哈爾濱工業大學（深圳）
HARBIN INSTITUTE OF TECHNOLOGY

实验报告

开课学期：大三上学期

课程名称：人工智能

实验名称：猴子摘香蕉

实验性质：设计

实验时间：2019.10.14 地点：T2109 实验台号 21

学生专业：计算机类

学生学号：SZ170110311

学生姓名：廖思瑀

评阅教师：

报告成绩：

实验与创新实践教育中心印制

2019 年 10 月

一. 简介/问题描述

1.1 待解决问题的解释

猴子摘香蕉：

一个房间里，天花板上挂有一串香蕉，有一只猴子可在房间里任意活动（到处走动，推移箱子，攀登箱子等）。设房间里还有一只可被猴子移动的箱子，且猴子登上箱子时才能摘到香蕉，问猴子在某一状态下（设猴子位置为 A，箱子位置为 B，香蕉位置在 C），如何行动可摘取到香蕉。

1.2 问题的形式化描述

猴子摘香蕉：

定义描述状态谓词

符号	表示	取值范围
Monkey	猴子的位置	{A,B,C}
Box	箱子的位置	{A,B,C}
On	是否站在箱子上	{0,1}
Banana	是否得到香蕉	{0,1}

初始状态(A, B, 0, 0) → 终止状态（C, C, 1, 1）

定义四个操作：

Move(x,y)：猴子从 x 处移向 y

Push(x,y)：猴子推着箱子从 v 走到 w 处

Climb 猴子爬上箱子

Get 猴子摘到香蕉

1.3 解决方案介绍（原理）

利用谓词逻辑表示法的原理，定义相应的初始状态和终止状态，并利用谓词单元，定义有关的动作，利用函数表示动作，最后，利用宽度优先搜索算法，使得从初始状态到达终止状态，实现猴子摘香蕉。

二. 算法介绍

2.1 所用方法的一般介绍

利用广度优先搜索算法：

广度优先搜索算法（Breadth-First Search，BFS）是一种盲目搜寻法，目的是系统地展开并检查图中的所有节点，以找寻结果。换句话说，它并不考虑结果的可能位置，彻底地搜索整张图，直到找到结果为止。

2.2 算法伪代码

（1）定义相应的状态谓词

（2）定义四个动作操作

（3）广度优先搜索实现功能

1）新状态入队列

2）判断队列是否为空

若非空

删除当前状态

执行四个动作操作

3）判断当前状态是否为终止状态

若非终止状态

将当前状态的前一状态入栈

4）判断当前栈是否为空

若非空

将栈中元素 pop 出栈

判断猴子是否摘到香蕉

打印当前状态信息

源代码：

```
public class Monkey {

    class State{
        Character monkey; //猴子的位置
        Character box;    //盒子的位置
        int on;  //是否在盒子上
        int getbanana; //是否摘到香蕉
        int no;     //当前状态的序号
        String op;  //操作

        public State(char monkey, char box, int onbox, int banana, int no, String op) {
            this.monkey = monkey;
            this.box = box;
            this.on = onbox;
            this.getbanana = banana;
            this.no = no;
            this.op = op;           //进行了什么操作
        }
        public State(){

        }

        @Override
        public String toString() {
            return "State{" + "monkey=" + monkey + ", box=" + box + ", on=" + on + ", getbanana=" + getbanana+"}";
        }



    }

   boolean flag = false;    //手中无香蕉
   private int num = 0;     //用来给一个状态编序号
   int[] record  = new int[100];  //用来记录某个状态的上一个状态，方便回溯找到到这一步的路径
   Queue<State> queue = new LinkedList();
   ArrayList<State> StateList = new ArrayList<>();//用来保存所有尝试过的状态
```

```
/**
 * 进行记录
 */
public void Record(State next,int pre){
    this.StateList.add(this.num, next);
    this.record[this.num] = pre;  //记录路径，state.no为上一个状态
    this.queue.add(next);         //将该状态加入队列
    System.out.println("新状态进入队列-"+next.toString());
}
/**
 * 猴子向某一处移动
 * @param state
 */

public void Move(State state){
    State next  = null;
    if(state.on == 0){
        if(state.monkey == 'A'){
            next = new State('B',state.box,0,0,++num,"Move from A to B");     //新状态猴子从A走向B

        }
        else if(state.monkey == 'B'){
            next = new State('A',state.box,0,0,++num,"Move from B to A");     //新状态猴子从B走向A
        }
    }
  if(next!=null){
      Record(next,state.no);
  }
}
/**
 * 推箱子
 * @param state
 */
public void push(State state){
    //推箱子的条件：猴子与盒子在同一个地方,手上没有香蕉，猴子没站在箱子上
    State next = null;
    if(state.monkey.equals(state.box)&&state.on==0&&state.getbanana==0){
        //只要能推盒子在任意地方都推向C
```

```
    //推箱子的条件：猴子与盒子在同一个地方，手上没有香蕉，猴子没站在箱子上
    State next = null;
    if(state.monkey.equals(state.box)&&state.on==0&&state.getbanana==0){
        //只要能推盒子在任意地方都推向c
        next = new State('C','C',0,0,++num,"pushBox from"+state.box.toString()+"to C");
    }
    if (next!=null){
        Record(next, state.no);
    }
}
/**
 * 是否能爬上箱子
 * @param state
 */
public void Climb(State state){
    State next = null;
    //猴子和箱子在同一个地方，并且猴子不站在箱子上
    if(state.monkey.equals(state.box)&&state.on==0){
        next = new State(state.monkey,state.box,1,0,++num,"Climb");
    }
    if (next!=null){
        Record(next, state.no);
    }
}
/**
 * 猴子得到香蕉
 * @param state
 */
public void Get(State state){
    State next = null;
    //猴子在c，箱子在c，猴子站在箱子上，猴子还没摘到香蕉
    if(state.box=='C'&&state.monkey=='C'&&state.on==1&&this.flag==false){
        next = new State('C','C',1,1,++num,"Get");
        this.flag = true;
    }
    if (next!=null){
        Record(next, state.no);
    }
}
/**
```

```
/**
 * 广度优先搜索，寻找解决方案
 */
public void findSolution(){
    State start = new State('A','B',0,0,num,"start");
    this.StateList.add(0,start);
    this.queue.add(start);
    System.out.println("新状态进入队列-"+start.toString());
    while(!queue.isEmpty()){
        State now = queue.poll();
        System.out.println("队列中删除状态-"+now.toString());
        Move(now);
        push(now);
        Climb(now);
        Get(now);
        if(this.flag){
            break;
        }
    }
    if(this.flag==false){
        System.out.println("impossible");
        return;
    }

    Stack<Integer> stack = new Stack();
    int now = this.num;      //结束状态的序号
    stack.push(now);
    while(now!=0){
        int pre = record[now];
        stack.push(pre);
        now = pre;
    }

    System.out.println("\n\n");
    while(!stack.isEmpty()){
         now = stack.pop();
         State nowState = this.StateList.get(now);
         System.out.println(nowState.op);
         System.out.println("当前状态:"+nowState.toString());

    }

}
 public static void main(String[] args) {
     Monkey test = new Monkey();
     test.findSolution();
 }
```

三． 算法实现

3.1 实验环境与问题规模

实验环境：java

问题规模：不需要占用过多的内存运行：

3.2 数据结构

数组：一维数组用于某个状态的上一个状态

栈：利用栈实现 bfs

队列：利用队列将摘香蕉的相应的状态进行连接

3.3 实验结果

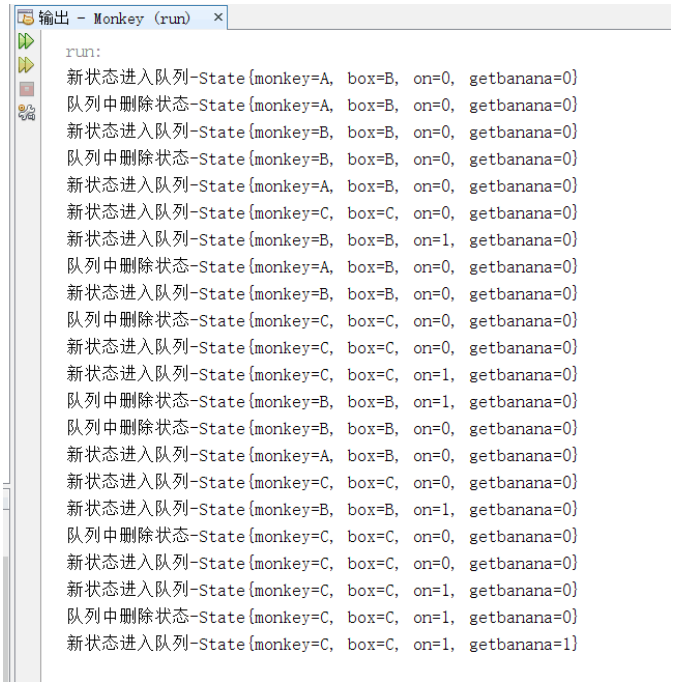
（1）从队列中弹出一个状态

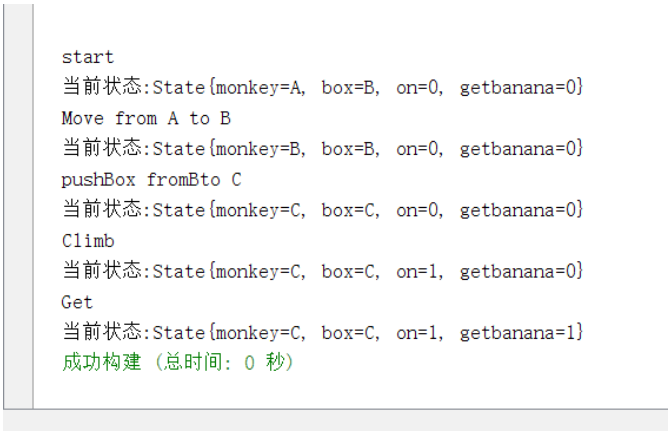
（2）将该状态可到达的状态压入队列中，并记录是通过什么操作到达该状态

（3）重复上述两个过程，直到完成任务

（4）打印操作路线

3.4 系统中间及最终输出结果（要求有屏幕显示）





在较短的耗时内成功摘到香蕉

四． 总结及讨论（对该实验的总结以及任何该实验的启发），

此次的猴子摘香蕉实验，主要考察我们对于谓词逻辑表示法的理解与实际运用，利用谓词逻辑表示法解决实际的问题。这次的实验难度上适中，同时还能考察我们对于人工智能相关知识的灵活使用，是一个很好的实验题目。