计算物理——Homework2

曾郅琛 PB20071431

摘要：利用C++语言解决以下问题：用16807产生器测试随机数序列中满足X(n-1)>X(n+1)>X(n)关系的比重并讨论Fibonacci延迟产生器中出现这种关系的比重。主要分为16807产生器比重计算和Fibonacci延迟产生器比重计算。

1 算法及实现

1.1 16807生成器及比重计算

在实验一的基础上，利用实验一的16807随机数生成器生成所需要的随机数，在函数`Sch_Test_weight(int N)`内，通过创建一个长度为3的动态数组，第一次读取前三个随机数，之后每一次多读取一个随机数，同时下一次比较时，将上一次的数组可用值传递给下一次使用：`tempArray[0]=tempArray[2]`,`tempArray[2]=tempArray[1]`.另外在每次比较时sum++，当满足X(n-1)>X(n+1)>X(n)关系时，num++，最后释放内存，返回num/sum比重。下为核心代码：

```
    int i =0,z= Seed();                          //生成种子
    int k[3]={0,2,1};            //n-1>n+1>n的排列，初始为0，2，1
    int sum=0,num=0;
    auto* tempArray = (double*)malloc(3 * sizeof(double));//创建动态数组存入每次比较的三个数字
    for(int j:k){                                                //初始三个数
        z= Sch_16807(z);
        tempArray[j]=double(z)/M;

    }
    if((tempArray[0]>tempArray[2])&&(tempArray[2]>tempArray[1])) num++;  //序号为n-1>n+1>n，计
数加一
    sum++;                                                                //每次比较加一
    for(i=3;i<N+2;i++){
        tempArray[0]=tempArray[2];                                      //继承上一次的n和n+1
        tempArray[2]=tempArray[1];
        z= Sch_16807(z);                                                 //生成新的随机数
        tempArray[1]=double(z)/M;
        if((tempArray[0]>tempArray[2])&&(tempArray[2]>tempArray[1])) num++; //序号为n-1>n+1>n
        sum++;                                                                    //同上
    }
 //测试数据    cout<<sum<<endl;
    free(tempArray);                                                     //释放内存
    return double(num)/sum;
```

1.2 Fibonacci延迟产生器及比重计算

Fibonacci延迟产生器描述如下，其中特殊符号可以为加、减、乘、XOR：

$$I_n = I_{n-p} \bigotimes I_{n-q} \ mod \ m$$

在此Fibonacci延迟产生器中，[p , q]表示延迟，即并非严格按Fibonacci数序列，在本次实验中将其作为变量传递到函数`fibonacci_dalayed(int N, int p, int q)`中执行。

另外，结合教材上给出的特殊的Fibonacci延迟产生器——带载减法产生器：

$$if(I_n > 0) : I_n = I_{n-p} - I_{n-q}$$
$$if(I_n <= 0) : I_n = I_{n-p} - I_{n-q} + 2^{32} - 5 - 1$$

取上述特殊符号为减法，进行算法运算。

同样的，首先利用Schrage生成器创建大小为100的动态数组，存入初始数值作为起始I[100]，之后迭代产生后续的I[n]，并判断是否小于0，从而加上设定的值O。每进行一次比较sum++，当满足X(n-1)>X(n+1)>X(n)关系时，num++，最后释放内存，返回num/sum比重。下为核心代码：

```
auto* tempArray = (long int*)malloc(100 * sizeof(long int));  //创建动态数组，存入初始数值
    double B[3]={0};
    for(j=0;j<100;j++){                                          //初始100个数，用来计算后续值大小
        z= Sch_16807(z);
        tempArray[j]=z;
    }
    for(i=0;i<N;i++){
        F= tempArray[99-p]-tempArray[99-q];
        if(F<0) F=F+O;                                           //o大小为2^32-5-1
        for(j=99;j>0;j--){tempArray[j]=tempArray[j-1];}
        tempArray[0]=F;
    //此时不用化为[0,1]内比大小,因为都是公约数
        if((tempArray[0]>tempArray[2])&&(tempArray[2]>tempArray[1])) num++; //序号为n-1>n+1>n
        sum++;                                                   //计数加加
    }
```

2 实验结果分析讨论

2.1 预期结果

由数学中的概率论知识可知，若三个数独立且任意大小关系，总共有3*2=6种大小关系，故出现X(n-1)>X(n+1)>X(n)关系的比重理论上应为1/6≈1.6666667，而鉴于生成随机数为伪随机数，存在一定的相关性，所以实验结果可能无法完全等于此值，必定存在偏差。

2.2 输入不同N，p，q计算两种算法的比重

```
while(true){
        cout<<"please input N,p,q: ";
        cin>>N>>p>>q;
        if(N!=0)
        {
            double a= s.Sch_Test_weight(N);
            double b= s.fibonacci_dalayed(N,p,q);
            cout<<"N= "<<N<<", 16807产生器满足序列比重:  "<<a<<endl;
            cout<<"N= "<<N<<", p= "<<p<<", q= "<<q<<", fibonacci产生器满足序列比重:  "
<<b<<endl;
            }
        else return 0;
    }
```

```
please input N,p,q: 100 22 43
N= 100, 16807产生器满足序列比重:  0.16
N= 100, p= 22, q= 43, fibonacci产生器满足序列比重:  0.15
please input N,p,q: 1000 22 43
N= 1000, 16807产生器满足序列比重:  0.156
N= 1000, p= 22, q= 43, fibonacci产生器满足序列比重:  0.167
please input N,p,q: 10000 22 43
N= 10000, 16807产生器满足序列比重:  0.1627
N= 10000, p= 22, q= 43, fibonacci产生器满足序列比重:  0.168
please input N,p,q: 100000 22 43
N= 100000, 16807产生器满足序列比重:  0.16586
N= 100000, p= 22, q= 43, fibonacci产生器满足序列比重:  0.16603
please input N,p,q: 1000000 22 43
N= 1000000, 16807产生器满足序列比重:  0.166794
N= 1000000, p= 22, q= 43, fibonacci产生器满足序列比重:  0.166385
please input N,p,q: 10000000 22 43
N= 10000000, 16807产生器满足序列比重:  0.166678
N= 10000000, p= 22, q= 43, fibonacci产生器满足序列比重:  0.166644
please input N,p,q: 100000000 22 43
N= 100000000, 16807产生器满足序列比重:  0.166642
N= 100000000, p= 22, q= 43, fibonacci产生器满足序列比重:  0.166683
```

```
please input N,p,q: 100 32 50
N= 100, 16807产生器满足序列比重:  0.2
N= 100, p= 32, q= 50, fibonacci产生器满足序列比重:  0.17
please input N,p,q: 1000 32 50
N= 1000, 16807产生器满足序列比重:  0.178
N= 1000, p= 32, q= 50, fibonacci产生器满足序列比重:  0.157
please input N,p,q: 10000 32 50
N= 10000, 16807产生器满足序列比重:  0.1696
N= 10000, p= 32, q= 50, fibonacci产生器满足序列比重:  0.1652
please input N,p,q: 100000 32 50
N= 100000, 16807产生器满足序列比重:  0.16648
N= 100000, p= 32, q= 50, fibonacci产生器满足序列比重:  0.16607
please input N,p,q: 1000000 32 50
N= 1000000, 16807产生器满足序列比重:  0.166786
N= 1000000, p= 32, q= 50, fibonacci产生器满足序列比重:  0.166894
please input N,p,q: 10000000 32 50
N= 10000000, 16807产生器满足序列比重:  0.166676
N= 10000000, p= 32, q= 50, fibonacci产生器满足序列比重:  0.166778
please input N,p,q: 100000000 32 50
N= 100000000, 16807产生器满足序列比重:  0.166609
N= 100000000, p= 32, q= 50, fibonacci产生器满足序列比重:  0.166722
```

```
please input N,p,q: 100000 2 3
N= 100000, 16807产生器满足序列比重:  0.16579
N= 100000, p= 2, q= 3, fibonacci产生器满足序列比重:  0.16682
please input N,p,q: 100000 5 9
N= 100000, 16807产生器满足序列比重:  0.16677
N= 100000, p= 5, q= 9, fibonacci产生器满足序列比重:  0.16874
please input N,p,q: 100000 21 15
N= 100000, 16807产生器满足序列比重:  0.16624
N= 100000, p= 21, q= 15, fibonacci产生器满足序列比重:  0.16651
please input N,p,q: 100000 42 30
N= 100000, 16807产生器满足序列比重:  0.16562
N= 100000, p= 42, q= 30, fibonacci产生器满足序列比重:  0.16789
```

2.3 16807产生器与Fibonacci产生器比较

	100	1000	10000	100000	1000000	10000000	100000000
16807产生器	0.16	0.156	0.1627	0.16586	0.166794	0.166678	0.166642
Fibonacci延迟产生器	0.15	0.167	0.168	0.16603	0.166385	0.166644	0.166683

如上表所示，取[p , q]=[22 , 43]，可以看出随着N的增大，无论是16807产生器还是Fibonacci延迟产生器，X(n-1)>X(n+1)>X(n)关系的比重越来越趋近于1/6≈1.6666667.即说明两种随机数产生器都很好地产生了一系列随机数，且随着数目增多而数据更加随机化。

2.4 不同[p , q]取值对Fibonacci产生器影响

```
please input N,p,q: 100000 2 3
N= 100000, 16807产生器满足序列比重:  0.16579
N= 100000, p= 2, q= 3, fibonacci产生器满足序列比重:  0.16682
please input N,p,q: 100000 5 9
N= 100000, 16807产生器满足序列比重:  0.16677
N= 100000, p= 5, q= 9, fibonacci产生器满足序列比重:  0.16874
please input N,p,q: 100000 21 15
N= 100000, 16807产生器满足序列比重:  0.16624
N= 100000, p= 21, q= 15, fibonacci产生器满足序列比重:  0.16651
please input N,p,q: 100000 42 30
N= 100000, 16807产生器满足序列比重:  0.16562
N= 100000, p= 42, q= 30, fibonacci产生器满足序列比重:  0.16789
```

从程序运行结果可以看出，不同[p , q]取值时，Fibonacci产生器产生随机数的大小按次序比重都很好地趋近于1/6，从而证明了Fibonacci产生器对于不同延迟效应的鲁棒性。

3 总结与收获

在homework2实验中，基于第一次作业的16807随机数生成器进一步探讨其生成随机数大小关系，增强自己的代码能力，同时采用不同的随机数生成器办法——Fibonacci产生器，研究发现当N越来越大时，随机数序列更趋向于随机独立的特性，这也非常符合我们对随机数的认知。实验结果表明，两种随机数生成器效果都非常好，在此基础上也验证了Fibonacci延迟产生器对不同数的“延迟”的效果。