**中 国 地 质 大 学**

**课程大作业封面**

课程名称 人工智能基础

学 生 姓 名 刘瑾瑾

学 生 学 号 20201000128

日 期 2023年1月1日

**大作业1：八数码问题的A\*搜索算法实现**

1. **内容：**

设计估价函数（编程语言不限），以八数码为例演示A\*算法的搜索过程，争取做到直观、清晰地演示算法，代码要适当加注释。

八数码问题：在3×3方格棋盘上，分别放置了标有数字1, 2, 3, 4, 5, 6, 7, 8的八张牌，初始状态S0根据题目要求设定，使用的操作有: 空格上移，空格左移，空格右移，空格下移。试采用A\*算法写程序实现这一搜索过程。

**二. 要求：**

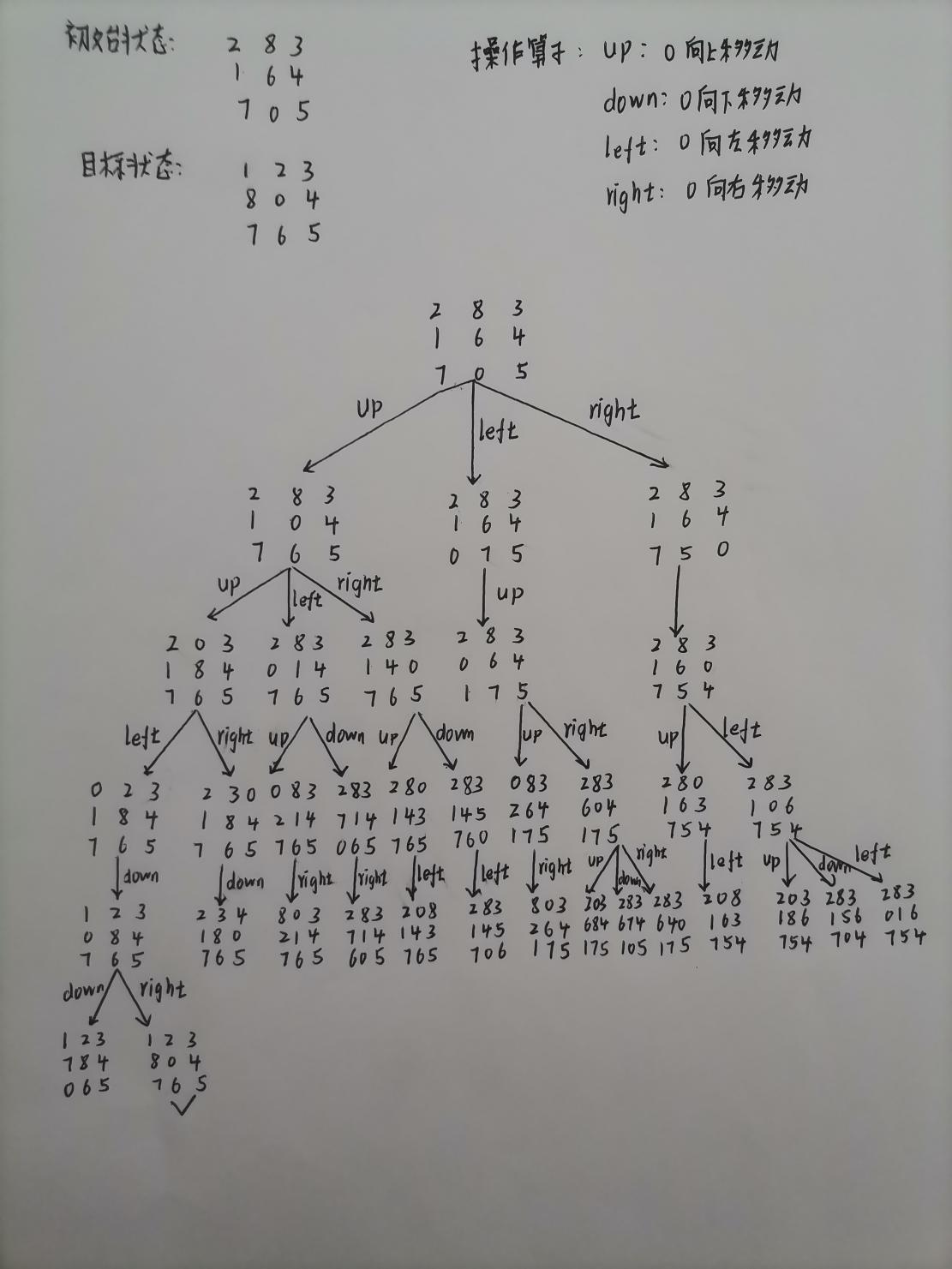
1. 设置相同的初始状态和目标状态，针对不同的估价函数,求得问题的解，比较它们对搜索算法性能的影响,包括扩展节点数、生成节点数等,填入表1。
2. 设置与上述1相同的初始状态和目标状态,用宽度优先搜索算法（即令估计代价*h*(*n*)=0的A\*算法）求得问题的解，以及搜索过程中的扩展节点数、生成节点数，填入表1。

表1不同启发函数*h*(*n*)求解8数码问题的结果比较

|  |  |  |  |
| --- | --- | --- | --- |
|  | 启发函数*h***(***n***)** | | |
| 不在位数 | 哈密顿距离 | 0 |
| 初始状态 | 1 2 3 8 0 5 4 6 7 | 1 2 3 8 0 5 4 6 7 | 1 2 3 8 0 5 4 6 7 |
| 目标状态 | 1 2 3 8 0 7 6 4 5 | 1 2 3 8 0 7 6 4 5 | 1 2 3 8 0 7 6 4 5 |
| 最优解 | 最佳路径长度为：16  初始状态：  1 2 3  8 0 5  4 6 7  第1层的状态：  1 2 3  8 6 5  4 0 7  第2层的状态：  1 2 3  8 6 5  4 7 0  第3层的状态：  1 2 3  8 6 0  4 7 5  第4层的状态：  1 2 0  8 6 3  4 7 5  第5层的状态：  1 0 2  8 6 3  4 7 5  第6层的状态：  0 1 2  8 6 3  4 7 5  第7层的状态：  8 1 2  0 6 3  4 7 5  第8层的状态：  8 1 2  6 0 3  4 7 5  第9层的状态：  8 1 2  6 7 3  4 0 5  第10层的状态：  8 1 2  6 7 3  0 4 5  第11层的状态：  8 1 2  0 7 3  6 4 5  第12层的状态：  0 1 2  8 7 3  6 4 5  第13层的状态：  1 0 2  8 7 3  6 4 5  第14层的状态：  1 2 0  8 7 3  6 4 5  第15层的状态：  1 2 3  8 7 0  6 4 5  第16层的状态：  1 2 3  8 0 7  6 4 5 | 最佳路径长度为：16  初始状态：  1 2 3  8 0 5  4 6 7  第1层的状态：  1 2 3  8 5 0  4 6 7  第2层的状态：  1 2 0  8 5 3  4 6 7  第3层的状态：  1 0 2  8 5 3  4 6 7  第4层的状态：  0 1 2  8 5 3  4 6 7  第5层的状态：  8 1 2  0 5 3  4 6 7  第6层的状态：  8 1 2  4 5 3  0 6 7  第7层的状态：  8 1 2  4 5 3  6 0 7  第8层的状态：  8 1 2  4 0 3  6 5 7  第9层的状态：  8 1 2  0 4 3  6 5 7  第10层的状态：  0 1 2  8 4 3  6 5 7  第11层的状态：  1 0 2  8 4 3  6 5 7  第12层的状态：  1 2 0  8 4 3  6 5 7  第13层的状态：  1 2 3  8 4 0  6 5 7  第14层的状态：  1 2 3  8 4 7  6 5 0  第15层的状态：  1 2 3  8 4 7  6 0 5  第16层的状态：  1 2 3  8 0 7  6 4 5 | 最佳路径长度为：16  初始状态：  1 2 3  8 0 5  4 6 7  第1层的状态：  1 2 3  8 6 5  4 0 7  第2层的状态：  1 2 3  8 6 5  4 7 0  第3层的状态：  1 2 3  8 6 0  4 7 5  第4层的状态：  1 2 0  8 6 3  4 7 5  第5层的状态：  1 0 2  8 6 3  4 7 5  第6层的状态：  0 1 2  8 6 3  4 7 5  第7层的状态：  8 1 2  0 6 3  4 7 5  第8层的状态：  8 1 2  6 0 3  4 7 5  第9层的状态：  8 1 2  6 7 3  4 0 5  第10层的状态：  8 1 2  6 7 3  0 4 5  第11层的状态：  8 1 2  0 7 3  6 4 5  第12层的状态：  0 1 2  8 7 3  6 4 5  第13层的状态：  1 0 2  8 7 3  6 4 5  第14层的状态：  1 2 0  8 7 3  6 4 5  第15层的状态：  1 2 3  8 7 0  6 4 5  第16层的状态：  1 2 3  8 0 7  6 4 5 |
| 扩展节点数  （不包括叶子节点） | 847 | 303 | 9219 |
| 生成节点数  （包含叶子节点） | 1377 | 492 | 14726 |
| 运行时间  （迭代次数） | 847 | 303 | 9219 |

**三. 问题分析**

1. 画出[2, 8, 3],[1, 6, 4],[7, 0, 5]推导至[1, 2, 3],[8, 0, 4],[7, 6, 5]的图解。

2. 分析不同的估价函数对A\*算法性能的影响。

A\*算法是目前最有影响的启发式图搜索算法，也成为最佳图搜索算法。A\*算法的启发函数由估价函数和实际代价函数组成，即f(n)=g(n)+h(n)，g(n)是从初始结点到n结点的实际代价，而h(n)是从n结点到目的结点的最佳路径的估计代价,定义h\*(n)为状态n到目的状态的最优路径的代价，则h(n)≤h\*(n)。如果某一问题有解，那么利用A\*搜索算法对该问题进行搜索则一定能搜索到解，并且一定能搜索到最优的解。

在八数码问题中，估价函数有三种：（1）不在位数：现有状态与目标状态不相同的位数；（2）哈密顿距离：现有状态各数码移动到目标状态所需要移动距离的综合；（3）0：即宽度优先搜索算法，盲目图搜索。启发函数的值越小，优先选择程序越高。通过不在位法和哈密顿距离法与宽度优先搜索比较，采用启发函数优于盲目搜索。当初始状态为1 2 3 8 0 5 4 6 7，目标状态为1 2 3 8 0 7 6 4 5时，搜索深度为16，此时不在位法扩展节点数位847，生成节点数位1377，迭代次数位847，而哈密顿距离法扩展结点为303，生成节点数为492，迭代次数为303，所以当搜索深度比较高时，相较于不在位法，哈密顿距离法所用时间更短，生成节点数更少，效率更高。

1. 根据宽度优先搜索算法和A\*算法求解8数码问题的结果,分析启发式搜索的特点。

宽度优先搜索最大的优点之处在于能够保证找到实际问题的解，但是在搜索的进行过程中必须需要存储大量的状态信息而会导致空间复杂度相当的大，从而容易产生“组合爆炸”的问题。

A\*算法是一种启发式的搜索，A\* 算法实现的过程中也需要保存一些搜索的状态,但是由于A\* 算法在搜索的过程中每次进行扩展时可以有启发地选择了有最优的节点进行扩展，因此从局部最优推至全局最优，大大的缩小了搜索的时间复杂度和空间复杂度，能够快速找到最优解。不同于A算法，A算法在启发信息给得越多即估价函数值越大,需要搜索处理的状态数就越少,其效率就越高，但也不是估价函数值越大越好,因为估价函数值太大会使A算法不一定能搜索到最优解，而使用A\* 算法在求解实际问题过程中一定能够找到最优解。

所以，启发式搜索通过启发信息选取估价函数，大大提高了搜索的效率，降低了时间复杂的和空间复杂度，启发信息给得越多即估价函数值越大,需要搜索处理的状态数就越少,其效率就越高，但估价函数过大容易导致无最优解（A算法），且获得的启发信息越多，需要耗费计算时间越长，所以可能抵消降低搜索空间复杂度和时间复杂度所带来的益处。

1. **心得体会**

通过对宽度优先搜索算法和启发式搜索算法的实际操作和应用，我对搜索算法的流程更加清晰，对于open表和close表的作用也有了更深刻的理解。启发式搜索相交于宽度优先搜索可以降低搜索过程中的时间和空间复杂度，对于解决计算量巨大、计算周期长的问题具有很大的优势，如围棋、象棋等。另外，通过对比不在位法和哈密顿距离法，我发现，选择合适的启发函数可以进一步提高搜索的效率，启发式的具体选择要根据应用场景具体选择。在本次操作过程中，我深刻的意识到仅仅只学习课本上的知识是远远不够的，我们不仅需要明白底层原理，更需要知道如何通过代码逻辑将其表示出来，即如何实际应用。总而言之，只有将理论知识应用到实践中，才能够真正掌握。

**大作业2：产生式系统设计**

**一．目的**

1. 熟悉知识的表示方法

2. 掌握产生式系统的运行机制

3. 产生式系统推理的基本方法

**二．内容**

设计并编程实现一个小型产生式系统（如分类、诊断等类型）。

**三．要求**

1. 具体应用领域自选，具体系统名称自定。

2. 用一阶谓词逻辑和产生式规则作为知识表示，利用产生式系统实验程序，建立知识库，分别运行正、反向推理。

**四．系统设计**

1. 系统设置，包括系统名称和系统谓词，给出谓词名及定义。

（1）系统名称：动物识别系统

（2）系统功能：对虎、金钱豹、斑马、长颈鹿、企鹅、鸵鸟和信天翁七种动物进行识别

2. 编辑知识库，通过输入规则或修改规则等，建立规则库。

R1：动物有毛发→ 哺乳类

R2：动物产奶 → 哺乳类

R3：动物有羽毛 → 鸟类

R4：动物会飞 ∧会下蛋 → 鸟类

R5：哺乳类∧动物吃肉→ 食肉动物

R6：动物有犬齿 ∧有爪 ∧眼睛盯前方→食肉动物

R7：哺乳类 ∧有蹄 →蹄类

R8：哺乳类 ∧反刍 → 蹄类

R9：哺乳类 ∧ 肉食类∧ 黄褐色 ∧ 有暗斑点→ 金钱豹

R10：哺乳类∧ 肉食类 ∧ 黄褐色 ∧ 有黑色条纹→虎

R11：蹄类 ∧ 长脖 ∧ 长腿 ∧ 有暗斑点→ 长颈鹿

R12：蹄类 ∧ 有黑色条纹→ 斑马

R13：鸟类 ∧长脖 ∧ 长腿 ∧ 不会飞∧黑白二色 →鸵鸟

R14：鸟类 ∧会游泳 ∧黑白二色 ∧ 不会飞 →企鹅

R15：鸟类 ∧善飞 →信天翁

3. 建立事实库（综合数据库），输入多条事实或结论。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | 描述 | 编号 | 描述 | 编号 | 描述 |
| 0 | 有毛 | 12 | 有暗斑点 | 24 | 企鹅 |
| 1 | 产奶 | 13 | 有黑色条纹 | 25 | 信天翁 |
| 2 | 有羽毛 | 14 | 长脖 | 26 | 鸵鸟 |
| 3 | 会飞 | 15 | 长腿 | 27 | 斑马 |
| 4 | 会下蛋 | 16 | 不会飞 | 28 | 长颈鹿 |
| 5 | 吃肉 | 17 | 会游泳 | 29 | 虎 |
| 6 | 有犬齿 | 18 | 黑白两色 | 30 | 金钱豹 |
| 7 | 有爪 | 19 | 善飞 |  |  |
| 8 | 眼睛盯前方 | 20 | 哺乳类 |  |  |
| 9 | 有蹄 | 21 | 鸟类 |  |  |
| 10 | 反刍 | 22 | 肉食类 |  |  |
| 11 | 黄褐色 | 23 | 蹄类 |  |  |

4. 运行推理，包括正向推理和反向推理，给出相应的推理过程、事实区和规则区。

（1）正向推理：

* 推理过程：将所有规则进行编号，组成规则库，遍历这些条件，根据用户给出的特征编号，进行匹配，同时计算每个条件的符合程度，推理出的特征加入到综合数据库中，重新遍历条件，更新符合度，如果符合度为1，则输出推理结果。
* 示例：

已知事实：产奶，有蹄，有暗斑点，长脖，长腿

综合数据库：1.产奶，9.有蹄，12.有暗斑点，14.长脖，15.长腿

从规则库取出规则，与综合数据库中的已知事实匹配

根据R2：动物产奶→ 哺乳类，更新综合数据库：

1.产奶，9.有蹄，12.有暗斑点，14.长脖，15.长腿，20.哺乳类

根据R7：哺乳类 ∧有蹄 →蹄类，更新综合数据库：

1.产奶，9.有蹄，12.有暗斑点，14.长脖，15.长腿，20.哺乳类，23.蹄类

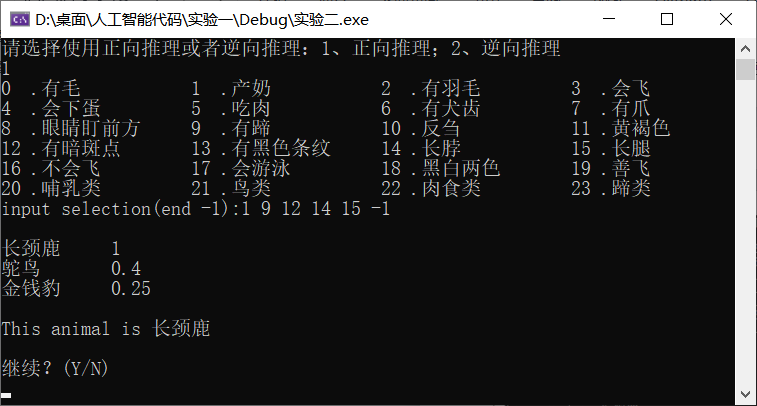
根据R11：蹄类 ∧ 长脖 ∧ 长腿 ∧ 有暗斑点→ 长颈鹿，更新综合数据库：

1.产奶，9.有蹄，12.有暗斑点，14.长脖，15.长腿，20.哺乳类，23.蹄类，

28.长颈鹿

检查综合数据库中的内容,发现要识别的动物长颈鹿包含在了综合数据库中,所以推出了“该动物是长颈鹿”这一最终结论。至此,问题的求解过程结束。

* 程序运行结果：

1. 逆向推理：

* 推理过程：输入一种动物，询问用户该动物满足的条件，得到条件后，正向推理进行验证。
* 示例：假设动物为虎

根据规则R10：哺乳类∧ 肉食类 ∧ 黄褐色 ∧ 有黑色条纹→虎

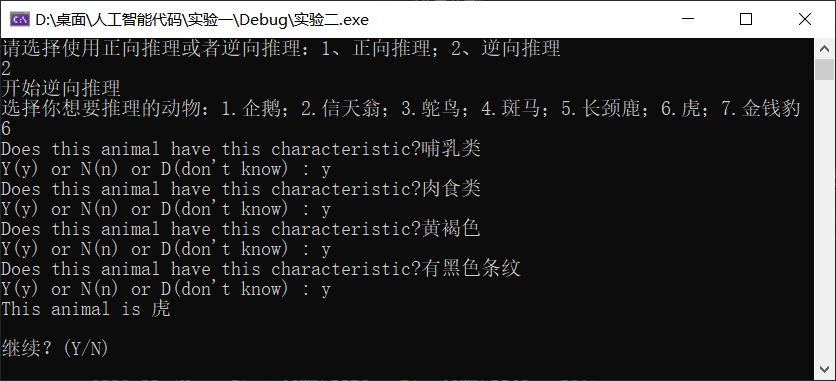
运行反向推理，逐条询问：1.该动物是否具有这个特征：哺乳类？

2.该动物是否具有这个特征：肉食类？

3.该动物是否具有这个特征：黄褐色？

4.该动物是否具有这个特征：有黑色条纹？

用户回答确认，若所有条件都成立，则动物即为虎。

* 程序运行结果：

1. 混合推理：

* 推理过程：将所有规则进行编号，组成规则库，遍历这些条件，根据用户给出的特征编号，进行匹配，同时计算每个条件的符合程度，推理出的特征加入到综合数据库中，重新遍历条件，更新符合度，如果符合度最终不为1，则进行逆向推理，补充询问可能的且没有在综合数据库中的条件，进行判断，将其加入综合数据库，重新进行正向推理。
* 示例：

已知事实：黄褐色，哺乳类，肉食类

综合数据库：11.黄褐色，20.哺乳类，22.肉食类

从规则库取出规则，与综合数据库中的已知事实匹配

R9：哺乳类 ∧ 肉食类∧ 黄褐色 ∧ 有暗斑点→ 金钱豹

R10：哺乳类∧ 肉食类 ∧ 黄褐色 ∧ 有黑色条纹→虎

由规则R9和R10知条件并不是全部满足，推出该动物为金钱豹或者虎的概率均为0.75

运行反向推理，询问用户补充条件:

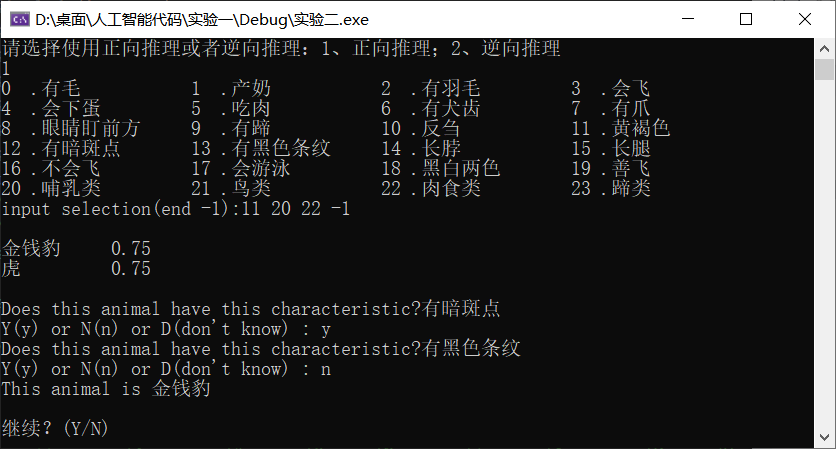
1.该动物是否具有这个特征：有暗斑点？是

2.该动物是否具有这个特征：有黑色条纹？否

得到用户回复后，补充条件：

综合数据库：11.黄褐色，12.暗斑点，20.哺乳类，22.肉食类

重新进行正向推理，得出该动物为金钱豹。

* 程序运行结果：

1. **心得体会**

在产生式系统的构建过程中，我实际构建了规则库、综合数据库和控制系统，设计了一个小型动物识别系统。在对程序的编写过程中，我遇到了很多问题：进行正向推理时，不知道如何保存推理后的结果，根据上网查找资料，通过数组记录标记将中间结论添加在综合数据库中；进行逆向推理时，更加给定动物推出的条件总存在差异，后面一步一步调试发现是规则和推导结果对应关系错误。

经过本次实际设计操作，我对产生式系统有了更深刻的认识，对正向推理、反向推理和混合推理的逻辑和过程更加清晰，巩固了课本相关知识，提高了我的实际应用能力。在遇到问题时，首先要进行独立思考，解决不了时再上网查找或者请教别人，这样可以提高我们解决问题的能力。