

评阅项1计分＿＿＿＿＿＿

评阅项2计分＿＿＿＿＿＿

评阅项3计分＿＿＿＿＿＿

评阅项4计分＿＿＿＿＿＿

实验得分＿＿＿＿＿＿

评阅日期＿＿＿＿＿＿

**编译原理实验**

实验一：正则表达式与有限自动机

实验选题：编程实现正则表达式转换为NFA

学生姓名：

学 号：

专业班级：

任课教师： 曹江莲

提交日期：

1. **实验目的**

通过本次实验，加深对正则表达式、NFA、DFA及其识别的语言的理解。

1. **实验内容**
2. DFA的输入与DFA的存储：确定DFA的数据结构以及存储格式。
3. DFA的正确性检查：DFA的五元组是否正确。
4. 输入任意一个整数N，DFA的能列表显示其识别的所有长度小于等于N的字符串。
5. DFA的规则字符串判定：输入（或用字符集随机生成）一个字符串，模拟DFA识别字符串的过程判定该字符串是否是规则字符串（属于DFA的语言集）。
6. **算法思想及问题对策**
7. 算法描述

采用数组存放DFA的状态转换图，由于这里的数组是用的普通数组，不能存放字符（或者要转换为ASCILL码），我采用的是将所有字符存放在一个数组e中，字符在e中的位置就代表着这个字符。

对于DFA五元组的校验做了简单的判断，首先就是始态和终态不能比输入的最大状态还大其次就是每个元素都要在字母表中，在就是如果DFA表的一个项目已经被填写输入了，这时再次要在这个项目中输入，如果输入的不是和已经存在的值相同，则这个DFA必然出错。

对于识别长度小于n的可识别串的输出，这里我采用递归搜索的形式进行，从始态开始搜索，按照元素在字母表中的位置进行，没搜索到一个就输出一个，当搜索到当前的长度大于输入的n的值的时候，其后面的必然都是长度大于n的字符串，此时返回，然后进行上一层的下一个元素的搜索。总结起来就是一个DFS。

对于字符串的识别，从始态开始，用一个指针记录识别到字符串的那个位置的字符，当当前状态和当前字符在DFA存储数组中有值时，状态转移到改值记录的状态，指针后移一位，如果中途发DFA数组中不存在值，则报错。若指针移到了字符串的末尾，此时若状态为终态则识别成功，否则识别失败。

1. 设计过程中的问题与对策

对于长度小于n的能识别字符串的递归搜索一直有问题，后来发现是玲姐条件的设置有问题，当长度到达了n-1就不需要再次的递归了。

对于字母表的存储，一开始是想用map，但是后来发现map不好去遍历，所以改成了一般的数组。

字符串识别的时候没有考虑字符串最后的状态是不是终态，而导致只要每一个元素都有下一状态就会识别成功，后来加入了一个函数判断最后一个状态是否为终态。

1. 本实验特色

首先当然就是递归搜索长度小于n的可识别字符串了。然后就是使用了动态数组来存储每一次识别的字符，用来充当一个栈的作用，这刚好和我们的递归想吻合。

用映射的下标来表示字母表中的元素，防止了ASCILL码转换的问题。

1. **具体实现**

1.数据结构以及变量的说明

|  |  |
| --- | --- |
|  | **int sum\_end = 0;表示终态的个数**  **int status = 0;表示DFA中终态数的个数**  **int elem = 0;代表元素的个数，即字母表中的元素个数**  **int start = 0;DFA的开始状态**  **int num = 0;用于完成功能识别长度小于n的字符串的n的输入保存**  **int dfa[100][100];存储DFA的矩阵**  **int enddfa[100];终态集合**  **char e[20];字母表的元素**  **vector<int> vec;识别长度小于n的可识别字符串时，方便存储输出。辅助输出** |

|  |  |
| --- | --- |
|  | **int getelemlocal(char \*a,char b)该函数输入字符串和字符，查找该字符是否在字符串中存在，存在则返回该字符在字符串中的位置，否则返回-1.用来获取字母表中的元素的映射位置。** |

|  |  |
| --- | --- |
|  | **int inend(int\* a,int b,int len)输入字符串a（终态集合）和长度len，输入b状态，用来判段该状态b是否是属于终态集，是则返回1，不是则返回0。**  **用于DFA的识别。** |

|  |  |
| --- | --- |
|  | **void search(int now\_status,int n,int now\_elem\_num)该函数用于递归搜索长度小于n的可识别字符串，并且打印出来。Now\_status是当前所在的状态，n是当前字符串的长度，now\_elem\_num是要输入的字母表元素（这里用他在字母表数组中的下标来表示）。**  **void search(int now\_status,int n,int now\_elem\_num)**  **{**  **if(n<num&&dfa[now\_status][now\_elem\_num]>=0)**  **{//num为输入的最大长度，如果此时的长度小于最大长度，而且在DFA数组中存在当前状态和元素的下一状态，则把这个元素压如动态数组vec中，说明可以往后移动。**  **vec.push\_back(now\_elem\_num);**  **if(inend(enddfa,dfa[now\_status][now\_elem\_num],sum\_end)&&n>=0)**  **{**  **//如果当前状态是终态，那么可以将动态数组中存储的元素输出。**  **for(int i = 0;i<vec.size();i++)**  **{**  **cout<<e[vec[i]];//因为存储的是元素在字母表中的下标，所以需要对应出来。**  **}**  **cout<<" #######\n";//这里是debug时留下的，没有意义**  **}**  **search(dfa[now\_status][now\_elem\_num],n+1,0);//多了一个元素//递归搜索**  **if(!vec.empty())**  **vec.pop\_back();//递归返回之后需要将元素压出，对应于算法就是一种回溯。**  **}**  **for(int i = now\_elem\_num+1;i<elem;i++)**  **{**  **if(dfa[now\_status][now\_elem\_num]>=0)**  **search(now\_status,n,i);**  **}//从字母表的下一元素开始继续进行递归搜索。**  **}** |

1. 对DFA的正确性判断

|  |  |
| --- | --- |
|  | **if(begin>=status||end>=status||(e,cor)<0)**  **{**  **cout<<"数据错误！";**  **return 0;**  **}//状态不能大于输入的最大状态，输入的状态和元素的转换之后的下一状态要在状态表中**  **if(dfa[begin][getelemlocal(e,cor)]!=-1&&end!=getelemlocal(e,cor))//转换图中有，而且和之前的不一样（不是重复输入）**  **{**  **cout<<"非DFA";**  **return 0;**  **}** |

1. 用于输出中文提示的宏

因为输出的比较简单，所没有设置宏，直接在输入之前用一个cout语句进行提示消息的输出。

1. 识别字符串

该步骤比较简单，在注释中也有所说明。

|  |  |
| --- | --- |
|  | **cin>>input;**  **int seei=0;**  **int startstatus = 0;**  **int flag = 0;**  **int now\_start=start;**  **for(int seei = 0;seei<strlen(input);seei++)**  **{//seei就是一个字符串的指针。**  **if(dfa[now\_start][getelemlocal(e,input[seei])]>=0)**  **{**  **cout<<now\_start<<endl;**  **now\_start = dfa[now\_start][getelemlocal(e,input[seei])];//如果能够转移则转移到下一状态**  **}**  **else**  **{**  **cout<<"该字符串不能识别！卡在了状态"<<now\_start<<endl;**  **flag = 1;**  **break;**  **}**  **}**  **if(!flag)**  **{**  **cout<<now\_start<<endl;**  **if(inend(enddfa,now\_start,sum\_end)) cout<<"字符串识别成功!\n";**  **else cout<<"该字符串不能识别,最后一个状态不是终态！\n";**  **}**  **cout<<"1识别长度小于n的字符串，2识别字符串，其他退出：";**  **}** |
|  |  |

1. **测试与运行情况**
2. 测试样例数据

测试用的DFA如图1所示。

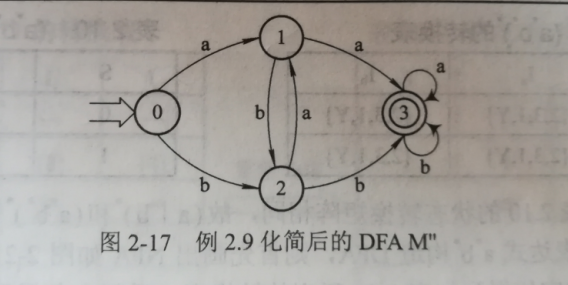


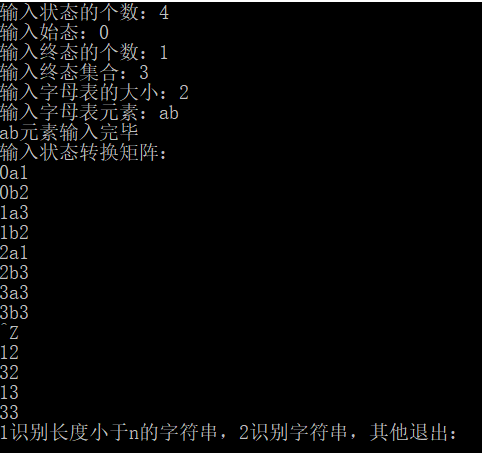
图1 测试用例

程序中输入数据如下：

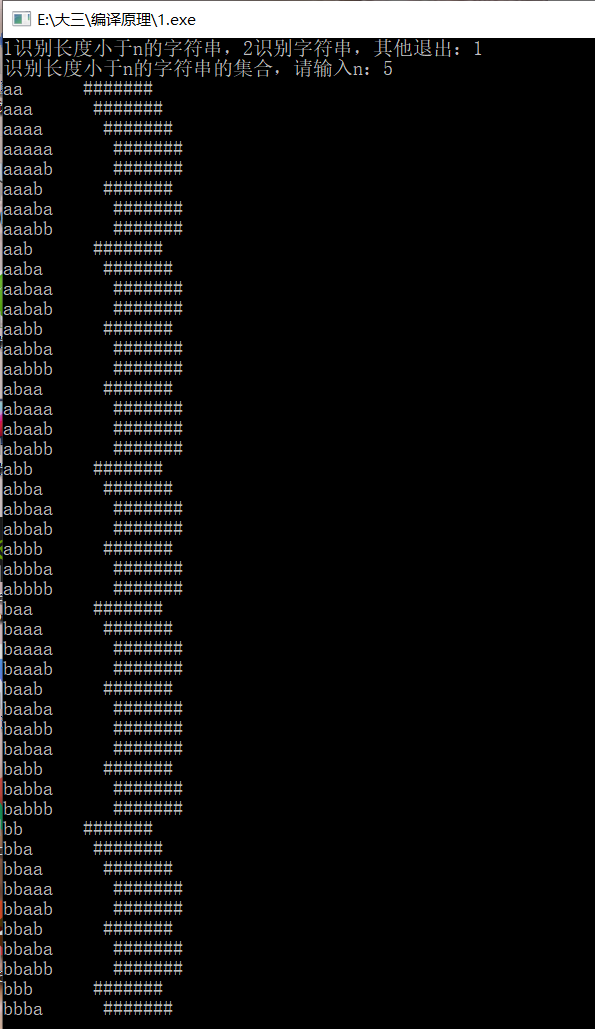
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | 4  0  1  3  2  ab  0a1  0b2  1a3  1b2  2a1  2b3  3a3  3b3  Ctrl+Z(这里要输入回车后执行)  Enter |

1. 程序执行结果

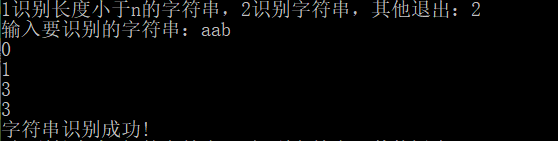
DFA的输入，程序的初始化



识别长度小于n的可识别字符串，并打印



识别特定的字符串



1. 程序执行结果分析

长度小于5的字符串经过网上查找对照发现和实验结果一致。

关于字符串aab的识别，根据DFA的直观图可以看出的确可以识别该字符串，所以程序的运行结果没有问题。

1. **实验总结**
2. 程序存在的问题
3. 对于DFA准确性的判断还是比较简单，需要更加严密的算法和判断条件。
4. 程序的变量的命名有些没有章法，需要加强
5. 对于存储数组的设置，由于设置的是一般数组，当数据量超过了设定的值时就会发生数组越界的问题，可以改成动态数组或者链表来解决。
6. DFA的存储还能够优化。

2. 收获

A）复习了DFA的有关知识，对DFA的了解更加的深刻，对于怎么判断DFA的准确性有了自己的一套方法

B）由于之前自己的算法功底不算深厚，对于递归程序的编写花了较大的心思，所以对于程序设计有了一些新的体会，对于以后要同类型的问题有了知识的储备。

C）对于DFA的状态转换图了解更加的深刻，学会了怎样去更加有效率地使用状态转换图。