**哈尔滨工业大学**

**人工智能实验二**

**报告**

**(2015年度秋季学期)**

**授课教师 赵铁军 郑德权**

**学生姓名 张 琦**

**学 号 15S003055**

**学 院 计算机科学与技术**

**一、问题描述**

**1.1 待解决问题的解释**

三枚钱币问题:

设有三枚钱币，其排列处在“正、正、反”状态，现允许每次翻动其中任意一个钱币，

问只许操作三次的情况下，如何翻动钱币使其变成“正、正、正”或“反、反、反”状态。

**1.2 状态转换图**

四元组(x,y,z,n):

x,y,z∈[0,1]，1表示钱币为正面，0表示钱币为反面。n=0,1,2,3，表示当前状态是经过n次翻钱币得到的。

(0,0,1,3)

(1,1,1,3)

(1,0,0,3)

(0,1,0,3)

(1,0,1,2)

(0,1,1,2)

(0,0,0,2)

(1,1,0,2)

(1,1,1,1)

(1,0,0,1)

(0,1,0,1)

(1,1,0,0)

**1.3 方法介绍**

将钱币的状态用四元组表示之后，问题转换为搜索问题，树的深度不超过3。

**二、算法介绍**

**2.1产生式系统表示**

1. 综合数据库  
   定义四元组：（x, y, z, n）

x,y,z∈[0,1]，1表示钱币为正面，0表示钱币为反面。n=0,1,2,3，表示当前状态是经过n次翻钱币得到的。

1. 规则库

r1: IF (x, y, z, n) THEN (~x, y, z, n+1)

r2: IF (x, y, z, n) THEN (x, ~y, z, n+1)

r3: IF (x, y, z, n) THEN (x, y, ~z, n+1)

其中~x表示对x取反。

1. 初始状态 (1, 1, 0, 0)
2. 结束状态 (1, 1, 1, n) 或者(0, 0, 0, n)，n<=3

**三、算法实现**

**3.1 实验环境与问题规模**

实验环境：windows10+codeblocks

编程语言：C++

问题规模 O(1)

**3.2 数据结构**

struct state{

int x;

int y;

int z;

int n;

};

struct point{

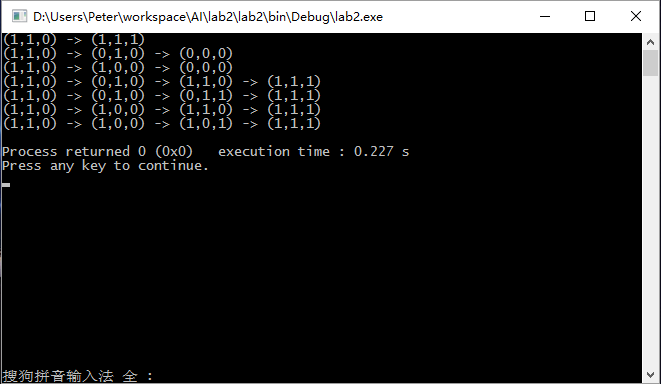
state s;

queue<state> path;

};

queue<point> q;

**3.3 最终输出结果**



**3.4 实验结果**

硬币有七种策略翻转可以在三次以内得到“正、正、正”或“反、反、反”的状态

令R1表示翻转第一枚硬币、R2表示翻转第二枚硬币、R3表示翻转第三枚硬币

1. R3=>“正、正、正”
2. R1,R2=>“反、反、反”
3. R2,R1=>“反、反、反”
4. R1,R1,R3=>“正、正、正”
5. R1,R3,R1=>“正、正、正”
6. R2,R2,R3=>“正、正、正”
7. R2,R3,R2=>“正、正、正”

**附录— 源代码及其注释**

#include <iostream>

#include <queue>

#include<stdlib.h>

using namespace std;

struct state{//状态四元组

int x;

int y;

int z;

int n;

};

struct point{//树节点，包括当前状态以及已有路径

state s;

queue<state> path;

};

queue<point> q;

void print\_state(state s)//打印状态的函数

{

cout<<"("<<s.x<<","<<s.y<<","<<s.z<<")";

}

void print\_path(point p)//打印路径的函数

{

while(!p.path.empty())

{

state tmp\_s = p.path.front();

p.path.pop();

print\_state(tmp\_s);

if (!p.path.empty())

cout<<" -> ";

}

}

int main()

{

point start;

start.s = {1,1,0,0};

start.path.push(start.s);

q.push(start);//将初始状态压入队列

while(!q.empty())

{

point tmp\_p = q.front();

q.pop();

//print\_state(tmp\_p.s);

//system("pause");

if((tmp\_p.s.x==1&&tmp\_p.s.y==1&&tmp\_p.s.z==1)||(tmp\_p.s.x==0&&tmp\_p.s.y==0&&tmp\_p.s.z==0))//如果为目标节点则打印路径，不再拓展节点

{

print\_path(tmp\_p);

cout<<endl;

continue;

}

if(tmp\_p.s.n>=3)//如果树的深度已经为3则不再拓展

continue;

point r1, r2, r3;//拓展的三个节点

r1=tmp\_p;

r1.s.x = 1- r1.s.x;

r1.s.n++;

r1.path.push(r1.s);

r2=tmp\_p;

r2.s.y = 1-r2.s.y;

r2.s.n++;

r2.path.push(r2.s);

r3=tmp\_p;

r3.s.z = 1-r3.s.z;

r3.s.n++;

r3.path.push(r3.s);

q.push(r1);

q.push(r2);

q.push(r3);

}

return 0;

}