Rabin-Miller素数测试

前提定理

1.费马小定理：若 n为质数，a<n，则 an−1≡1 (mod n)；若任意整数x满足ax−1≡1 (mod x)，则x有很大概率为质数；

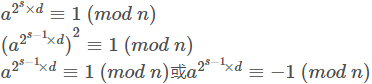
2.定理2：若p为大于2的质数，则a2≡1 (mod p)的解只能为a≡1或a≡−1；

Rabin-Miller素数测试

若待测数n为大于2的质数，则n−1为偶数

设n−1=2s×d

随机一个小于n的整数a



若为1，还可以继续递归，直到得到-1或者得到 ad≡±1 (mod n)

实现时，要倒着来

首先计算出ad，判断其是否为±1

然后不停的平方，得到

直到中途某一步得到了−1，则通过测试。

如果中途在得到-1之前，得到了1，则说明该数通过某一个不等于±1的数平方得到了1，不满足定理2，则该数测试不通过

一般都要随机取很多a，测试多次才能提高概率。

但是在OI中，用这一串a就行了

{2,3,5,7,11,13,17,19,23}{2,3,5,7,11,13,17,19,23}

它保证3,825,123,056,546,413,0513,825,123,056,546,413,051以内的数计算正确（已经超过long long）

代码

版题：HDU2138

#include<cstdio>

const int TEST\_VAL[]={2,3,5,7,11,13,17,19,23,29,31,37,41},TEST\_NUM=13;

long long PowMod(long long a,long long b,long long p)

{

long long res=1LL;

while(b)

{

if(b&1LL)

res=1LL\*res\*a%p;

a=1LL\*a\*a%p;

b>>=1LL;

}

return res;

}

bool MillerRabin(long long a,long long n,long long d)

{

long long x=PowMod(a,d,n);

if(x==1||x==n-1)

return true;

while(d<n-1)

{

x=1LL\*x\*x%n;

d=d\*2LL;

if(x==1)

return false;

if(x==n-1)

return true;

}

return false;

}

bool isPrime(long long x)

{

if(x<10)

{

if(x==2||x==3||x==5||x==7)

return true;

return false;

}

long long d=x-1;

while((d&1LL)==0)

d>>=1LL;

for(int i=0;i<TEST\_NUM;i++)

if(TEST\_VAL[i]<x&&!MillerRabin(TEST\_VAL[i],x,d))

return false;

return true;

}

int main()

{

int n,ans;

long long x;

while(scanf("%d",&n)!=EOF)

{

ans=0;

while(n--)

{

scanf("%I64d",&x);

ans+=isPrime(x);

}

printf("%d\n",ans);

}

return 0;

}