Introduction To The Theory Of Computation Michael Sipser
计算理论导论

LAB A	实验 A—-ADFA 的可判定性
Name	屈德林
Student No.	201808010522
Class	计算机科学与技术 1805
Department	CSEE
Email	qdl.cs@qq.com
Date	2021 年 5 月 15 日

湖南大學

目录

1 Problem description

A_{DFA}=< B,w >|B 是 DFA，w 是串，B 接收 w 证明：A_{DFA} 是可判定的。实验方法: 编写一个算法/程序，对于任意给定的输入，可以判定 ADFA。

1.1 Input

有多个测试序列，测试结束于测试文件结束；每个测试序列的第一行为几个正整数 n m t a 分别表示有 n 个状态，从 a 开始 m 个小写字母组成的字符集，第一个状态默认为起始状态。t 个接受状态和 a 个测试串, 接下来为一个 n 行 m 列的矩阵 S, 其中 S[i][j] 表示第 i 行第 j 列，意义为状态 i 经过字母 j 到达状态 S[i][j]。接下来有 t 个数字, 表示 t 个接受状态值，然后是 a 行，每行一个串表示待测试的串。

1.2 Output

对于每个字符串输出 YES 表示该 DFA 接受该串，NO 表示不接受。

1.3 Sample Input

```
3 3 1 2
2 3 2
3 3 3
3 3 3
2
a
b
```

1.4 Sample Output

```
YES
NO
```

2 Lab Environment 环境

- 操作系统：Arch Linux
- 程序运行环境：gcc (GCC) 10.2.0
- 报告编写环境：TeX Live 2020
- 开发工具：VSCode

3 Lab Steps 步骤

3.1 分析问题

题目理解：矩阵的行代表状态：状态 1,2,3；矩阵的列代表字母第 1 列就是字母 a，第 2 列是字母 b，以此类推。倒数第三行表示的一个接受状态值 2. 即状态 2 是接受状态。

3.2 算法思想

1. 首先将输入的状态转移矩阵保存在 S 数组中，其中其中 S[i][j] 表示第 i 行第 j 列，意义为状态 i 经过字母 j 到达状态 S[i][j];
2. 对每一个输入的串 W, 从 after（after 表示每次转换后的状态，初始为起始状态）开始，按照每一个字符，得到相应的后继状态，保存在 after 中。
3. 最后判断 accept[after] 的值，即串在 DFA 上运行之后最终状态是否可接受。

3.2.1 算法伪代码表述：

经过上述分析，算法伪代码可以表述为：

```
int main(){
    while(scanf("%d%d%d%d",&n,&m,&t,&a)!=EOF){              // 多个序列输入
        memset(s,0,sizeof(s));   // 初始化为0
        for(int i = 1;i<=n;i++) // 状态转换矩阵
        for(int i = 0;i<t;i++)   // 输入接收状态F
        while(a--){
            // 输入测试串
            int cur = 1;
            for(int i = 0;i<temps.length();i++){
                cur = s[cur][temps[i]-'a'+1]; // 转移
            }
            if(F[cur]==1)
                printf("YES\n");
            else
                printf("NO\n");
        }
    }
}
```

4 Lab Results 结果

4.1 实验结果

HUNAN UNIVERSITY ACM/ICPC Judge Online

Solution 697436's Status

User: jsll201808010522 , **Problem :** 12595
Language : GNU C++ , **Judge Result:** Accepted

Source Code

```
#include <iostream>
#include <cstring>
using namespace std;
int n,m,t,a;
int s[1000][1000],F[1000];                              // 状态转换矩阵和接受状态集
int main(){
        while(scanf("%d%d%d%d",&n,&m,&t,&a)!=EOF){                              // 多个序列输入
                memset(s,0,sizeof(s));                          // 初始化为0
                memset(F,0,sizeof(F));
                // 状态转换矩阵
                for(int i = 1;i<=n;i++){
                        for(int j = 1;j<=m;j++)
                                scanf("%d",&s[i][j]);
                }
                // 输入接收状态F
                for(int i = 0;i<t;i++){
                        int cur;
                        cin>>cur;
                        F[cur] = 1;
                }
                // 输入测试串
                while(a--){
                        string temps;
                        cin>>temps;
                        int cur = 1;
                        for(int i = 0;i<temps.length();i++){
                                // 转移
                                cur = s[cur][temps[i]-'a'+1];
                        }
                        if(F[cur]==1)
                                printf("YES\n");
                        else
                                printf("NO\n");
                }
        }
        return 0;
}
```

图 1: http://acm.hnu.cn/online 提交结果

在 http://acm.hnu.cn/online 提交代码，AC 通过. SolutionID 697436 ，User jsll201808010522，Memory 5056KB Time Used 15ms，实验正确。

5 Lab Experience 心得

5.1 实验心得

1. 审题出现了问题：我因为看到样例输入就只有一个 DFA，所以写的代码就只能判定一个 DFA，但实际上程序应该能连续识别多个 DFA 的测试。

2. Runtime error 有三个原因，不过我现在只记得两个了：一是除 0，二是爆栈。在我的程序中除 0 是不可能的了，只能是因为爆栈了。因为我是用 C++ 的 new 去动态分配数组的，而且我在程序中还用了 while 循环（因为要测定多个 DFA），所以一定要切记 new 完要自己去 delete 释放内存空间！如果一开始换用静态数组也没这些问题。

A 附录 1：Solution

```
#include <iostream>
```

```
#include <cstring>
using namespace std;
int n,m,t,a;
int s[1000][1000],F[1000];                    // 状态转换矩阵和接受状态集
int main(){
    while(scanf("%d%d%d%d",&n,&m,&t,&a)!=EOF){ // 多个序列输入
        memset(s,0,sizeof(s));                // 初始化为0
        memset(F,0,sizeof(F));
        // 状态转换矩阵
        for(int i = 1;i<=n;i++){
            for(int j = 1;j<=m;j++)
                scanf("%d",&s[i][j]);
        }
        // 输入接收状态F
        for(int i = 0;i<t;i++){
            int cur;
            cin>>cur;
            F[cur] = 1;
        }
        // 输入测试串
        while(a--){
            string temps;
            cin>>temps;
            int cur = 1;
            for(int i = 0;i<temps.length();i++){
                // 转移
                cur = s[cur][temps[i]-'a'+1];
            }
            if(F[cur]==1)
                printf("YES\n");
            else
                printf("NO\n");
        }
    }
    return 0;
}
```