

《数据结构与算法分析》

实验报告

实验名称 基于字符串模式匹配算法的病毒感染检测问题

姓名 曾庆文

学号 23060218

院系 自动化学院

专业 人工智能

班级 23061011

实验时间 2024.4

一、实验目的

1. 掌握字符串的顺序存储表示方法。
2. 掌握字符串模式匹配算法 BF 算法和 KMP 算法的实现。

二、实验设备与环境

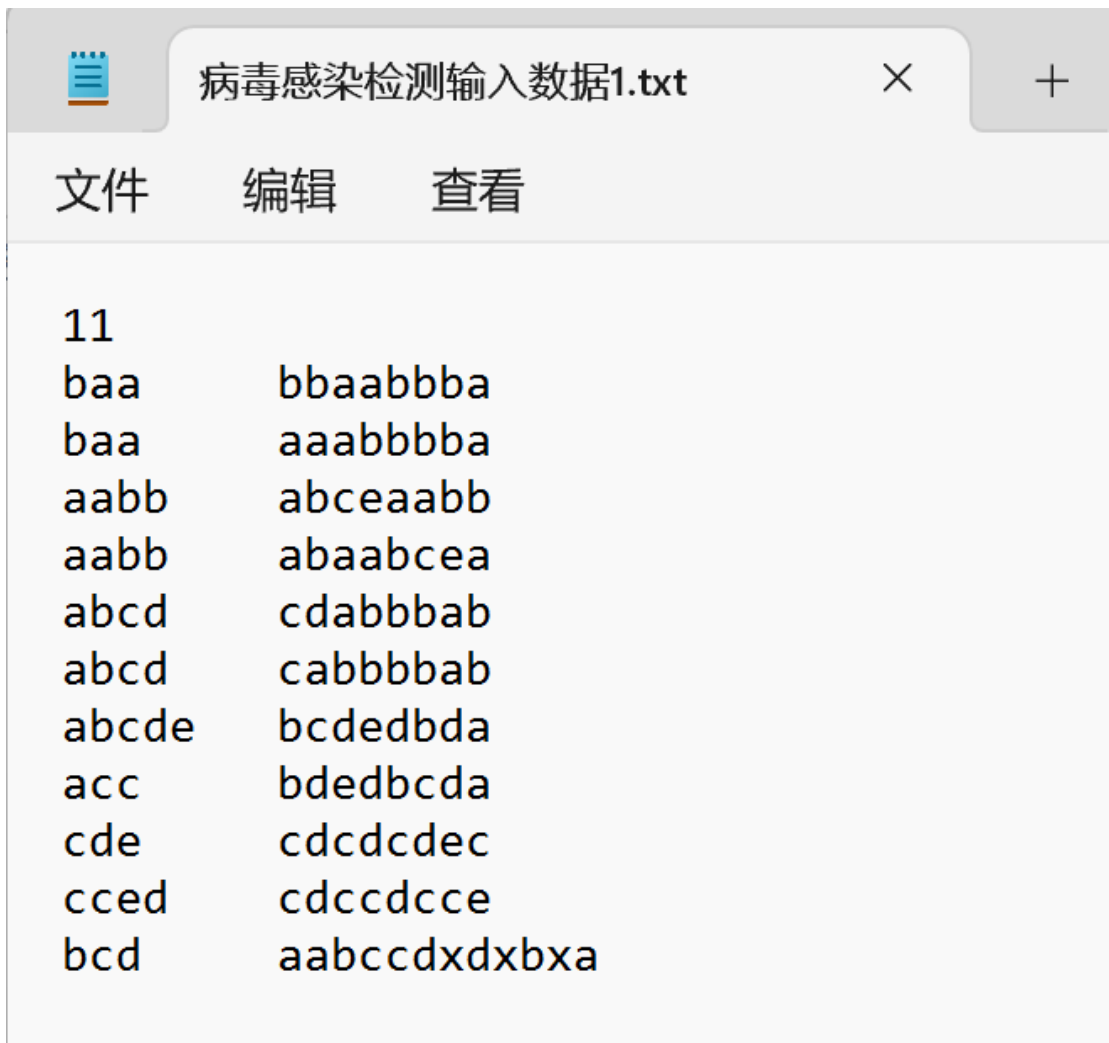
编辑器：Visual Studio Code

编译器：gcc

三、实验内容与结果

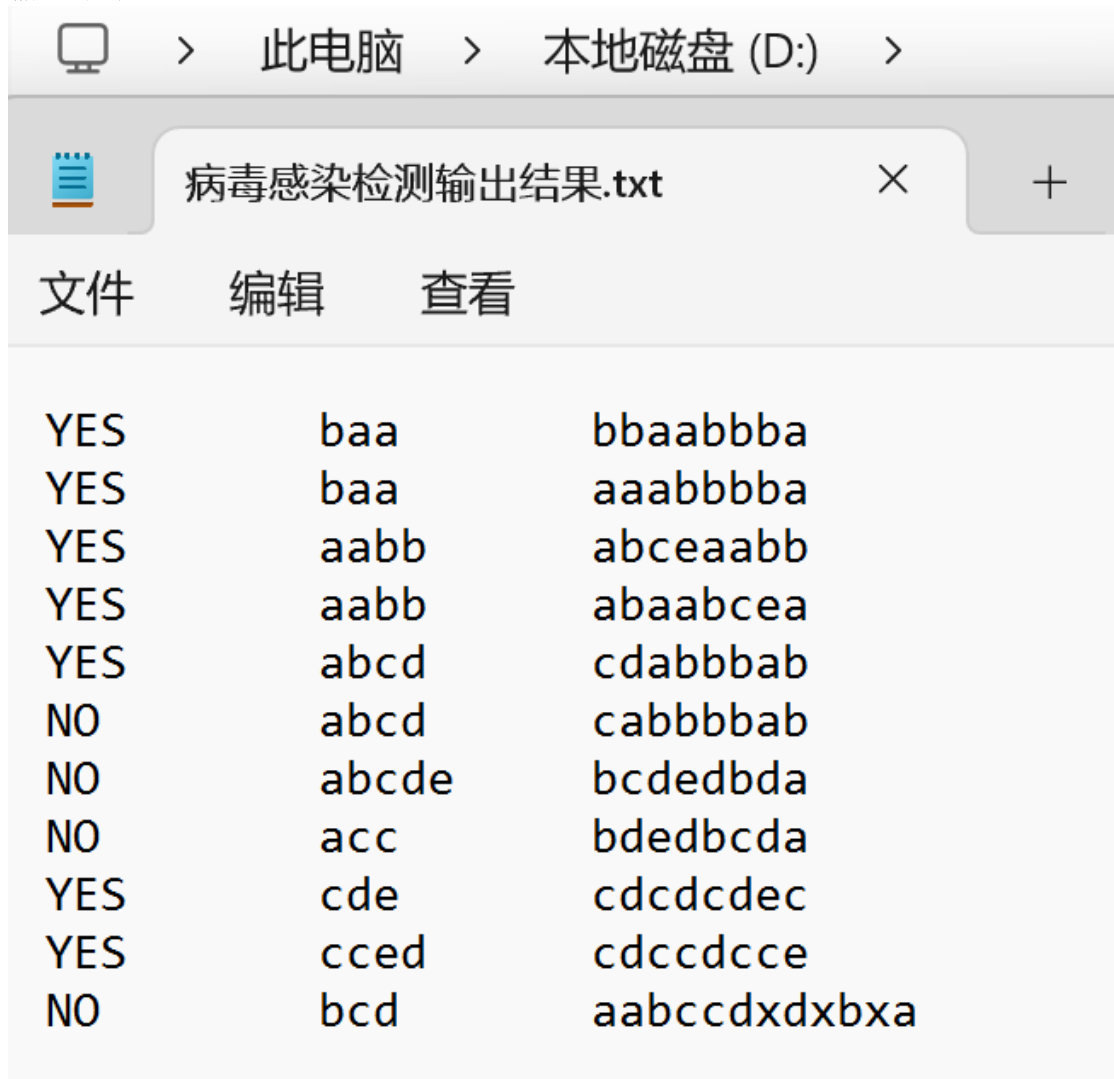
基本操作的输入输出结果如下：

1. 输入数据 1:

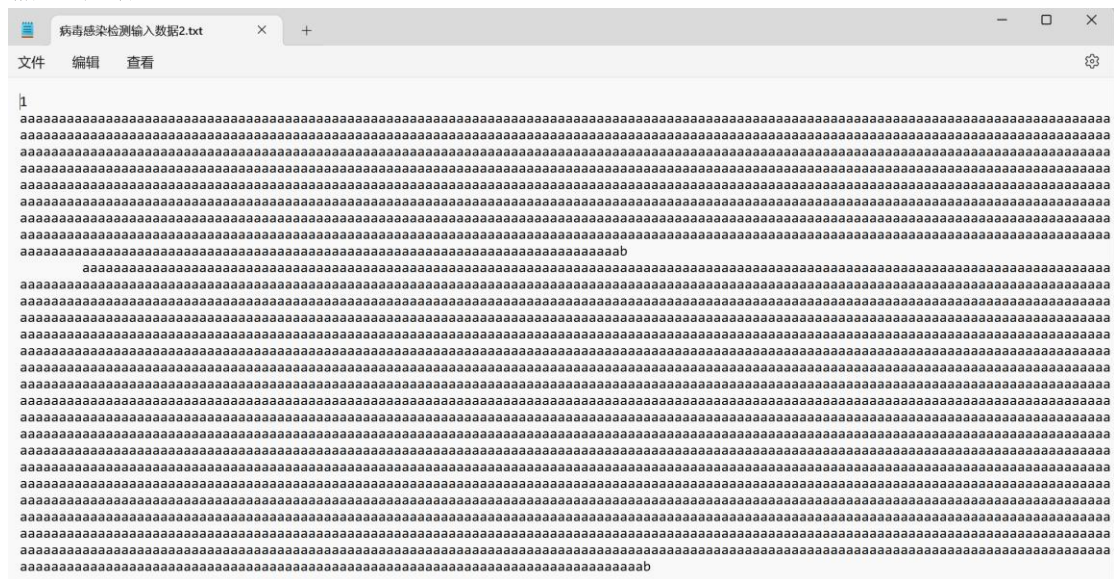


```
11
baa      bbaabbba
baa      aaabbbba
aabb     abceaabb
aabb     abaabcea
abcd     cdabbbab
abcd     cabbbbab
abcde    bcdedbda
acc      bdedbcda
cde      cdcdcdec
cced     cdccdcce
bcd      aabccdxdbxa
```

输出结果:



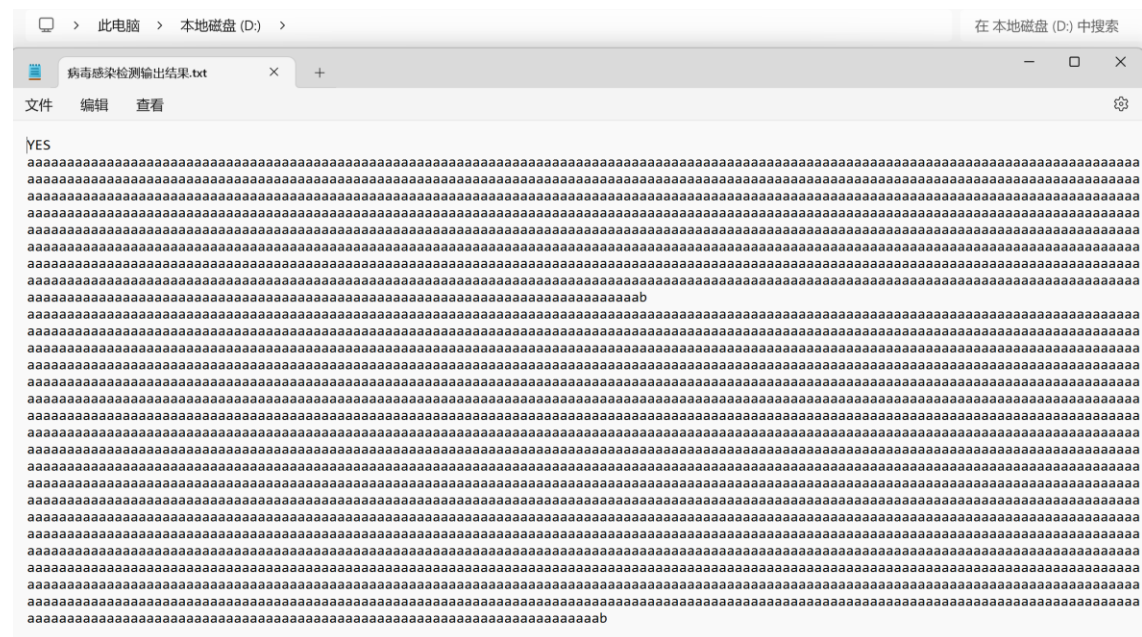
2. 输入数据 2:



输出结果:

注: (1).由于输入数据过大,为防止数组越界,故运行程序前需将 BF 算法和 KMP 算法代码第四行改为#define MAXLEN 2600。

(2).为保证输出结果美观,将 BF 算法 85-93 行和 KMP 算法 110-118 行每次输出后加上换行符。



实验全部代码如下：

(一)、BF 算法：

```
1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <string.h>
4. #define MAXLEN 255
5.
6. //串的定长顺序存储
7. typedef struct
8. {
9.     char ch[MAXLEN+1];
10.    int length;
11. }SString;
12.
13. //BF 算法
14. int Index_BF(SString S,SString T,int pos)
15. {
16.     int i=pos;
17.     int j=0;
18.     //课本代码从1开始作为下标计数，本程序从0开始
19.     while(i<S.length && j<T.length)
20.     {
21.         if(S.ch[i]==T.ch[j]){
22.             i++;
23.             j++;
24.         }
25.         else
26.         {
27.             i=i-j+1;
28.             j=0;
29.         }
30.     }
31.     if(j>=T.length)
32.     {
33.         return i-T.length+1;
34.     }
35.     else
36.     {
37.         return 0;
38.     }
39. }
40. //利用BF算法实现病毒检测
41. int main()
```

```

42.{
43.    SString Virus;
        //病毒的DNA 序列
44.    SString Person;
        //人的DNA 序列
45.    SString temp;
        //暂存病毒DNA 每次的环状序列
46.    char Vir[MAXLEN];
        //暂存病毒的初始DNA 序列
47.    int flag=0;
        //用来标识是否匹配, 初始为0, 匹配后为非0
48.    int m;
        //病毒DNA 序列的长度
49.    int num;
        //待检测的任务数
50.    FILE *inFile=fopen("D:\\病毒感染检测输入数据
    1.txt","a+");        //输入的数据
51.    FILE *outFile=fopen("D:\\病毒感染检测输出结
    果.txt","w+");        //输入的数据
52.    rewind(inFile);
53.    fscanf(inFile,"%d",&num);
54.    while (num--)
55.    {
56.        fscanf(inFile,"%s",Virus.ch);
            //读取病毒的序列
57.        fscanf(inFile,"%s",Person.ch);
            //读取人的序列
58.        Virus.length=strlen(Virus.ch);
59.        Person.length=strlen(Person.ch);

60.        temp.length=strlen(Virus.ch);
61.        strcpy(Vir,Virus.ch);
62.        flag=0;
63.        m=Virus.length;
64.        for(int i=m,j=0;j<m;j++)
65.        {
66.            Virus.ch[i]=Virus.ch[j];
                //将病毒字符串长度扩大两倍
67.            i++;
68.        }
69.        Virus.ch[2*m]='\0';
            //添加结束符
70.        for(int i=0;i<m;i++)
            //取每个长度为m 的病毒DNA 环状串 temp

```

```

71.     {
72.         for(int j=0;j<m;j++)
73.         {
74.             temp.ch[j]=Virus.ch[i+j];
75.         }
76.         temp.ch[m]='\0';
           //添加结束符
77.         flag=Index_BF(Person,temp,0);
           //BF 算法模式匹配
78.         if(flag)
79.         {
80.             break;
81.         }
82.     }
83.     if(flag)
84.     {
85.         fprintf(outFile,"%-10s","YES");
86.         fprintf(outFile,"%-10s",Vir);
87.         fprintf(outFile,"%-10s\n",Person.ch);
88.     }
89.     else
90.     {
91.         fprintf(outFile,"%-10s","NO");
92.         fprintf(outFile,"%-10s",Vir);
93.         fprintf(outFile,"%-10s\n",Person.ch);
94.     }
95. }
96. }

```

(二)、KMP 算法:

```

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <string.h>
4. #define MAXLEN 255
5.
6. //串的定长顺序存储
7. typedef struct
8. {
9.     char ch[MAXLEN+1];
10.    int length;

```

```

11. }SString;
12.
13. int next[MAXLEN];
14.
15. //计算next 数组函数值
16. void get_next(SString T,int next[])
17. {
18.     int i=0;
19.     int j=-1;
20.     next[0]=-1;
21.     while (i<T.length-1)
22.     {
23.         if(j==-1 || T.ch[i]==T.ch[j])
24.         {
25.             i++;
26.             j++;
27.             next[i]=j;
28.         }
29.         else
30.         {
31.             j=next[j];
32.         }
33.     }
34. }
35.
36. //KMP 算法
37. int Index_KMP(SString S,SString T,int pos)
38. {
39.     int i=pos;
40.     int j=0;
41.     while (i<S.length && j<T.length)
42.     {
43.         if(j==-1 || S.ch[i]==T.ch[j])
44.         {
45.             i++;
46.             j++;
47.         }
48.         else
49.         {
50.             j=next[j];
51.         }

```



```

52.     }
53.     if(j>=T.length)
54.     {
55.         return i-T.length+1;
56.     }
57.     else
58.     {
59.         return 0;
60.     }
61. }
62.
63. // 利用 KMP 算法实现病毒检测
64. int main()
65. {
66.     memset(next, 0, sizeof(next));
67.     SString Virus;
        // 病毒的 DNA 序列
68.     SString Person;
        // 人的 DNA 序列
69.     SString temp;
        // 暂存病毒 DNA 每次的环状序列
70.     char Vir[MAXLEN];
        // 暂存病毒的初始 DNA 序列
71.     int flag=0;
        // 用来标识是否匹配, 初始为 0, 匹配后为非 0
72.     int m;
        // 病毒 DNA 序列的长度
73.     int num;
        // 待检测的任务数
74.     FILE *inFile=fopen("D:\\病毒感染检测输入数据
        1.txt", "a+");        // 输入的数据
75.     FILE *outFile=fopen("D:\\病毒感染检测输出结
        果.txt", "w+");        // 输入的数据
76.     rewind(inFile);
77.     fscanf(inFile, "%d", &num);
78.     while (num--)
79.     {
80.         fscanf(inFile, "%s", Virus.ch);
        // 读取病毒的序列
81.         fscanf(inFile, "%s", Person.ch);
        // 读取人的序列
82.         Virus.length=strlen(Virus.ch);
83.         Person.length=strlen(Person.ch);

```

```

84.         temp.length=strlen(Virus.ch);
85.         strcpy(Vir,Virus.ch);
86.         flag=0;
87.         m=Virus.length;
88.         for(int i=m,j=0;j<m;j++)
89.         {
90.             Virus.ch[i]=Virus.ch[j];
91.             //将病毒字符串长度扩大两倍
92.             i++;
93.         }
94.         Virus.ch[2*m]='\0';
95.         //添加结束符
96.         for(int i=0;i<m;i++)
97.         {
98.             //取每个长度为m的病毒DNA环状串temp
99.             temp.ch[j]=Virus.ch[i+j];
100.        }
101.        temp.ch[m]='\0';
102.        //添加结束符
103.        get_next(temp, next);
104.        flag=Index_KMP(Person,temp,0);
105.        //KMP 算法模式匹配
106.        if(flag)
107.        {
108.            break;
109.        }
110.    }
111.    if(flag)
112.    {
113.        fprintf(outFile,"%-10s","YES");
114.        fprintf(outFile,"%-10s",Vir);
115.        fprintf(outFile,"%-10s\n",Person.ch);
116.    }
117.    else
118.    {
119.        fprintf(outFile,"%-10s","NO");
120.        fprintf(outFile,"%-10s",Vir);
121.        fprintf(outFile,"%-10s\n",Person.ch);
122.    }
123. }

```

四、实验总结及体会

本次实验对我个人来说有以下几个难点：

1. 文件的输入与输出：文件相关知识点在上学期学完后使用得很少，而这次实验的输入输出都要用到文件，因此我花了一点时间复习文件。从文件指针的使用方式，到文件输入输出函数的选择和它们各自的参数，这些都要在实验中熟练掌握。
2. 病毒模式串的匹配问题：病毒的 DNA 序列是环状的，因此有多种排列方式，在模式匹配中所有排列方式都要表达清楚，否则会漏掉情况，导致得到错误的结果。这里我将病毒模式串复制一份接到原来串的后面，这样每次匹配拼接串的一部分就能将环状序列的所有情况考虑到。
3. 关于变量的初值问题：课本上的 BF 算法和 KMP 算法的代码都将模式串数组，next 数组的初始下标记为 1，但是我按照这种方法编写代码，所得到的输出结果却不完全正确。因此我将这些数组的初始下标还是按照 0 来计算，再将后续语句进行调整，得到了正确答案。

总结：总的来说这次实验并不是很难，但是 KMP 算法是这门课遇到的第一个较难理解的算法，加上文件的输入输出不够熟练，所以也是花了一番时间才完成这次实验。