实　验　报　告

学　　号	20020006107	姓　　名	王义钧	专业班级	2020 级通信工程
课程名称	自然语言处理			学期	2022 年 春 季学期
任课教师	仲国强	完成日期	2022/6/7	上机课时间	2022/6/7
实 验 名 称	NLP 实验五：形态分析				

一、 实验目的

学习有限状态自动机的状态转移，并利用自定义的有限状态自动机实现对英语单词的形态还原。

1) -ed 结尾的动词过去时，去掉 ed；
*ed → * (e.g., worked → work)
*ed → *e (e.g., believed → believe)
*ied → *y (e.g., studied → study)

2) -ing 结尾的现在分词，
*ing → * (e.g., developing → develop)
*ing → *e (e.g., saving → save)
*ying →*ie (e.g., die →dying)

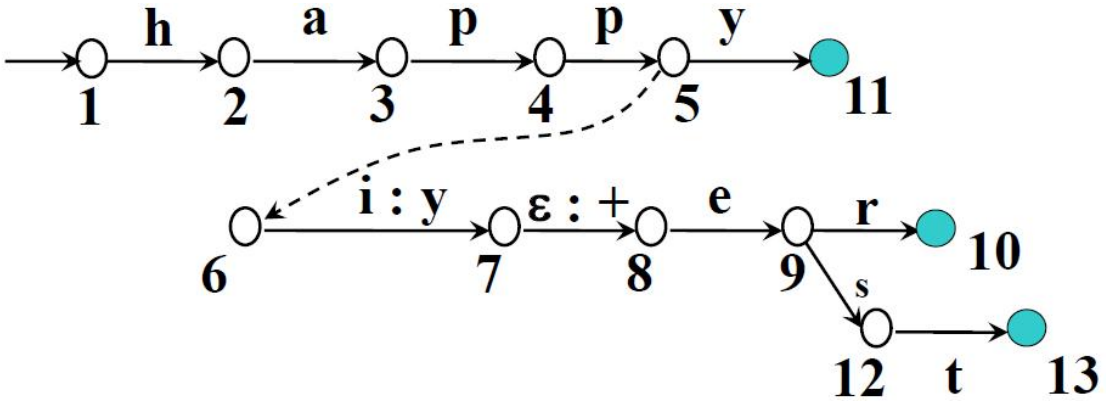
3) -s 结尾的动词单数第三人称；
*s → * (e.g., works → work)
*es → * (e.g., discuss → discusses)
*ies → *y (e.g., studies → study)

4) -ly 结尾的副词
*ly → * (e.g., hardly → hard)

5) –er/est 结尾的形容词比较级、最高级
*er → * (e.g., cold → colder)
*ier → *y (e.g., easier → easy)
……

二、 实验内容：

请实现以下有限自动机的状态转移过程,通过它的状态转移过程可以识别happy的原型，比较级happier，最高级happiest，可以将单词的比较级和最高级转换为“原型+后缀”的形式，进行单词形态的还原，其中<i：y>为输入输出标签对，即输入i不仅进行状态转移，同时会输出y，可以理解为把i转换为y，另外ε为空输入，即不需要输入，可以输出+，可以理解为这一步转移不需要识别字符，直接输出+。状态转移过程需要自己实现。



最终结果：

输入：happy

输出：happy→happy

输入：happier

输出：happier→happy+er

输入：happiest

输出：happiest→happy+est

三、 实验过程：

1、将自动机转换为邻接矩阵。

建立一个邻接矩阵，按照给定好的自动机形式，将其输入到矩阵中。每两个点间的权值大小为读取到“happy”字符串的值。因此，在主函数中，进行形式语言自动机的跳转，即通过读取邻接矩阵的每个可以匹配节点的列值，将其列值作为下一个行值继续查找。从第 0 个到第 4 个节点的值分别为“happ”,从第 4 个节点开始设置分支,从第 5 个节点到第 8 个节点分别设置“y+e”,第 8 个节点设置分支，一直到第 12 个节点设置权值“st”。对于结束节点 9,10，12，则会在主函数中进行判断。此外，其他节点的权值设置为全 0。

```
void initMatrix() {
    matrix[0][1] = 'h';
    matrix[1][2] = 'a';
    matrix[2][3] = 'p';
    matrix[3][4] = 'p';
    matrix[4][5] = 'y';
    matrix[4][10] = 'y';
    matrix[5][6] = 'y';
    matrix[6][7] = '+';
    matrix[7][8] = 'e';
    matrix[8][9] = 'r';
    matrix[8][11] = 's';
    matrix[11][12] = 't';
}
```

2、设置检索字符串限制条件。

我们想要规定输入的格式分别为“happy”，“happier”，“happiest”，至于其他格式则不允许输入，提示“格式错误！”。

```
while(scanf("%s", &s) != EOF) {
        flag = 0;
        if(!strcmp(s, "happy"))
            flag = 1;
        else if(!strcmp(s, "happier"))
            flag = 1;
        else if(!strcmp(s, "happiest"))
            flag = 1;

        if(!flag) {
            cout << "格式错误！" << endl;
        }
```

3、根据自动机还原英语单词。

声明整型变量 line，表示每个节点的行数，j 为列数。如果可以匹配到字符，那么令 line 等于其列数，可以查找到该点对应的下一个节点的行位置。同时，在下一层循环中，令 j = line + 1。这里是因为我给邻接矩阵添加的权值基本都分布在对角线上，所以令其为 line + 1 可以充分减少时间复杂度。

但字符匹配不是所有字符都可以匹配，比如“i”。当检索到“i”时，需要设置一个跳转标志 flag1。令 flag1 = 1，进行单独的跳转检索。再按照上面的方法，每一次循环均输出对应邻接矩阵的值。一直到节点 8 为止。

如果输入字符串为“happiest”，此时无需再设置跳转符。只需令 flag = 0，然后继续进行匹配检索即可。

```
if(flag) {
            cout << s << "->";

            int line = 0;
            int tmp = 0;
            int flag1 = 0;
            int breakCircle = 0;
```

```
            for(int j = line + 1; j < 13; j++) {
                if(breakCircle)
                    break;
                for(int k = tmp; k < strlen(s); k++) {
                    if(s[k] == 'i' || flag1) {
                        flag1 = 1;
                        if(line == 4) {
                            line++;
                            break;
                        }

                        cout << matrix[line][j];
                        line++;
                        if(line == 8) {
                            flag1 = 0;
                            tmp = tmp + 2;
                        }
                        break;
                    }
                    else if(s[k] == matrix[line][j]) {
                        cout << matrix[line][j];
                        if(j == 10 || j == 9 || j == 12) {
                            breakCircle = 1;
                            break;
```

```
                    }
                    line = j;
                    tmp++;
                    break;
                }
                else{
                    tmp = k;
                    break;
                }
            }
        }
    }
```

四、 结果展示：

五、 心得体会：

该次实验比较简单，主要学习了如何利用形式语言自动机实现对英语单词happy不同形态的辨别。基于实验一的实现思路，我首先将图的相关内容转变为矩阵的相关内容来实现，也进一步熟练掌握自动机的编程实现能力。希望在以后的实验中继续锻炼自己的能力。