《计算机算法设计与分析》第七章作业

姓名：任宇　　学号：33920212204567

算法分析题 7-3

7-3　随机产生 m 个整数。

试设计一个算法，随机地产生范围在 $1\sim n$ 的 m 个随机整数，且要求这 m 个随机整数互不相同。

答：

算法设计

1. 初始化序列：

- 创建一个长度为 n 的数组 sequence。
- 填充这个数组，使其包含从 1 到 n 的所有整数。

2. 打乱序列：

- 对于数组 sequence 中的每个索引 i（从 0 开始到 n-1），生成一个 0 到 1 的 double 随机数 k，计算出索引 i 为 n*k+j。
- 交换 sequence[i]和 sequence[j]的值。这一步是“洗牌”算法的关键部分,j 自增。

3. 选取前 m 个元素：

- 创建一个新数组 randomSelection。
- 从 sequence 中复制前 m 个元素到 randomSelection。

4. 输出结果：

- 返回 randomSelection 作为最终结果，它包含了范围在 1 到 n 之间的 m 个互不相同的随机整数。

时间复杂度分析

整体时间复杂度：O(n)。因为填充初始数组的时间复杂度为 O(n)，“洗牌”算法的时间复杂度为 O(n)，从打乱的数组中选取前 m 个元素的时间复杂度为 O(1)。

算法分析题 7-4

7-4 集合大小的概率算法。

设 X 是含有 n 个元素的集合，从 X 中均匀地选取元素。设第 k 次选取时首次出现重复。

（1）试证明当 n 充分大时，k 的期望值为 $\beta\sqrt{n}$。其中，$\beta\sqrt{\pi/2}=1.253$。

（2）由此设计一个计算给定集合 X 中元素个数的概率算法。

答：（1）

$$P(k)=\frac{n}{n}\times\frac{n-1}{n}\times\frac{n-2}{n}\times\cdots\times\frac{n-k+1}{n}=\frac{\binom{n}{k-1}(k-1)!(k-1)}{n^k}$$

$$E(k)=\sum_{k=1}^{n}\frac{\binom{n}{k-1}(k-1)!(k-1)k}{n^k}$$

用斯特林公式 $n!=\sqrt{2\pi n}\left(\frac{n}{e}\right)^n\left(1+\frac{1}{12n}+\Theta\left(\frac{1}{n^2}\right)\right)$

代入 $E(k)$ 计算可得 $E(k)=\sqrt{\frac{\pi n}{2}}-\frac{1}{3}+\Theta\left(\frac{1}{\sqrt{n}}\right)\approx 1.253\sqrt{n}$

（2）

算法设计

1. 初始化：

设置一个空集合 selectedElements 用于存储已选择的元素。

初始化一个计数器 k 为 0，用于记录尝试次数。

2. 随机选择元素：

- 不断重复以下步骤，直到出现重复元素：
  - 从集合 X 中随机选择一个元素。
  - 增加计数器 k 的值（k += 1）。
  - 检查这个元素是否已存在于 selectedElements 中。
    - 如果是，停止选择过程。
    - 如果不是，将该元素添加到 selectedElements 中。

3.　估计元素数量并返回：

使用公式估算集合 X 中元素的数量，返回估算的集合大小 $\frac{2k^2}{\pi}$。

时间复杂度分析

最坏情况时间复杂度：O(n)，因为在最坏的情况下，可能需要遍历整个集合才能找到一个重复的元素。

算法分析题 7-5

7-5　生日问题。

试设计一个随机化算法计算 $365!/340!365^{25}$，并精确到 4 位有效数字。

答：

在生日问题中，K个人中至少两人生日相同的概率是 $1-\frac{365!}{(365-k)!\,365^k}$

题目是 $\frac{365!}{340!\,365^{25}}$，也就是25人中~~至~~每人的生日各不一样的概率

本题可以采用蒙特卡罗算法：

算法设计

1. 初始化参数，比如 numSimulation=1000000（次数越多越精确）

设定人数 n=25，一年天数 ~~天数~~ daysInYear=365

2. 进行模拟

· 初始化一个计数器 count 用于记录没有重复生日的模拟次数

· 对1到 numSimulation 每一个模拟：

创建一个空集合 birthdays 存储生日

生成n个随机生日添加到集合中并判断是否重复，不重复则count加1

3. 计算概率

~~返回~~ 返回 count/numSimulation的值，并精确到四位小数。

时间复杂度：主要由模拟次数决定，为O(numSimulation)，每次模拟中生成和检查n个生日的时间复杂度为O(n)，因此总复杂度为O(n*numSimulation)

算法分析题 7-9

7-9 n 后问题解的存在性。

如果对于某个 n 值，n 后问题无解，则算法将陷入死循环。

（1）证明或否定下述论断：对于 $n\geq 4$，n 后问题有解。

（2）是否存在正数 δ，使得对所有 $n\geq 4$ 算法成功的概率至少是 δ?

答：（1）证明：

对于 n ≥ 4 的情况，可以通过构造性证明来展示解的存在：

偶数 n（n≥4）：将第一个皇后放在第一行的第二列，第二个皇后放在第二行的第四列，以此类推，直到放置最后一个皇后在倒数第二行的第一列和最后一行的第三列。这种放置方法确保了所有皇后都不在同一行、同一列或同一对角线上。

奇数 n（n>4）：对于奇数 n，可以先解决 n-1 的问题，然后在最后一行和列添加一个皇后。

因此，对于所有 n ≥ 4，n 皇后问题总是有解的。

（2）

由（1）可知，对于所有 n ≥ 4，n 皇后问题总是有解的。对所有 n ≥ 4：要声明存在一个正数 δ，使得对于所有 n ≥ 4，理论上，可能存在这样的正数 δ，但这个 δ 可能非常非常小，特别是对于大的 n。随机算法的一个主要问题是缺乏系统性，它们可能会在不成功的配置上浪费大量时间。特别是对于大的 n，随机算法可能无法在合理的时间内找到解。解决大规模 n 后问题，更常用的方法是采用启发式或优化算法（如回溯算法、遗传算法等），这些方法在寻找有效解方面更加高效和可靠。

算法分析题 7-12

7-12 重复 3 次的蒙特卡罗算法。

设 mc(x)是一致的 75%正确的蒙特卡罗算法，考虑下面的算法：

```
mc3(x) {
  int  t, u, v;
  t = mc(x);
  u = mc(x);
  v = mc(x);
  if ((t == u) || (t == v))
    return t;
  return v;
}
```

（1）试证明上述算法 mc3(x)是一致的 27/32 正确的算法，因此是 84%正确的。

（2）试证明如果 mc(x)不是一致的，则 mc3(x)的正确率有可能低于 71%。

答：

(1) 重复3次的蒙特卡罗算法各次正确的分布为：

000, 001, 010, 100, 011, 101, 110, 111. 其中返回正确解有4种情况，即011, 101, 110, 111，其概率为：$\frac{1}{4}\times\frac{3}{4}\times\frac{3}{4}+\frac{3}{4}\times\frac{1}{4}\times\frac{3}{4}+\frac{3}{4}\times\frac{3}{4}\times\frac{1}{4}+\frac{3}{4}\times\frac{3}{4}\times\frac{3}{4}=\frac{27}{32}$

$=84.375\%$

(2) 如果mc(x)是一致的，则011不能保证返回正确解，返回正确解概率降为 $\frac{27}{32}-\frac{1}{4}\times\frac{3}{4}\times\frac{3}{4}=\frac{45}{64}\approx 70.3125\% < 71\%$

算法分析题 7-14

7-14 由蒙特卡罗算法构造拉斯维加斯算法。

设算法 A 和 B 是解同一判定问题的两个有效的蒙特卡罗算法。算法 A 是 p 正确偏真算法，算法 B 是 q 正确偏假算法。试利用这两个算法设计一个解同一问题的拉斯维加斯算法，并使所得到的算法对任何实例的成功率尽可能高。

答：当算法 A 返回真时，整体算法返回真：算法 A 是 p 正确偏真算法，这意味着当实际答案是真时，算法 A 正确的概率很高（p）。由于算法 A 在是实例（答案为真）时表现良好，因此当它返回真时，我们可以相对有信心地认为整个问题的答案确实是真。当算法 B 返回真时，整体算法返回假：算法 B 是 q 正确偏假算法，这意味着当它返回真（即认为答案是假）时，我们可以相对有信心地认为整个问题的答案确实是假。

用伪代码表示为：

```
bool LasVegasAlgorithm(ST X):{
    While（TRUE）{
    if （AlgorithmA(X)） return true;
    if （! AlgorithmB(X)）return false;
    }
}
```

算法实现题 7-3

7-3 集合相等问题。

问题描述：给定两个集合 *S* 和 *T*，试设计一个判定 *S* 和 *T* 是否相等的蒙特卡罗算法。

算法设计：设计一个拉斯维加斯算法，对于给定的集合 *S* 和 *T*，判定其是否相等。

数据输入：由文件 input.txt 给出输入数据。第 1 行有 1 个正整数 *n*，表示集合的大小。接下来的 2 行，每行有 *n* 个正整数，分别表示集合 *S* 和 *T* 中的元素。

结果输出：将计算结果输出到文件 output.txt。若集合 *S* 和 *T* 相等则输出“YES”，否则输出“NO”。

答：

算法设计

算法步骤

1. 随机化检查元素：

随机选择 S 中的元素，并检查它是否也存在于 T 中。这个步骤可以重复进行若干次。

同样，随机选择 T 中的元素，并检查它是否也存在于 S 中。

2. 判断相等性：

如果在所有随机选择中，S 和 T 中的元素都能在对方集合中找到，则返回“相等”。如果存在任何一个元素在对方集合中找不到，则返回“不相等”。

3. 重复或终止：

如果需要提高算法的确定性，可以增加随机选择元素的次数。

时间复杂度分析：遍历集合需要的时间复杂度为 O(n)，假设我们进行了 k 次随机检查，则时间复杂度为 O(k*n)。

算法实现题 7-4

7-4 逆矩阵问题。

问题描述： 给定两个 $n \times n$ 矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$，试设计一个判定 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 是否互逆的蒙特卡罗算法（算法的计算时间应为 $O(n^2)$）。

算法设计： 设计一个蒙特卡罗算法，对于给定的矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$，判定其是否互逆。

数据输入： 由文件 input.txt 给出输入数据。第 1 行有 1 个正整数 n，表示矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 为 $n \times n$ 矩阵。接下来的 $2n$ 行，每行有 n 个实数，分别表示矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 中的元素。

结果输出： 将计算结果输出到文件 output.txt。若矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 互逆，则输出“YES”，否则输出“NO”。

答：

算法设计

1. 初始化：

设定实验次数，例如 k 次。

2. 进行随机实验：

对于每次实验：

- 随机选择一个索引 i（在 0 到 n-1 之间）。
- 计算矩阵 A 第 i 行，遍历矩阵 B 的每一列，如果所有的计算都符合 AB=I 的条件（即对角线上的点积等于 1，非对角线上的点积等于 0），则继续下一次实验。否则，返回 false。

3. 返回结果：

如果所有实验都是一致的，则认为两个矩阵互逆。

时间复杂度分析

每轮的计算次数是 n，时间复杂度为 O(n)，一共有 k * n 轮，因此总时间复杂度为 $O(n^2)$。