**南京理工大学计算机科学与工程学院**

**智能科学技术导论 作业**

**班 级 9181069502**

**学生姓名 黄海浪**

**学 号 9181040G0818**

**起止时间 2020.05.12-2020.05.13**

**教 师 袁夏**

目录

[一. 问题描述与作业要求 2](#_Toc40502039)

[问题1 2](#_Toc40502040)

[问题2 2](#_Toc40502041)

[二. 问题分析与流程设计 2](#_Toc40502042)

[对于问题1 2](#_Toc40502043)

[对于问题2 3](#_Toc40502044)

[三. 代码与结果 5](#_Toc40502045)

[问题1代码 5](#_Toc40502046)

[问题1结果截图 6](#_Toc40502047)

[问题2代码 6](#_Toc40502048)

[问题2结果截图 9](#_Toc40502049)

## 问题描述与作业要求

### 问题1

实现 揲蓍成卦 算法。

揲蓍成卦 算法：揲蓍（shé shī）古代问卜的一种方式，用手抽点蓍草茎的数目，以决定吉凶祸福。其法为，大衍之数五十，其用四十有九，分而为二以象两，挂一以象三，揲之以四以象四时，归奇于劫以象闰，五岁再闰，故再营而后卦。是故四营而成易（爻），十有八变而成卦。八卦而小成，引而伸之，触类而长揲，天下之能事毕矣。

### 问题2

如何构造一个能够计算一个字符串在另一个字符串中出现次数的算法，请用伪码来表示所给出的算法，并分析算法的计算复杂性。

## 问题分析与流程设计

### 对于问题1

**1.算法文字说明：**

如果强调精确性，那么实际卦象生成算法如下（重复如下三变步骤之六遍，共计十八变）：

（1）大衍之数50，其用49：50根蓍草，去其1而不用，令可用之数s=49，余数之和t=0，循环如下操作三变：

（2）分二（分而为二以象两）：余49根随意分为左右两堆（天意）

（3）挂一（挂一以象三）：从右堆中取出1根

（4）揲四（揲之以四以象四时）：对左右两堆均以四根一组数之，令其为新的可用之数s

（5）归奇（归奇于扐以象闰）：t=t+两堆得出的余数之和

（6）（故四营而成易）：不到三变转（2）

（7）根据y=（49-t）/4来决定一个爻画：y=7（少阳）、8（少阴）、9（老阳）、6（老阴）

**2.算法流程图：**



**3.算法分析：**

\*分析后 最后s有4种情况 36、32、28或24

\*将这4个结果分别除以4，所得的商分别是9、8、7、6

\*9为老阳（概率为3/16），6为老阴（概率为1/16）

\*7为少阳（概率为5/16），8为少阴（概率为7/16）

### 对于问题2

**1.算法文字说明：**

算法要求获得子字符串在主字符串中出现次数，那么可以使用KMP模式匹配算法，每次获得第一次出现的位置，然后进行计数，之后再将主字符串拆分出来再次使用KMP算法。

**2.算法流程图：**



**3.算法伪代码：**

算法getCnt(str,strt)

输入：主字符串str，子串strt

输出：子串strt在主字符串str出现的次数

1. index <- KMP(str,strt)       //获得strt第一次在str出现的位置

2. cnt <- 0                     //次数0

3. while(index != -1)   then    //如果有出现

4.      cnt <- cnt + 1          //+1

5.      str <- str[index:str.Length]    //主字符串改变

6.      index <- KMP(str,strt)          //重新计算

7. end while

8. return cnt

**4.算法分析：**

一次KMP算法时间复杂度

设str长度为n, strt的长度为m

**Next数组的求取**：为O(m)

**总的时间复杂度为**：O(n+m)

由于问题规模改变 出现次数是不定的 但是出现次数不可能大于n

故 时间复杂度为O(n+m)

## 代码与结果

### 问题1代码

#include<iostream>

#include<stdlib.h>

#include<time.h>

#include<string>

using namespace std;

const string ways[4] = {"老阴","少阳","少阴","老阳"};

int main() {

    srand(time(0));

    int s = 50 - 1;

    for(int i=0; i<3; ++i) {

        //左边一堆 右边一堆 （避免有一边为0，不然右边为0，0-1=-1）

        int left = (rand()%(s-1)) + 1 , right = s - left - 1;

        //计算余数

        int t\_l = left%4 , t\_r = right%4;

        //检测是否有余数0，有的话令其为4 则 s -= 4

        t\_l || (s -= 4);

        t\_r || (s -= 4);

        //重新计算s s-=1 因为右边拿出1

        s -= t\_l + t\_r + 1;

    }

    /\*

     \*分析后 最后s有4种情况 36、32、28或24

     \*将这4个结果分别除以4，所得的商分别是9、8、7、6

     \*9为老阳（概率为3/16），6为老阴（概率为1/16）

     \*7为少阳（概率为5/16），8为少阴（概率为7/16）

     \*/

    cout<<ways[(s/4-6)]<<endl;

    system("pause");

    return 0;

}

### 问题1结果截图



### 问题2代码

#include<iostream>

#include<string>

using namespace std;

class strUtil {

    private:

        static void getNext(string str,int \*next);                  //获得 KMP所需 next

    public:

        static int strCmp(string str,string strt,int index = 0);    //KMP模式匹配

        static int getCnt(string str,string strt);

};

int main() {

    string strt = "123";

    string str = "qwe123w";

    cout<<"请输入子串: " ;

    while(cin.peek()=='\n'){

        cin.get();

    }

    getline(cin,strt);

    cout<<"请输入主串: " ;

    while(cin.peek()=='\n'){

        cin.get();

    }

    getline(cin,str);

    cout<<"输入的字符串分别为："<<endl<<strt<<endl<<str<<endl<<endl;

    cout<<"第一次出现位置："<<strUtil::strCmp(str,strt)<<endl;

    cout<<"出现总次数："<<strUtil::getCnt(str,strt)<<endl;

    system("pause");

    return 0;

}

int strUtil::getCnt(string str,string strt) {

    int len = strt.length(),cnt = 0,len2 = str.length();

    for(int index = strUtil::strCmp(str,strt); index!=-1 && index+len <= len2; index = strUtil::strCmp(str,strt,index+len)) {

        ++cnt;

    }

    return cnt;

}

int strUtil::strCmp(string str,string strt,int index) {

    int strLen = str.length() , strtLen = strt.length();//长度

    if(index>strLen) return -1;

    //将index--len个拷贝

    str.assign(str,index,strLen);

    int i = 0,j = 0; //i为主串index  j为子串回溯index

    //用于获取next

    int \*next = new int [strtLen];

    strUtil::getNext(strt,next);

    while(i<strLen && j<strtLen) {

        if(j == -1 || str[i] == strt[j]) {

            //不断匹配

            ++i;

            ++j;

        } else {

            //不同则下一个

            j = next[j];

        }

    }

    delete next;

    if(j>=strtLen) {

        return index + i - strtLen;

    } else {

        return -1;

    }

}

void strUtil::getNext(string str,int \*next) {

    int i = 0,j = -1;

    next[i] = -1;

    int len = str.length();

    while(i<len) {

        //匹配

        if(j==-1||str[i]==str[j]) {

            ++i;

            ++j;

            //相同则表明前面出现过

            if(str[i]==str[j]) {

                next[i] = next[j];

            } else {

                //未出现过 新的

                next[i] = j;

            }

        } else {

            // 回溯

            j = next[j];

        }

    }

}

### 问题2结果截图

