算法分析第7次作业

小组编号：23

本次作业负责人:李嘉琪

1 算法分析题7-3 答案：

7-3 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。

算法设计：

首先，创建一个包含从 1 到 n 的整数列表，称为 `numbers` 。然后初始化一个空列表，称为 `random_numbers` ，用于存储生成的随机整数。

通过循环从 0 到 m-1，使用当前迭代的索引 i。在每次迭代中，生成一个随机整数，记为 index，其范围从 i 到 n-1（包括 i 和 n-1）。通过交换 `numbers` 列表中索引 i 和 index 位置的元素，确保每个索引位置 i 处的元素都是随机选择的。然后将 `numbers` 列表中此时索引 i 处的元素添加到 `random_numbers` 列表中，作为一个生成的随机整数。

循环结束后，就生成了 m 个不重复的随机整数。最终，返回 `random_numbers` 列表作为结果。

算法分析：

时间复杂度：该算法的时间复杂度主要取决于循环的次数 m。每次循环中生成随机数、交换数字、插入数字的操作均为 O(1) 的时间，因此算法的时间复杂度为 O(n)。

空间复杂度：该算法需要 `numbers` 数组和 `random_numbers` 数组来存储原始数字以及生成的随机整数，因此空间复杂度为 O(n)。

本题分工：小组共同讨论，黄勖编写

2 算法分析题7-4 答案：

7-4 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。

解答：

（1）证明：

从n个元素的集合X中均匀选取元素，假设第k次选取时首次出现重复，则对于前面$k-1$次不能重复

前面$k-1$个位置所有排列可能数量为A_n^{k-1}，每个位置都可以选择1到n

因此前面$k-1$次不重复的概率为

$$\frac{A_n^{k-1}}{n^{k-1}}$$

第k次必须重复，可以从n个元素中选择前面出现的$k-1$个数字的某个，概率为$\frac{k-1}{n}$

因此第k次出现重复的概率为

$$\frac{A_n^{k-1}}{n^{k-1}} \times \frac{k-1}{n} = \frac{A_n^{k-1}(k-1)}{n^k}$$

$k \in [1, n], k \in Z$，因此k的期望值为

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

根据$Stirling$公式

$$n! = \sqrt{2\pi n}(\frac{n}{e})^n(n \to \infty)$$

代入$E(k)$可得

$$E(k) = \sqrt{\frac{\pi n}{2}}$$

得证

（2）由（1）证明可得，若E(k)已知，则

$$n = \frac{2 * E(k) * E(k)}{\pi}$$

算法描述:

因此可以设计随机算法如下

初始化空集合S，循环处理:

每次随机选择集合X中的一个元素a，同时用k记录次数

- 若$a \in S$，则返回$n = \frac{2*k*k}{\pi}$，算法结束
- 若$a \notin S$，则将a加入S，且$k++$，算法继续

算法分析:

假设第k次首次出现重复，则每次选择元素a时遍历判断a是否在集合S中，算法时间复杂度为

$$O(k^2)$$

本题分工：小组共同讨论，李嘉琪编写

3 算法分析题7-5 答案：

7-5 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。

算法设计：

首先，化简原计算公式为

$$(\frac{365!}{(340!\cdot 365^{25})}=\frac{341}{365}\cdot\frac{342}{365}\cdot\dots\cdot\frac{364}{365})$$

每次随机试验由 24 次独立的子试验组成，每次子试验的随机值取自区间[1, 365]。在每次子试验中，将随机值与一个特定的阈值 K 进行比较。如果在 24 次子试验中，所有的随机值都落在区间[1, K]内，那么判定此次事件为真；否则，判定此次事件为假。为了估计事件的发生概率，重复进行多次测试，每次测试中都进行 24 次子试验。通过统计在所有测试中事件被判定为真的次数以及测试的总次数，得到对事件发生概率的估计。

算法思路是通过多次随机试验模拟事件发生情况，并根据试验结果进行判断。通过重复测试并统计结果，可以估计事件发生的概率。具体实现思路为：

- 创建一个随机数生成器对象，命名为 `birth`，用于生成随机数。
- 初始化一个变量 `testingTimes`，表示测试的总次数。
- 进入一个循环，循环变量 `i` 从 1 递增到 `testingTimes`：计算当前测试次数 `total`，即 `i` 乘以一个常数，表示当前测试的总次数。初始化一个计数变量 `count`，初始值为 `total`。
- 进入一个嵌套循环，循环变量 `j` 从 0 递增到 `total`：在内部嵌套循环中，根据设定的条件（随机数与阈值的比较），更新计数变量 `count`。计算概率的估计值 `rate`，即 `count` 除以 `total`。输出当前测试次数 `total` 和计算结果 `rate`。

算法分析：

① 时间复杂度分析：算法的时间复杂度主要从内外两层循环考虑。外层循环由变量 `testingTimes` 决定，循环次数为 `testingTimes`。内层循环由变量 `total` 决定，循环次数为 `total`。内层循环中存在两个嵌套的循环，其中一个由变量 `j` 决定，循环次数为 `total`。另一个由变量 `k` 决定，循环次数为 25 次（从 364 到 340）。在最内层循环，调用了 `birth.Random(1, 365)` 生成随机数并进行一次比较操作。假设生成随机数和比较操作的时间复杂度为常数时间，则算法的总体时间复杂度近似为

$$O(\text{testingTimes}\times\text{total}\times 25)$$

② 空间复杂度分析：算法中使用的额外空间主要是随机数生成器对象 `birth` 和几个整型变量。因此，算法的空间复杂度为 $O(1)$。

本题分工：小组共同讨论，黄勖编写

4 算法分析题7-9 答案:

7-9 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?

解答:

（1）证明:

假设x_{ij}表示在棋盘格子(i,j)出放置皇后的状态

若$x_{ij} == 1$，表示在棋盘格子(i,j)处放置一个皇后

若$x_{ij} == 0$，表示在棋盘格子(i,j)处未放置皇后

使用随机算法解决n后问题时，要么求到解，要么到某个位置无法得到解

$$\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij}$$

因此转变为如下$0-1$规划问题

$$\max \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij} \leq n$$

要证明$n \geq 4$时n后问题有解，即证明$n \geq 4$时，$\max \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij} = n$

- 当$n \geq 4, n\%2 = 0, (n-2)\%6 > 0$时，对$1 \leq j \leq \frac{n}{2}$，令$x(j, 2j) = 1, x(\frac{n}{2}+j, 2j-1) = 1$
- 当$n \geq 4, n\%2 = 0, n\%6 > 0$时，对$1 \leq j \leq \frac{n}{2}$，令$x(j, k(j)) = 1, x(n+1-j, n-(k(j)) = 1$，且$k(j) = (\frac{n}{2} + 2(j-1) - 1)\%n$
- 当$n > 4, n\%2 = 1$时，取$x(n,n) = 1$,转换为$(n-1)\times(n-1)$棋盘问题，采用上述构造

对于$n \geq 4$，能构造出n后问题的解

（2）不存在，δ取值不能确定

本题分工：小组共同讨论，李嘉琪编写

5 算法分析题7-12 答案:

7-12 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%。

解答:

（1）记 mc(x)所得解正确为 1，错误为 0，则 mc3(x)算法存在8种情况:

000，010，001，011，100，110，101，111，其中 011，101，110，111 四种情况所求解正确，则正确率为

$$\frac{1}{4}\cdot\frac{3}{4}\cdot\frac{3}{4}+\frac{3}{4}\cdot\frac{1}{4}\cdot\frac{3}{4}+\frac{3}{4}\cdot\frac{3}{4}\cdot\frac{1}{4}+\frac{3}{4}\cdot\frac{3}{4}\cdot\frac{3}{4}=\frac{27}{32}\approx 0.84$$

（2）如果 `mc(x)` 不是一致的，情况 `110` 中，`mc(x)` 返回给 `t` 与 `u` 的值不一定会相等，即无法保证得到正确解，则正确率应为:

$$\frac{1}{4}\cdot\frac{3}{4}\cdot\frac{3}{4}+\frac{3}{4}\cdot\frac{1}{4}\cdot\frac{3}{4}+\frac{3}{4}\cdot\frac{3}{4}\cdot\frac{3}{4}=\frac{45}{64}\approx 0.703$$

因此，如果算法 `mc3(x)` 不是一致的，正确率有可能低于 71%。

本题分工：小组共同讨论，黄勖编写

6 算法分析题7-14 答案:

7-14 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。

算法描述:

蒙特卡罗算法对于任意输入x

$A(x)$为真时，算法一定为真

$B(x)$为假时，算法一定为假

设计$LasVegas$算法如下:

初始化$success = false$，设置$while$循环：

- 若$A(x) == true$，则设置$success = true$，退出循环，算法结束
- 若$B(x) == false$，则设置$success = true$，退出循环，算法结束
- 其他情况，$success = false$，算法继续

算法分析:

假设算法运行k次得到正确解，则算法获得正确解的概率P为

$$P \geq 1 - [(1-p)(1-q)]^k$$

相比单独使用A或者B

$$P \geq 1 - [(1-p)(1-q)]^k \geq 1 - (1-t)^k \; t \in \{p, q\}$$

本题分工：小组共同讨论，李嘉琪编写

7　算法实现题7-3 答案：

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”。

算法设计：

分析该问题得知，从 S 数组中随机选择一个元素，若该元素存在于 T 集合中，则不一定表明 S 和 T 相等；若该元素不存在于 T 集合中，则 S 和 T 一定不相等。因此，设计出来的蒙特卡罗算法是一个偏假的蒙特卡罗算法。

首先进入一个有限循环，循环次数为一个足够大的数。从 S 中随机挑选一个数，判断这个数是否在 T 中。若不在 T 中，算法直接返回 false；若在 T 中，说明 S 和 T 可能相等，进入下一次循环。循环结束后，可以近似认为 S 和 T 相等。

算法分析：

① 时间复杂度分析：在最坏情况下（即每次从 S 中选出来的数都在 T 中），该算法的时间复杂度主要取决于循环次数。然而，分析时间复杂度在这种情况下可能失去意义，因为蒙特卡罗算法的特点是通过随机抽样来近似解决问题，循环次数的影响难以精确估计。

② 空间复杂度：算法需要存储两个数组，分别用于存储 S 和 T 两个集合，因此空间复杂度为 O(n)。

本题分工：小组共同讨论，黄勖编写

8 算法实现题7-4 答案：

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”。

蒙特卡罗算法运行一次一定有解但不能保证一定是正确的解

算法描述:

判断条件：若矩阵A、矩阵B互为逆矩阵，当且仅当$AB = E$

对于偏假的$MonteCarlo$算法设计思路如下：

随机选择确定i，且$i \in [1, n], i \in Z$

- 若$\sum_{j=1}^{n} A[i][j] \times B[j][i] == 1$，返回$true$
- 若$\sum_{j=1}^{n} A[i][j] \times B[j][i] \neq 1$，返回$false$

设置出错率$\epsilon \in (0, 1)$，计算循环次数$k = log_2 \frac{1}{\epsilon}$

循环k次$MonteCarlo$算法：

- 若返回$false$，一定是正确解，算法结束
- 若返回$true$，不一定是正确解，算法继续

算法分析:

$MonteCarlo$算法是一个偏假的$p = \frac{1}{2}$正确的$MonteCarlo$算法

重复k次调用$MonteCarlo$算法得到正确解的概率为

$$1 - (1 - p)^k$$

每次计算$MonteCarlo$算法的时间复杂度为$O(n)$，需要循环k次，因此时间复杂度为

$$O(n) \times k = O(nlog\frac{1}{\epsilon})$$

本题分工：小组共同讨论，李嘉琪编写