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
计科三班	计算机科学与技术	21307185	张礼贤

一、实验题目

补全pddl文件，完成积木世界和十五数码问题

二、实验内容

1、算法原理

积木世界问题

1. 论域的定义：领域名称为“blocks”，该领域需要满足以下要求：
    - :strips 表示只考虑基本条件和动作
    - :typing 表示可以定义类型
    - :equality 表示可以使用等于符号(=)
    - :universal-preconditions 表示所有前提必须都成立
    - :conditional-effects 表示效果可以使用条件
    - :negative-preconditions 表示允许否定前提 (后续写代码时加上去的，因为需要用到负文字的前提条件)
2. 类型声明：声明了一个类型 physob 表示积木的物理对象，涵盖了积木和桌子对象
3. 谓词声明：
    - ontalbe 表示物体 x 放在桌子上

- clear 表示物体 x 的上面没有其他物体
- on 表示物体 x 放在物体 y 的上面

4. 动作声明：
   - move
     - 含义：表示将一个物体 x 移到另一个物体 y 上面
     - 参数：?x 和 ?y，都是physob类型，表示物体 x 和物体 y
     - 前提：(clear ?x) 表示物体 x 的上面没有其他物体，(clear ?y) 表示物体 y 的上面没有其他物体，(not(= ?x ?y)) 表示物体 x 和物体 y 不相等
     - 效果：(forall (?z - physob)(when (on ?x ?z) (and (not (on ?x ?z)) (clear ?z)))) 表示对于所有在物体 z 如果有物体 x 在 z 上面，则清除 x 在 z 上面的状态并且更新 z 的状态为 clear； (on ?x ?y) 表示物体 x 在物体 y 上面。 (not (clear ?y)) 表示物体 y 的上面有物体了，从而实现状态的转换
   - movetotable
     - 含义：表示将一个物体移动到桌面上
     - 参数：?x，类型为physob，表示物理类型的变量
     - 前提：(clear ?x) 表示物体 x 的上面没有其他物体，(not (ontable ?x)) 表示物体 x 不在桌子上
     - 效果：(forall (?y - physob) (when (on ?x ?y) (and (not (on ?x ?y)) (clear ?y))))表示对于所有在物体 y 如果有物体 x 在 y 上面，则清除 x 并且更新 y 的状态为 clear；(ontable ?x) 表示物体 x 放在桌子上。
5. 问题的定义：
   - 声明了是在blocks论域下进行问题的求解的
   - 声明了一个物体类型 physob，包含了 A、B、C、D、E、F 六个具体的物体
   - 初始化部分表示初始状态，其中包括了 A 在 B 上面，B 在 C 上面，C 放在桌面上，D 放在桌面上，F 放在桌面上，E 在 D 上面，E 和 F 上面没有其他物体
   - 目标状态是 F 在 A 上面，A 在 C 上面，C 放在桌面上，E 在 B 上面，B 在 D 上面，D 放在桌面上，且 F、E、A、B、C、D 的上面没有其他物体
   - 通过初始化问题和定义问题的目标，通过一系列的规划操作达到问题的求解的目的

十五数码问题

1. 论域的定义：论域名称为puzzle，需要满足以下条件：
   - :strips 表示只考虑基本条件和动作
   - :typing 表示可以定义类型
   - :equality 表示可以使用等于符号(=)
2. 类型声明：声明了两个类型：

- num类型：表示数字
- loc类型：表示位置

3. 谓词定义：
   - at ?n - num ?l - loc 表示数字 n 在位置 l 上。
   - adjacent ?x ?y - loc 表示位置 x 和位置 y 相邻。
   - empty ?x - loc 表示位置 x 是空的，用于判断该位置是否可以进行交换
4. 动作定义：
   - slide
     - 含义：表示将滑块移动到空的位置（相当于与空位置滑块进行交换）
     - 参数：?t 表示数字 t，?x 和 ?y 表示位置 x 和位置 y。
     - 前提：(and (at ?t ?x) (empty ?y) (adjacent ?x ?y)) 表示数字 t 在位置 x 上，位置 y 是空的，位置 x 和位置 y 相邻。
     - 效果：(and (at ?t ?y) (not (empty ?y)) (empty ?x) (not(at ?t ?x))) 表示数字 t 在位置 y 上，位置 x 变为空，位置 y 不再为空，数字 t 不再在位置 x 上。
5. 问题的定义：
   - 声明了是在puzzle下论域进行问题的求解
   - 定义了P1 ~ P16 十六个位置，类型为loc；定义了N1 ~ N16 十六个数字，类型为num
   - 初始化问题：
     - 定义每个位置的相邻状态（这里注意是要对称，P1与P2相邻，P2也与P1相邻）
     - 根据问题的给定将每个数字填充到相应的位置，如果位置为0则表示为空，置为empty
   - 设置目标状态：位置相邻的情况与问题的初始化无异，但是目标变成了每个数字对应每一个位置，最后一个数位为空

2、关键代码及注释

积木世界问题

- 积木世界domain声明程序

```
(define (domain blocks)
  (:requirements :strips :typing :equality
                 :universal-preconditions
                 :conditional-effects :negative-preconditions)
  (:types physob)
  (:predicates
    (ontable ?x - physob)
    (clear ?x - physob)
    (on ?x ?y - physob))

  (:action move
    :parameters (?x ?y - physob)
    :precondition (and (clear ?x) (clear ?y) (not(= ?x ?y)) )
    :effect (and (forall (?z - physob)(when (on ?x ?z) (and (not (on ?x ?
z)) (clear ?z)))) (on ?x ?y) (not (clear ?y)))
  )

  (:action moveToTable
    :parameters (?x  - physob)
    :precondition (and   (clear ?x) (not (ontable ?x) ))
    :effect (and (forall (?y - physob)(when (on ?x ?y) ( and (not (on ?x
?y)) (clear ?y)))) (ontable ?x) )
  )
)
```

- 积木世界problem定义程序

```
(define (problem prob)
 (:domain blocks)
 (:objects A B C D E F - physob)
 (:init (clear A)(on A B)(on B C)(ontable C) (ontable D)
  (ontable F)(on E D)(clear E)(clear F)
)
 (:goal  (and (clear F) (on F A) (on A C) (ontable C)(clear E) (on E B)
         (on B D) (ontable D)) )
 )
```

十五数码问题

- 十五数码domain声明程序

```
(define (domain puzzle)
  (:requirements :strips :equality:typing)
  (:types num loc)
  (:predicates
    (at ?n - num ?l - loc)
    (adjacent ?x ?y - loc)
    (empty ?x - loc))

(:action slide
      :parameters (?t - num  ?x ?y - loc)
      :precondition (and (at ?t ?x) (empty ?y) (adjacent ?x ?y))
      :effect (and (at ?t ?y) (not (empty ?y)) (empty ?x) (not(at ?t ?x))
)
 )
)
```

- 十五数码problem定义程序

```
(define (problem prob)
 (:domain puzzle)
 (:objects P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15 P16 - loc
     N1 N2 N3 N4 N5 N6 N7 N8 N9 N10 N11 N12 N13 N14 N15 N16 - num
 )
 (:init (adjacent P1 P2) (adjacent P2 P3) (adjacent P3 P4)
        (adjacent P5 P6) (adjacent P6 P7) (adjacent P7 P8)
        (adjacent P9 P10) (adjacent P10 P11) (adjacent P11 P12)
        (adjacent P13 P14) (adjacent P14 P15) (adjacent P15 P16)
        (adjacent P1 P5) (adjacent P5 P9) (adjacent P9 P13)
        (adjacent P2 P6) (adjacent P6 P10) (adjacent P10 P14)
        (adjacent P3 P7) (adjacent P7 P11) (adjacent P11 P15)
        (adjacent P4 P8) (adjacent P8 P12) (adjacent P12 P16)

        (adjacent P2 P1) (adjacent P3 P2) (adjacent P4 P3)
        (adjacent P6 P5) (adjacent P7 P6) (adjacent P8 P7)
        (adjacent P10 P9) (adjacent P11 P10) (adjacent P12 P11)
        (adjacent P14 P13) (adjacent P15 P14) (adjacent P16 P15)
        (adjacent P5 P1) (adjacent P9 P5) (adjacent P13 P9)
        (adjacent P6 P2) (adjacent P10 P6) (adjacent P14 P10)
        (adjacent P7 P3) (adjacent P11 P7) (adjacent P15 P11)
        (adjacent P8 P4) (adjacent P12 P8) (adjacent P16 P12)

        (at N11 P1) (at N3 P2) (at N1 P3) (at N7 P4)
        (at N4 P5) (at N6 P6) (at N8 P7) (at N2 P8)
        (at N15 P9) (at N9 P10) (at N10 P11) (at N13 P12)
        (at N14 P13) (at N12 P14) (at N5 P15) (empty P16)
)

 (:goal  (and (adjacent P1 P2) (adjacent P2 P3) (adjacent P3 P4)
        (adjacent P5 P6) (adjacent P6 P7) (adjacent P7 P8)
        (adjacent P9 P10) (adjacent P10 P11) (adjacent P11 P12)
        (adjacent P13 P14) (adjacent P14 P15) (adjacent P15 P16)
        (adjacent P1 P5) (adjacent P5 P9) (adjacent P9 P13)
        (adjacent P2 P6) (adjacent P6 P10) (adjacent P10 P14)
        (adjacent P3 P7) (adjacent P7 P11) (adjacent P11 P15)
        (adjacent P4 P8) (adjacent P8 P12) (adjacent P12 P16)

        (adjacent P2 P1) (adjacent P3 P2) (adjacent P4 P3)
        (adjacent P6 P5) (adjacent P7 P6) (adjacent P8 P7)
        (adjacent P10 P9) (adjacent P11 P10) (adjacent P12 P11)
        (adjacent P14 P13) (adjacent P15 P14) (adjacent P16 P15)
```

```
        (adjacent P5 P1) (adjacent P9 P5) (adjacent P13 P9)
        (adjacent P6 P2) (adjacent P10 P6) (adjacent P14 P10)
        (adjacent P7 P3) (adjacent P11 P7) (adjacent P15 P11)
        (adjacent P8 P4) (adjacent P12 P8) (adjacent P16 P12)

        (at N1 P1) (at N2 P2) (at N3 P3) (at N4 P4)
        (at N5 P5) (at N6 P6) (at N7 P7) (at N8 P8)
        (at N9 P9) (at N10 P10) (at N11 P11) (at N12 P12)
        (at N13 P13) (at N14 P14) (at N15 P15) (empty P16))
        )
)
```

三、实验结果与分析

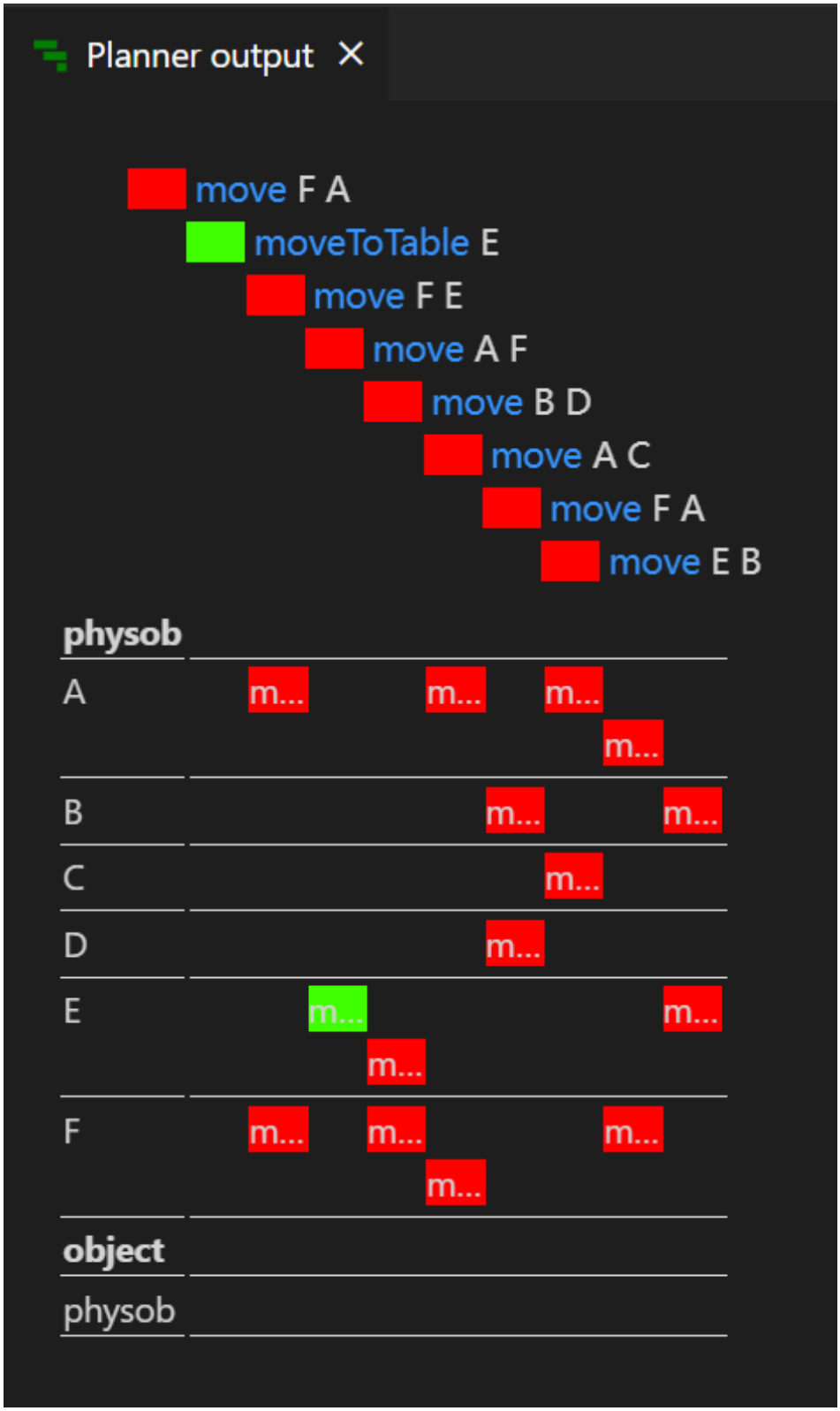
实验结果展示实例

积木世界blocks

- 利用规划器进行求解，运行后的移动步数如下所示：

```
Plan found:
0.00100: (move f a)
0.00200: (movetotable e)
0.00300: (move f e)
0.00400: (move a f)
0.00500: (move b d)
0.00600: (move a c)
0.00700: (move f a)
0.00800: (move e b)
Metric: 0.008
Makespan: 0.008
States evaluated: undefined
Planner found 1 plan(s) in 6.482secs.
```

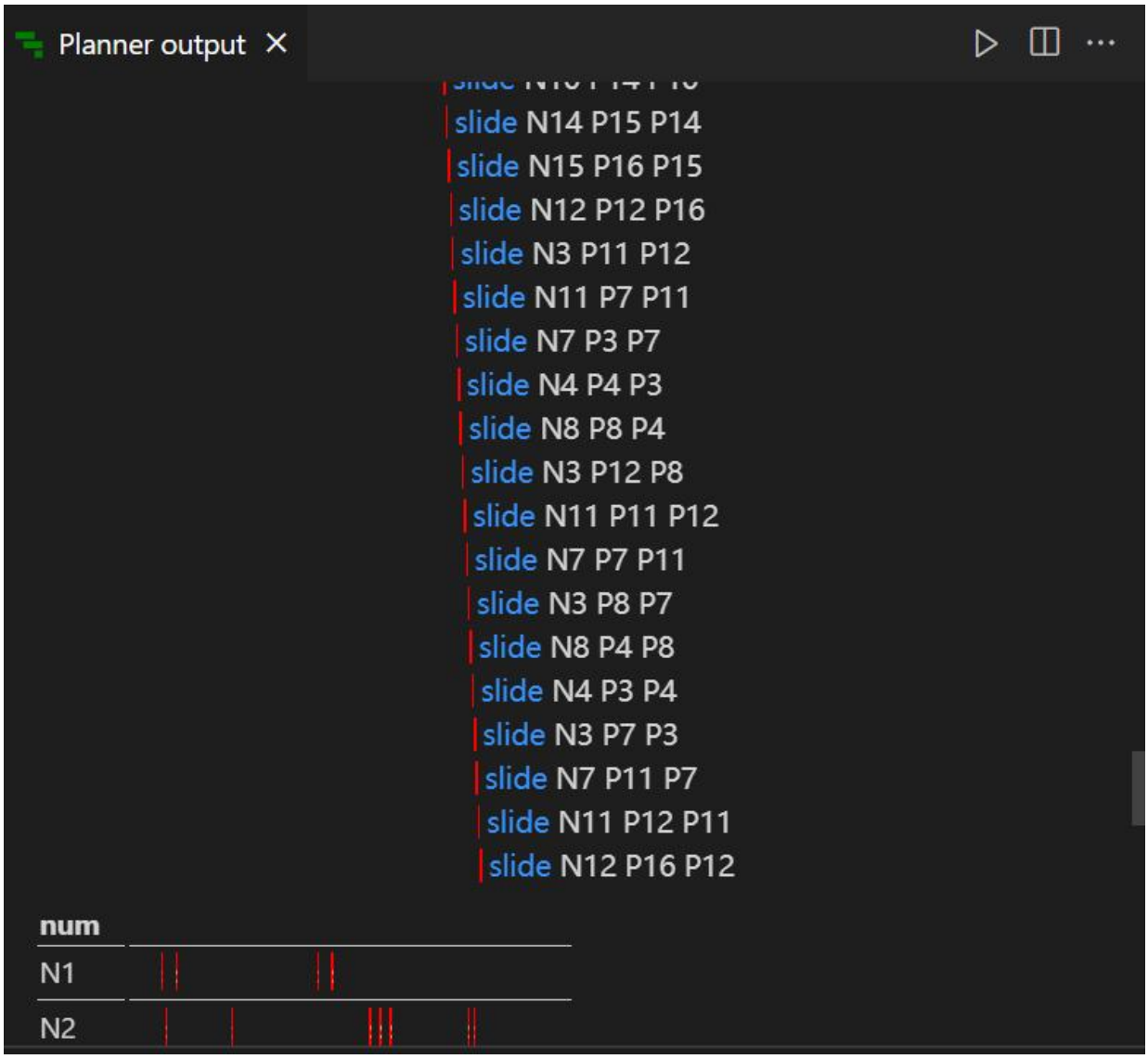
- 下图为规划器给出的图示：



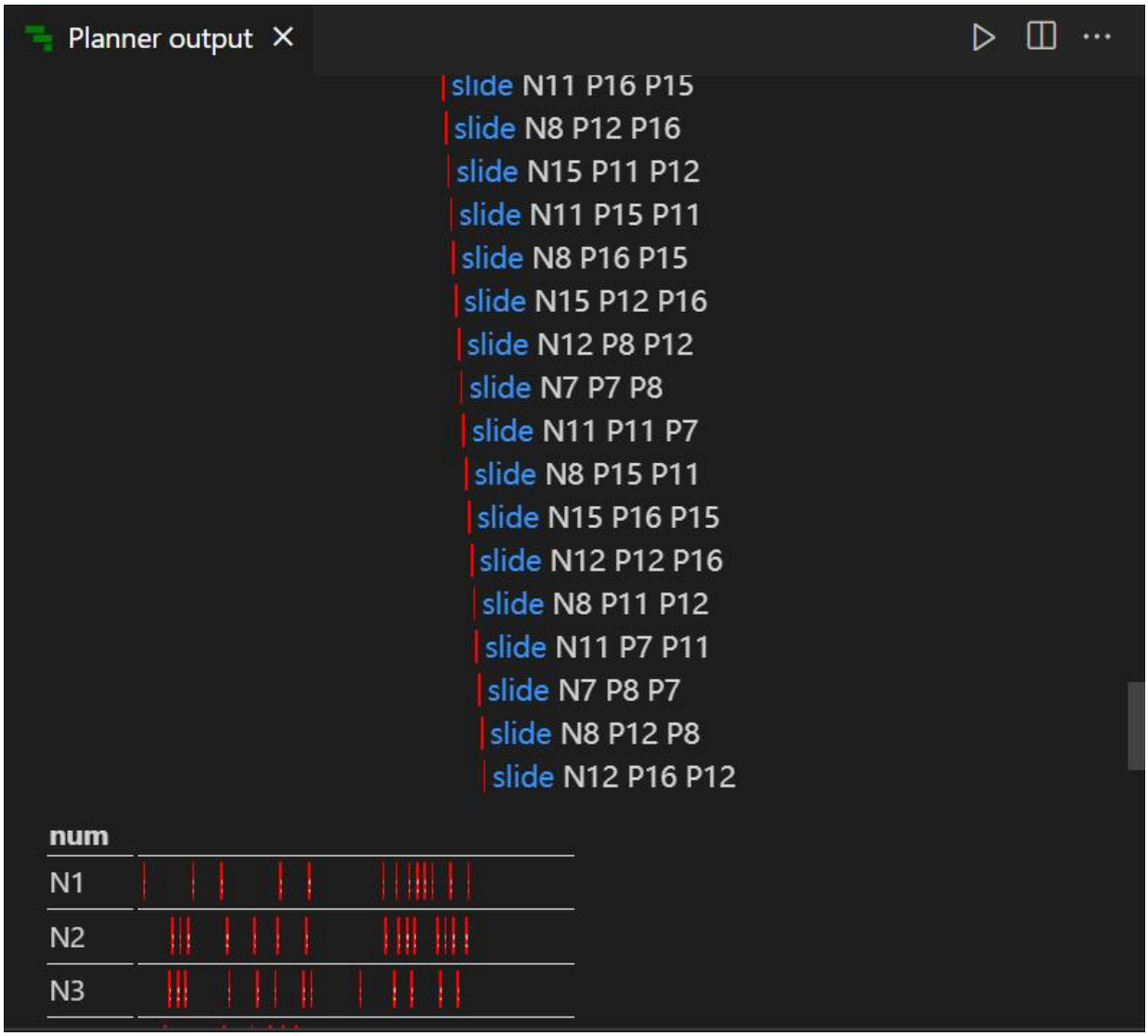
十五数码puzzle

- puzzle1 ~ 4的结果如下，由于移动的步数较多，所以只截取了最后的移动步数

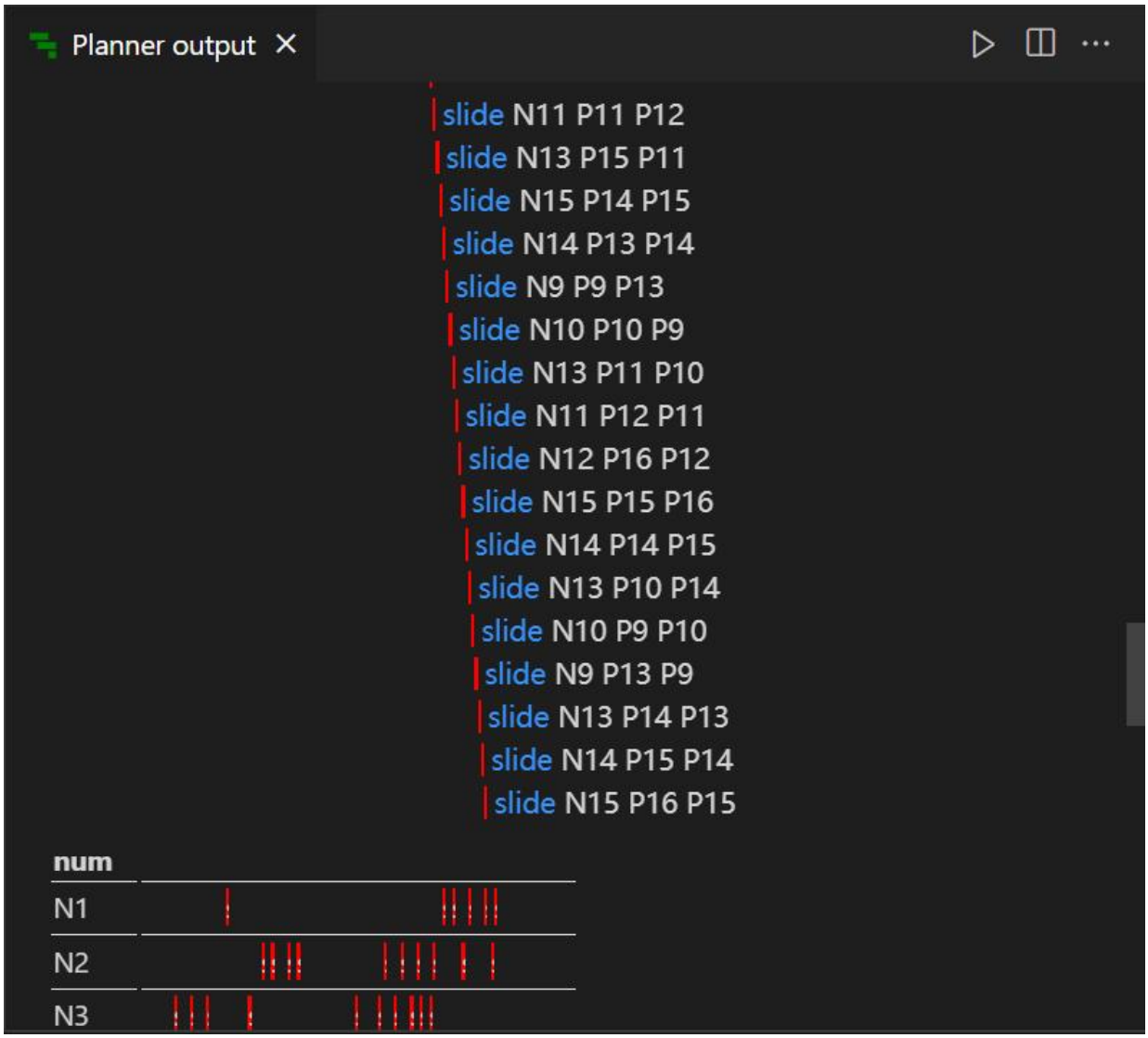
- puzzle1:


Planner output
slide N14 P15 P14
slide N15 P16 P15
slide N12 P12 P16
slide N3 P11 P12
slide N11 P7 P11
slide N7 P3 P7
slide N4 P4 P3
slide N8 P8 P4
slide N3 P12 P8
slide N11 P11 P12
slide N7 P7 P11
slide N3 P8 P7
slide N8 P4 P8
slide N4 P3 P4
slide N3 P7 P3
slide N7 P11 P7
slide N11 P12 P11
slide N12 P16 P12
num
N1
N2

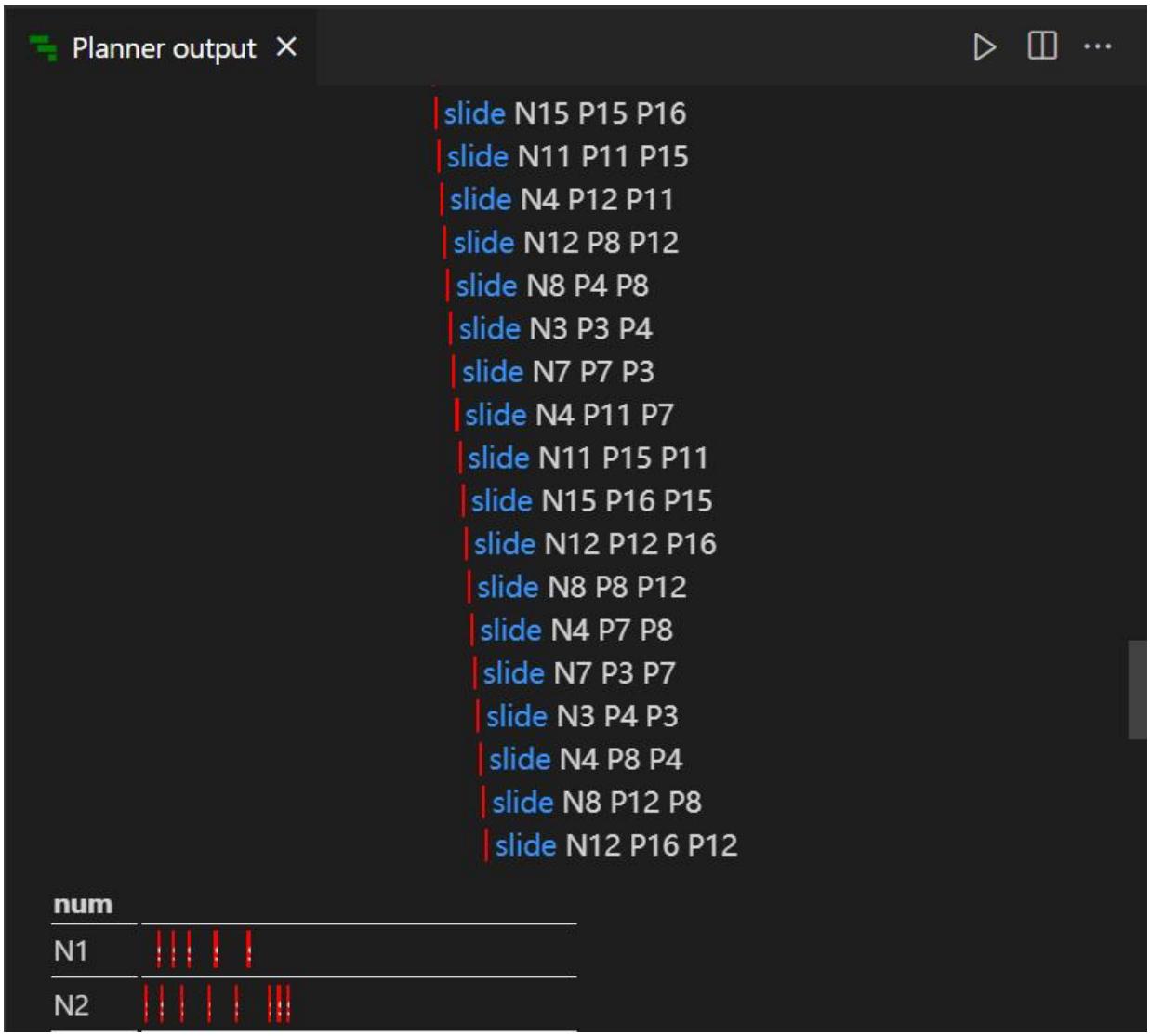
- puzzle2:


Planner output
slide N11 P16 P15
slide N8 P12 P16
slide N15 P11 P12
slide N11 P15 P11
slide N8 P16 P15
slide N15 P12 P16
slide N12 P8 P12
slide N7 P7 P8
slide N11 P11 P7
slide N8 P15 P11
slide N15 P16 P15
slide N12 P12 P16
slide N8 P11 P12
slide N11 P7 P11
slide N7 P8 P7
slide N8 P12 P8
slide N12 P16 P12
num
N1
N2
N3

- puzzle3:


Planner output
slide N11 P11 P12
slide N13 P15 P11
slide N15 P14 P15
slide N14 P13 P14
slide N9 P9 P13
slide N10 P10 P9
slide N13 P11 P10
slide N11 P12 P11
slide N12 P16 P12
slide N15 P15 P16
slide N14 P14 P15
slide N13 P10 P14
slide N10 P9 P10
slide N9 P13 P9
slide N13 P14 P13
slide N14 P15 P14
slide N15 P16 P15
num
N1
N2
N3

- puzzle4:



- 由于求解过程中的步数较多而导致无法看出其正确性，但是可以采用程序验证的方法进行验证。可以设置循环，不断地交换规划器给出的可交换位置的数码，直到所有步数用尽，比较最终状态是否与题目中的要求相同，其验证程序如下：

```
def swap(matrix,x,y,m,n):
    temp = matrix[x][y]
    matrix[x][y] = matrix[m][n]
    matrix[m][n] = temp

f = open('puzzle4_test.txt')
store = []
for line in f:
    s = line.strip().split(' ')
    store.append(s[1:])
#matrix = [[11, 3, 1, 7],[4 ,6 ,8 ,2],[15 ,9 ,10, 13],[14, 12, 5, 0]]
#第一个puzzle的初始状态
#matrix = [[14 ,10 ,6 ,0],[4 ,9 ,1 ,8],[2 ,3 ,5 ,11],[12 ,13 ,7 ,15]]
#第二个puzzle的初始状态
#matrix = [[0 ,5 ,15 ,14],[7, 9, 6 ,13],[1 ,2 ,12 ,10],[8 ,11 ,4 ,3]]
#第三个puzzle的初始状态
matrix = [[6 ,10 ,3, 15],[14, 8, 7 ,11],[5 ,1 ,0 ,2],[13, 12, 9, 4]]
#第四个puzzle的初始状态
for s in store:
    x = (int(s[0])-1) // 4
    y = (int(s[0])-1) % 4
    m = (int(s[1])-1) // 4
    n = (int(s[1])-1) % 4
    swap(matrix,x,y,m,n)
print(matrix)
```

- 其 1 ~ 4 的程序运行结果如下所示：

```
[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 0]]
PS D:\桌面\py_work> python -u "d:\桌面\puzzle\puzzle_test.py"
[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 0]]
PS D:\桌面\py_work> python -u "d:\桌面\puzzle\puzzle_test.py"
[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 0]]
PS D:\桌面\py_work> python -u "d:\桌面\puzzle\puzzle_test.py"
[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12], [13, 14, 15, 0]]
PS D:\桌面\py_work>
```

实验结果分析

- 对于规划器给出的积木世界的步骤，我们可以看到有很多没有必要且重复的步骤，因此其步骤并不具备最优性

- 同理，十五数码问题的步骤平均高达两三百步，存在很多重复且不必要的动作，不具备最优性