第 10 章　概率算法

1. 用随机投点法计算圆周率。

C/C++代码：

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

double random_double() {
    // 生成一个 [-1, 1] 范围的随机数
    return 2.0 * rand() / RAND_MAX - 1.0;
}

int main() {
    int total_points, inside_circle =0;
    double x, y, pi_estimate;
    // 读取输入
    scanf("% d", &total_points);
    // 初始化随机数生成器
    srand(time(NULL));
    // 投点模拟
    for (int i = 0; i < total_points; i++) {
        // 生成随机点 (x, y)
        x = random_double();
        y = random_double();

        // 判断点是否在圆内
        if (x * x + y * y <= 1) {
            inside_circle++;
        }
    }
    // 估算圆周率
    pi_estimate =4.0 * inside_circle / total_points;
    // 输出结果
    printf("% .5f\n", pi_estimate);
    return 0;
}
```

Python 代码：

```
import random

def random_double():
    # 生成一个 [-1, 1] 范围的随机数
    return 2.0 * random.random() - 1.0

def estimate_pi(total_points):
    inside_circle =0
    for _ in range(total_points):
        # 生成随机点 (x, y)
        x = random_double()
        y = random_double()

        # 判断点是否在圆内
        if x * x + y * y <= 1:
            inside_circle +=1
    # 估算圆周率
    pi_estimate =4.0 * inside_circle / total_points
    return pi_estimate
if __name__=="__main__":
    # 读取输入
    total_points =int(input())
    # 估算并输出结果
    pi_estimate = estimate_pi(total_points)
    print(f"{pi_estimate: .5f}")
```

2. 利用随机选择策略改进快速排序算法。

C/C++代码：

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

//交换两个元素
void swap(int* a, int* b) {
    int temp = * a;
    * a = * b;
    * b = temp;
}
```

```
//分区函数(以末尾元素为枢轴)
int partition(int arr[], int low, int high) {
    int pivot = arr[high]; // 枢轴值
    int i = low -1;         // 小于枢轴的区域边界

    for (int j = low; j < high; j++) {
        if (arr[j] <= pivot) {
            i++;
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i +1], &arr[high]); // 将枢轴放到正确位置
    return i + 1;
}

//随机选择枢轴并排序
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        // 随机选择枢轴
        int rand_idx = low + rand() % (high - low +1);
        swap(&arr[rand_idx], &arr[high]);

        // 分区并获取枢轴位置
        int pi = partition(arr, low, high);

        // 递归排序左右子数组
        quickSort(arr, low, pi -1);
        quickSort(arr, pi +1, high);
    }
}

int main() {
    // 初始化随机数种子
    srand(time(NULL));
    int n;
    scanf("% d", &n);
    int arr[n];
    for (int i = 0; i < n; i++) {
        scanf("% d", &arr[i]);
```

```
    }
    // 调用随机快速排序
    quickSort(arr, 0, n - 1);
    // 输出结果
    for (int i = 0; i < n; i++) {
        printf("% d", arr[i]);
        if (i < n - 1) printf(" ");
    }
    printf("\n");
    return 0;
}
```

Python 代码：

```
import random

#交换两个元素
def swap(arr, i, j):
    arr[i], arr[j] = arr[j], arr[i]

#分区函数(以末尾元素为枢轴)
def partition(arr, low, high):
    pivot = arr[high]  # 枢轴值
    i = low -1          # 小于枢轴的区域边界

    for j in range(low, high):
        if arr[j] <= pivot:
            i +=1
            swap(arr, i, j)

    swap(arr, i +1, high)  # 将枢轴放到正确位置
    return i + 1

#随机选择枢轴并排序
def quick_sort(arr, low, high):
    if low < high:
        # 随机选择枢轴
        rand_idx = random.randint(low, high)
        swap(arr, rand_idx, high)
        # 分区并获取枢轴位置
```

```
        pi = partition(arr, low, high)
        # 递归排序左右子数组
        quick_sort(arr, low, pi -1)
        quick_sort(arr, pi +1, high)

def main():
    # 输入数组大小
    n =int(input())
    # 输入数组元素
    arr =list(map(int, input().split()))
    # 调用随机快速排序
    quick_sort(arr, 0, n - 1)
    # 输出结果
    print(" ".join(map(str, arr)))

if __name__ == "__main__":
    main()
```

3. 给定能随机生成整数 1~5 的函数 rand5( ), 写出能随机生成整数 1~7 的函数 rand7( )。

C/C++代码：

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

//通过 rand5()* 5+rand5()产生 6、7、8 ..... 28、29、30 这 25 个数, 每个整数出现的概
//率相等, 取前面 21 个整数, 舍弃后面的 4 个整数,
//将{6, 7, 8}转化成 1, 将{9, 10, 11}转化成 2,
//{12, 13, 14}转化成 3, {15, 16, 17}转化成 4,
//{18, 19, 20}转化成 5, {21, 22, 23}转化成 6,
//{24, 25, 26}转化成 7.
int rand5(){
    return rand()% 5+1;
}
int main(){
    int a, i, j, n;
    srand((unsigned int)time(NULL)); //设置当前时间为种子
    scanf("% d", &n);
    for(i =1; i<=n; i++){
        while(1){
```

```
            a = rand5()* 5+rand5();
            if(a<=26)
                break;
        }
        j = a/3-1;
        printf("% d ", j);
    }
    //system("pause");
    return 0;
}
```

Python 代码：

```
import random
import time

#模拟 C 中的 rand5 函数
def rand5():
    return random.randint(1, 5)

#主程序
if __name__ == "__main__":
    # 设置随机种子, 基于当前时间
    random.seed(int(time.time()))
    # 读取输入 n
    n =int(input())
    # 生成并打印 n 个随机数
    for i in range(1, n + 1):
        while True:
            a = rand5() * 5 + rand5()
            if a <= 26:
                break
        j = a //3 - 1  # 使用 // 表示整数除法
        print(j, end=" ")
    # 换行(C 中 printf 后的空格不会自动换行, 这里显式添加)
    print()
```