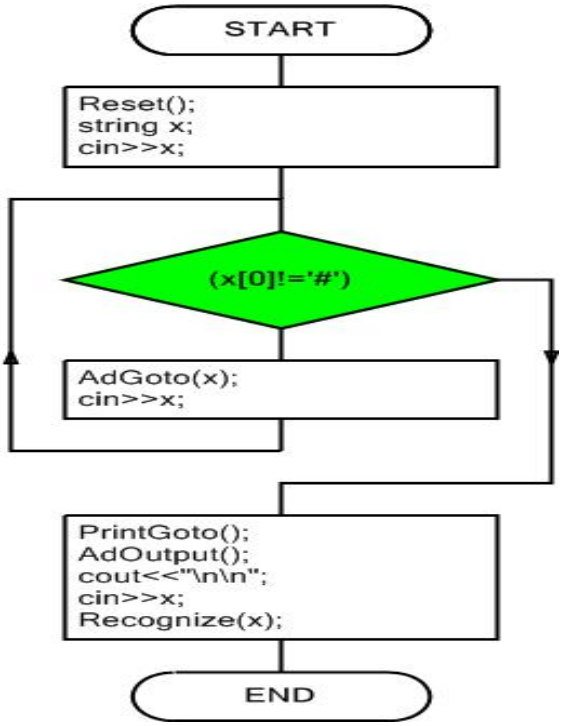
恶意软件特征代码法检测

一、程序流程

```
void Reset();/*初始化*/
void AdGoto(string x);/*更新 Goto&&Output&&D*/
void AdOutput();/*更新 Fail&&Output*/
void Recognize(string x);/*匹配目标文本*/
void PrintGoto();/*输出 Goto*/
```



二、定义的数据结构及功能

1.int Goto[M][28](转向函数表)

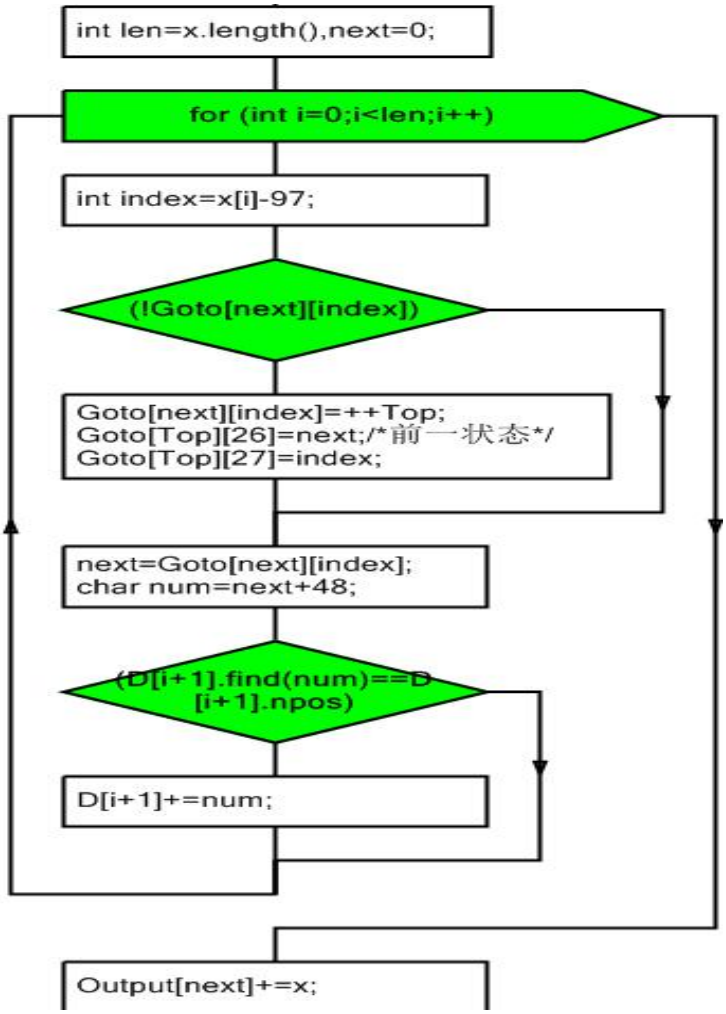
Goto[i][j](j<26)存储的内容为：在状态 i 时，输入字符 H(j)所转向的状态
其中 H(0)=a、H(1)=b……H(25)=z

Goto[i][26]存储状态 i 的前一级状态
Goto[i][27]存储前一级状态转向当前状态 i 的输入字符 x 所对应的整数 F(x)
其中 F(a)=0、F(b)=1……F(z)=25

Goto[i][j](j=0、1……27)初始化时为 0

2.int Fail[M](失效函数表)：Fail[i]存储状态 i 的失效状态,初始化时为 0
3.int Top=0：表示当前的状态总数，初始化时为 0
4.string Output[M](输出函数表)：Output[i]表示状态 i 的输出，初始化时为""
5.string D[M]：D[i]表示深度为 i 时所对应的状态集合，初始化时为""

三、转向函数的构建过程：void AdGoto(string x)



1.从状态 0 开始更新，从左到右按字符依次读入某个模式

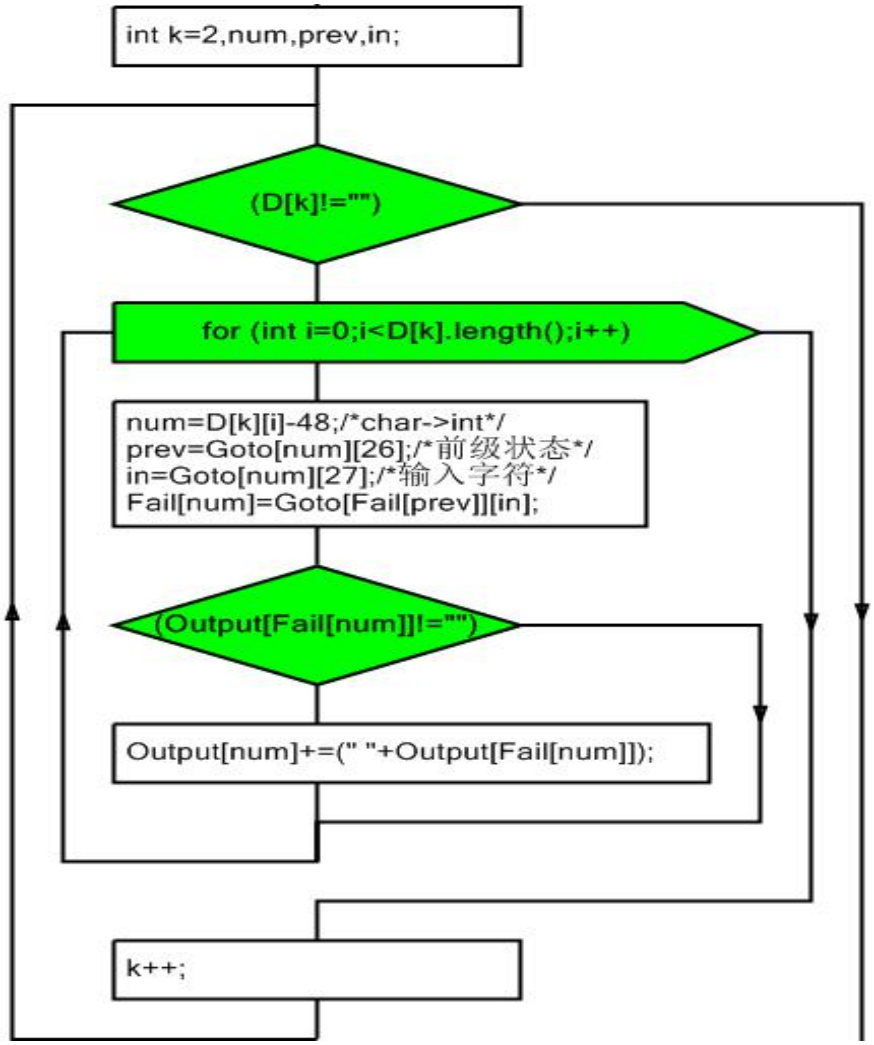
1.1.将当前字符 y 转换为 index=F(y)；若 Goto[next][F(y)]=0，则作如下更新
①.添加新的状态，即 Goto[next][F(y)]=++Top
②.更新 Goto[Top][26]和 Goto[Top][27]

1.2.转向下一状态

1.3.若更新 Goto 表，则将新状态 num 加入 D[i+1]，其中 i 为字符 x 的下标
其中 num=(char)next

2.读完该模式后，更新 Output 表

四、失效函数和输出函数的构建过程：void AdOutput()



从第二层 D[2]开始更新失效函数，直到 D[k]=""：

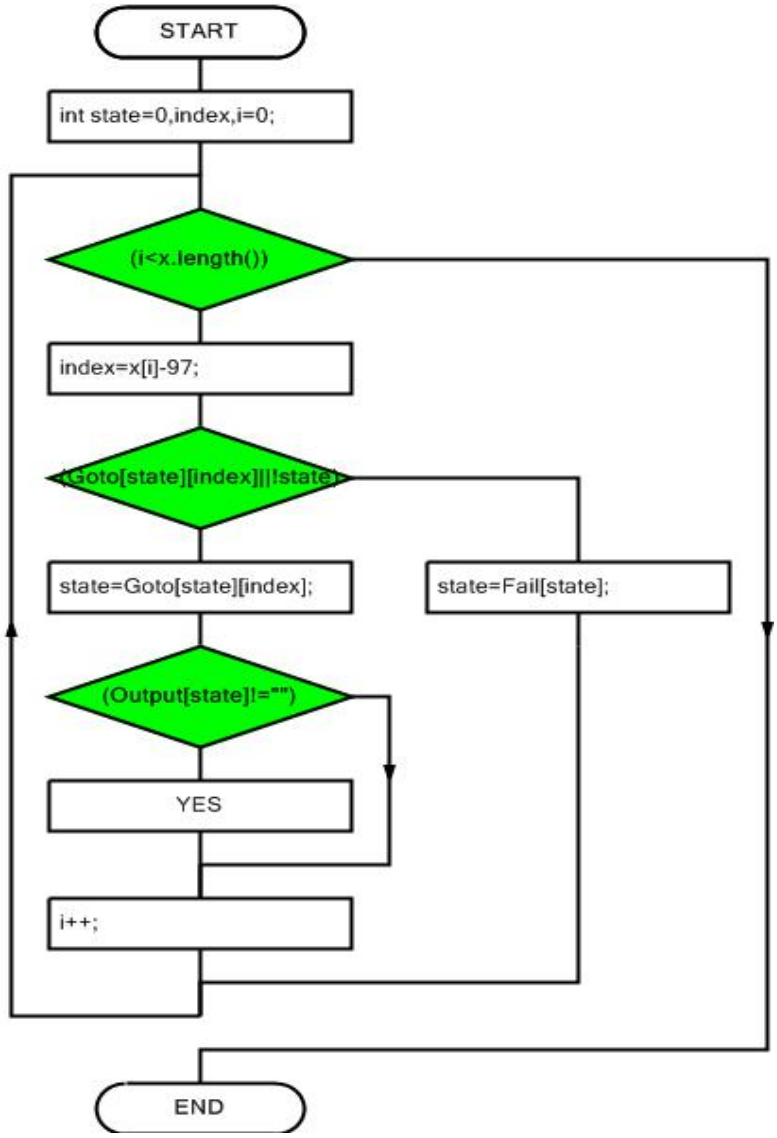
1.对于第 k 层中的每个状态 num=(int)D[k][i]，查询前级状态 prev 和输入字符 H(in)

2.根据公式 Fail[num]=Goto[Fail[prev]][F(H(in))]=Goto[Fail[prev]][in]更新 Fail 函数

3.若 Fail(s)=s'，将状态 s 的输出合并到状态 s'的输出

4.更新完第 k 层后，开始更新第 k+1 层

五、特征扫描流程：void Recognize(string x)



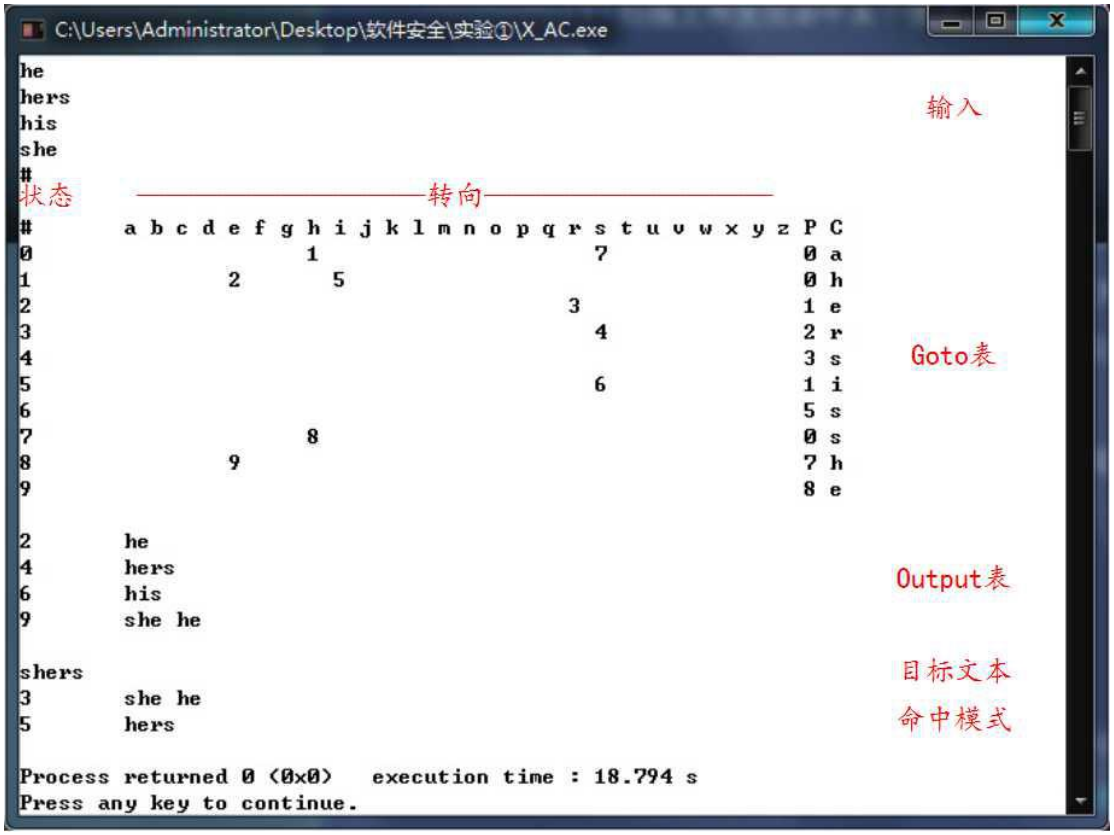
从目标文本 x 的下标 i=0 开始扫描，对于当前字符 y，有 index=F(y)

1. 若输入 y 后转向状态不为 0 或当前状态为 0，则
    1.1. 转向新状态 Goto[state][index]
    1.2. 若状态 state 有输出，则输出 Output[state]
    1.3. 下标 i=i+1

2. 否则根据失效函数进入新状态，即

3. 重复 1、2 直到读完该文本

六、实验结果



七、代码

```
#include<iostream>
#include<string.h>
#define M 20//State Number
using namespace std;
int Goto[M][28],Fail[M],Top=0;
string Output[M],D[M];

void Reset();/*初始化*/
void AdGoto(string x);/*更新 Goto&&Output&&D*/
void AdOutput();/*更新 Fail&&Output*/
void Recognize(string x);/*匹配*/
void PrintGoto();/*输出 Goto*/

int main()
{   Reset();
    string x;
    cin>>x;
    while(x[0]!='#')
    {   AdGoto(x);
```

```
        cin>>x;}
    PrintGoto();
    AdOutput();
    cout<<"\n\n";
    cin>>x;
    Recognize(x);}

void Reset()/*初始化*/
{   for(int i=0;i<M;i++)
    {   for(int j=0;j<28;j++)
            Goto[i][j]=0;
        Fail[i]=0;
        Output[i]=D[i]="";}
    D[0]+='0';}

void AdGoto(string x)/*更新Goto&&Output&&D*/
{   int len=x.length(),next=0;
    for(int i=0;i<len;i++)
    {   int index=x[i]-97;/*char[26]->int[26]*/
        if(!Goto[next][index])
        {   Goto[next][index]=++Top;
            Goto[Top][26]=next;/*前一状态*/
            Goto[Top][27]=index;/*输入字符*/}
        next=Goto[next][index];
        char num=next+48;/*int->(char)int*/
        if(D[i+1].find(num)==D[i+1].npos)/*记录深度*/
            D[i+1]+=num;}
    Output[next]+=x;/*更新输出*/}

void AdOutput()/*更新Fail&&Output*/
{   int k=2,num,prev,in;
    while(D[k]!="")
    {   for(int i=0;i<D[k].length();i++)
        {   num=D[k][i]-48;/*char->int*/
            prev=Goto[num][26];/*前级状态*/
            in=Goto[num][27];/*输入字符*/
            Fail[num]=Goto[Fail[prev]][in];
            if(Output[Fail[num]]!="")
                Output[num]+=(" "+Output[Fail[num]]);}
        k++;}
    cout<<'\n';
    for(int i=0;i<=Top;i++)
        if(Output[i]!="")
            cout<<'\n'<<i<<'\t'<<Output[i];}
```

```
void Recognize(string x)/*匹配*/
{   int state=0,index,i=0;
    while(i<x.length())
    {   index=x[i]-97;/*char[26]->int[26]*/
        if(Goto[state][index]||!state)
        {   state=Goto[state][index];
            if(Output[state]!="")
                cout<<i+1<<'\t'<<Output[state]<<'\n';
            i++;}
        else
            state=Fail[state];}}

void PrintGoto()/*输出 Goto*/
{   cout<<"\n#\ta b c d e f g h i j k l m n o p q r s t u v w x y z P C";
    for(int i=0;i<=Top;i++)
    {   cout<<'\n'<<i<<'\t';
        for(int j=0;j<26;j++)
        {   if(Goto[i][j])
                cout<<Goto[i][j]<<' ';
            else
                cout<<"  ";}
            cout<<Goto[i][26]<<' '<<(char)(Goto[i][27]+97);}}
```