# 简介/问题描述

## 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。

## 问题的形式化描述

Define state predicates as following:  
 Site(x,y): x is at y;  
 Hang(w,y): w is hanging on y;  
 On(z): z is standing on the box;  
 Holds(z): bananas are held by z;

Define 4 operations:  
 Goto(u,v): monkey goes to v from u;  
 PushBox(v,w): monkey goes to w from v with boxes;  
 ClimeBox: monkey climbs up box;  
 Grasp: monkey gets the banana;  
  
Define initial state:  
 S\_0 = Site(Monkey, a)  
 && Hang(banana, b)  
 && Site(Box, c)  
 && !On(Monkey)  
 && !Holds(Monkey)  
  
Define final state:  
 S\_g = Site(Monkey, b)  
 && !Hang(banana, b)  
 && Site(Box, b)  
 && On(Monkey)  
 && Holds(Monkey)

## 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。在具体代码实现中，将某个物体的状态表示为某个类的属性；将操作定义为一个函数，条件部分用if语句表示，动作部分用来执行具体的操作。

# 算法介绍

## 所用方法的一般介绍

在定义了对应的谓词、操作后，使用回溯算法, 暴力搜索所有状态空间。在搜索过程中，Goto操作和PushBox操作一定是由Monkey进行的，因此起点一定是Monkey.site，所以可以对搜索空间进行剪枝，使时间复杂度从O(n3)变为O(n2)。

## 算法伪代码

1. 定义初始状态，状态空间place
2. for i in place:  
    Monkey go to i  
    tempMonkey = monkey  
    tempBox = box
3. for j in place:  
    tempMonkey1 = monkey  
    tempBox1 = box  
    monkey pushes box to j

Monkey climbs up box   
 Monkey tries to get banana

1. if Monkey get the banana

结束  
 monkey = tempMonkey1  
 box = tempBox1

返回3

1. monkey = tempMonkey  
    box = tempBox
2. 返回2
3. 没有答案

# 算法实现

## 实验环境与问题规模

实验环境: Python 3.8, pycharm

问题规模: O(n3)

## 数据结构

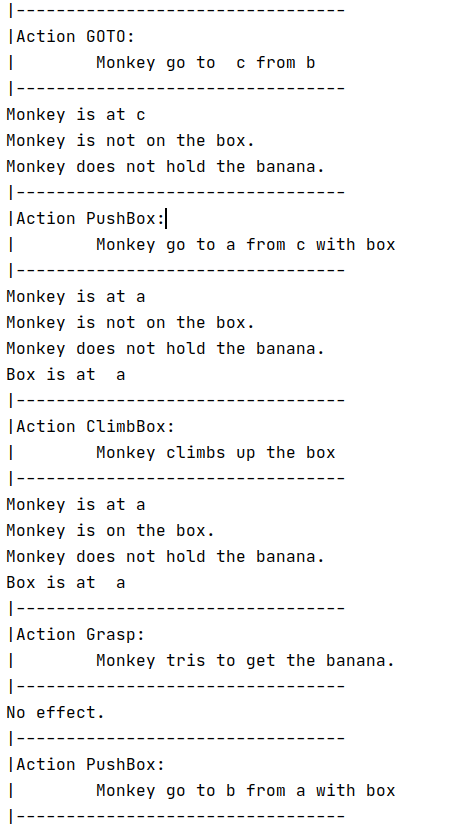
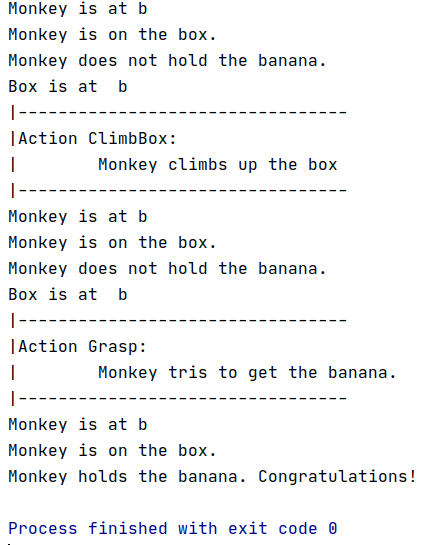
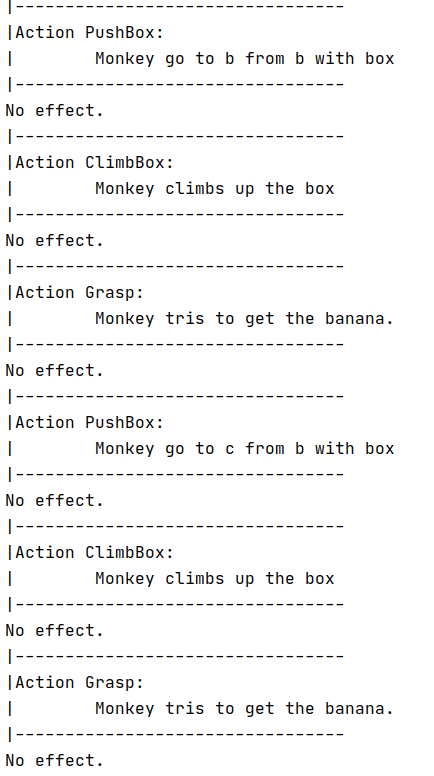
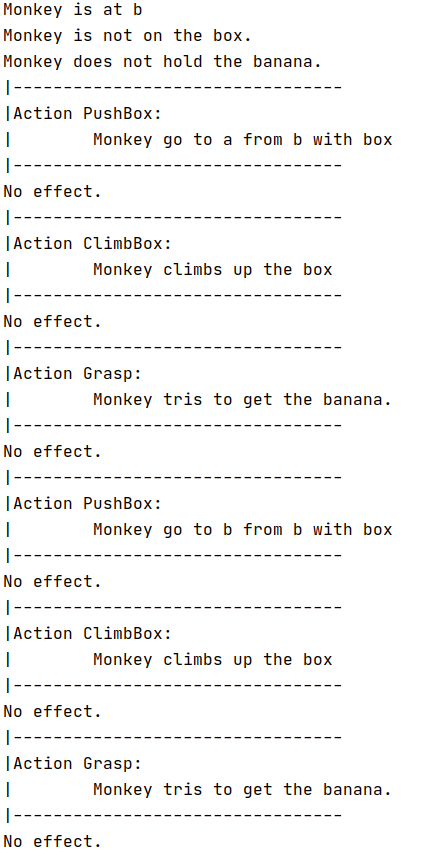
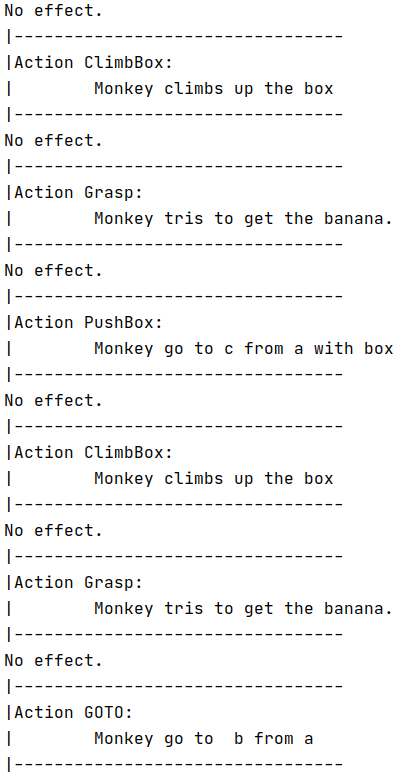
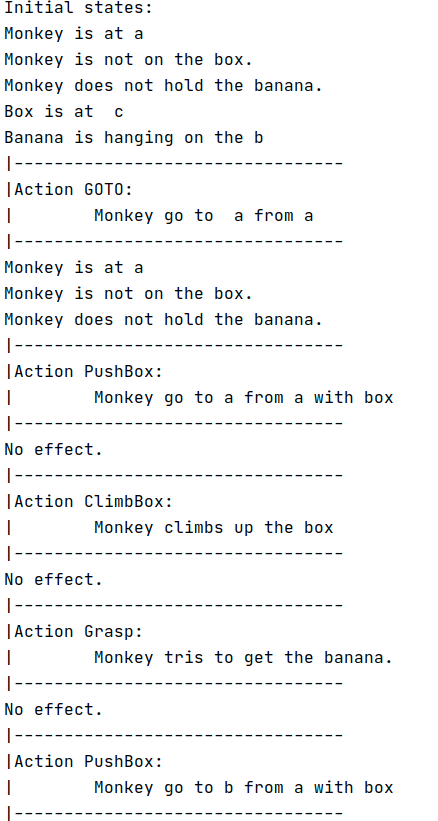
数组

## 实验结果

Monkey holds the banana. Congratulations!

成功搜索到结果

## 系统中间及最终输出结果（要求有屏幕显示）



# 总结及讨论（对该实验的总结以及任何该实验的启发）

该实验以谓词公式表示在编程难度上其实并不高，更为重要的是理解人工智能中如何进行表示知识的过程。实验题目的状态空间很小, 因此执行速度依旧很快; 但如果猴子的位置的可能变得更多, 箱子的位置、香蕉的位置也变得更多，就会出现组合爆炸的情况，效率相对于其他表示方式更低。并且相较于其他表达方式，知识表示能力更差。

# 参考文献

无

# 附录—源代码及其注释（纸质版不需要打印）

*"""  
Define state predicates as following:  
 Site(x,y): x is at y;  
 Hang(w,y): w is hanging on y;  
 On(z): z is standing on the box;  
 Holds(z): bananas are held by z;  
Define 4 operations:  
 Goto(u,v): monkey goes to v from u;  
 PushBox(v,w): monkey goes to w from v with boxes;  
 ClimeBox: monkey climbs up box;  
 Grasp: monkey gets the banana;  
  
Define initial state:  
 S\_0 = Site(Monkey, a)  
 && Hang(banana, b)  
 && Site(Box, c)  
 && !On(Monkey)  
 && !Holds(Monkey)  
  
Define final state:  
 S\_g = Site(Monkey, b)  
 && !Hang(banana, b)  
 && Site(Box, b)  
 && On(Monkey)  
 && Holds(Monkey)  
"""  
  
  
# Define the monkey class*class monkey:  
 def \_\_init\_\_(self, site, On, Holds):  
 self.site = site  
 self.On = On  
 self.Holds = Holds  
 self.report()  
  
 def report(self):  
 print(**"Monkey is at"**, self.site)  
 if self.On:  
 print(**"Monkey is on the box."**)  
 else:  
 print(**"Monkey is not on the box."**)  
 if self.Holds:  
 print(**"Monkey holds the banana. Congratulations! "**)  
 else:  
 print(**"Monkey does not hold the banana."**)  
  
  
*# Define the box class*class box:  
 def \_\_init\_\_(self, site):  
 self.site = site  
 self.report()  
  
 def report(self):  
 print(**"Box is at "**, self.site)  
  
  
*# Define the banana class*class banana:  
 def \_\_init\_\_(self, y):  
 self.hang = y  
 print(**"Banana is hanging on the "** + y)  
  
  
*# Monkey goes to u from v*def Goto(monkey, u, v):  
 print(**"|---------------------------------"**)  
 print(**"|Action GOTO:"**)  
 print(**"| Monkey go to "**, v, **"from"**, u)  
 print(**"|---------------------------------"**)  
 if (~monkey.On) and (monkey.site == u):  
 monkey.site = v  
 monkey.report()  
 else:  
 print(**"No effect."**)  
  
  
*# Monkey goes to v from w with box*def PushBox(monkey, box, v, w):  
 print(**"|---------------------------------"**)  
 print(**"|Action PushBox:"**)  
 print(**"| Monkey go to"**, w, **"from"**, v, **"with box"**)  
 print(**"|---------------------------------"**)  
 if ~monkey.On and monkey.site == v and box.site == v:  
 monkey.site = w  
 box.site = w  
 monkey.report()  
 box.report()  
 else:  
 print(**"No effect."**)  
  
  
*# Monkey climbs up box*def ClimbBox(monkey, box):  
 print(**"|---------------------------------"**)  
 print(**"|Action ClimbBox:"**)  
 print(**"| Monkey climbs up the box"**)  
 print(**"|---------------------------------"**)  
 if monkey.site == box.site and ~monkey.On:  
 monkey.On = True  
 monkey.report()  
 box.report()  
 else:  
 print(**"No effect."**)  
  
  
*# Monkey tris to get banana*def Grasp(monkey, box, banana):  
 print(**"|---------------------------------"**)  
 print(**"|Action Grasp:"**)  
 print(**"| Monkey tris to get the banana."**)  
 print(**"|---------------------------------"**)  
 if monkey.On and box.site == banana.hang:  
 monkey.Holds = True  
 monkey.report()  
 else:  
 print(**"No effect."**)  
  
  
def determine(monkey, box, banana, lace):  
 flag = False  
 for i in place:  
 Goto(monkey, monkey.site, i)  
 tempMonkey = monkey  
 tempBox = box  
 for j in place:  
 tempMonkey1 = monkey  
 tempBox1 = box  
  
 PushBox(monkey, box, monkey.site, j)  
 ClimbBox(monkey, box)  
 Grasp(monkey, box, banana)  
 if monkey.Holds and box.site == monkey.site and monkey.site == banana.hang:  
 flag = True  
 return flag  
  
 monkey = tempMonkey1  
 box = tempBox1  
 monkey = tempMonkey  
 box = tempBox  
 if not flag:  
 print(**"No solution."**)  
  
  
if \_\_name\_\_ == **"\_\_main\_\_"**:  
 print(**"Initial states:"**)  
 Monkey = monkey(**"a"**, False, False)  
 Box = box(**"c"**)  
 Banana = banana(**"b"**)  
 place = [**"a"**, **"b"**, **"c"**]  
 determine(Monkey, Box, Banana)