

第 9 章

其他题目

从 5 随机到 7 随机及其扩展

【题目】

给定一个等概率随机产生 1~5 的随机函数 rand1To5 如下：

```
public int rand1To5() {  
    return (int) (Math.random() * 5) + 1;  
}
```

除此之外，不能使用任何额外的随机机制，请用 rand1To5 实现等概率随机产生 1~7 的随机函数 rand1To7。

【补充题目】

给定一个以 p 概率产生 0，以 $1-p$ 概率产生 1 的随机函数 rand01p 如下：

```
public int rand01p() {  
    // 可随意改变 p  
    double p = 0.83;  
    return Math.random() < p ? 0 : 1;  
}
```

除此之外，不能使用任何额外的随机机制，请用 rand01p 实现等概率随机产生 1~6 的随机函数 rand1To6。

【进阶题目】

给定一个等概率随机产生 $1 \sim M$ 的随机函数 rand1ToM 如下：

```
public int rand1ToM(int m) {
    return (int) (Math.random() * m) + 1;
}
```

除此之外，不能使用任何额外的随机机制。有两个输入参数，分别为 m 和 n ，请用 rand1ToM(m)实现等概率随机产生 $1 \sim n$ 的随机函数 rand1ToN。

【难度】

原问题 尉 ★★☆☆

补充问题 尉 ★★☆☆

进阶问题 校 ★★★★★

【解答】

先解决原问题，具体步骤如下：

1. rand1To5()等概率随机产生 1,2,3,4,5。
2. rand1To5()-1 等概率随机产生 0,1,2,3,4。
3. (rand1To5()-1)*5 等概率随机产生 0,5,10,15,20。
4. (rand1To5()-1)*5+(rand1To5()-1)等概率随机产生 0,1,2,3,...,23,24。注意，这两个 rand1To5()是指独立的两次调用，请不要化简。这是“插空儿”的过程。
5. 如果步骤 4 产生的结果大于 20，则重复进行步骤 4，直到产生的结果在 0~20 之间。同时可以轻易知道出现 21~24 的概率，会平均分配到 0~20 上。这是“筛”过程。
6. 步骤 5 会等概率随机产生 0~20，所以步骤 5 的结果再进行%7 操作，就会等概率的随机产生 0~6。
7. 步骤 6 的结果再加 1，就会等概率地随机产生 1~7。

具体请参看如下代码中的 rand1To7 方法。

```
public int rand1To5() {
    return (int) (Math.random() * 5) + 1;
}

public int rand1To7() {
    int num = 0;
    do {
```

```

        num = (rand1To5() - 1) * 5 + rand1To5() - 1;
    } while (num > 20);
    return num % 7 + 1;
}

```

然后是补充问题。虽然 rand01p 方法以 p 的概率产生 0，以 $1-p$ 的概率产生 1，但是 rand01p 产生 01 和 10 的概率却都是 $p(1-p)$ ，可以利用这一点来实现等概率随机产生 0 和 1 的函数。具体过程请参看如下代码中的 rand01 方法。

```

public int rand01p() {
    // 可随意改变 p
    double p = 0.83;
    return Math.random() < p ? 0 : 1;
}

public int rand01() {
    int num;
    do {
        num = rand01p();
    } while (num == rand01p());
    return num;
}

```

有了等概率随机产生 0 和 1 的函数后，再按照如下步骤生成等概率随机产生 1~6 的函数：

1. rand01()方法可以等概率随机产生 0 和 1。
2. rand01()*2 等概率随机产生 0 和 2。
3. rand01()*2+rand01()等概率随机产生 0,1,2,3。注意，这两个 rand01()是指独立的两次调用，请不要化简。这是“插空儿”过程。

步骤 3 已经实现了等概率随机产生 0~3 的函数，具体请参看如下代码中的 rand0To3 方法：

```

public int rand0To3() {
    return rand01() * 2 + rand01();
}

```

4. rand0To3()*4+rand0To3()等概率随机产生 0,1,2,...,14,15。注意，这两个 rand0To3()是指独立的两次调用，请不要化简。这还是“插空儿”过程。

5. 如果步骤 4 产生的结果大于 11，则重复进行步骤 4，直到产生的结果在 0~11 之间。那么可以知道出现 12~15 的概率会平均分配到 0~11 上。这是“筛”过程。

6. 因为步骤 5 的结果是等概率随机产生 0~11，所以用第 5 步的结果再进行%6 操作，就会等概率随机产生 0~5。

就会等概率随机产生 0~5。

7. 第 6 步的结果再加 1，就会等概率随机产生 1~6。

具体请参看如下代码中的 rand1To6 方法。

```
public int rand1To6() {
    int num = 0;
    do {
        num = rand0To3() * 4 + rand0To3();
    } while (num > 11);
    return num % 6 + 1;
}
```

最后是进阶问题。如果读者真正理解了“插空儿”过程和“筛”过程，就可以知道，只要给定某一个区间上的等概率随机函数，就可以实现任意区间上的随机函数。所以，如果 $M \geq N$ ，直接进入如上所述的“筛”过程；如果 $M < N$ ，先进入如上所述“插空儿”过程，直到产生比 N 的范围还大的随机范围后，再进入“筛”过程。具体地说，是调用 k 次 rand1ToM(m)，生成有 k 位的 M 进制数，并且产生的范围要大于或等于 N 。比如随机 5 到随机 7 的问题，首先生成 0~24 范围的数，其实就是 $0 \sim (5^2 - 1)$ 范围的数。在把范围扩到大于或等于 N 的级别之后，如果真实生成的数大于或等于 N ，就忽略，也就是“筛”过程。只留下小于或等于 N 的数，那么在 $0 \sim N - 1$ 上就可以做到均匀分布。具体请参看如下代码中的 rand1ToN 方法。

```
public int rand1ToM(int m) {
    return (int) (Math.random() * m) + 1;
}

public int rand1ToN(int n, int m) {
    int[] nMSys = getMSysNum(n - 1, m);
    int[] randNum = getRanMSysNumLessN(nMSys, m);
    return getNumFromMSysNum(randNum, m) + 1;
}

// 把 value 转成 m 进制数
public int[] getMSysNum(int value, int m) {
    int[] res = new int[32];
    int index = res.length - 1;
    while (value != 0) {
        res[index--] = value % m;
        value = value / m;
    }
    return res;
}

// 等概率随机产生一个 0~nMSys 范围的数，只不过是使用 m 进制表达的
```

```

public int[] getRanMSysNumLessN(int[] nMSys, int m) {
    int[] res = new int[nMSys.length];
    int start = 0;
    while (nMSys[start] == 0) {
        start++;
    }
    int index = start;
    boolean lastEqual = true;
    while (index != nMSys.length) {
        res[index] = randlToM(m) - 1;
        if (lastEqual) {
            if (res[index] > nMSys[index]) {
                index = start;
                lastEqual = true;
                continue;
            } else {
                lastEqual = res[index] == nMSys[index];
            }
        }
        index++;
    }
    return res;
}

// 把 m 进制数转成十进制数
public int getNumFromMSysNum(int[] mSysNum, int m) {
    int res = 0;
    for (int i = 0; i != mSysNum.length; i++) {
        res = res * m + mSysNum[i];
    }
    return res;
}

```

一行代码求两个数的最大公约数

【题目】

给定两个不等于 0 的整数 M 和 N ，求 M 和 N 的最大公约数。

【难度】

士 ★★☆☆

【解答】

一个很简单的求两个数最大公约数的算法是欧几里得在其《几何原本》中提出的欧几