FLA_turing实验报告

191220136　计算机科学与技术系 许汤智

一、分析与设计思路

命令行分析

首先需要处理命令行中输入的指令。根据手册，输入命令格式为`turing [-v|--verbose] [-h|--help] <tm> <input>`，按照此格式进行`argv`的分别分析。

`argv[1]`的第一位为`-`的话，即为`-h|--help|-v|--verbose`中的一种，其他的直接退出。

```
string arg = argv[i];
if (arg[0] == '-'){
    if (arg == "-h" || arg == "--help"){
        cout << usage << endl;
        exit(0);
    }
    else if (arg == "-v" || arg == "--verbose"){
        verbose = true;
    }
    else{
        cerr << usage << endl;
        exit(0);
    }
}
```

若`argv[1]`的第一位不是`-`的话，即为`xxx.tm`，直接判断是否为tm文件并分别读入文件名和之后的 输入，其他情况如缺失等直接退出。

```
if (filename == ""){
        string argrev = arg;
        reverse(argrev.begin(),argrev.end());
        if (argrev[0] == 'm' && argrev[1] == 't' && argrev[2] == '.')
            filename = arg;
        else{
            cerr << usage << endl;
            exit(0);
        }
}
else if (input == "")
{
    input = arg;
}
else
{
    cerr << usage << endl;
    exit(0);
}
```

解析器

在命令行分析中得到文件名和输入后，初始化图灵机。

图灵机的类定义如下，需要分别初始化其中的成员。

```
class turing
{
private:
    //grammer
    set<string> state;                    //Q
    set<char> input_symbol;               //S
    set<char> tape_symbol;                //G
    set<string> final_state;              //F
    string start_state;                   //q0
    char blank;                           //B
    int N;                                //N
    map<string, vector<delta> > deltafunc; //delta

    //tape
    vector<list<unit> > tapes;
    vector<list<unit>::iterator> heads;

    //command
    string filename;
    bool verbose;
public:
    //initialize
    turing(string fname, bool v);
};
struct delta
{
    string old_state;
    string old_symbol;
    string new_symbol;
    string direction;
    string new_state;
};
struct unit
{
    int index;
    char symbol;
};
```

首先打开文件，若失败则退出，若成功则循环读入文件中每一行。

防止行内有注释影响解析，先对每一行进行清理处理。找到行末的空格和`;`，再截取空格前的有效子串返回。

```
//删除行内注释
int tagindex = -1;
for (int i = 0; i < line.size(); i++){
    if (line[i] == ';'){
        tagindex = i;
        break;
    }
}
```

```
//删除行末空格
int spaceindex = line.size() - 1;
if (tagindex == -1){
    for (int i = line.size() - 1; i > 0; i--){
        if (line[i] != ' '){
            spaceindex = i;
            break;
        }
    }
}
else{
    for (int i = tagindex - 1; i > 0; i--){
        if (line[i] != ' '){
            spaceindex = i;
            break;
        }
    }
}
//返回删除后子串
string s = line.substr(0, spaceindex + 1);
return s;
```

然后对清理完的行进行分析。

如果行首位为`;`或行为空，直接跳过。

如果行首位为`#`，则为部分变量的初始化信息。由于该部分格式为`#X = xxx|{xx,xx}`，直接找出串中`=`号后部分并去除空格和`{}`，根据`{}`和空格的位置，来找出有效子串。

```
int first_space = s.find_first_of(' ');
int last_space = s.find_last_of(' ');
int begin = s.find("{");
int end = s.find("}");
//return {,,,}
else if (begin != string::npos && end != string::npos)
{
    string temp = s.substr(begin + 1, end - begin - 1);
    return temp;
}
//return q0 = 0
else if (begin == string::npos && end == string::npos)
{
    string temp = s.substr(last_space + 1, s.size() - last_space);
    return temp;
}
```

若读出行第二位为`Q,S,G,F`，则对有效子串进行`,`的子串分割，并将每部分放入向量中。

```
vector<string> v;
string::size_type pos1, pos2;
pos2 = s.find(c);
pos1 = 0;
while (string::npos != pos2)
{
    v.push_back(s.substr(pos1, pos2 - pos1));

    pos1 = pos2 + c.size();
```

```
    pos2 = s.find(c, pos1);
}
if (pos1 != s.length())
    v.push_back(s.substr(pos1));
return v;
```

对其中每个部分进行合法测试，测试通过则直接加入对应状态集合中，否则报错退出。

若读出行第二位为`q0,B,N`，则直接进行合法测试后赋值给对应成员变量。

如果行首位为其他的，则为状态转移函数，对行内容进行按空格分割，合法分析后直接插入map对应键的向量中。

```
delta d(v[0], v[1], v[2], v[3], v[4]);
deltafunc[v[0]].push_back(d);
```

由于我实现的是转移函数可以比状态集更早输入，所以在读完文件后对转移函数中状态合法性分析。

```
map<string, vector<delta>>::iterator iter;
//判断新旧状态是否在Q中
for (iter = deltafunc.begin(); iter != deltafunc.end(); iter++){
    vector<delta> value = iter->second;
    for (int i = 0; i < value.size(); i++){
        if (state.count(value[i].old_state) == 0
||state.count(value[i].new_state) == 0)
            error(line, 7);
    }
}
//判断新旧符号组是否在G中
for (iter = deltafunc.begin(); iter != deltafunc.end(); iter++){
    vector<delta> value = iter->second;
    for (int i = 0; i < value.size(); i++){
        for (int j = 0; j < value[i].old_symbol.size(); j++)
            if (tape_symbol.count(value[i].old_symbol[j]) == 0)
                error(line, 10);
        for (int j = 0; j < value[i].new_symbol.size(); j++)
            if (tape_symbol.count(value[i].new_symbol[j]) == 0)
                error(line, 10);
    }
}
```

至此已经初始化了图灵机中的所有部分。

模拟器

接下来就是图灵机针对输入的串进行运行。为方便报告，均按照verbose模式进行报告。

首先针对所有的tape进行初始化，初始化为`(0,_)`，head指向tape头。

```
//初始化tape
for (int i = 0; i < N; i++)
{
    list<unit> tape(1, unit(0, '_'));
    tapes.push_back(tape);
    heads.push_back(tapes[i].begin());
}
```

然后检验输入串，合法后将输入串输入到第一条tape上。

```
//初始化第一条tape并检验合法
if (input != ""){
    tapes[0].clear();
    for (int i = 0; i < input.size(); i++){
        if (input_symbol.count(input[i]) == 0){
            error(to_string(i) + input, 12);
        }
        else{
            tapes[0].push_back(unit(i, input[i]));
        }
    }
    heads[0] = tapes[0].begin();
}
```

接着开始运行图灵机。

首先需要打印当前步骤的各条tape中的信息。初始化各种输出串。

```
string index = "";
string tape = "";
string head = "";
```

因为打印不是所有部分都打印，若头指向非空符号外的空符号，则需要打印对应空符号，否则只需打印非空符号。通过循环找到需要打印tape的左右区间。

```
auto left = tapes[i].begin();
auto right = tapes[i].end();
for (auto iter = tapes[i].begin(); iter != heads[i]; iter++){
    if ((*iter).symbol != blank)
        break;
    left++;
}
for (auto iter = --tapes[i].end(); iter != heads[i]; iter--){
    if ((*iter).symbol != blank)
        break;
    right--;
}
```

在左右区间内打印字符。

首先是`index`。因为负序列号需要去除负号，另外设置一个变量，若相应序列号为负，则乘`-1`。然后将该`index`添加到`index`串后。

```
int indexj = (*j).index;
if ((*j).index < 0)
    indexj *= -1;
string indexstr = to_string(indexj);
index = index + indexstr + " ";
```

`tape`串需要与`index`的数字左对齐，所以要根据`index`的长度添加不同长度的空格。

```
int num = indexstr.size();
tape = tape + (*j).symbol + string(num, ' ');
```

当`heads`指向`index`等于当前位`index`，则在该位置加一个`^`。

```
if ((*j).index == (*heads[i]).index)
    head = head + '^';
head = head + string(num + 1, ' ');
```

最后用制表符对齐输出各行信息。

```
cout << "Index" << i << "\t: " << index << endl
    << "Tape" << i << "\t: " << tape << endl
    << "Head" << i << "\t: " << head << endl;
```

打印完各条tape中信息后，若当前状态为终止状态，或者当前状态没有对应转移函数，则直接停机输出。

```
if (final_state.count(cur_state) == 1)
    return result();
if (deltafunc.find(cur_state) == deltafunc.end())
    return result();
```

若有对应转移函数，则找到当前的符号组。

```
string cur_symbols = "";
for (int i = 0; i < N; i++){
    char ch = (*heads[i]).symbol;
    cur_symbols += ch;
}
```

根据当前状态和当前符号组，找到对应的转移函数，若找不到则直接停机。

```
//找到对应的转移函数
vector<delta> tapen = deltafunc[cur_state];
int funcindex = -1;
for (int i = 0; i < tapen.size(); i++){
    if (tapen[i].old_state == cur_state && tapen[i].old_symbol == cur_symbols){
        funcindex = i;
        break;
    }
}
//找不到转移函数，停机
if (funcindex == -1){
    return result();
}
```

首先需要改变当前指向的符号。

```
(*heads[i]).symbol = tapen[funcindex].new_symbol[i];
```

然后改变头指向的位置。碰到最左边或最右边需要进行扩展并加入空符号。

```
switch (tapen[funcindex].direction[i])
{
    case 'l':
        //如果最左边省略负号(后面再乘-1)
        if (heads[i] == tapes[i].begin()){
            tapes[i].push_front(unit((*heads[i]).index - 1, blank));
            heads[i]--;
        }
        else
            heads[i]--;
        break;
    case 'r':
        //如果最右边扩展
        if (heads[i] == --tapes[i].end()){
            tapes[i].push_back(unit((*heads[i]).index + 1, blank));
            heads[i]++;
        }
        else
            heads[i]++;
        break;
    case '*':
        break;
    default:
        error(tapen[funcindex].direction, 9);
        break;
}
```

最后改变当前状态。

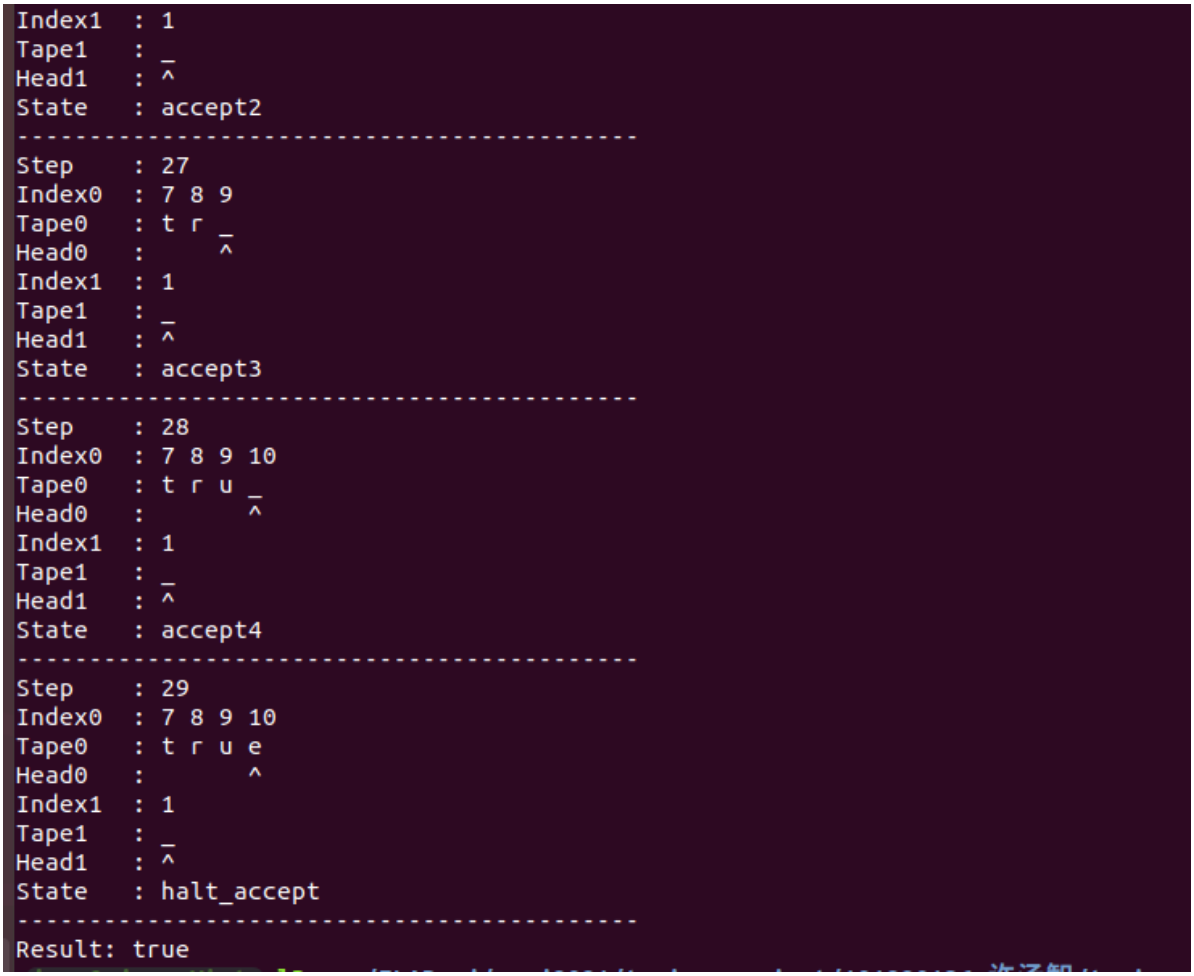
```
cur_state = tapen[funcindex].new_state;
```

最后停机后输出结果，只要将tape上从左到右输出非空符号即可。

```
string res = "";
for (auto i = tapes[0].begin(); i != tapes[0].end(); i++)
{
    char ch = (*i).symbol;
    if (ch != blank)
        res = res + ch;
}
return res;
```

至此，图灵机全部完成。

运行结果如下：



多带图灵机程序

实现最大公约数，采用辗转相减法，用两个带来进行互相比较并循环删除1的数量，直至最后1数量相等即得到结果。

首先将初始状态到copy状态，若输入开头就为0，则进入error状态。

```
q0 1_ 1_ ** copy
;若第一个读到0，说明没有第一个数，错误
q0 0_ __ r* error
```

error状态输出`error`。

```
;处理错误
error 1_ __ r* error
error 0_ __ r* error
error __ r_ l* error1
error1 __ o_ l* error2
error2 __ r_ l* error3
error3 __ r_ l* error4
error4 __ e_ ** halt_error
```

copy状态是将输入复写到第二条带上,复写完后进入beginindex状态。

```
copy 1_ 11 rr copy
copy 0_ 00 rr copy
copy __ __ ll beginindex
```

beginindex状态将第二条带的指针指到开头,之后进入find状态。

```
beginindex 11 11 *l beginindex
beginindex 10 10 *l beginindex
beginindex 01 01 *l beginindex
beginindex 00 00 *l beginindex
beginindex 1_ 1_ *r find
beginindex 0_ 0_ *r find
```

find状态即进行辗转相减，指针一起移动直到一个到0，根据情况进入erase状态，若同时到0即a=b，进入accept_erase状态。

```
;指针一起移动直到一个到0
find 11 11 lr find
;b>a,然后进行删除
find 10 10 rl erase1
;a>b，然后进行删除
find 01 01 rl erase2
;a=b,进行接受前的处理
find 00 00 ** accept_erase
```

若第一条带上1多，则删除对应数量的1，否则同理。然后进入recover状态。

```
;b=b-a，删除第一条上a个1
erase1 11 _1 rl erase1
erase1 __ __ lr recover

;a=a-b，删除第二条上b个1
erase2 11 1_ rl erase2
erase2 __ __ lr recover
```

recover状态是用来将指针调整到find之前的位置，以便进行辗转相减的循环。

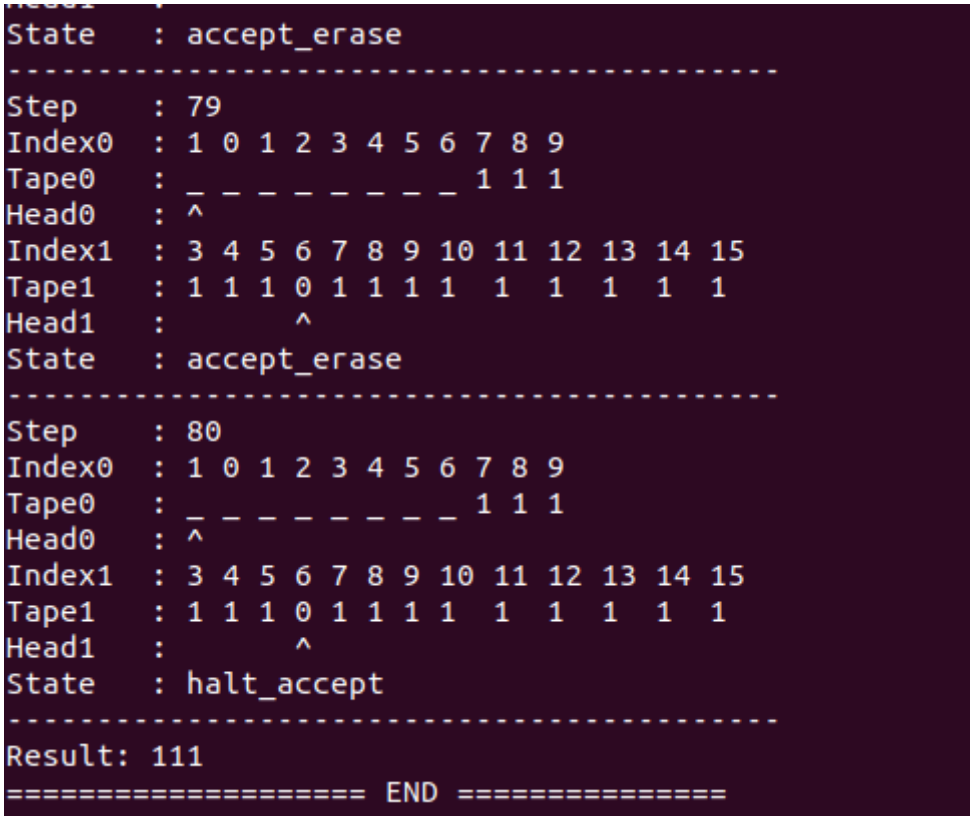
```
recover _1 _1 l* recover
recover 1_ 1_ *r recover
recover 11 11 ** find
```

到accep_erase状态时，需要将第一条tape中除了结果以外的其他符号全置为空格符号。

```
accept_erase 00 _0 l* accept_erase
accept_erase 10 _0 l* accept_erase
accept_erase _0 _0 ** halt_accept
```

运行结果如下：

```
njucs@njucs-VirtualBox:~/FLAProj/proj2021/turing-project/191220136-许汤智/turing-project$ ./turing -v ~/FLAPro
j/proj2021/turing-project/191220136-许汤智/programs/gcd.tm 1111110111111111
```



6和9的最大公约数为3，符合预期。

二、实验完成度

完成实验全部内容，包括普通模式和verbose模式。

三、实验中遇到的问题及解决方案

1、转移函数的变量和tape上字符如何保存问题，使用了多种方案，最后使用了`map`和`vector<list>`来储存信息。

2、判断转移函数中状态的合法性，一开始是边读边判定，这必须要求转移函数在状态集初始化后输入，较为局限，后改为读完文件后进行判定。

3、没有实现通配符导致调试所给tm文件时无法运行，修改tm文件，将所给文件通配符改为分类的几种情况，后调试运行。

4、遇到文件中一行后有空格，加入清理函数来删去空格。

5、空行不能用`` `line[0]== ' ' ``来识别，应该用`line==" "`来识别。

6、map和list只能用迭代器进行访问，而不能用类似数组下标的方式进行访问。

7、停机情况包括进入终止状态，当前状态没有转移函数和当前状态转移函数中没有与当前符号组对应的转移函数。

8、index去掉符号输出问题，采用输出时用另外变量减去负号，而不是在tape上将index置为正数。

9、输出的对其问题，需要控制好空格的输出个数，以及使用制表符。

四、总结感想

这次实验让我对确定性图灵机的运行流程有了更深入的认识，了解了确定性图灵机的运行流程，对理论知识也有了更透彻的理解。在实验实践中，对C++的STL容器有了更熟练的应用，包括在字符串的处理以及迭代器的运用等。编写图灵机程序时，能够做到思路较为清晰，获得了长足的进步。

五、对课程和实验的意见与建议

无