**算法分析第7次作业**

算法分析题

7-3 准备好1~n的数列，循环m次产生m个相应下标之后的随机数，随机数对应的下标再与当前下标进行交换，最后数组的前m个数即为所需要的随机数。

vector<int> randomM(int *n*,int *m*)

{

    RandomNumber ran;

    vector<int> arr(*n*);

    for(int i=0;i<*n*;i++)

    {

        arr[i]=i+1;

    }

    for(int i=0;i<*m*;i++)

    {

        int k=i+(*n*-i-1)\*ran.fRandom();

        int temp=arr[i];

        arr[i]=arr[k];

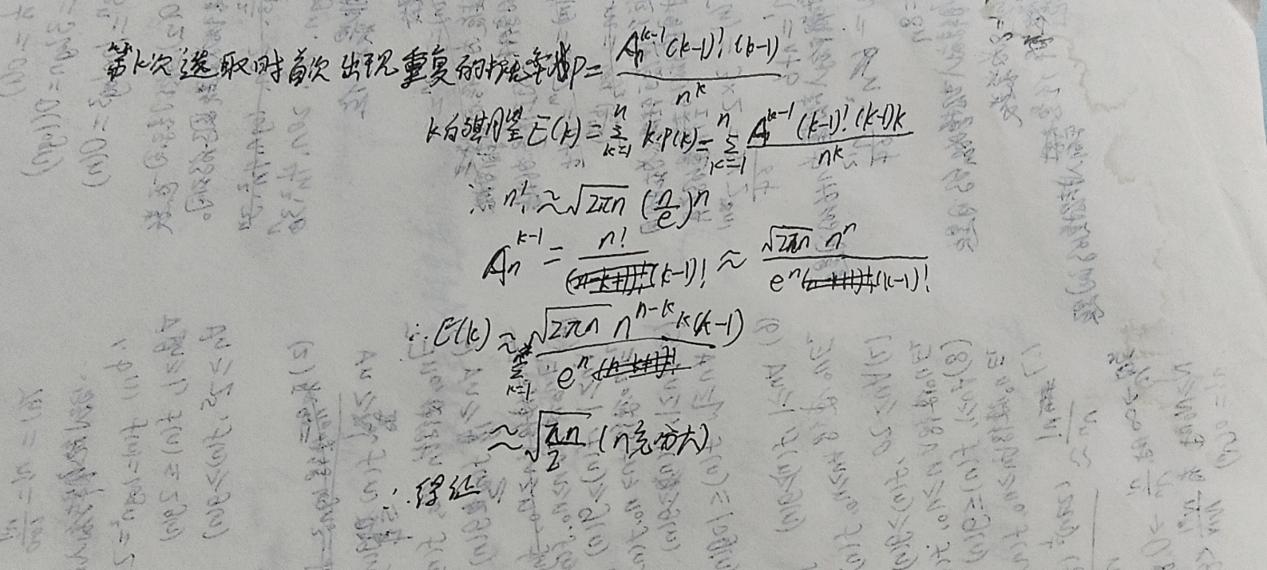
        arr[k]=temp;

    }

    return vector<int>(arr.begin(),arr.begin()+*m*);

}

初始化时遍历了数组，时间复杂度为O(n),空间复杂度为O(n)

7-4 （1）

（2）从集合中随机选取一个数，如果出现重复，则退出循环，循环次数即为k的期望，否则标记该数，继续循环。最后根据公式返回n的值。最差时间复杂度为O(n),最差空间复杂度为O(n)。

int countSet(set<int> &*s*)

{

    int k=0;

    set<int> x;

    RandomNumber ran;

    set<int>::iterator it=*s*.begin();

    advance(it,(*s*.size()-1)\*ran.fRandom());

    while(true)

    {

        x.insert(\*it);

        k++;

        advance(it,(*s*.size()-1)\*ran.fRandom());

        if(x.find(\*it)!=x.end())

        {

            break;

        }

    }

    return 2\*k\*k/3.1415926;

}

7-5 题目可以解释为25人中生日各不相同的概率。化简后为341\*342\*...\*364/(365^24)。传入参数为循环的迭代次数，只有当所有随机值都小于阈值k时，才算命中，最终返回的概率即为未命中的概率。

double birthdayQuestion(int *times*)

{

    RandomNumber ran;

    int cnt=*times*;

    for(int j=0;j<*times*;j++)

    {

        for(int k=340;k<=364;k++)

        {

            if(ran.Random(365)+1>k)

            {

                cnt--;

                break;

            }

        }

    }

    return (double)cnt/*times*;

}

7-9 （1）设xij棋盘格子（i，j）放置皇后的状态，如果为1表示放置，为0则表示没有放置。所以放置后<=n，要证明n>=4后有解，即证明n>=4时，=n。可分为三种情况讨论：

当n>=4,n%2=0,(n-2)%6>0时，对1<=j<=n/2，令x(j,2j)=1,x(n/2+j,2j-1)=1

当n>=4,n%2=0,n%6>0时，对1<=j<=n/2，令x(j,k(j))=1,x(n+1-j,n-k(j)+1)=1,其中，k(j)=(n/2+2\*(j-1)-1)%n

当n>4,n%2==1时，取x(n,n)=1,转换成(n-1)\*(n-1)棋盘问题，采用上述方法能够求解

算法实现如下

void build(int \**board*,int *n*)

{

    if(*n*<4)

        return;

    if(*n*&1==1)

    {

*board*[*n*]=*n*;

*n*--;

    }

    if((*n*-2)%6>0)

    {

        int k=*n*/2;

        for(int i=1;i<=k;i++)

        {

*board*[i]=2\*i;

*board*[k+i]=2\*i-1;

        }

    }

    else

    {

        for(int i=1;i<=*n*/2;i++)

        {

*board*[i]=(2\*(i-1)+*n*/2-1)%*n*;

*board*[*n*+1-i]=*n*-(2\*(i-1)+*n*/2-1)%*n*+1;

        }

    }

}

综上所述，对于n>=4，n后问题有解。

1. 不存在。

7-12 （1） 设mc(x)正确为1，错误为0，重复三次可得结果000,001,010,011,100,101,110,111，其中011,101,110,111返回正确结果，概率p=1/4\*3/4\*3/4+3/4\*1/4\*3/4+3/4\*3/4\*1/4+3/4\*3/4\*3/4=54/64=27/32≈84%。

得证。

（2） 如果mc(x)是不是一致的，则110不能保证返回正确结果，因此概率p=1/4\*3/4\*3/4+3/4\*1/4\*3/4+3/4\*3/4\*3/4=45/64≈70.3%<71%。

得证。

7-14 使算法A为真时一定为真，算法B为假时，一定为假。

bool las(T x)

{

While(true)

{

if(A(x))

return true;

if(!B(x))

return false;

}

}

算法实现题

7-3 循环有限次，随机从集合S中选取一个数，如果该数不在集合T中，则直接返回False，否则继续循环，表示不能确定两个集合相等，如果正常退出循环，可近似认为两个集合相等，因此这个算法是一个偏假的蒙特卡罗算法。传入的参数中包括循环的次数，时间复杂度最差为O(times)，空间复杂度为O(m+n)。

bool isSameSet(set<int> &*s*,set<int> &*t*,int *times*)

{

    RandomNumber ran;

    set<int> x;

    set<int>::iterator it=*s*.begin();

    for(int i=0;i<*times*;i++)

    {

        advance(it,(*s*.size()-1)\*ran.fRandom());

        if(*t*.find(\*it)==*t*.end())

        {

            return false;

        }

    }

    return true;

}

7-4 若A、B互为逆矩阵，当且仅当AB=I。循环有限次数，取随机值1<=i<=n,1<=j<=n，如果，i≠j，或，i=j，则直接返回false，否则继续循环，表示不能确定AB是否互为逆矩阵，如果正常退出循环，有较大概率AB互为逆矩阵，返回true，因此该算法是偏假的蒙特卡罗算法。传入的参数中包括循环的次数，时间复杂度最差为O(times\*n)，空间复杂度为O(n^2)。

bool isReverMatrix(int \*\**a*,int \*\**b*,int *n*,int *times*)

{

    RandomNumber ran;

    for(int t=0;t<*times*;t++)

    {

        int i=ran.Random(*n*)+1,j=ran.Random(*n*)+1;

        int temp=0;

        for(int k=1;k<=*n*;k++)

        {

            temp+=*a*[i][k]\**b*[k][j];

        }

        if(i==j&&temp!=1||i!=j&&temp!=0)

        {

            return false;

        }

    }

    return true;

}