学号 WA2214014 专业 人工智能 姓名 杨跃浙
实验日期 6 月 3 号 教师签字 成绩

实验报告

【实验名称】串、数组和广义表

【实验目的】

串是内容受限的线性表，它限定了表中的元素为字符。串有两种基本存储结构：顺序存储和链式存储，但多采用顺序储存结构。串的常用算法是模式匹配算法，主要有 BF 算法和 KMP 算法。掌握 BF 算法和 KMP 算法的具体实现，明确 KMP 对 BF 的改进之处，掌握 KMP 算法中 next 数组和 nextval 数组的计算方法。

【实验原理】

串的常用算法是模式匹配算法，主要有 BF 算法和 KMP 算法。KMP 算法中 next 数组和 nextval 数组的计算方法。

【实验内容】

1.目标串为“abcaabbabcabaacbacba”，模式串为“abcabaa”，分别采用BF算法和KMP算法寻找到模式串在目标串中首次出现的位置 KMP 算法中请分别计算和输出模式串的 next 函数值和 nextval 函数值

2.设计一个 KMP 算法实现病毒序列与宿王序列间的匹配，注意病毒序列是环形的，可在任意一个位置断裂

病毒序列是 CCCATGATCC

宿主序列是 ATCCGTACGAATCGATCCCCCATGC

```
#include <iostream>
#include <cstring>
#pragma warning (disable:4996)
using namespace std;
#define MAXLEN 255
#define OK 1
#define ERROR 0
#define OVERFLOW -2
typedef int Status;
typedef struct
{
	char ch[MAXLEN + 1];
	int length;
}SString;
int nextp[MAXLEN + 1] = { 0 };
int nextval[MAXLEN + 1] = { 0 };
Status StringAssign(SString& S, const char* str)
{
	strcpy(S.ch + 1, str);
	S.length = strlen(str);
	return OK;
}
int Index_BF(SString S, SString T, int pos)
{
	int i = pos; int j = 1;
	while (i <= S.length && j <= T.length)
	{
		if (S.ch[i] == T.ch[j])
		{
```

```
                ++i;
                ++j;
            }
            else
            {
                i = i - j + 2;
                j = 1;
            }
        }
        if (j > T.length)
            return i - T.length;
        else return 0;
}
void next_get(SString T,int next[])
{
        int i = 1; next[1] = 0; int j = 0;
        while (i < T.length)
        {
            if (j == 0 || T.ch[i] == T.ch[j])
            {
                ++i;
                ++j;
                next[i] = j;
            }
            else
                j = next[j];
        }
}
void nextval_get(SString T, int nextval[])
{
        int i = 1; nextval[1] = 0; int j = 0;
        while (i < T.length)
        {
            if (j == 0 || T.ch[i] == T.ch[j])
            {
                ++i;
                ++j;
                if (T.ch[i] != T.ch[j])
                    nextval[i] = j;
                else nextval[i] = nextval[j];
            }
            else j = nextval[j];
        }
}
int Index_KMP(SString S, SString T, int pos)
```

```
{
    int i = pos; int j = 1;
    while (i <= S.length && j <= T.length)
    {
        if (j == 0 || S.ch[i] == T.ch[j])
        {
            i++;
            j++;
        }
        else
            j = nextp[j];
    }
    if (j > T.length)
        return i - T.length;
    else return 0;
}
void print(int len)
{
    for (int i = 1; i <= len;i++)
        cout << "next[" << i << "]=" << nextp[i] << "\t";
    cout << endl;
    for (int i = 1; i <= len;i++)
        cout << "nextval[" << i << "]=" << nextval[i] << "\t";
    cout << endl;
}
void strmcpy(SString &P, SString &S, int m)
{
    for (int i = 1; i<=P.length; i++)
    {
        S.ch[i]=P.ch[m];
        m++;
    }
    S.length = P.length;
}
void Matching()
{
    SString T, P,S;
    StringAssign(T, "ATCCGTACGAATCGATCCCCCATGC");
    StringAssign(P, "CCCATGATCC");
    StringAssign(S, "");
    for (int i = 1; i <= P.length; i++)
        P.ch[i + P.length] = P.ch[i];
    int i;
    for (i = 1; i <= P.length; i++)
    {
```

```
            strmcpy(P, S, i);
            next_get(S, nextp);
            nextval_get(S, nextval);
            if (Index_KMP(T, S, 1))
            {
                cout <<"Yes!" << endl<<Index_KMP(T, S, 1) << endl;
                break;
            }
        }
        if (i == P.length + 1) cout << "No!";
}
int main()
{
        SString Target, Pattern;
        StringAssign(Target, "abcaabbabcabaacbacba");
        StringAssign(Pattern, "abcabaa");
        cout << Index_BF(Target, Pattern, 1) << endl;
        next_get(Pattern,nextp);
        nextval_get(Pattern, nextval);
        print(Pattern.length);
        cout << Index_KMP(Target, Pattern, 1) << endl;
        Matching();
}
```

【小结或讨论】

通过该次实验我掌握了串的模式匹配算法，主要有 BF 算法和 KMP 算法，并能够应用模式匹配算法解决实际问题，如病毒序列与宿王序列间的匹配等。