人工智能第五次作业
迷宫问题的两种解法

姓名：王丹

学号：2120151036

目录

一、实验目的.....3
二、实验内容.....3
三、实验环境.....3
四、实验原理.....3
1.迷宫寻路问题.....3
2.A*算法.....4
3.一阶谓词逻辑.....4
五、实验步骤及算法实现.....5
1.A*算法流程及伪代码.....5
2.一阶谓词逻辑步骤及伪代码.....7
六、实验结果及分析.....8
1.A*算法.....9
2.一阶谓词逻辑.....10
七、实验小结.....11
八、参考文献.....11

一、实验目的

1.熟悉和掌握 A*算法，实现迷宫寻路功能，掌握启发式函数的编写以及各启发式函数效果的比较。

2.熟悉一阶谓词逻辑，掌握基于规则推理的基本方式。

二、实验内容

1.A*算法

（1）学习 A*算法的原理，思考求解过程，画出 A*算法求解迷宫最短路径的流程图。

（2）设置不同的地图，以及不同的起点和终点，记录 A*算法的求解结果。

（3）对于相同的起点和终点，设计不同的启发式函数，比较不同启发式函数对迷宫寻路问题的影响。

2.一阶谓词逻辑

（1）学习一阶谓词的原理，结合迷宫问题，完成迷宫问题事实库和规则库的建立。

（2）规矩建立的事实库和规则库，写出迷宫问题的求解过程。

（3）分析影响一阶谓词逻辑求解路径的因素。

三、实验环境

1.编程语言：JAVA

2.编程平台：NetBeans8.1

3.Java 环境：JRE1.8.0

4.系统环境：ubuntu 14.04 LTS

四、实验原理

1.迷宫寻路问题

迷宫寻路问题可以表述为：一个二维网格，0 表示点不允许通过，1 表示点允许通过，点用(x, y)表示，寻找从某一个给定的起始单元格出发，经由行相邻或列相邻的单元格（允许通过的），最终可以到达目标单元格的、所走过的单元格序列。在任一个单元格中，都只能看到与它邻近的 4 个单元格（如果位于底边，则只有 3 个；位于 4 个角上，则只有两个临近单元格）。

2.A*算法

A*算法是一种静态路网中求解最短路径最有效的方法。公式表示为：$f(n)=g(n)+h(n)$，其中$f(n)$是节点n从初始点到目标点的估价函数，$g(n)$是在状态空间中从初始节点到n节点的实际代价，$h(n)$是从n到目标节点最佳路径的估计代价。保证找到最短路径（最优解）条件，关键在于估价函数$h(n)$的选取：估价值$h(n)$小于等于n到目标节点的距离实际值$h^*(n)$，这种情况下，搜索的点数多，搜索范围大，效率低，但能得到最优解。如果股价大于实际值，搜索的点数少，搜索范围小，效率高，但不能保证得到最优解。

在迷宫寻路问题中，A*算法的解是全局最优解。

假设在一个 n*m 的迷宫里，入口坐标和出口坐标分别是（0，0）和（4，4），每一个坐标点有两种可能：0 和 1，其中 0 表示该位置不允许通过，1 表示该位置允许通过。

如地图：

(x,y)	0	1	2	3	4
0	1(入口)	1	1	1	1
1	0	1	0	0	1
2	1	1	0	0	1
3	1	0	1	1	1
4	1	1	1	0	1(出口)

最短路径应该是

(x,y)	0	1	2	3	4
0	A(入口)	B	C	D	E
1	0	1	0	0	F
2	1	1	0	0	G
3	1	0	1	1	H
4	1	1	1	0	I(出口)

即：(0,0)→(0,1)→(0,2)→(0,3)→(0,4)→(1,4)→(2,4)→(3,4)→(4,4)。

3.一阶谓词逻辑

谓词由谓词名和个体组成，用于表示事物或概念的性质、状态，以及事物或概念之间的相互关系。这里，个体表示独立存在的事物或者抽象的概念，谓词名指出个体的某种性质、状态，或个体之间的某种关系，其语义根据需要人为确定。设P为谓词名，$x_1,x_2,\ldots\ldots,x_n$为 n 个个体，则谓词的一般形式为

$$P(x_1,x_2,\ldots,x_n)$$

n 被称为谓词的元数，n=1 时为 1 元谓词，n=2 时为二元谓词……谓词中，个

体可以是常量，也可以是变量和函数。个体常量、个体变量和函数统称为项。个体变量的取值范围被称为个体域。个体域可以是有限的，也可以是无限的。如果一个为此的个体都是常量、变量或函数，则称为一阶谓词。

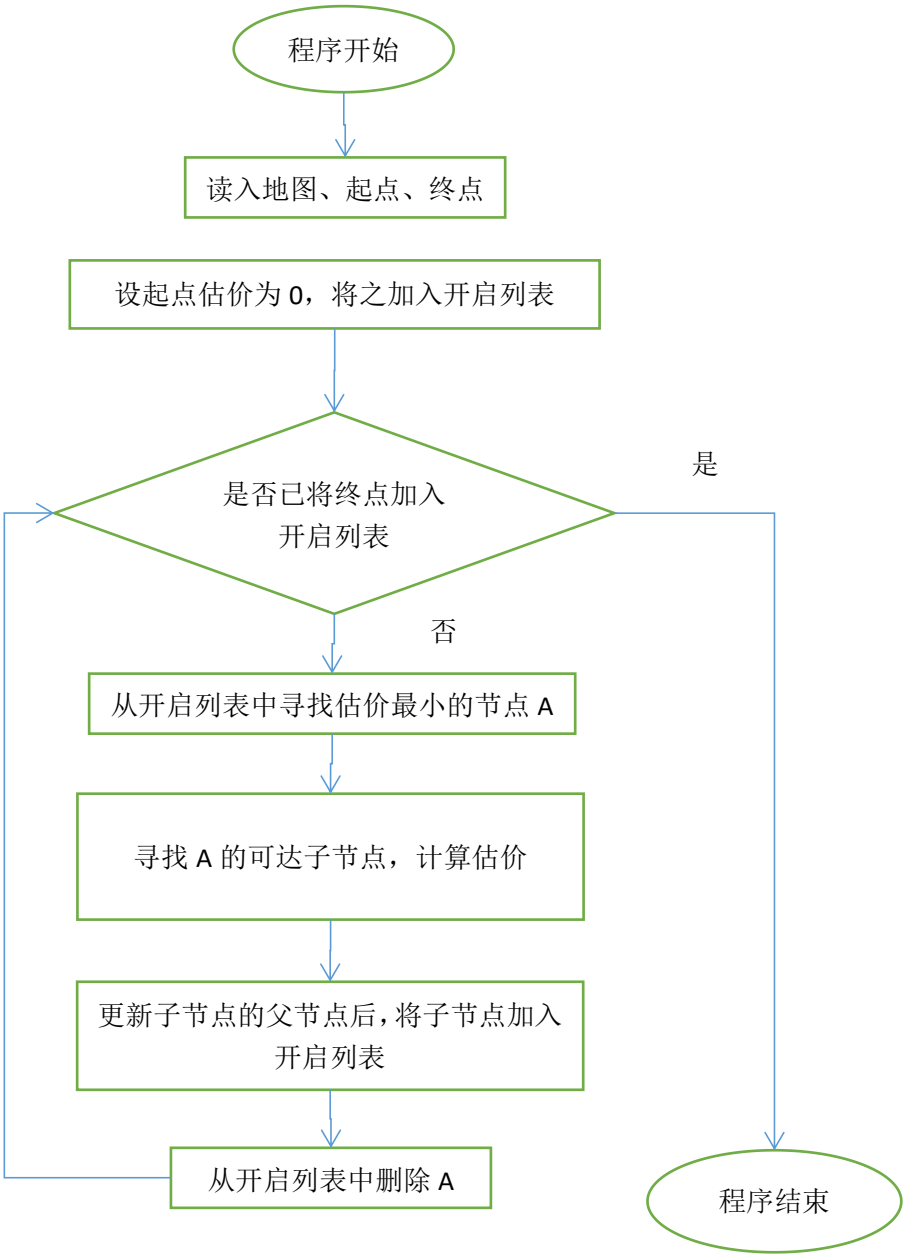
一般来说，谓词公式是用连接词$(\neg,\vee,\wedge,\rightarrow,\leftrightarrow)$和量词$(\forall,\exists)$将谓词连接起来所构成的公式。

当用谓词表示知识时，首先定义原子谓词，然后用连接词或量词把这些谓词连结起来，形成相应的谓词公式。谓词不仅可以用来表示事实性知识，也可以用来表示过程性知识。对于事实性知识，通常用否定、析取或合取符号连接起来的谓词公式表示。对于过程性知识，通常用蕴含式表示。

五、实验步骤及算法实现

1.A*算法流程及伪代码

1.1A*算法解决迷宫问题流程图


程序开始
读入地图、起点、终点
设起点估价为 0，将之加入开启列表
是否已将终点加入
开启列表
是
否
从开启列表中寻找估价最小的节点 A
寻找 A 的可达子节点，计算估价
更新子节点的父节点后，将子节点加入
开启列表
从开启列表中删除 A
程序结束

1.2 伪代码实现

输入：地图大小 n*m,地图矩阵，起点坐标，终点坐标
输出：从起点到终点的坐标序列
算法过程：

```
初始化地图矩阵 mazeMatrix、起点 enterPoint、终点 exitPoint；
初始化开启列表 openTable，关闭列表 closeTable；
openTable.add(enterPoint);
While(!openTable.contain(exitPoint)){
    currentPoint = min(openTable);
    sonPoints = currentPoint.findSonPoints();//包含 f、g、h 的计算
    for point:sonPoints
        point.setFatherPoint(currentPoint);
        If(openTable.contain(point))
            point.CompareandUpdateFatherPoint();
            openTable.updatePoint();
        else
            openTable.addPoint();
}
```

2.一阶谓词逻辑步骤及伪代码

2.1 一阶谓词逻辑解决迷宫问题步骤

（1）建立事实库。

$Road(x)$：x 是一个允许通过的节点。

$OneDis(x,y)$： x 和 y 之间的曼哈顿距离是 1。

$Neighbor(x,y)$： x 和 y 是邻居节点。

$Available(x_1,x_2,\ldots,x_n)$： x_1 和 x_n 之间路径可达。

（2）编辑知识库，输入规则，完成整个规则库的建立。

$\forall x\exists y(Road(x)\wedge Road(y)\wedge OneDis(x,y))\rightarrow Neighbor(x,y)$；

$\forall x_1\ldots x_n\exists x_{n+1}(Available(x_1,\ldots,x_n)\wedge Neighbor(x_n,x_{n+1}))\rightarrow Available(x_1,\ldots,x_n,x_{n+1})$

（3）确定问题状态空间。

初始状态： $Available(enterPoint)$

目标状态： $Available(enterPoint,x_1,\ldots,x_m,ExitPoint)$

（4）运行推理，给出相应的推理过程、事实区和规则区。

2.2 伪代码实现

输入：迷宫地图矩阵、起点坐标、终点坐标
输出：从起点到终点的路径序列
算法实现：

Step1:读入地图矩阵、起点坐标、终点坐标；

Step2:构建通路事实库：根据地图矩阵生成 $Road(x_i)$，x_i 是地图矩阵中的一个点。

Step3:构建距离事实库：生成 x_i 的 $OneDis(x_i, x_j)$

Step4:

While(! $Available(enterPoint, x_1, \ldots, x_m, ExitPoint)$){

```
Step4.1: lastPoint = Available(……);
Step4.2: {
        lastPoint.findNeighbor();
        If(lastPoint.Neighbor().size==0)
            Remove lastPoint from Available(……);
            Goto Step4.1;
        }
Step4.3: currentLastPoint = lastPoint.Neighbor().get(0);
Step4.4: lastPoint.Neighbor().remove(0);
Step4.5: {
        Add currentLastPoint into Available(……);
        }
}
```

六、实验结果及分析

本实验中使用的迷宫矩阵如下所示：

(x,y)	0	1	2	3	4	5
0	1	1	1	1	1	1
1	0	1	0	1	0	1
2	1	1	0	1	0	1
3	1	0	1	1	1	1
4	1	1	1	0	1	0

其中，0 表示不允许通过，1 表示允许通过。

1.A*算法

（1）相同的启发函数，不同的起点和终点

设置启发函数 $f(n)=g(n)+h(n)$，其中 $g(n)=g(n-1)+10, s.t. g(0)=0$，

$h(n)=(|x_n-x_{exit}|+|y_n-y_{exit}|)*10$。

运行结果如下：

入口	出口	最短路径
(0,0)	(4,4)	(0,0)→(0,1)→(0,2)→(0,3)→(1,3)→(2,3)→(3,3)→(3,4)→(4,4)
(2,0)	(2,5)	(2,0)→(2,1)→(1,1)→(0,1)→(0,2)→(0,3)→(0,4)→(0,5)→(1,5)→(2,5)
(0,3)	(4,0)	(0,3)→(1,3)→(2,3)→(3,3)→(3,2)→(4,2)→(4,1)→(4,0)

部分截图如下：

```
入口节点：0;3
出口节点4;0
stats函数启动
第0轮迭代: 当前选择的节点是0;3  第1轮迭代: 当前选择的节点是1;3  它的父节点是0;3
第2轮迭代: 当前选择的节点是2;3  它的父节点是1;3
第3轮迭代: 当前选择的节点是3;3  它的父节点是2;3
第4轮迭代: 当前选择的节点是3;2  它的父节点是3;3
第5轮迭代: 当前选择的节点是4;2  它的父节点是3;2
第6轮迭代: 当前选择的节点是4;1  它的父节点是4;2
已经找到出口！！
(4,0)<==(4,1)<==(4,2)<==(3,2)<==(3,3)<==(2,3)<==(1,3)<==(0,3)运行时间：5毫秒
成功构建 (总时间: 0 秒)
```

（2）相同的起点和终点，不同的启发函数

起点坐标：$(2,0)$；终点坐标：$(2,5)$

启发函数 a：设置启发函数 $f(n)=g(n)+h(n)$，其中 $g(n)=g(n-1)+10, s.t. g(0)=0$，

$h(n)=(|x_n-x_{exit}|+|y_n-y_{exit}|)*10$。

启发函数 b：设置启发函数 $f(n)=g(n)+h(n)$，其中 $g(n)=g(n-1)+10, s.t. g(0)=0$，

$h(n)=\sqrt{(x_n-x_{exit})^2+(y_n-y_{exit})^2}*10$。

启发函数	最短路径	运行时间 (ms)
a	(2,0)→(2,1)→(1,1)→(0,1)→(0,2)→(0,3)→(0,4)→(0,5)→(1,5)→(2,5)	3
b	(2,0)→(3,0)→(4,0)→(4,1)→(4,2)→(3,2)→(3,3)→(3,4)→(3,5)→(2,5)	4

截图如下：

```
入口节点：2;0
出口节点2;5
stats函数启动
第0轮迭代: 当前选择的节点是2;0  第1轮迭代: 当前选择的节点是3;0  它的父节点是2;0
第2轮迭代: 当前选择的节点是4;0  它的父节点是3;0
第3轮迭代: 当前选择的节点是2;1  它的父节点是2;0
第4轮迭代: 当前选择的节点是4;1  它的父节点是4;0
第5轮迭代: 当前选择的节点是4;2  它的父节点是4;1
第6轮迭代: 当前选择的节点是1;1  它的父节点是2;1
第7轮迭代: 当前选择的节点是3;2  它的父节点是4;2
第8轮迭代: 当前选择的节点是0;1  它的父节点是1;1
第9轮迭代: 当前选择的节点是3;3  它的父节点是3;2
第10轮迭代: 当前选择的节点是0;2  它的父节点是0;1
第11轮迭代: 当前选择的节点是0;0  它的父节点是0;1
第12轮迭代: 当前选择的节点是2;3  它的父节点是3;3
第13轮迭代: 当前选择的节点是3;4  它的父节点是3;3
第14轮迭代: 当前选择的节点是4;4  它的父节点是3;4
第15轮迭代: 当前选择的节点是0;3  它的父节点是0;2
第16轮迭代: 当前选择的节点是1;3  它的父节点是0;3
第17轮迭代: 当前选择的节点是0;4  它的父节点是0;3
第18轮迭代: 当前选择的节点是0;5  它的父节点是0;4
第19轮迭代: 当前选择的节点是1;5  它的父节点是0;5
已经找到出口！！
(2,5)<==(1,5)<==(0,5)<==(0,4)<==(0,3)<==(0,2)<==(0,1)<==(1,1)<==(2,1)<==(2,0)运行时间：4毫秒
```

```
入口节点：2;0
出口节点2;5
stats函数启动
第0轮迭代: 当前选择的节点是2;0  第1轮迭代: 当前选择的节点是3;0  它的父节点是2;0
第2轮迭代: 当前选择的节点是4;0  它的父节点是3;0
第3轮迭代: 当前选择的节点是4;1  它的父节点是4;0
第4轮迭代: 当前选择的节点是4;2  它的父节点是4;1
第5轮迭代: 当前选择的节点是2;1  它的父节点是2;0
第6轮迭代: 当前选择的节点是1;1  它的父节点是2;1
第7轮迭代: 当前选择的节点是3;2  它的父节点是4;2
第8轮迭代: 当前选择的节点是3;3  它的父节点是3;2
第9轮迭代: 当前选择的节点是3;4  它的父节点是3;3
第10轮迭代: 当前选择的节点是4;4  它的父节点是3;4
第11轮迭代: 当前选择的节点是3;5  它的父节点是3;4
已经找到出口！！
(2,5)<==(3,5)<==(3,4)<==(3,3)<==(3,2)<==(4,2)<==(4,1)<==(4,0)<==(3,0)<==(2,0)运行时间：4毫秒
成功构建 (总时间: 0 秒)
```

（3）分析

经过如上实验，可以发现，A*算法能找到迷宫问题的解路径，并且是全局最优解。当启发函数不同时，估价值不同，产生的最优解不同。此时，为确保产生全局最优解，应保证 $h(n)$ 小于等于真实值 $h^*(n)$。运行时间与启发函数有关，因为不同启发函数带来的寻路次数不同，启发函数自身计算时间不同等等。

2.一阶谓词逻辑

（1）起点坐标： $(0,0)$

终点坐标：(4,4)

运行结果如下：

```
第0次循环
(0,0)==>(0,1)==>第1次循环
(0,0)==>(0,1)==>(0,2)==>第2次循环
(0,0)==>(0,1)==>(0,2)==>(0,3)==>第3次循环
(0,0)==>(0,1)==>(0,2)==>(0,3)==>(0,4)==>第4次循环
(0,0)==>(0,1)==>(0,2)==>(0,3)==>(0,4)==>(0,5)==>第5次循环
(0,0)==>(0,1)==>(0,2)==>(0,3)==>(0,4)==>(0,5)==>(1,5)==>第6次循环
(0,0)==>(0,1)==>(0,2)==>(0,3)==>(0,4)==>(0,5)==>(1,5)==>(2,5)==>第7次循环
(0,0)==>(0,1)==>(0,2)==>(0,3)==>(0,4)==>(0,5)==>(1,5)==>(2,5)==>(3,5)==>第8次循环
(0,0)==>(0,1)==>(0,2)==>(0,3)==>(0,4)==>(0,5)==>(1,5)==>(2,5)==>(3,5)==>(3,4)==>第9次循环
(0,0)==>(0,1)==>(0,2)==>(0,3)==>(0,4)==>(0,5)==>(1,5)==>(2,5)==>(3,5)==>(3,4)==>(3,3)==>第10次循环
(0,0)==>(0,1)==>(0,2)==>(0,3)==>(0,4)==>(0,5)==>(1,5)==>(2,5)==>(3,5)==>(3,4)==>(3,3)==>(3,2)==>第11次循环
(0,0)==>(0,1)==>(0,2)==>(0,3)==>(0,4)==>(0,5)==>(1,5)==>(2,5)==>(3,5)==>(3,4)==>(3,3)==>(3,2)==>(4,2)==>第12次循环
(0,0)==>(0,1)==>(0,2)==>(0,3)==>(0,4)==>(0,5)==>(1,5)==>(2,5)==>(3,5)==>(3,4)==>(3,3)==>(3,2)==>(4,2)==>(4,1)==>第13次循环
(0,0)==>(0,1)==>(0,2)==>(0,3)==>(0,4)==>(0,5)==>(1,5)==>(2,5)==>(3,5)==>(3,4)==>(3,3)==>(3,2)==>(4,2)==>(4,1)==>(4,0)==>第14次循环
(0,0)==>(0,1)==>(0,2)==>(0,3)==>(0,4)==>(0,5)==>(1,5)==>(2,5)==>(3,5)==>(3,4)==>(3,3)==>(3,2)==>(4,2)==>(4,1)==>(4,0)==>(3,0)==>第15次循环
(0,0)==>(0,1)==>(0,2)==>(0,3)==>(0,4)==>(0,5)==>(1,5)==>(2,5)==>(3,5)==>(3,4)==>(3,3)==>(3,2)==>(4,2)==>(4,1)==>(4,0)==>(3,0)==>(2,0)==>第16次循环
(0,0)==>(0,1)==>(0,2)==>(0,3)==>(0,4)==>(0,5)==>(1,5)==>(2,5)==>(3,5)==>(3,4)==>(3,3)==>(3,2)==>(4,2)==>(4,1)==>(4,0)==>(3,0)==>(2,0)==>(2,1)==>第17次循环
(0,0)==>(0,1)==>(0,2)==>(0,3)==>(0,4)==>(0,5)==>(1,5)==>(2,5)==>(3,5)==>(3,4)==>(3,3)==>(3,2)==>(4,2)==>(4,1)==>(4,0)==>(3,0)==>(2,0)==>(2,1)==>(1,1)==>第18次循环
(0,0)==>(0,1)==>(0,2)==>(0,3)==>(0,4)==>(0,5)==>(1,5)==>(2,5)==>(3,5)==>(3,4)==>(3,3)==>(2,3)==>第19次循环
(0,0)==>(0,1)==>(0,2)==>(0,3)==>(0,4)==>(0,5)==>(1,5)==>(2,5)==>(3,5)==>(3,4)==>(3,3)==>(2,3)==>(1,3)==>第20次循环
(0,0)==>(0,1)==>(0,2)==>(0,3)==>(0,4)==>(0,5)==>(1,5)==>(2,5)==>(3,5)==>(3,4)==>(4,4)==>程序结束
```

找到解路径：

(0,0)→(0,1)→(0,2)→(0,3)→(0,4)→(0,5)→(1,5)→(2,5)→(3,5)→(3,4)→(4,4)

对循环过程解释如下：（以第 5 次循环为例）

$$Available((0,0),(0,1),(0,2),(0,3),(0,4)) \wedge Neighbor((0,4),(0,5))$$
$$\rightarrow Available((0,0),(0,1),(0,2),(0,3),(0,4),(0,5))$$

（2）分析：

一阶谓词逻辑编程求解迷宫问题的主要目的是体会一阶谓词逻辑的形式和推理方式，所以在编程中并未加入寻求最优解的步骤。编写的代码最后找到的只是解空间中的一个解，未必是全局最优解。这带来的后果是代码编写会对求解产生影响。一种解决方法是在 findNeighbor 步骤中加入权重，以此为据找到合取式的 Nerghbor 项。编写一阶谓词逻辑程序要注意的是在定义谓词过程避免产生冗余谓词，冗余谓词会给推理过程带来不利影响。

七、实验小结

本次实验，我理解并编程写出了迷宫寻路问题的 2 种解法：A*算法和一阶谓词逻辑。此外，能够辨别启发式搜索和穷举式搜索也是收获之一。

八、参考文献

[1]刘峡壁，人工智能导论：方法与系统，北京：国防工业出版社，2008.8；
[2]人工智能及其应用：实验指导书，浙江工业大学，2011.9