算法分析第七次作业

算法分析题

**7-3题答案：**

**答：**

思路：使用 rand() 函数生成 1 到 n 之间的随机整数，通过循环检查确保生成的随机数不重复，将符合条件的随机数存入结果向量。

vector<int> solution(int m, int n) {

vector<int> ans;

for (int i = 0; i < m; i++) {

int key = rand() % n + 1;

while (find(ans.begin(), ans.end(), key) != ans.end()) {

key = rand() % n + 1;

}

ans.push\_back(key);

}

return ans;

}

**7-4题答案：**

**答：**

**思路：**化简原计算公式：

以24次随机试验为一次事件。第(365-K)次试验随机值取值范围是［1，365］，若实验值在［1，K］则为真，否则为假。若24次试验值都为真，则此次事件为真，否则为假，代码如下：

int solution(){

static default\_random\_engine eng;

static uniform\_int\_distribution<int> dis(1, 365);

return dis(eng);

}

int main(){

int t = 10;

for(int i = 1; i <= t; ++i){

int total = i \* 1e7;

int cnt = total;

for(int j = 0; j < total; ++j){

for(int k = 340; k <= 364; ++k){

if(solution() > k){

cnt --;

break;

}

}

}

cout << "total: " << total << endl << "answer: " << double(cnt)/total << endl;

}

return 0;

}

**7-5题答案：**

**答：**

思路：

如果n足够大，那么k的期望趋近为：

≈1.253

所以*=*=k

所以集合X的大小 |X|(也就是n)的估计值为:

n=

int main(int argc,char \*argv[]){

if(argc != 2){

fprintf(stderr,"usage %s <n> \n",argv[0]);

exit(0);

}

srand((unsigned int)time(NULL));

int n = atoi(argv[1]);

int num=5; //运行num次取平均值

int i,k=0;

for(i=0;i<num;i++){

memset(flag,0,sizeof(flag));

while(1){

k++;

int tmp = rand()%n + 1;//取{1,2,3,...,n}上的随机数

if(flag[tmp]==1)

break;

flag[tmp]=1;

}

}

k=k/num;

printf("the real value = %d, the predicted value = %d\n",n,(int)(2\*k\*k/PI));

return 0;

}

**7-9题答案：**

**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. 不存在，取值不能确定

**7-12题答案：**

**答：**

（1.证明

是一致的话考虑出错的概率：

t错，u错的情况下概率1/16。

t错，v错的情况下概率1/16。

除去三者同错的情况，1/16\*2-1/64 = 7/64。

第二种出错的情况只有t对，u错，且v错。概率为3/4\*1/4\*1/4= 3/64

所以正确的概率为1-7/64-3/64=27/32。

（2.

如果m(x)不是一致的，则110不能保证返回正确解（即a的m(x)与b的m(x)值有可能不相同），则正确概率有：

1/4·3/4·3/4+3/4·1/4·3/4+3/4·3/4·3/4=45/64=0.70

正确率有可能低于0.71。

**7-14题答案：**

**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，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，可以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，可以认为整个问题的答案确实是假。

LasVegasAlgorithm(ST X):{

While（TRUE）{

if （AlgorithmA(X)） return true；

if （！AlgorithmB(X)）return false；

}

}

算法实现题

**7-3题答案：**

**答：**

随机选择集合S中的元素与集合T中的元素进行比较，若随机选择很多次都能从集合T中找到与之对应的相等，则集合S和T相等。

函数设计:

f1()：从S中随机选择的数组元素x,测试集合T中是否有与之相等的元素，若有算法返回true，否则返回 false,表明集合S和T不相等。

f2()：重复调用函数f1()，调用过程中若f1()返回true则继续调用，否则可以判定集合S和T不相等，直接退出测试。

bool f1() {//单独一次

srand((unsigned)time(NULL));//用随机函数rand求0~n-1的随机数index

int index=rand()%n-1;//数组索引

for(int j=0; j<n; j++) {//找寻数组 T 中是否含有S[index]

if(T[j]==S[index])

return true;

}

return false;

}

bool f2() {

int x=(int)ceil(log(1/e));

for(int i=1; i<=x; i++) {//找很多次

if(!f1())//只要找到一个不相符的结果，说明Ｓ和T就是不想等的

return false;

}

return true;

}

**7-4题答案：**

**答：**

据矩阵互逆定义知道:AB=I

两个矩阵相乘为单位矩阵, 单位矩阵特点就是对角线元素为1, 其他未知元素为0。设计蒙特卡罗算法, 随机选取矩阵的A的第J行, 随机选取矩阵B的第K列, 对应元素相乘并进行累加。ans=AJ1×B1k+AJ2×B2k+...+AJN×BNk, 如果J==k,则表示对角线上元素为1,即判断 ans是否为 1;如果 J!=K,判断 ans是否为0。

int main()

{

srand(time(NULL));

int n; cin >> n;

for(int i = 1; i <= n; i++)

{

for(int j = 1; j <= n; j++) cin >> A[i][j];

}

for(int i = 1; i <= n; i++)

{

for(int j = 1; j <= n; j++) cin >> B[i][j];

}

int flag = 0;

for(int i = 1; i <= n; i++)

{

int J = rand() % n + 1, K = rand() % n + 1; // 蒙特卡罗随机选择

double cur = 0;

for(int j = 1; j <= n; j++)

{

cur += (A[J][j] \* B[j][K]);

}

if(J == K)

{

if(cur - 1 <= 1e-6) continue;

else flag = 1;

}

else

{

if(cur <= 1e-6) continue;

else flag = 1;

}

}

if(flag) cout << "NO" << endl;

else cout << "YES" << endl;

return 0;

}