伪随机数及其在JAVA程序中的应用探讨

王 宇
(重庆文理学院 重庆 402160)

摘 要 本文分析了生成伪随机数的一般原理,讨论了采用线性同余法获取伪随机数的方法,结合JAVA语言产生伪随机数的机制,通过一个游戏程序实例说明伪随机数的应用。

关键词 线性同余法,伪随机数生成,JAVA程序设计

1 引言

随机数是蒙特卡罗(Monte-Carlo)方法的应用基础,它在程序设计中扮演着十分重要的角色,尤其是在实践环境模拟和测试等领域中有着广泛的应用。例如,构建随机模型、大型游戏、计算机仿真、动画设计、数字图书保护、数字水印、分形图形技术、数据加密等都要大量使用随机数。

所谓随机数,是指满足如下两个条件的数值序列:

①满足一个确定的概率分布;

②序列的任意数值之间相互独立。

产生随机数的方法有两种:

①用物理方法产生真正的随机数。一般而言,硬件应该是随机数的最佳来源,早期生成的随机数,都是来自于各种物理装置和设备。但是生成随机数的硬件耗资不菲,并且存在生成速度慢、效率低、需占用大量存储空间、不可重现等问题,因此其使用价值很有限;

②用数学方法(算法)产生一组满足一定统计规律的迭代随机序列,即伪随机数。由于它不可能是真正的随机数,必须对其进行检验。

从实用的角度看,获取这种数的最简单和最自然的方法是利用各种计算机语言的函数库提供的随机数发生器。当然不同的开发环境提供的生成随机数的函数和方法不一样。典型情况下,它会输出一个均匀分布在[0,1]区间内的伪随机变量的值。

2 伪随机数的生成机制

根据随机数的定义,凡是用算法生成的都不是真正的随机数。因此使用算法只能生成伪随机数。伪随机数并非假随机数,是指计算机产生的伪随机数既是随机的又是有规律的,这为通过算法产生随机数提供了基础。

2.1 伪随机数(Psaudo Random Number)的生成方法

产生伪随机数的方法很多,比如线性同余法、平方取中法、取小数法、斐波那契(Fibonecei)法、非线性同余法、乘同余法、移位寄存器序列、混合同余法等,都能够在特定的情况下构造优良的伪随机数。

伪随机数可以使用递推公式产生,其通用公式为:

$$a_{n+1}=f(a_n) \quad n=1,2,\cdots\cdots \tag{1}$$

显而易见,当计算方法(递推公式)及初值确定后,整个随机数序列就唯一地确定了。但是这与随机数的独立性要求相矛盾;同时,由于伪随机数总要出现周期性循环,这也和真的随机数有区别。因此,在计算机上生成的伪随机数必须满足下列要求:

①分布的均匀性:即同一范围内的任何一个随机数的出现的概率是相同的,即必须尽可能地接近U(0,1)分布。

②统计上的独立性:产生的随机数序列的相关性越小越好。

③足够长的周期:在其达到重复(循环)之前,能生成足够多的、不重复的随机数。

④占用的内存尽可能少,产生随机数的速度足够快。

前两条要求,可以使用统计方法对随机数序列进行检验。

2.2 线性同余法 LCG(Linear Congruence Generator)

在各种程序设计语言中,最常用的还是线性同余法。这种方法是Lehmer于1951年提出的:对于一个正整数M和任意自然数n_0,a,b,由递推公式:

$$n_{i+1}=(af(n_i)+b)\bmod m$$
$$i=0,1,2,\cdots,m-1 \tag{2}$$

生成的数值序列称为是同余序列。当函数f(n)为线性函数时,即得到线性同余序列:

$$n_{i+1}=(a*n_i+b)\bmod m$$
$$i=0,1,2,\cdots,m-1 \tag{3}$$

其中a为乘子(常数),b为增量(常数),m为模,n_0为该随机序列的种子。如何选取该方法中的

常数 a、b 和 m 直接关系到所产生的随机序列的随机性能。

线性同余法的如下特点：

①$0 \leqslant n_i \leqslant m-1$，即 n_i 只能从 0～m－1 这 m 个整数中取值；

②适当地选择 m、a、b，可以保证 n_i 在[0，m－1]区间上的均匀性。

线性同余法不仅与种子有关，而且与参数 m、a、c 的选取有关，实例如表 1 所示：

表 1　种子与参数的关系

a	b	n_0	m	随机序列	周期
7	7	7	10	6, 9, 0, 7, 6, 9, …	4
5	3	7	16	6, 1, 8, 11, 10, 5, 12, 15, 14, 9, 0, 3, 2, 13, 4, 7, 6, …	16
5	1	1	8	6, 7, 4, 5, 2, 3, 0, 1, 6, 7…	8

为了保证其统计性质，提高 n_i 的均匀性，要求 m 足够大，例如可取 m 为机器大数，或者把它取为 2^w，W 是计算机的字长。并且在计算过程中，为了防止整数溢出，还需要进行一些算法处理。

如果伪随机数发生器具有满周期，可以预计所得到的 n_i 在[0，1]区间上是均匀分布的，且取值是相当密集的，这样就足够近似于均匀分布 U(0，1)。

2.3　线性同余法生成伪随机数的算法

```
RandomNum(n,m,seed,a,b){
  n0=seed;
  for(i=1;i<=n;i++)
  ni=(a*ni-1+b)mod m
}
```

其中种子参数 seed 的初始值可以任意选择，人们经常使用系统测定技术来计算随机“种子”，然后将它放入伪随机数发生器，而种子的值用快速计数寄存器或移位寄存器来生成。

为了尽量降低重复率，既可以使用主板序列号、硬盘序列号、网卡序列号、当前时间等作种子数，也可以采样用户的键盘或鼠标事件，并根据时间或位置数据来派生出随机性。

3　Java 程序实例

3.1　Java 生成伪随机数的机制

在 Java 中有两种生成伪随机数的方式：Random 类和 Math. random( )方法：

Java 实用工具类库中的类 java. util. Random 提供了产生各种类型随机数的方法。当创建 Random 对象，就可以得到一系列的随机数，它可以产生 int、long、float、double 以及 Goussian 等类型的伪随机数。

而 java. lang. Math 中 Random( )只能产生 double 型的随机数。

事实上，Random 类是一个伪随机数发生器，它有两个构造方法和六个普通方法。

构造方法：public Random( )、public Random (long seed)

Java 默认使用当前的系统时间为种子来初始化 Random 对象。因为没有任何时刻的时间是相同的，所以就可以减少随机数序列相同的可能性。

普通方法：public synonronized void setSeed (long seed)、public int nextInt( )、public long nextLong( )、public float nextFloat( )、public double nextDouble( )、public synchronized double nextGoussian()

Java 设计者在 Random 类的 Ramdom()构造方法中使用当前的系统时间来初始化 Random 对象。因为没有任何时刻的时间是相同的，所以就可以减少随机数序列相同的可能性。

3.2　Java Random 类的实现

对于下面两条语句：

```
Random randomGenerator = new Random( );
int randomNumber = randomGenerator. nextInt(n);
```

其中用到了 Random 类的 nextInt 方法。以下是 nextInt 方法实现的主要代码：

```
public int nextInt(int n){
  int bits,val;
  do{
    bits=next(31);
    val=bits%n;
    }while(bits-val+(n-1)<0);
  return val;
}
```

通过 nextInt 方法产生的随机数的范围应根据特定应用程序需要的不同而不同。可按照下列形式为 nextInt 方法传递一个参数：value=1+randomGenerator. nextInt(6)；

该语句产生一个从 1 到 6 范围内的随机整数。当给 nextInt 方法传递一个参数时，由 nextInt 方法所返回的值的范围为 0 到参数值减 1。

3.3　应用实例

创建一个模拟掷骰子游戏的应用程序。每个骰子有六个面，分别为 1，2，3，4，5，6 这六个点。当游戏者掷出的两个骰子停止滚动，计算其面上的点数和。如果投掷的点数之和为 7 或 11，游戏者获胜并显示“Win”；如果投掷的点数之和为 2，3 或 12，游戏失败并显示“Lose”；如果点数之和为 4，5，6，7，9 或

10，则将该值作为游戏者的“点数”。

为了能在掷骰子游戏应用程序中使用随机数，需要导入 java. util 包中的 Random 类。

该应用程序中有几个地方都需要滚动和显示两个骰子。因此需要创建两个方法：rollDice 方法用于滚动骰子；displayDice 用于显示骰子的图片。其主要代码如下：

```
private int rollDice( ){
  int dice1=1+randomObject. nextInt(6);
  int dice2=1+randomObject. nextint(6);
  displayDice(dice1JLabel,dice1);
  displayDice(dice2JLabel,dice2);
  return dice1+dice2;
} //end method rollDice
```

该段代码将变量 dice1 和 dice2 分别设置为 1～6（包括边界值）之间的一个 int 型随机数。之后对 displayDice 方法进行了两次调用。displayDice 方法用于显示对应于 1～6 之间的骰子图片，其中第 1 个参数代表所显示图片的 JLabel，第 2 个参数代表骰子的点数值。

```
private void displayDice(JLabel picDiceJLabel,int face){
  imageIcon image=new ImageIcon(FILE_PREFIX + face
+ FILE_SUFFIX);
  picDiceJLabel. setIcon(image);
  } // end method displayDice
```

displayDice 方法显示与用 rollDice 方法产生的随机骰子数相对应的图片。其中参数 int 用来取得一幅图片，JLablel 用来显示该图片。FILE_PREFIX 和 FILE_SUFFIX 是两个已得到声明的常量，字符串 FILE_PREFIX + face + FILE_SUFFIX 用来指明某个图片文件的位置（FILE_PREFIX 和 FILE_SUFFIX 都定义为 private final String 型，其值分别为“Images/dice”和“. png”）。如果 face 的值为 1，则该表达式将变为“Images/dice1. png”，表示表面点数为 1 的图片的位置。该程序的运行界面如图 1 所示。

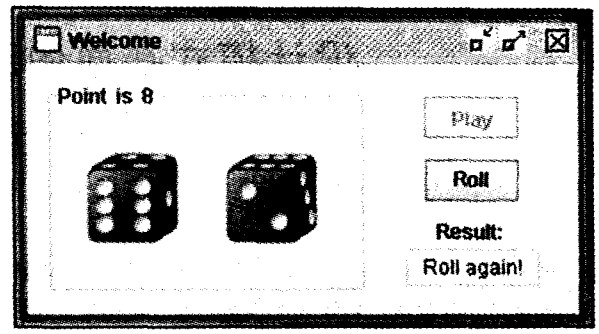

图 1　游戏程序运行界面

当单击 Play 按钮后，该应用程序将生成两个介于 1～6 之间的随机数，并将相应的骰子图片显示在指定位置，计算出两个骰子表面上的数值之和。

结束语　伪随机数在软件开发和程序设计中应用较为广泛，如何获得质量高、性能佳的随机数是软件开发者追求的目标之一。JAVA 作为一种主流的面向对象程序设计语言，提供了多种生成伪随机数的途径，以满足不同的设计要求。应当指出，不同算法产生的伪随机数的随机性还是有差别的，应当根据实际需要加以选择。

参 考 文 献

1　盛骤，谢式千. 概率论与数理统计[M]. 北京：高等教育出版社，2000

2　李芝兴，杨瑞龙. Java 程序设计之网络编程[M]. 北京：清华大学出版社，2006

3　王晓东. 算法设计与分析[M]. 北京：电子工业出版社（第 2 版），2004

（上接第 186 页）

开发人员可以集中关注业务；

（4）采用 Hibernate 的 ORM 映射机制，抽象了数据库，使得程序与数据库的交互透明化。

总结　面对需求无法确定和不断变化的难题，传统软件开发方法面临很大挑战，可复用快速原型法由于其快速开发、修改方便、快速相应需求和高可扩展的特点迎合了这种需求。本文提出的基于 J2EE 的 Struts ＋Spring＋Hibernate 架构，利用 MVC，建立了一个结构清晰的系统框架，利用 IOC，使各个模块之间松散耦合，利用 AOP，用简单的方法完成了对系统横切点的考虑，利用 ROM，高效地实现了与数据库的交互。在可复用快速原型法的指导下，《重庆铁路物流电子商务平台——集装箱运营系统》很优雅地实现了。

参 考 文 献

1　史济民，顾春华. 软件工程原理、方法与应用. 高等教育出版社，2002

2　乔非，严隽薇，等. 企业模型体系的建模与优化方法，2000

3　孙卫琴编著. 基于 MVC 的 java web 设计和开发[M]. 北京：电子工业出版社，2004

4　Johnson R. EXPERT ONE-ON-ONE J2EE DEVELOPMENT WITHOUT EJB. 北京：电子工业出版社，2005

5　Johnson R. Spring Reference Documentation[EB/OL]. http ://www. springframework. org/ ,2004

6　Hibernate，官方 . Hibernate Reference Documentation [EB/OL]. http ://www. hibernate. org/ hib docs/ v3/ reference/ en/html single/ ,2005