Discrete Mathematics(1)

王浩算法 Lab

[wang-zx23@mails.tsinghua.edu.cn](mailto:wang-zx23@mails.tsinghua.edu.cn) November 2, 2024

# 运行结果

项目组织如下，运行结果如图。

* proj1
  + main.py # 程序入口
  + input\_proc.py # 输入处理函数
  + binary\_tree.py # 构建表达式二叉树
  + state.py # 存储逻辑连接词
  + infer.py # 王浩算法的实现（10条推理规则+1条公理）
  + test.py
  + test.txt
  + result.txt
  + result.pdf # 将txt文件前30条推理结果复制到typora后导出的pdf

实现了以下功能点：

1. 将以 markdown 格式输入的中缀表达式转换为后缀表达式。
2. 从得到的后缀表达式构建表达式二叉树。
3. 初始化存储表达式树的前后件列表，根据推理规则进行推理。
4. 输出推理步骤和结果。

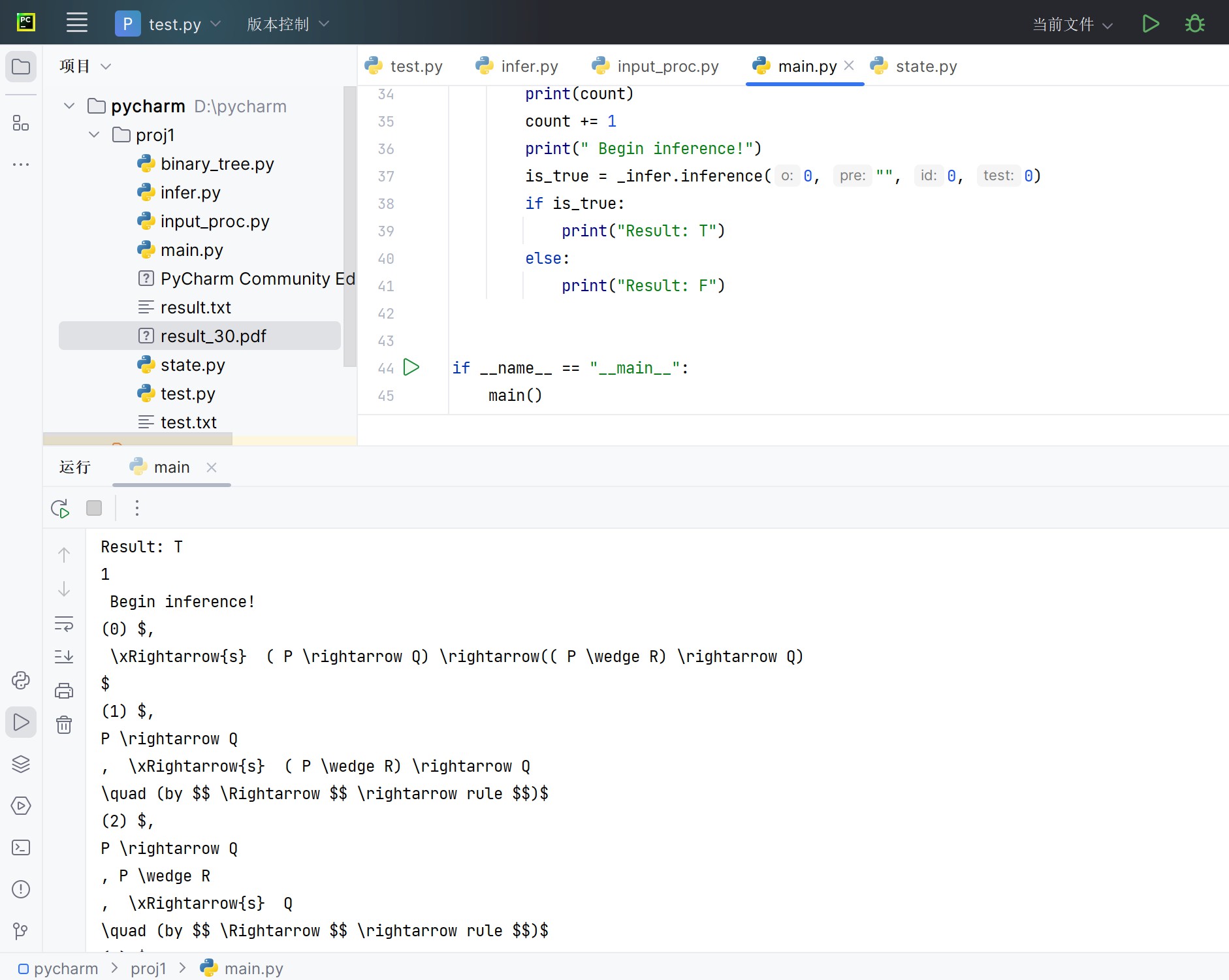


图 1: Enter Caption

# 实现思路

## Input Processing

首先接受合式中缀表达式的输入, 转换为逆波兰表达式（后缀表达式），使用栈结构的优先级处理确保运算符顺序。然后将得到的后缀表达式转换为二叉树存储，实现树建立、插入、删除、遍历等操作。

**Algorithm 1** Convert Infix to Reverse Polish Notation

1: Initialize *res* and *stk* as empty

2: **for** each char *t* in input **do** 3: **if** *t* is an operand **then** 4: Add *t* to *res*

5: **else if** *t* is an operator **then**

6: **while** *stk* not empty and priority(top) ≥ priority(*t*) **do**

7: Pop *stk* to *res*

8: **end while**

9: Push *t* onto *stk* 10: **else if** *t* is ’(’ **then** 11: Push *t* onto *stk* 12: **else if** *t* is ’)’ **then**

13: **while** top of *stk* is not ’(’ **do**

14: Pop *stk* to *res*

15: **end while**

16: Pop ’(’ from *stk*

17: **end if**

18: **end for**

19: **while** *stk* not empty **do**

20: Pop *stk* to *res*

21: **end while**

**Algorithm 2** BinaryTree Initialization and Traversal

1: Initialize an empty stack *stk*

2: **for** each value in input **do**

3: Create a new node *node* with value

4: **if** value starts with ’\’ **then** 5: *node.right* → Pop from *stk* 6: **if** *stk* is not empty **then**

7: *node.left* → Pop from *stk*

8: **end if**

9: **end if**

10: Push *node* onto *stk*

11: **end for**

12: **if** length of *stk* is 1 **then**

13: Return True

14: **end if**

## Inference by Wanghao Algorithm

实例化“推理”类，初始化空的前件列表，后件列表初始化为只包含待推理的表达式二叉树，根据王浩算法进行合式推理。

**Algorithm 3** Inference Algorithm

1: cnt → 0

2: **while** True **do**

3: print\_prove(o, pre, id, test)

4: **for** each *v* ∈ LHS **do**

5: **if** v.root.value[0] **then**

6: *o* → int(*v.root.value*[1])

7: **if** *o* = 1 **then**

8: RHS.append(subtree from v.root.right)

9: **else if** *o* = 2 **then**

10: LHS.append(subtree from v.root.right) 11: LHS.append(subtree from v.root.left) 12: **end if**

13: LHS.remove(v)

14: **end if**

15: **end for**

16: **for** each *v* ∈ RHS **do**

17: **if** v.root.value[0] = ‘//’ **then**

18: *o* → int(*v.root.value*[1])

19: **if** *o* = 1 **then**

20: LHS.append(subtree from v.root.right)

21: **else if** *o* ∈ {3*,* 4} **then**

22: RHS.append(subtree from v.root.right)

23: **if** *o* = 4 **then**

24: LHS.append(subtree from v.root.left)

25: **end if**

26: **end if**

27: RHS.remove(v)

28: **end if**

29: **end for**

30: **if** common trees found in LHS and RHS **then**

31: **RETURN** True

32: **end if**

33: **end while**

## Data Structure

class BinaryTree:

def init (self,

\_input):

self.root = None self.nodes = []

self.L = []

self.R = [] self.\_input = \_input

self.is\_valid = self.init\_tree()

class Inference:

def init (self, l, r, class\_tree\_instance, """

rules=S,):

:param:LHS存储前件所有命题公式对应的表达式二叉树

:param:RHS存储后件所有命题公式对应的表达式二叉树 """

self.rules = rules self.cnt = 0 self.LHS = l self.RHS = r self.nodes = []

self.class\_tree = class\_tree\_instance

# acknowledgement

《数理逻辑与集合论》（第二版）