实验报告 冯雨桐 221240009

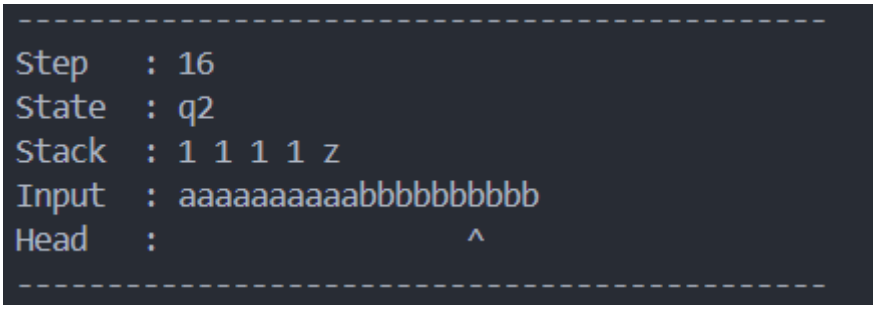
完成度

实验内容: 已全部完成

Bonus: PDA-verbose 模式已完成

(PDA-verbose 模式输出类似 TM-verbose, 如下图所示, head 指向正在读取的字符)

(运行方式也同 TM, 加上 -v/--verbose 参数即可)



设计思路 && 问题及方案

1. 数据结构: 转移函数

PDA 和 TM 的核心都在于转移函数 δ, 这里选择 std::unordered_map 来模拟函数映射

然而 std::unordered_map 并不原生支持 pair/tuple 作为 key, 需要为 pair/tuple 实现 hash 函数

```cpp
namespace std {
    template <>
    struct hash<std::pair<string, string>> {
        size_t operator()(const std::pair<string, string>& p) const {
            size_t h1 = hash<string>()(p.first);
            size_t h2 = hash<string>()(p.second);
            return h1 ^ (h2 << 1);
        }
    };
}
```

2. 数据结构: 纸带

由于纸带需要支持负数下标访问, 直接使用 vector 模拟负数下标有些繁琐

这里选择使用 std::map<int, char> 来模拟纸带并支持负数下标访问

3. 模拟器实现

模拟过程中, 将当前状态(包括栈顶符号/当前纸带符号等)作为 key, 在 unordered_map 中查询对应的 value, 若找不到则说明当前状态下没有可能的转移

4. TM 的 * 符号

TM 转移的新旧符号中可能出现 *, 处理方法如下:

对于旧符号含有 * 的转移, 将 key 中的 * 替换为每一个非 _ 的符号并添加转移

对于新符号含有 * 的转移, 写纸带时什么都不做即可

5. 程序架构

PDA/TM 的实现分别位于 pda.h/tm.h 中, 各自实现了class PDA, class TM

两个类对外开放 simulate(input) 接口以及构造函数

通过将文件路径+verbose传入构造函数来初始化 PDA/TM

通过调用 simulate(input) 接口, 在 input 串上模拟 PDA/TM 的运行

自动化测试

编写测试脚本和一些测试用例, 可以一键编译测试运行

```
cd build && make && cd ../bin
echo ========== ANBN ==============
./fla ../pda/anbn.pda aaaaaabbbbbb
./fla ../pda/anbn.pda aaabbb
./fla ../pda/anbn.pda aabb
./fla ../pda/anbn.pda ab
echo -----
./fla ../pda/anbn.pda aaabb
./fla ../pda/anbn.pda abbb
./fla ../pda/anbn.pda ba
./fla ../pda/anbn.pda a
./fla ../pda/anbn.pda b
./fla ../pda/anbn.pda ""
echo -----
./fla ../pda/anbn.pda
./fla ../pda/anbn.pda acb
./fla ./pda/anbn.pda ab
./fla ./pda/anbn.pd ab
echo ========== CASE1 ==============
./fla ../pda/case.pda ""
./fla ../pda/case.pda "()"
./fla ../pda/case.pda "(())"
./fla ../pda/case.pda "(()()()())()(())()"
./fla ../pda/case.pda "((()()()())()(())())((()()(())())(())())()()
((()))()"
echo -----
./fla ../pda/case.pda "("
./fla ../pda/case.pda ")"
./fla ../pda/case.pda "())"
./fla ../pda/case.pda "((()()()))(()(())()())()()(((()))()"
./fla ../pda/case.pda "((()()()()()(()))()((()()(((()))(()))()))()()(((()))()"
echo -----
./fla ../pda/case.pda "(a)()"
./fla ../pda/case.pda "-"
echo ========== PALINDROME =============
```

```
./fla ../tm/palindrome_detector_2tapes.tm 1
./fla ../tm/palindrome_detector_2tapes.tm 0
./fla ../tm/palindrome_detector_2tapes.tm 101
./fla ../tm/palindrome_detector_2tapes.tm 111
./fla ../tm/palindrome_detector_2tapes.tm 1001
./fla ../tm/palindrome_detector_2tapes.tm 1111
./fla ../tm/palindrome_detector_2tapes.tm 10001
./fla ../tm/palindrome_detector_2tapes.tm 10101
./fla ../tm/palindrome_detector_2tapes.tm 100001
./fla ../tm/palindrome_detector_2tapes.tm 111111
./fla ../tm/palindrome_detector_2tapes.tm 1000001
./fla ../tm/palindrome_detector_2tapes.tm 1010101
./fla ../tm/palindrome_detector_2tapes.tm 111000111
./fla ../tm/palindrome_detector_2tapes.tm 1001001
./fla ../tm/palindrome_detector_2tapes.tm 101010101
./fla ../tm/palindrome_detector_2tapes.tm 100010001
./fla ../tm/palindrome_detector_2tapes.tm 111010111
./fla ../tm/palindrome_detector_2tapes.tm 101000101
./fla ../tm/palindrome_detector_2tapes.tm 10011001
./fla ../tm/palindrome_detector_2tapes.tm 10101010101
./fla ../tm/palindrome_detector_2tapes.tm 10000000001
./fla ../tm/palindrome_detector_2tapes.tm 110000011
./fla ../tm/palindrome_detector_2tapes.tm ""
echo -----
./fla ../tm/palindrome_detector_2tapes.tm 1000010001
./fla ../tm/palindrome_detector_2tapes.tm 111011111
./fla ../tm/palindrome_detector_2tapes.tm 101110101
./fla ../tm/palindrome_detector_2tapes.tm 101001101
./fla ../tm/palindrome_detector_2tapes.tm 111100111
./fla ../tm/palindrome_detector_2tapes.tm 111001111
./fla ../tm/palindrome_detector_2tapes.tm 10010001
./fla ../tm/palindrome_detector_2tapes.tm 010110
./fla ../tm/palindrome_detector_2tapes.tm 11100
./fla ../tm/palindrome_detector_2tapes.tm 10
./fla ../tm/palindrome_detector_2tapes.tm 01
./fla ../tm/palindrome_detector_2tapes.tm 110
./fla ../tm/palindrome_detector_2tapes.tm 01101
./fla ../tm/palindrome_detector_2tapes.tm 11111111111110
echo -----
./fla ../tm/palindrome_detector_2tapes.tm 10000140001
./fla ../tm/palindrome_detector_2tapes.tm a0000140001
echo ========== CASE1 ==========
./fla ../tm/case1.tm ab
./fla ../tm/case1.tm aabb
./fla ../tm/case1.tm aaabb
./fla ../tm/case1.tm aaaabbb
echo -----
./fla ../tm/case1.tm ""
./fla ../tm/case1.tm abc
./fla ../tm/case1.tm abbba
./fla ../tm/case1.tm bab
./fla ../tm/case1.tm aba
echo ========== CASE2 ==========
./fla ../tm/case2.tm ""
./fla ../tm/case2.tm 111
./fla ../tm/case2.tm 11111
```

```
./fla ../tm/case2.tm 11111111
./fla ../tm/case2.tm 11111111111
./fla ../tm/case2.tm 1111111111111
./fla ../tm/case2.tm 11111111111111111111111111
./fla ../tm/case2.tm 1111111111111111111111111111111111111111
echo -----
./fla ../tm/case2.tm 1
./fla ../tm/case2.tm 1111
./fla ../tm/case2.tm 111111111
./fla ../tm/case2.tm 1111111111111111
./fla ../tm/case2.tm 11111111111111111111111
./fla ../tm/case2.tm 1111111111111111111111111111111111111111
./fla ../tm/case2.tm 11111111111111111111111111111111111111111111111111
echo ================ TEST DONE ================
```

感想与建议

感觉实验内容还是很有意思的, 但大部分时间都花在了一些 corner case 的处理上, 比如说语法错误, TM/PDA 的定义细节, 以及 verbose 下的输出等等 (还有阅读大家的海量问题). 而核心功能 (解析器+模拟器) 的代码量和耗时其实并不大

以后或许可以考虑开放 OJ ? 这样大家可以直观地看到自己程序的正确性, 也不会太过纠结于一些细枝末节的问题