不是完备的，例如在罗马尼亚问题中，会陷入 A-S-A-S 的死循环。又如果状态空间无限大，深度优先搜索就会失败。可以通过深度受限搜索(Depth-limited)以及迭代加深的深度优先搜索(Iterative deepening)来解决这个问题。（具体例子看习题 3.15）

此外，迭代加深的深度优先搜索在空间复杂度上具有优势（O(bd)）



深度优先算法示意图

3.4 双向搜索(Bidirectional search)

双向搜索的思想是同时运行两个搜索：一个从初始状态向前搜索同时另一个从目标状态向后搜索，如果两个进程在中间相遇，搜索终止。在之前的搜索算法中，目标检测都是检测是否是目标结点，双向搜索将目标检测改为了两个方向的搜索边缘结点是否相交。这种算法大大降低了时间空间复杂度。

以上所有算法的性能对比可以总结为一张表：

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes[a]	Yes[a,b]	No	No	Yes[a]	Yes[a,d]
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$
Optimal?	Yes[c]	Yes	No	No	Yes[c]	Yes[c,d]

3.5 相关题目解答

3.15 Consider a state space where the start state is number 1 and each state k has two successors: numbers $2k$ and $2k+1$.

a. Draw the portion of the state space for states 1 to 15.

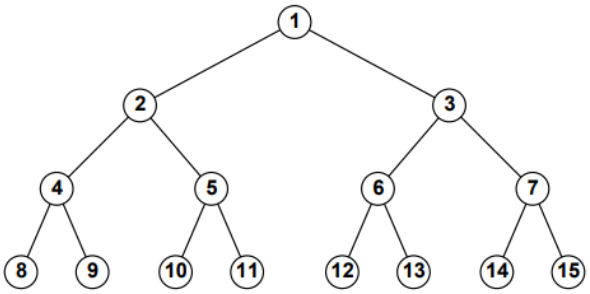
b. Suppose the goal state is 11. List the order in which nodes will be visited for breadth-first search, depth-limited search with limit 3, and iterative deepening search.

c. How well would bidirectional search work on this problem? What is the branching factor in each direction of the bidirectional search?

d. Does the answer to (c) suggest a reformulation of the problem that would allow you to solve the problem of getting from state 1 to a given goal state with almost no search?

e. Call the action going from k to $2k$ Left, and the action going to $2k+1$ Right. Can you find an algorithm that outputs the solution to this problem without any search at all?

答：a.



b. 宽度优先搜索：1-2-3-4-5-6-7-8-9-10-11

深度界限为 3 的深度受限搜索：1-2-4-8-9-5-10-11

迭代加深搜索：1-1-2-3-1-2-4-5-3-6-7-1-2-4-8-9-5-10-11

c. 该状态空间非常适合用双向搜索求解，因为沿逆向搜索时，结点 n 的后继只有 1 个，为[n/2]，需要扩展的结点大大减少。正向求解的分支因子（branching factor）为 2，逆向的分支因子为 1.

d. 在该状态空间中，如使用双向搜索，由于逆向搜索的结点 n 只有一个后继，因此从目标结点 11 逆向搜索只有一条确定的路径，因此几乎不需要任何的搜索。

e. 一个巧妙的方法：可以将目标结点的数字用二进制表示，0 代表向左，1 代表向右，二进制数字的最低位对应最深层的 action，依次往前推，每一层都对应一位数字。Eg。目标数字是 11，写成二进制是 1011，则解为左-右-右。

3.17 On page 90, we mentioned **iterative lengthening search**, an iterative analog of uniform cost search. The idea is to use increasing limits on path cost. If a node is generated whose path cost exceeds the current limit, it is immediately discarded. For each new iteration, the limit is set to the lowest path cost of any node discarded in the previous iteration.

a. Show that this algorithm is optimal for general path costs.

b. Consider a uniform tree with branching factor b, solution depth d, and unit step costs. How many iterations will iterative lengthening require?

c. Now consider step costs drawn from the continuous range $[\epsilon, 1]$, where $0 < \epsilon < 1$. How many iterations are required in the worst case?

d. Implement the algorithm and apply it to instances of the 8-puzzle and traveling salesperson problems. Compare the algorithm's performance to that of uniform-cost search, and comment on your results.

答：a. 依照该算法的规则，每次迭代扩展的结点是按照路径代价排列的，因此当搜索到第一个目标结点时，一定是所有解中路径代价最小的，因此是最优的。

b. 与迭代加深算法相似，需要 O(b^d)次迭代才能找到解

c. 当每一步的路径代价比较小时，每次迭代的路径代价的限制值就会变化的比较慢，每次只能扩展较少的新结点，因此找到解所需的迭代次数就会增多。最坏的情况，即次迭代时路径代价的限制值都只增加ϵ，但解路径的每一步的路径代价都取最大值 1，则找到解所需的迭代次数为$\frac{1}{\epsilon} \times d = \frac{d}{\epsilon}$。

3.18 Describe a state space in which iterative deepening search performs much worse than depth-first search (for example, $O(n^2)$ vs. $O(n)$).

答：这样的状态空间很好找到，如果每个结点只有一个后继，那么使用深度优先搜索只需要 n 步，但使用迭代加深搜索就需要 1+2+3+…+n=O(n^2)步。da

知识点四：有信息（启发式）搜索策略(informed search)

有信息搜多的一般算法成为最佳优先搜索。最佳优先搜索的结点时基于评价函数 f(n)值被选择扩展的。评估函数被看作时代价估计，因此评估值最低的结点被选择首先进行扩展。

大多数的最佳优先搜索算法的 f 由启发函数构成：

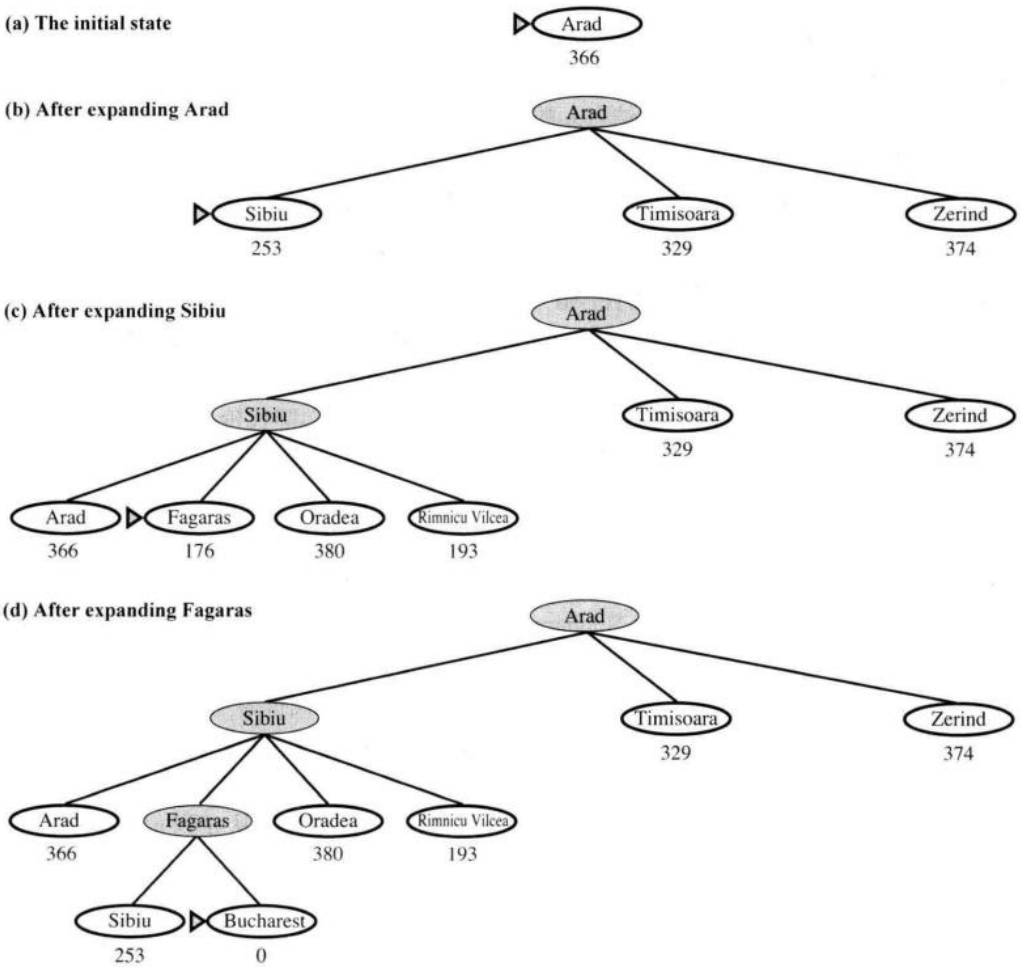
h(n)=结点 n 到目标节点的最小代价路径的代价估计值。例如，在贯穿本章的罗马尼亚问题中，可以用当前城市距离目标城市的直线距离来作为 h(n)。

4.1 贪婪最佳优先搜索(Greedy best-first search)

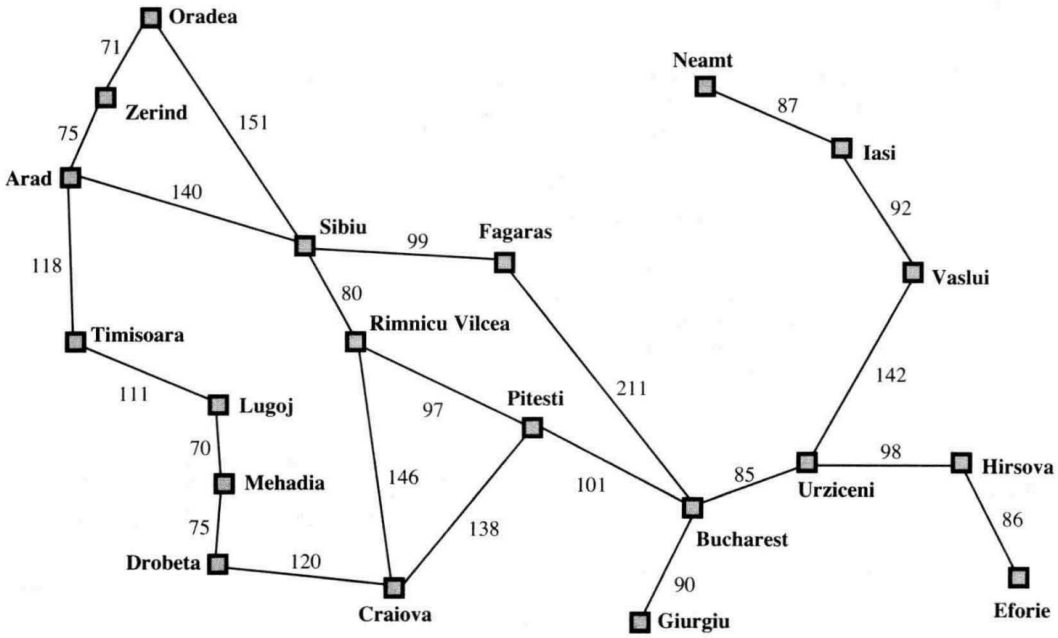
贪婪最佳优先搜索试图扩展离目标最近的结点，理由是这样可能可以很快找到解。因此，它只使用启发式信息，即 **f(n)=h(n)**。将此算法应用在罗马尼亚问题中：使用**直线距离**启发式，记作h_{LSD}，h_{LSD}是和实际路程相关的。下表列出了在该问题中h_{LSD}的值：

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

下图为使用直线距离启发的贪婪最佳优先搜索寻找从 Arad(出发城市)到 Bucharest(目标城市)的路的过程。



可以看到，贪婪最佳优先搜索的每一步都试图找到离目标最近的结点，但它也有一些问题：仍旧以罗马尼亚问题为例，考虑从 Iasi 到 Fagaras，贪婪搜索算法会最先扩展 Neamt，但 Neamt 无法去到任何地方，又会返回 Iasi，形成死循环。可以看到该算法不是完备的。

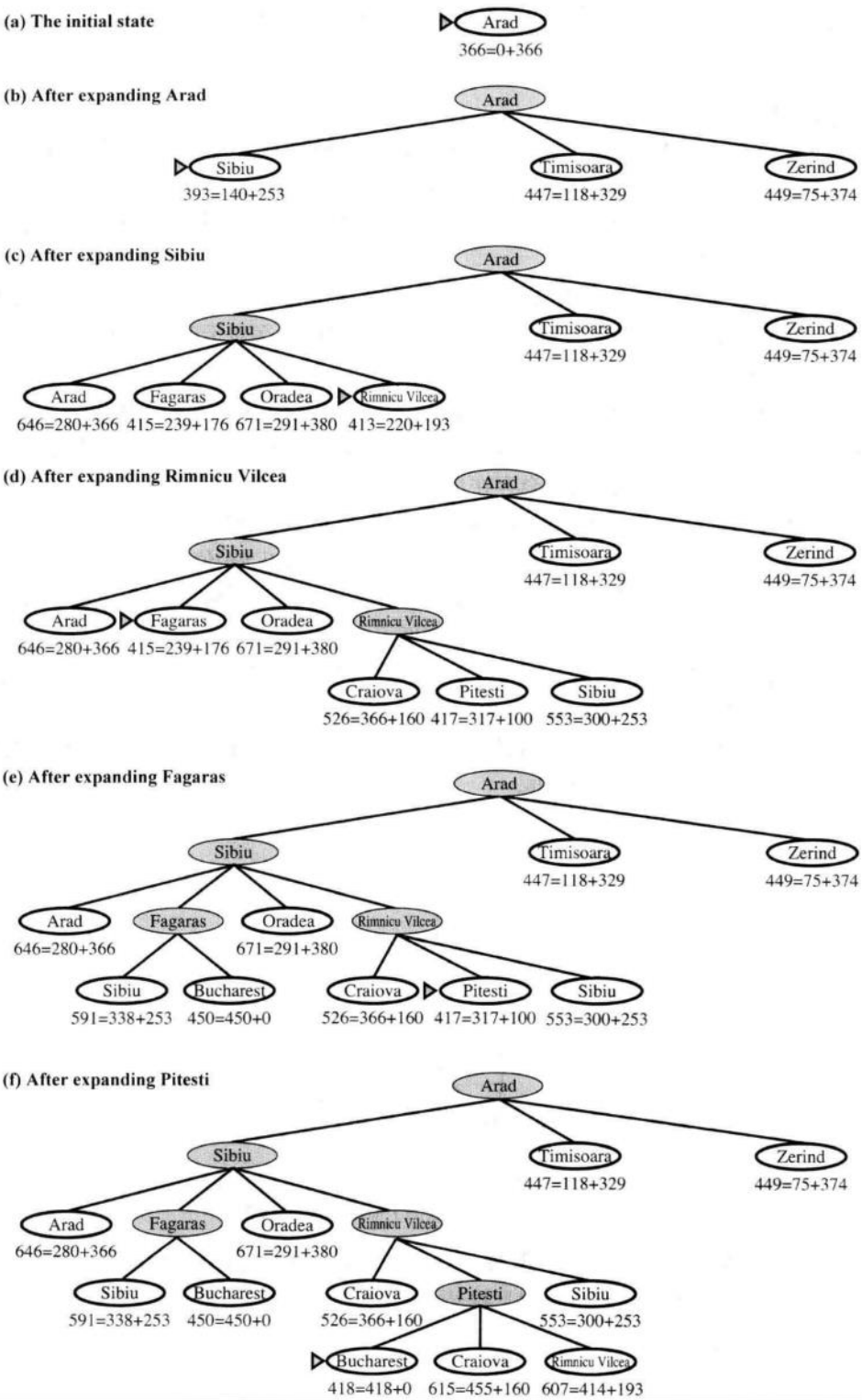


4.2 A*搜索

上述问题可以通过完备的算法 A*来解决。在该算法中，对结点的评估函数结合了到达此结点已经花费的代价 g(n)，和从该结点到目标结点需要花费的代价：

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

由于 g(n)是从开始结点到结点 n 的路径代价，而 h(n)是从结点 n 到目标结点的最小代价估计值，因此 f(n)就是经过结点 n 的最小代价解的估计值。通过 A*算法求解罗马尼亚问题的过程如下图：

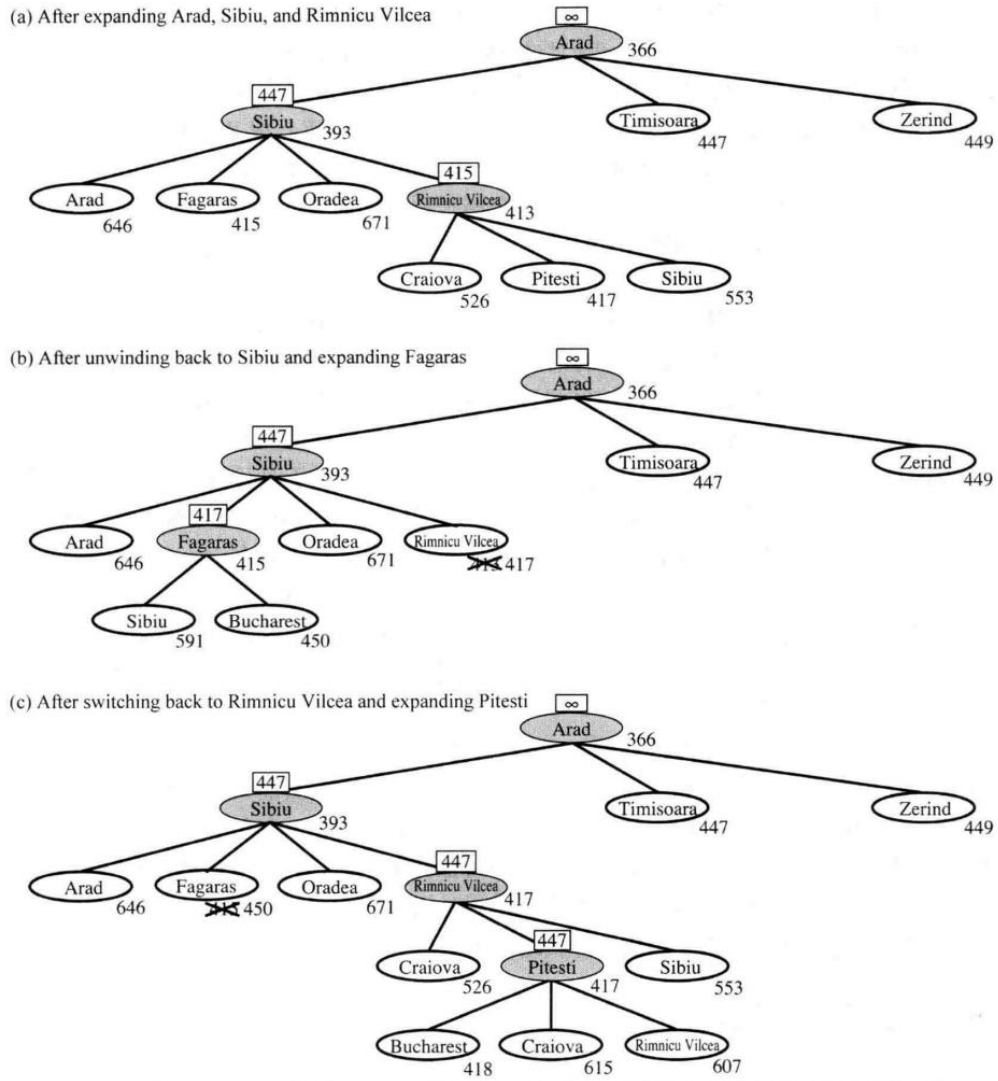


通过证明可以得到 A*算法的最优性结论。

4.3 存储受限的启发式搜索(RBFS)

递归最佳优先搜索(RBFS)试图模仿标准的贪婪搜索算法，只使用**线性的存储空间**（选择探索一条路径就会忘记别的子树）。它用变量 f_limit 跟踪记录从当前结点的祖先可得到的最佳可选路径的 f 值，如果当前结点超过了这个限制，递归将回到可选路径上。如果递归回溯，

则对当前路径上的每个结点，RBFS 用其子结点的最佳 f 替换其 f 值。这样，RBFS 就能记住被它以往的子树中最佳叶节结点的 f 值，以决定以后是否要重新扩展该子树。下面仍用罗马尼亚问题举例：



在图(a)中，Sibiu 的祖先节点是 Arad，除过 Sibiu 本身外 Arad 可探索的最佳路径为 Timisoara，因此将 Timisoara 的 f 值作为 Sibiu 的 f_limit；同理，Rimnicu Vilcea 的祖先结点有 Sibiu 和 Arad，这些结点可得到的最佳路径为 A-S-F（城市名简写），则 R 的 f_limit 就等于 Fagaras 的 f 值 415。扩展完 R 后，发现最佳叶结点的 f 值为 417，大于 f_limit，因此在下一步（图(b)）中发生了递归回溯，转而探索 Fagaras 而遗忘 R，并把被遗忘的最佳叶结点的 f 值（417）回填到 R…

4.4 启发式函数(heuristic function)

显然启发式搜索算法的质量由启发式函数 h(n)的设计决定。启发式函数的最优性有下面两个条件：

可采纳（admissable）：h(n)从不会过高地估计到达目标的代价，因此 f(n)从不会超过经过结点 n 的解的实际代价。例如在罗马尼亚问题中，启发式函数 h(n)采用的是直线距离h_{LSD}。直线距离是可采纳的启发式，因为两点之间直线最短，因此肯定不会高估。

一致性/单调性(consistency)：对于每个结点 n 和通过任一行动 a 生成的 n 的每个后继结点 n'，从结点 n 到达目标的估计代价不大于从 n 到 n'的单步代价与从 n'到达目标的估计代价之和：

$$h(n) \leq c(n,a,n') + h(n')$$

这是一个三角不等式，保证了三角形两边之和大于第三边。

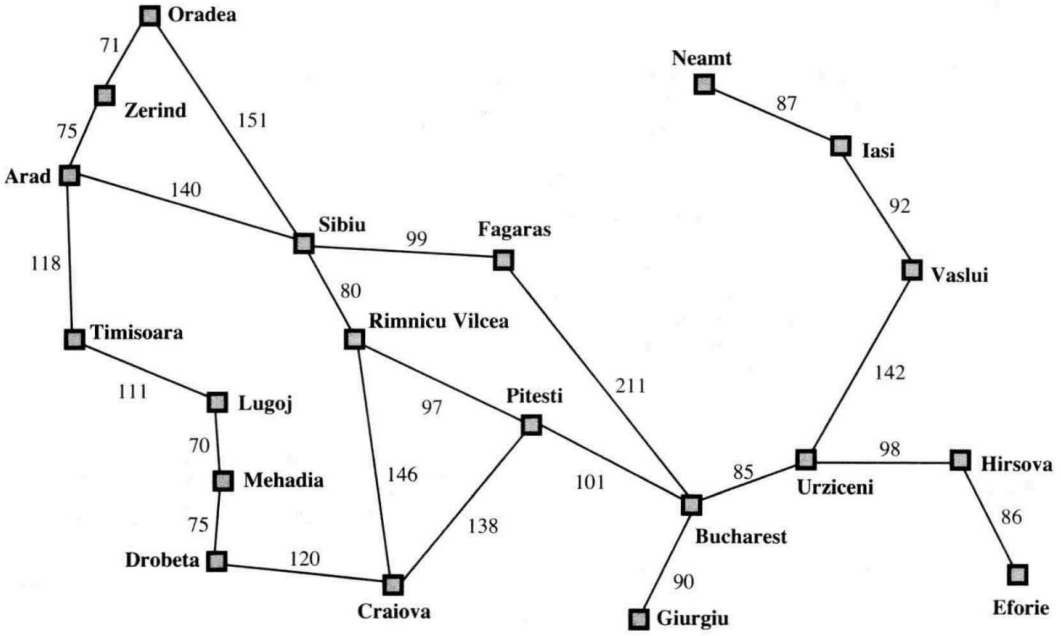
一致的启发式都是可采纳的。一致性的要求比可采纳性更严格。

除了直线距离h_{LSD}，常用的启发函数还有曼哈顿距离、对角线距离、欧几里得距离等。

4.4 相关题目解答

3.23 Trace the operation of A* search applied to the problem of getting to Bucharest from Lugoj using the straight-line distance heuristic. That is, show the sequence of nodes that the algorithm will consider and the f, g, and h score for each node.

答：将相关信息贴在下面方便查阅：

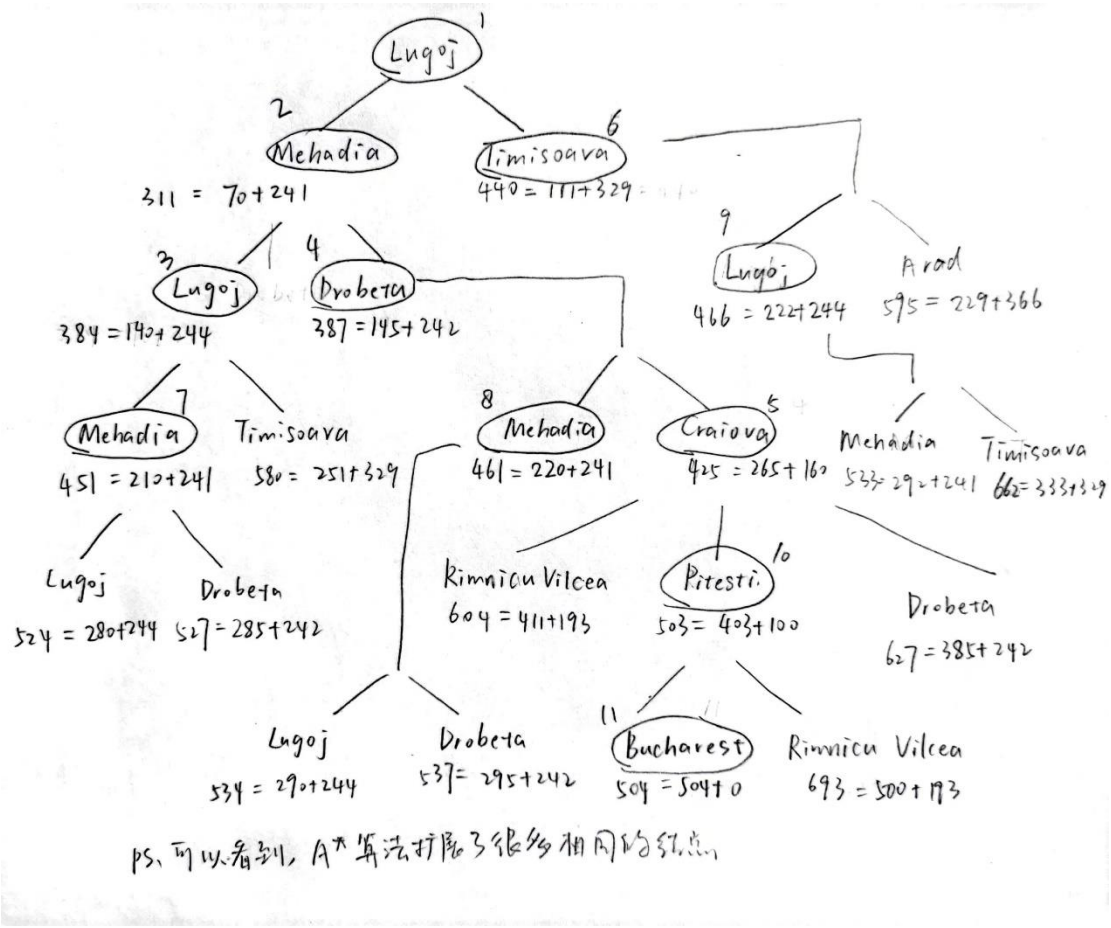


地图

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

h_{LSD}的值

按照 A*算法探索结点的步骤如下图：



3.25 The **heuristic path algorithm** (Pohl, 1977) is a best-first search in which the evaluation function is $f(n) = (2 - w)g(n) + wh(n)$. For what values of w is this complete? For what values is it optimal, assuming that h is admissible? What kind of search does this perform for $w = 0$, $w = 1$, and $w = 2$?

答：要让算法最优，g(n)必须对 f(n)产生影响，因此$0 \le w < 2$时可以保证算法最优。

当 w=0 时，f(n)=2g(n)，忽略因子 2，该算法等同于一致代价搜索。

当 w=1 时，f(n)=g(n)+h(n)，为 A*搜索

当 w=2 时，f(n)=2h(n)，此时 g(n)不再影响 f(n)，等同于贪婪搜索。

3.26 Consider the unbounded version of the regular 2D grid shown in Figure 3.9. The start state is at the origin, (0,0), and the goal state is at (x, y).

a. What is the branching factor b in this state space?

b. How many distinct states are there at depth k (for $k > 0$)?

c. What is the maximum number of nodes expanded by breadth-first tree search?

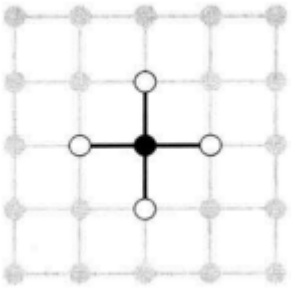
d. What is the maximum number of nodes expanded by breadth-first graph search?

e. Is $h = |u - x| + |v - y|$ an admissible heuristic for a state at (u, v)? Explain.

f. How many nodes are expanded by A* graph search using h?

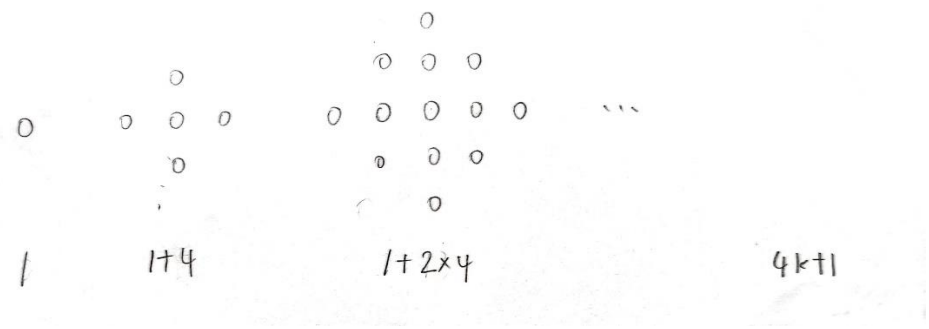
g. Does h remain admissible if some links are removed?

h. Does h remain admissible if some links are added between nonadjacent states?



答：a. 状态空间的分支因子为 4（每个节点都有上下左右四个邻居结点）

b. 状态扩展的图样如下，是一个旋转了 45° 的正方形：



可以看到，每次扩展后正方形的外围就会增加一圈，第一次增加 4 个状态，第二次增加 8 个状态，知道深度为 k 时，将会有 4k+1 个状态。

（另外，这个题的答案不是4^k，要分清“状态”和“结点”的概念上的区别，这也是作者在问题形式化那一节中强调过的！）

c. 达到(0,1)需要两次扩展，达到(1,1)需要三次扩展，(2,2)需要五次扩展，可以总结规律：达到(x,y)需要 x+y+1 次扩展。因此在树搜索中，扩展的结点数最大为

$$\frac{4^{x+y+1}-1}{4-1}-1=\frac{4^{x+y+1}-1}{3}-1$$

d. 如采用宽度优先的图搜索，需要搜索的深度仍为 x+y+1，则可能扩展的结点数最大为 2(x+y)(x+y+1)-1。

e. 是可采纳的。该启发式函数明显为曼哈顿距离。

f. 如果采用 A*算法，每次每个结点的四个子结点中只有两个子结点会得到扩展，其余的结点都会因为偏离方向使曼哈顿距离稍大，因此最后总共会扩展$4^{x+y+1}-2$个结点

g. 一些连接断开不会影响 h 的可采纳性，因为连接断开实际的代价会变大，h 仍不会对代价进行高估

h. 如果加入了一些新的连接，那么 h 可能就不是可采纳的了。例如如果在初始状态和目标状态间添加一条直线路径，两点之间直线最短，那么 h 总会高估路径代价，不再是可采纳的了

3.27 n vehicles occupy squares $(1,1)$ through $(n,1)$ (i.e., the bottom row) of an $n \times n$ grid. The vehicles must be moved to the top row but in reverse order; so the vehicle i that starts in $(i,1)$ must end up in $(n-i+1,n)$. On each time step, every one of the n vehicles can move one square up, down, left, or right, or stay put; but if a vehicle stays put, one other adjacent vehicle (but not more than one) can hop over it. Two vehicles cannot occupy the same square.

a. Calculate the size of the state space as a function of n.

b. Calculate the branching factor as a function of n.

c. Suppose that vehicle i is at (x_i, y_i); write a nontrivial admissible heuristic h_i for the number of moves it will require to get to its goal location $(n-i+1,n)$, assuming no other vehicles are on the grid.

d. Which of the following heuristics are admissible for the problem of moving all n vehicles to their destinations? Explain.

(i) $\sum_{i=1}^{n} h_i$.
(ii) $\max\{h_1, \ldots, h_n\}$.
(iii) $\min\{h_1, \ldots, h_n\}$.

答：a. 共有 n 量车、n^2个格子，粗略计算共有$(n^2)^n = n^{2n}$个状态空间

b. 每辆车每一步都有五种选择（向上，向下，向左，向右，不动），共有 n 辆车，将会扩展出5^n个状态，因此扩展因子为5^n。

c. 该问题为网格搜索，网格搜索最常用的启发式函数为曼哈顿距离。即

$$h(n) = |(h-i+1) - x_i| + |n - y_i|$$

d. 只有(iii)中的启发式函数是可采纳的。

令 W 等于达到目标后每一辆车的路径代价总和，则有：$W \geq \sum_{i=1}^{n} h_i \geq n \cdot min\{h_1, \ldots, h_n\}$

且每一步可走的步数不会超过 n（一辆车动时另一辆不会动），因此完成目标所需要的步数一定不会超过$n \cdot min\{h_1, \ldots, h_n\}/n = min\{h_1, \ldots, h_n\}$。

因此$min\{h_1, \ldots, h_n\}$一定不会高估代价，是可采纳的。