中山大学计算机学院人工智能本科生实验报告（2022学年春季学期）

课程名称：Artificial Intelligence

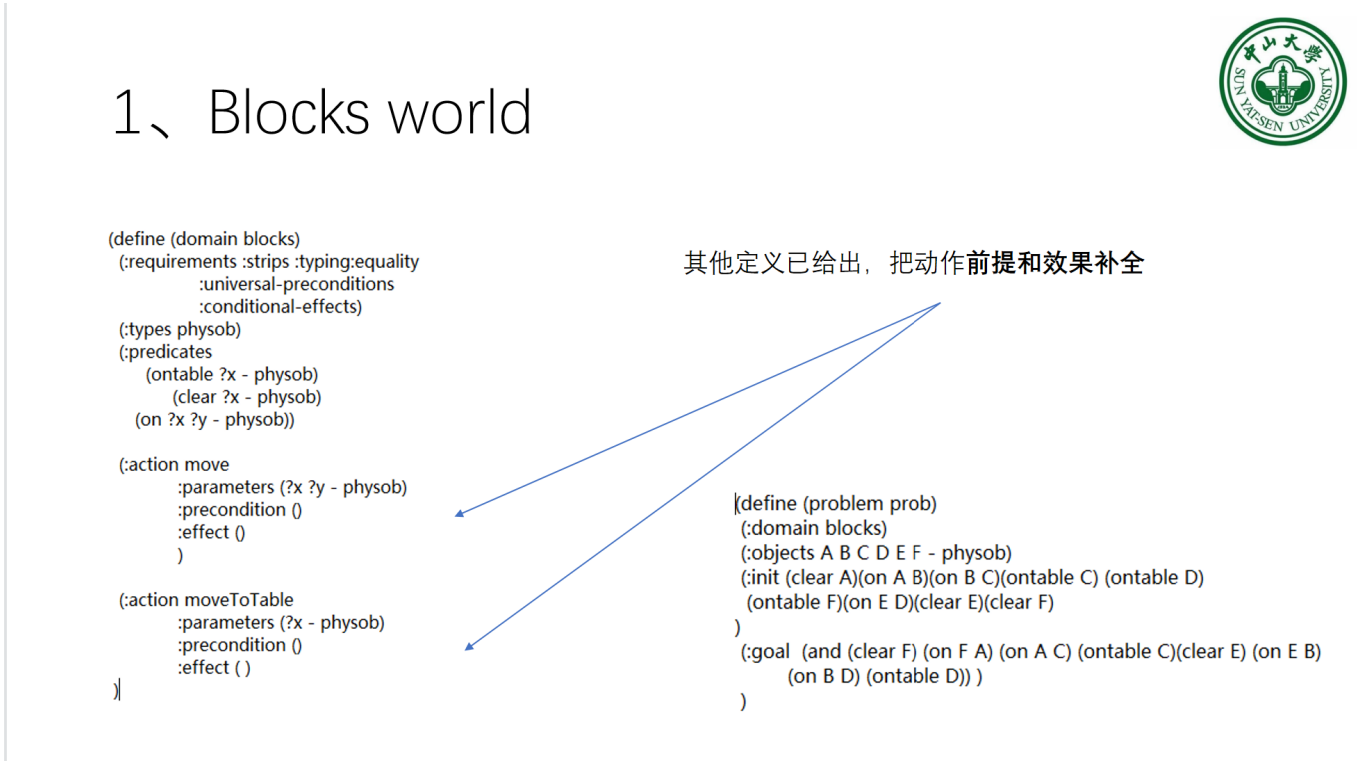
教学班级	专业（方向）	学号	姓名
2班	计算机科学与技术	21307174	刘俊杰

一、实验题目

PDDL实验

二、实验内容

1. 积木世界：



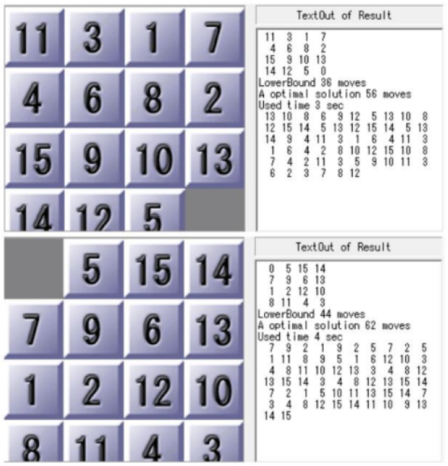
- 补全动作(可能需要forall和when)

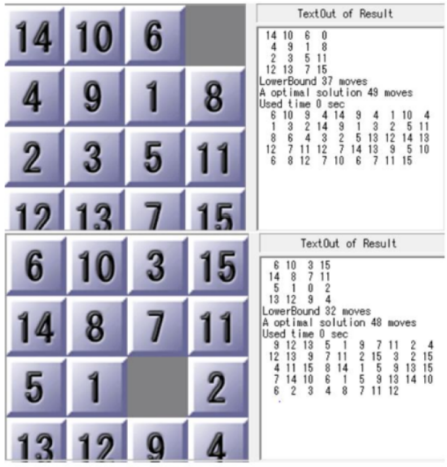
2. 15数码

2、15数码问题

- A*和IDA*中PPT上的四个15数码问题

11 3 1 7
4 6 8 2
15 9 10 13
14 12 5 0





14 10 6 0
4 9 1 8
2 3 5 11
12 13 7 15

0 5 15 14
7 9 6 13
1 2 12 10
8 11 4 3

6 10 3 15
14 8 7 11
5 1 0 2
13 12 9 4

- 不需要考虑解的最优性！！
- 补充完整两个文件，只允许slide这一个动作，其他自行定义

1. 算法原理

(1)规划

规划作为搜索问题

- 给定一个CW-KB表示初始状态 一组STRIPS或ADL操作 一个目标条件 (用公式表示)
- 规划问题是: 确定一个动作序列，当应用于初始的CW-KB时，将产生满足目标的更新的CW-KB。 这就是所谓的经典规划任务。

(2)PDDL

PDDL= Planning Domain Definition Language是对人工智能规划语言进行标准化的一种尝试。PDDL提供了标准化，好处是能使研究更易于重用和比较。 与针对特定论域的系统相比，这要付出一些表达能力的代价

(3)PDDL 相关的语法

PDDL 相关的语法介绍实验PPT已经讲的十分详细了，这里就不再赘述了

2.流程图

无

3.关键代码展示

(1)积木世界

1. 动作move

- 作用：move(x,y)将积木x移到积木y上
- 参数:x、y都是积木

- pre:

```
(1)x上无积木，即clear ?x
(2)上y无积木，即clear ?y
(3)x 和 y不是同一块积木，即(not (= ?x ?y))
说明：前提是(1)(2)(3)的合取，前提条件(3)不是必须的，但是加上可以帮助我们减少一些不必要的操作规划(如将一块积木放在自己上卖弄)
```

```
 :precondition (and(clear ?x)(clear ?y)(not(= ?x ?y)));前提有(都是合取)：(1)x上无积木 (2)y上无积木 (3)x和y不是同一块积木
```

- adds:(1)x在z上
- dels:(1)对所有x在z上的z，clear ?z=true and on ?x ?z=false (2)clear ?y = false (3) ontable ?x=false

```
:effect (and(forall(?z - physob)(when(on ?x ?z )(and(clear ?z)(not(on ?x ?z))))(on ?x ?y)(not(clear ?y))(not (ontable ?x)))
            ;效果(都是合取)：(1)对所有x在z上的z，clear ?z=true and on ?x ?z=false (2)x在z上(3)clear ?y = false (4) ontable ?x=false
```

-动作move完整代码如下

```
(:action move;将x移动到y上
            :parameters (?x ?y - physob);x、y都是积木
            :precondition (and(clear ?x)(clear ?y)(not(= ?x ?y)));前提有(都是合取)：(1)x上无积木 (2)y上无积木 (3)x和y不是同一块积木
            :effect (and(forall(?z - physob)(when(on ?x ?z )(and(clear ?z)(not(on ?x ?z)))))(on ?x ?y)(not(clear ?y))(not (ontable ?x)))
            ;效果(都是合取)：(1)对所有x在z上的z，clear ?z=true and on ?x ?z=false (2)x在z上(3)clear ?y = flase (4) ontable ?x=false
            )
```

2. 动作moveToTable

- 作用：moveToTable(x)将x放在桌上
- 参数:x都是积木
- pre:(都是合取)：(1)ontable ?x=false (2)x上无积木
- del:(1)对所有x在y上的y on ?x ?y =false
- add:(1)ontable ?x= true
- moveToTable完整代码如下

```
  (:action moveToTable;将x移到桌子上
            :parameters (?x - physob);x是积木
            :precondition (and(not (ontable ?x))(clear ?x));前提(都是合取)：(1)ontable ?x=false (2)x上无积木
            :effect (and(forall(?y - physob)(when(on ?x  ?y )(and(clear ?y)(not (on ?x ?y))))))(ontable ?x))
            ;效果(都是合取)：(1)对所有x在y上的y on ?x ?y =false (2)ontable ?x= true
  )
```

(2)十五数码问题

1.谓词和类的定义：

我们分为两类：数字num和位置loc 定义的谓词如下：

- (neighbor ?x ?y - loc) x、y相邻
- (at ?x - num ?y - loc) 数字x在位置y上
- (zero_at ?x - loc) 数字0在位置x上

```
(define (domain puzzle)
  (:requirements :strips :equality
                 :typing:universal-preconditions
                 :conditional-effects)
  (:types num loc)
  (:predicates  (neighbor ?x ?y - loc);x、y相邻
                (at ?x - num ?y - loc);数字x在位置y上
                (zero_at ?x - loc);数字0在位置x上

  )
```

2.slide动作

- 作用：slide(n,x,y)将数字n从位置x移动到位置y上
- 参数：n为数字，x、y为位置
- pre:(都是合取):(1)0在位置y上 (2)n在位置x上 (3)位置x、y相邻
- add: (1)0在位置x上 (2)数字n在位置y上
- del: (1)0不在位置y上 (2)数字n不在位置x上

```
(:action slide;滑动数字n
             :parameters (?n - num ?x ?y -loc);n为数字，x、y为位置
             :precondition (and(zero_at ?y)(at ?n ?x)(neighbor ?x ?y));前提(都是合取):(1)0在位
置y上 (2)n在位置x上 (3)位置x、y相邻
             :effect (and(not(zero_at ?y))(zero_at ?x)(at ?n ?y)(not (at ?n ?x)));效果(都是合
取):(1)0不在位置y上 (2)0在位置x上 (3)数字n在位置y上 (4)数字n不在位置x上
)
)
```

3.初始状态定义和目标状态定义

```
(define (problem prob)
 (:domain puzzle)
 (:objects num_1 num_2 num_3 num_4 num_5 num_6 num_7 num_8 num_9 num_10 num_11 num_12 num_13
num_14 num_15 - num;1-15 数字
           pos_1 pos_2 pos_3 pos_4 pos_5 pos_6 pos_7 pos_8 pos_9 pos_10 pos_11 pos_12 pos_13
pos_14 pos_15 pos_16 - loc ;16个位置
 )
 (:init (neighbor pos_1 pos_2)(neighbor pos_1 pos_5);位置的相邻情况
        (neighbor pos_2 pos_1)(neighbor pos_2 pos_3)(neighbor pos_2 pos_6)
        (neighbor pos_3 pos_2)(neighbor pos_3 pos_4)(neighbor pos_3 pos_7)
        (neighbor pos_4 pos_8)(neighbor pos_4 pos_3)
```

```
        (neighbor pos_5 pos_1)(neighbor pos_5 pos_6)(neighbor pos_5 pos_9)
        (neighbor pos_6 pos_5)(neighbor pos_6 pos_7)(neighbor pos_6 pos_2)(neighbor pos_6
pos_10)
        (neighbor pos_7 pos_6)(neighbor pos_7 pos_8)(neighbor pos_7 pos_3)(neighbor pos_7
pos_11)
        (neighbor pos_8 pos_7)(neighbor pos_8 pos_4)(neighbor pos_8 pos_12)
        (neighbor pos_9 pos_5)(neighbor pos_9 pos_10)(neighbor pos_9 pos_13)
        (neighbor pos_10 pos_9)(neighbor pos_10 pos_11)(neighbor pos_10 pos_14)(neighbor
pos_10 pos_6)
        (neighbor pos_11 pos_10)(neighbor pos_11 pos_12)(neighbor pos_11 pos_7)(neighbor
pos_11 pos_15)
        (neighbor pos_12 pos_11)(neighbor pos_12 pos_8)(neighbor pos_12 pos_16)
        (neighbor pos_13 pos_9)(neighbor pos_13 pos_14)
        (neighbor pos_14 pos_13)(neighbor pos_14 pos_15)(neighbor pos_14 pos_10)
        (neighbor pos_15 pos_14)(neighbor pos_15 pos_16)(neighbor pos_15 pos_11)
        (neighbor pos_16 pos_15)(neighbor pos_16 pos_12)
        ;15个数字和0在对应的起始位置上
        (at num_11 pos_1)
        (at num_3 pos_2)
        (at num_1 pos_3)
        (at num_7 pos_4)
        (at num_4 pos_5)
        (at num_6 pos_6)
        (at num_8 pos_7)
        (at num_2 pos_8)
        (at num_15 pos_9)
        (at num_9 pos_10)
        (at num_10 pos_11)
        (at num_13 pos_12)
        (at num_14 pos_13)
        (at num_12 pos_14)
        (at num_5 pos_15)
        (zero_at  pos_16)

; 11 3 1 7
;4 6 8 2
;15 9 10 13
;14 12 5 0


  )
 (:goal (and(at num_1 pos_1)(at num_2 pos_2)(at num_3 pos_3);目标
        (at num_4 pos_4)
        (at num_5 pos_5)
        (at num_6 pos_6)
        (at num_7 pos_7)
        (at num_8 pos_8)
        (at num_9 pos_9)
        (at num_10 pos_10)
        (at num_11 pos_11)
        (at num_12 pos_12)
        (at num_13 pos_13)
        (at num_14 pos_14)
        (at num_15 pos_15)
        (zero_at  pos_16)))
)
```

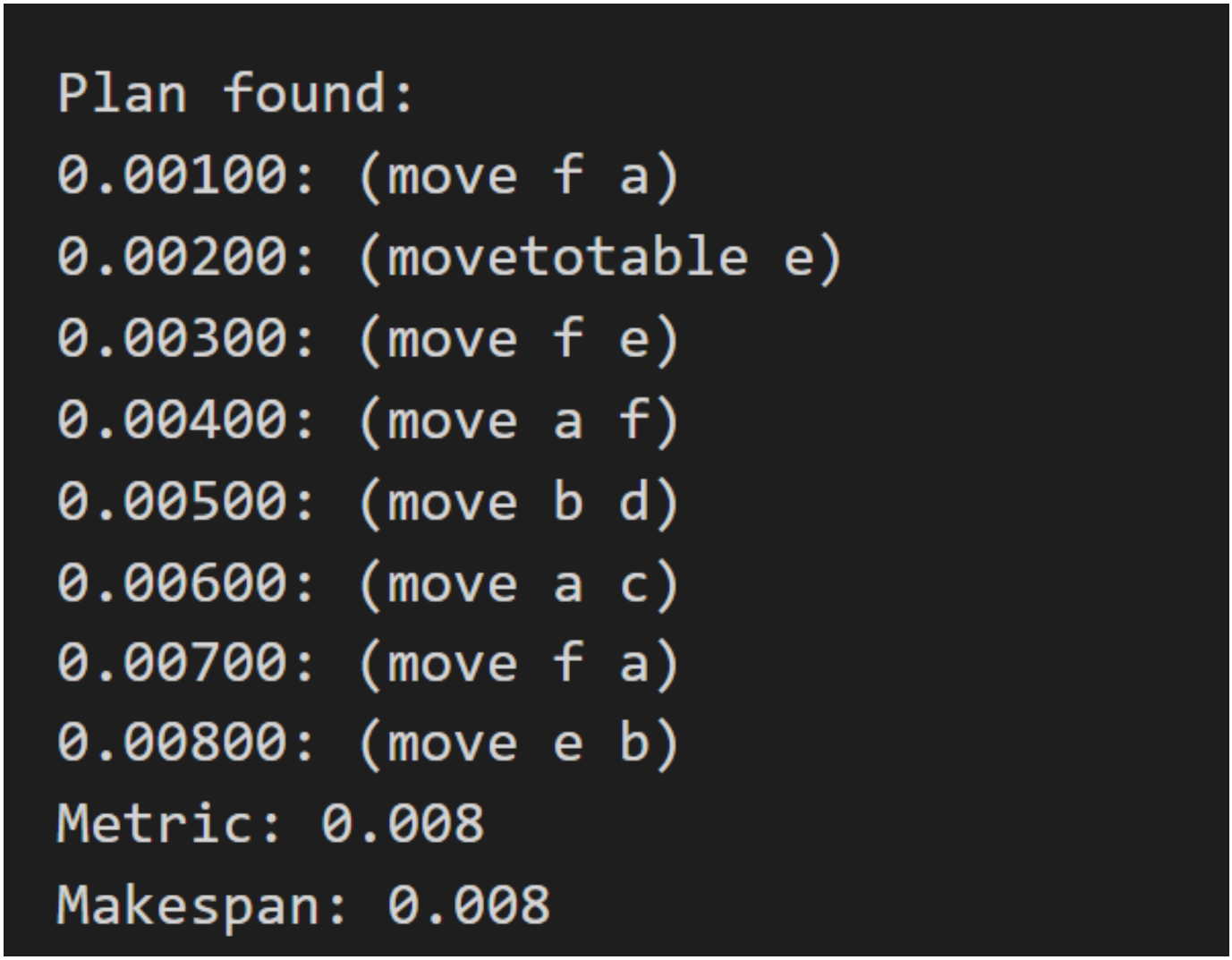
4.创新点&优化

无

三、实验结果及分析

1. 实验结果展示示例

(1)积木世界



经过验证，得到输出的结果是正确的

(2)十五数码问题

十五数码问题的结果较长，已经将结果放在了result文件夹中 同时为了验证是否正确，我编写了如下的代码验证

```
# coding=gb2312
matrix=[[0]*4 for i in range(4)]
index=0#记录数字0在数码表的下标
def initialize(matrix,f):#初始数码表赋值
     for i in range (4):
        tmp=((f.readline()).strip()).split()
        for j in range(len(tmp)):
            matrix[i][j]=int(tmp[j])
            if matrix[i][j]==0:
                global index
                index=i*4+j
def swap(l,n):#交换函数
```

```
    for i in range(len(l)):
        for j in range(len(l[i])):
            if l[i][j]==n:
                global index
                l[index//4][index%4]=l[i][j]
                l[i][j]=0
                if abs(index-i*4-j)!=1 and abs(index-i*4-j)!=4:
                    print("Wrong")#若移动不符合规则则打印错误
                index=i*4+j
                return
def print_matrix(l):#打印输出数码表
    for i in range(len(l)):
        for j in range(len(l[i])):
            print(l[i][j],end=' ')
        print()
f = open(r"E6\puzzle_input\puzzle4_input.txt",'r')#读取初始状态
initialize(matrix,f)
f = open(r"E6\result\result4.txt",'r')#读取移动的步骤
for line in f:#进行数码表的移动
    tmp=line.strip().split(" ")
    swap(matrix,int(tmp[0]))
print_matrix(matrix)#打印输出移动后的数码表
```

通过验证得出输出的结果都是正确的

2. 评测指标展示及分析

(1)积木世界

积木世界的所做的动作不一定是最优解

(2)十五数码问题

A * 和 IDA * 求解结果

实例	答案步数	未优化 A* 算法运行时间	优化后 A* 算法运行时间	A* 搜索节点个数	优化前 IDA* 算法运行时间	优化后 IDA* 算法运行时间
PPT1	56	内存超出限制	5007.7800552845s	28838241	运行时间过长，未运行出结果	20110.355590343475s
PPT2	49	132.3359088897705s	43.0419979095459s	435953	507.0727288722992s	245.15968203544617s
PPT3	62	内存超出限制	8537.527533054352s	4118629	运行时间过长，未运行出结果	42842.5228061676s
PPT4	48	249.5972819328308s	255.44039845466614s	2494136	2672.5074956417084s	465.87745571136475s

pddl搜索结果

序号	步数	搜索时间
第1个	166	4.488s
第2个	117	2.792s
第3个	242	4.023secs

序号	步数	搜索时间
第4个	218	1.97s

分析

通过对十五数目的A * 和 IDA * 求解结果 和 pddl搜索结果 对比 我们可以得出：

1. A * 和 IDA * 求解的结果是最优解，所耗费的时间很长(有的甚至要几千或几万秒 也就是花费了几个小时取求解)并且空间也很大(为优化前56步和62步的超出内存)

2. pddl求解是非最优解，于是它所花费的时间很快，及几秒就能找到最好的结果

3. 如果不考虑最优解使用A * 和 IDA * 算法求解，也可以达到很快的效果，这就需要选择适合的启发式函数(与我们班一样做 IDA* 和 A * 的其他班的一位同学，他们班不要求最优解，那位同学自己写了一个启发式 可以用 IDA * 和 A * 在大概零点几秒到几秒找出56步和62步的解)

四、思考题

1.目标状态如何表示?

目标状态可以表示为对应的数字在对应的位置上

2.是否一定要写全十六个位置的状态，15个可以吗，为什么?

可以用15个表示，可以对剩下的位置做特殊的判断，但是会使得代码变得更加复杂和麻烦

五、参考资料

pddl实验PPT