

哈尔滨工业大学

软件安全课程实验报告

实验三

## 学院：计算机学院

## 班级：1403202

## 学号：1140320206

## 姓名：霍峻杰

# 实验四 恶意软件检测方法

1. 实验内容
2. 基于特征匹配的误用检测技术原理和方法
3. 掌握多模式匹配方法
4. 掌握基于双数组-自动机原理的多模式匹配原理
5. 基于双数组-AC算法的多模式特征匹配算法实现
6. 双数组-自动机的预处理：Next表、Base表、Check表、失效函数、输出函数构建
7. 双数组-自动机的特征扫描流程
8. 合理的数据结构
9. 利用构建的自动机扫描目标文件

（1）扫描文件发现命中某个模式，需明确给出命中的模式和该模式在文件中的位置

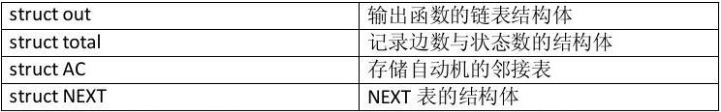
（2）如命中多个模式，需全部列出

1. 实验要求
2. 实验数据准备。实验数据应简单实用：利用ASCII字符集做为输入集，不考虑多字节编码的中文、英文字符集。模式和待匹配文本可以只包含英文。
3. 程序的输入部分（模式、待匹配文本）需以文件方式提供或者其它非固定的输入途径。
4. 实验结果和实验数据一起给出：包括设定的模式有哪些？Next表、Base表、Check表是什么？待匹配文本是什么？输出结果是什么。注意：仅给出匹配结果是不够的，必须在输入多模式后，给出Next表、Base表、Check表内容，Next表中空间占用的百分比(Next表中状态占用的空间/Next表大小)，然后输入待检测文本，再输出检测结果。
5. 程序本身需要提交。
6. 实验结果

1. 设计的算法流程图



2. 关键的数据结构





3.数据结构及各函数解释

1. 转向函数

转向函数的生成是NEXT ,BASE,CHECK 三个表协作来实现的来实现。循环查找所有head，找到next 不为NULL（不是叶节点），算出NEXT 。如果找到的next==null（是叶节点），就找下一个状态。

NEXT 表：

从根节点开始创建，同时计算BASE 表中的值，并填充CHECK 表中的值，每一层结束后在遍历左侧的邻接表进行下一层的NEXT 表创建，直到全部状态创建完成为止。Next 为转向函数表（数组、链表），下标是位置偏移量，输出是状态值。

Base 表：

下标是状态值，输出是Base 值。Next 表中当前状态为s ，输入为c 时，假设应跳转为状态t ，状态t 在Next 表中的位置=状态S 的位置+状态S 的Base 值+输入c 的ASCII 码值。

Check 表：

下标是状态值，输出是下标状态的父状态的值。在创建的过程中，每一层的状态可能要作为下一层状态的父状态，因此在邻接表中记录NEXT 中的pos ，存储在AC 中，方便最后特征码的检测。

2. 失效函数

Step1：state = f(r)。

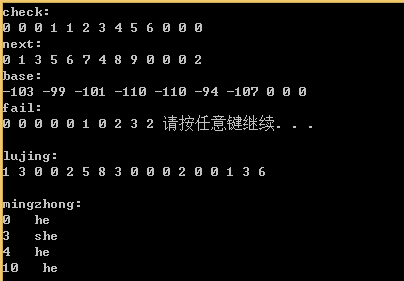
Step2：f(s) = g(state, a)

其中a 根据CHECK 表得出，并现先行确定初始状态和第一层失效状态的失效函数为f(s)=0。

3. 输出函数

输出函数以链表的方式储存。要求遍历所有从根状态到叶状态的通路，在每个分支节点处将此状态压栈，在遍历完这一个分之后弹栈，并遍历此状态的另一分支，同时在进行到每一节点的同时将字符压栈，并与模式集进行比较，若存在此模式则链接到输出链表上。在每一个状态不仅要比较本身的字符串与模式集，还需要比较此状态的失效状态字符串与输出链表，最终创立一个单向链表，保存所有需要的输出函数。

4. 实验结果



5. 程序源码

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <cstring>

#include <cstdio>

#include <cstdlib>

#include <queue>

#include <stack>

using namespace std;

struct node

{

int currentCharacter;

int fatherCharacter;

} modelSet[100][100];

struct nextNode

{

int state;

int character;

} next[256];

struct outputNode

{

int state;

char str[100];

bool available;

} output[256];

struct zhongyu

{

int pos;

int state;

}result[100];

queue<int>q;

int characterAppear[256];

int base[256];

int check[256];

int fail[256];

int currentState = 0;

int fatherState = 0;

int readyToInsert[256];

int index = 0;

int curNodeNum = 0;

int nextNodeNum = 0;

int inputNum;

char input[100][100];

int tongji = 0;

int currentPos;

void buildTable();

int gotoFunc(int state, int c);

void buildFailTable();

void buildOutputTable();

int main()

{

freopen("C:/Users/lovebear96/Desktop/1.txt", "r", stdin);

memset(input, 0, sizeof(input));

scanf("%d", &inputNum);

int rowLen;

int maxRowLen = -1;

for (int i = 0; i<inputNum; i++)

{

scanf(" %s", input[i]);

rowLen = strlen(input[i]);

if (rowLen>maxRowLen) maxRowLen = rowLen;

for (int j = 0; j < rowLen; j++)

{

modelSet[i][j].currentCharacter = input[i][j];

//printf("cur=%c ",modelSet[i][j].currentCharacter);

if (!j) modelSet[i][j].fatherCharacter = 0;

else modelSet[i][j].fatherCharacter = modelSet[i][j - 1].currentCharacter;

//printf("fat=%c\n",modelSet[i][j].fatherCharacter);

}

}

memset(characterAppear, 0, sizeof(characterAppear));

for (int i = 0; i < inputNum; i++)

{

characterAppear[modelSet[i][0].currentCharacter] = 1;

}

for (int j = 0; j < 256; j++)

{

if (characterAppear[j])

{

q.push(j);

readyToInsert[index++] = j;

}

}

curNodeNum = index;

buildTable();

int k = 1;

while (!q.empty())

{

memset(characterAppear, 0, sizeof(characterAppear));

int qhead = q.front();

q.pop();

curNodeNum--;

for (int i = 0; i < inputNum; i++)

{

if (modelSet[i][k].currentCharacter == 0) continue;

if (qhead == modelSet[i][k].fatherCharacter)

{

characterAppear[modelSet[i][k].currentCharacter] = 1;

}

}

for (int j = 0; j < 256; j++)

{

if (characterAppear[j])

{

q.push(j);

readyToInsert[index++] = j;

nextNodeNum++;

}

}

buildTable();

if (!curNodeNum)

{

curNodeNum = nextNodeNum;

k++;

nextNodeNum = 0;

}

}

printf("check:\n");

for (int i = 0; i <= 12; i++)

{

printf("%d ", check[i]);

}

printf("\nnext:\n");

for (int i = 0; i <= 12; i++)

{

printf("%d ", ::next[i].state);

}

printf("\nbase:\n");

for (int i = 0; i <= currentState; i++)

{

printf("%d ", base[i]);

}

buildFailTable();

printf("\nfail:\n");

for (int i = 0; i <= currentState; i++)

{

printf("%d ", fail[i]);

}

buildOutputTable();

system("pause");

printf("\nlujing:\n");

//char testchar[100]="yasherhs";

char testchar[100];

getchar();

fgets(testchar,100,stdin);

int nowState = gotoFunc(0, testchar[0]);

printf("%d ", nowState);

result[tongji].state = nowState;

result[tongji++].pos = 1;

for (currentPos = 1; currentPos < (int)strlen(testchar); currentPos++)

{

nowState = gotoFunc(nowState, testchar[currentPos]);

printf("%d ", nowState);

result[tongji].state = nowState;

result[tongji++].pos = currentPos + 1;

}

printf("\n\nmingzhong:\n");

for (int j = 0; j < tongji; j++)

{

for (int i = 0; i <= currentState; i++)

{

if (output[i].available == 1 && output[i].state == result[j].state)

printf("%d %s\n", result[j].pos - strlen(output[i].str), output[i].str);

}

}

fclose(stdin);

return 0;

}

void buildTable()

{

if (!index)return;

int j = 1;

for (; j < 256; j++)

{

if (!::next[j].state) break;

}

base[fatherState] = j - (readyToInsert[0]);

//adjust base[]

bool flag = 1;

while (flag)

{

int p = 0;

for (; p < index; p++)

{

if ((::next[base[fatherState] + readyToInsert[p]].state) != 0) break;

}

if (p == index) flag = 0;

else base[fatherState]++;

}

//deal check & next table

for (int i = 0; i < index; i++)

{

::next[base[fatherState] + readyToInsert[i]].state = ++currentState;

::next[base[fatherState] + readyToInsert[i]].character = readyToInsert[i];

check[currentState] = fatherState;

}

memset(readyToInsert, 0, sizeof(0));

index = 0;

fatherState++;

}

void buildFailTable()

{

for (int i = 0; i <= currentState; i++)

{

if (!check[i]) continue;

else

{

for (int j = 0; j < 256; j++)

{

if (::next[j].state == i)

{

fail[i] = gotoFunc(fail[check[i]], ::next[j].character);

// printf("\ni=%d j=%d %d %c\n",i,j,fail[check[i]],next[j].character);

break;

}

}

}

}

}

int gotoFunc(int state, int c)

{

int t = ::next[base[state] + c].state;

if (check[t] == state) return t;

else if (state == 0) return 0;

else

{

printf("%d ", fail[state]);

result[tongji].state = fail[state];

result[tongji++].pos = currentPos;

return gotoFunc(fail[state], c);

}

}

void buildOutputTable()

{

char keng1[256];

for (int i = 1; i <= currentState; i++)

{

memset(keng1, 0, sizeof(keng1));

output[i].state = i;

int temp = i;

int num = 0;

while (temp)

{

for (int j = 0; j < 256; j++)

{

if (temp == ::next[j].state)

{

keng1[num++] = ::next[j].character;

break;

}

}

temp = check[temp];

}

int k = 0;

for (int p = num - 1; p >= 0; p--)

{

output[i].str[k++] = keng1[p];

}

for (int q = 0; q < inputNum; q++)

{

if (!strcmp(input[q], output[i].str))

{

output[i].available = 1;

break;

}

}

}}