

计算理论项目报告

161130118 尹浚宇

分析与设计思路

1. 图灵机程序解析器

由于项目中规定了图灵机程序的语法格式, 因此具体解析过程只需要判断读取到的内容对应于语法中的哪一部分内容, 再将其中内容解析并存储起来即可.

观察到图灵机的语法格式以行作为单位, 所以这里每次都读取一整行, 并根据读取到的结果进行处理.

对于存储图灵机程序的数据结构, 除迁移函数外, 各部分要么为集合, 要么为单个字符, 于是在 C++ 中可以很自然地用 set 和 char 进行处理. 对于迁移函数, 观察到其参数在 C++ 中对应 2 个 string, 其返回值对应 3 个 string, 且参数和返回值具有一一对应的关系, 于是可用 struct 分别封装参数和返回值, 再用 map 实现迁移函数的存储. 特别地, 为了实现通配符功能, 这里还用 multimap 存储了迁移函数参数中的状态-读取符号.

对于注释而言, 整行注释可以直接跳过, 且这里认为行内注释一定出现在每行的有效内容之后. 于是每次读取完有效内容之后就继续读取下一行即可.

2. 图灵机模拟器

由于计算机内存有限, 这里对于图灵机中纸带的模拟, 采用大数组加基准量的方式实现. 具体而言, 用一个 char tape[SIZE] 来模拟纸带, 用 int base 标明纸带起始位置. 对于多带, 这里使用如下方式实现:

```
vector<char[TAPE_SIZE]> tapes(numTape);
```

由图灵机的定义可知, 起始时, 图灵机纸带上的内容即为对应的输入, 每次运行, 根据当前状态以及纸带上的内容, 查迁移函数表, 获取新状态、该往纸带上写入什么内容以及读写头移动的方向, 直到达到终止状态. 因此, 实现的关键其实就是实现迁移函数查表, 然后根据查表结果进行对应的修改即可.

这里迁移函数使用的数据结构 map 天然就对应于 key-value 类型数据, 所以查表过程很简单. 对于通配符的处理, 借助额外的数据结构 multimap, 这里也能简单处理.

3. 图灵机程序

对于程序 1, 大致思路是使用三带图灵机(下标从 0 开始)实现, 其交替使用带 1 和带 2 计算斐波那契数, 当计算完成后, 将结果和输入进行对比, 并根据对比结果进行下一步处理, 具体见项目中的 test.tm.

对于程序 2, 同样使用三带图灵机实现, 其先将输入拷贝至带 2, 并在带 1 上利用额外的标记 X 找到输入串的中间位置, 当找到中间位置后, 简单对比前半段字符串和后半段字符串即可, 具体见项目中的 test.tm.

实验完成度

完成了要求的所有三个任务.

实验中遇到的问题及解决方案

由于本项目要求比较简单，所以写代码过程中基本没有遇到困难。但由于我选择首先使用 vs 在 windows 下进行编码和测试，然后再移植到 linux 中测试，所以遇到了行尾换行符的问题。解决方案是在 linux 下使用命令 `dos2unix` 将输入文档和图灵机程序文档中的换行符从 windows 风格换为 linux 风格。

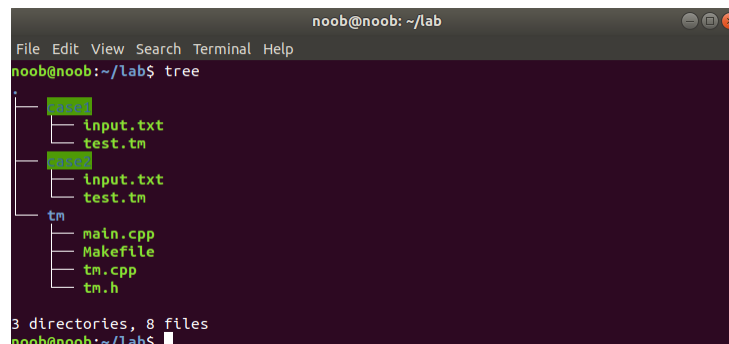
编译运行说明

在 makefile 文件的同级目录下使用 `make` 指令即可编译。同时 makefile 中还提供了 `make test` 和 `make clean` 两个命令分别用于执行测试和清除中间文件。

特别注意：由于不同系统对行尾换行符的处理不同，若在 windows 下执行测试，则输入文件需要在 windows 下创建且编辑；若在 linux 下执行测试，则输入文件需要在 linux 下创建且编辑。

演示

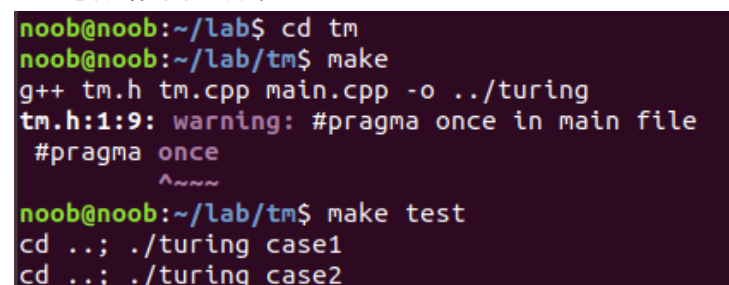
首先观察到目录下的初始状态为：



```
noob@noob: ~/lab
File Edit View Search Terminal Help
noob@noob:~/lab$ tree
.
├── case1
│   ├── input.txt
│   └── test.tm
├── case2
│   ├── input.txt
│   └── test.tm
└── tm
    ├── main.cpp
    ├── Makefile
    ├── tm.cpp
    └── tm.h

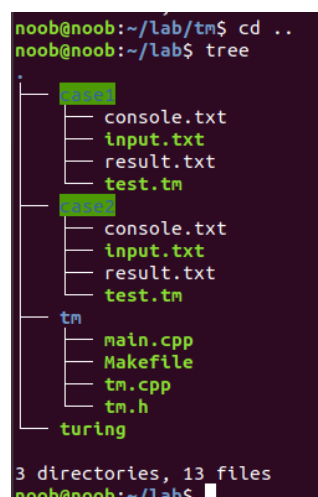
3 directories, 8 files
noob@noob:~/lab$
```

之后使用 makefile 进行编译和测试：



```
noob@noob:~/lab$ cd tm
noob@noob:~/lab/tm$ make
g++ tm.h tm.cpp main.cpp -o ../turing
tm.h:1:9: warning: #pragma once in main file
#pragma once
      ^~~~~~
noob@noob:~/lab/tm$ make test
cd ../; ./turing case1
cd ../; ./turing case2
```

然后再观察目录情况：



```
noob@noob:~/lab/tm$ cd ..
noob@noob:~/lab$ tree
.
├── case1
│   ├── console.txt
│   ├── input.txt
│   ├── result.txt
│   └── test.tm
├── case2
│   ├── console.txt
│   ├── input.txt
│   ├── result.txt
│   └── test.tm
├── tm
│   ├── main.cpp
│   ├── Makefile
│   ├── tm.cpp
│   └── tm.h
└── turing

3 directories, 13 files
noob@noob:~/lab$
```

这里观察 case1 输入和对应的结果来说明正确性:

[illegible]

```
noob@noob:~/lab/case1$ cat result.txt
False
True
True
True
False
True
False
False
True
False
False
False
False
False
True
False
False
False
False
False
False
False
False
True
Error
Error
Error
True
noob@noob:~/lab/case1$
```

可以注意到结果是完全正确的，之后再展示 console.txt，来说明输出格式的正确性：

Step :	32																			
Index0 :	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Tape0 :	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Head0 :	^																			
Index1 :	0	1	2																	
Tape1 :	0	0	0																	
Head1 :	^																			
Index2 :	0	1	2	3																
Tape2 :	0	0	0																	
Head2 :				^																
State :	append2																			

Step :	33																			
Index0 :	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Tape0 :	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Head0 :	^																			
Index1 :	0	1	2																	
Tape1 :	0	0	0																	
Head1 :			^																	
Index2 :	0	1	2	3	4															
Tape2 :	0	0	0	0																
Head2 :				^																
State :	append2																			

Step :	34																			
Index0 :	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Tape0 :	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Head0 :	^																			
Index1 :	0	1	2	3																
Tape1 :	0	0	0																	
Head1 :				^																
Index2 :	0	1	2	3	4	5														
Tape2 :	0	0	0	0	0															
Head2 :					^															
State :	append2																			

总结感想

经过一学期的奋斗，本门课程终于结束了。个人认为这门课程是一门很硬核的课程，每次作业都很难很花时间，有些题甚至需要一整天才能想出答案。但过程越痛苦，收获就越多，总体而言我觉得这门课算是一门非常好的课程。就是本学期最后两次作业的时间都没有给够两周，加上其他课的任务，所以完成起来有些吃力。这里给出我对于课程的一些建议。我觉得作业的布置可以不用等到对应内容都讲完，可以在开新内容的时候就把相应的作业布置下去，在内容结束后的下一周收作业，这样完成作业的时间就会比较充裕，而且可以边学边做相应的题，起到巩固课堂内容的效果。