**2018 ACM-ICPC 中国大学生程序设计竞赛线上赛**

宁夏那场的网络赛

题目大意就是给你一个偶数，让你把偶数分成两个素数和，输出任意一组答案即可

打表发现可能的分解情况中，较小的那个素数非常小，最大不过在一万左右

所以现在问题就变成了如何快速的判断一个数是否为素数

所以要用“Miller-Rabin素数检测算法”，具体参加如下博客

要注意的是题目给的数非常大，即使用long long存稍微算一下加法也会炸

所以要用unsigned long long 运算，输出用 %llu

#include<iostream>

#include<cstdio>

#include<cstdlib>

#include<time.h>

using namespace std;

typedef unsigned long long ll;

const int S=8;

//随机算法判定次数，一般8~10就够了

// 计算ret = (a\*b)%c a,b,c < 2^63

ll mult\_mod(ll a,ll b,ll c)

{

a%=c;

b%=c;

ll ret=0;

ll tmp=a;

while(b)

{

if(b&1)

{

ret+=tmp;

if(ret>c) ret-=c;//直接取模慢很多

}

tmp<<=1;

if(tmp>c) tmp-=c;

b>>=1;

}

return ret;

}

// 计算 ret = (a^n)%mod

ll pow\_mod(ll a,ll n,ll mod)

{

ll ret=1;

ll temp=a%mod;

while(n)

{

if(n&1) ret=mult\_mod(ret,temp,mod);

temp=mult\_mod(temp,temp,mod);

n>>=1;

}

return ret;

}

// 通过 a^(n-1)=1(mod n)来判断n是不是素数

// n-1 = x\*2^t 中间使用二次判断

// 是合数返回true, 不一定是合数返回false

bool check(ll a,ll n,ll x,ll t)

{

ll ret=pow\_mod(a,x,n);

ll last=ret;

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

ret=mult\_mod(ret,ret,n);

if(ret==1&&last!=1&&last!=n-1) return true;

last=ret;

}

if(ret!=1) return true;

else return false;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Miller\_Rabin算法

// 是素数返回true,(可能是伪素数)

// 不是素数返回false

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

bool miller\_rabin(ll n){

if(n<2) return false;

if(n==2) return true;

if((n&1)==0) return false;

ll x=n-1;

ll t=0;

while((x&1)==0) {

x>>=1;

t++;

}

srand(time(NULL));

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

long long a=rand()%(n-1)+1;

if(check(a,n,x,t))

return false;

}

return true;

}

int main()

{

int T;

ll n;

scanf("%d",&T);

while(T--)

{

scanf("%llu",&n);

for(ll i=0;i<n/2;i++)

if(miller\_rabin(i) && miller\_rabin(n-i))

{

printf("%llu %llu\n",i,n-i);

break;

}

}

return 0;

}