**算法训练 麦森数**

时间限制：1.0s   内存限制：256.0MB

问题描述

　　形如2*P*-1的素数称为麦森数，这时P一定也是个素数。但反过来不一定，即如果P是个素数，2*P*-1不一定也是素数。到1998年底，人们已找到了37个麦森数。最大的一个是P=3021377，它有909526位。麦森数有许多重要应用，它与完全数密切相关。  
　　任务：从文件中输入P（1000<P<3100000），计算2*P*-1的位数和最后500位数字（用十进制高精度数表示）

输入格式

　　文件中只包含一个整数P（1000<P<3100000）

输出格式

　　第一行：十进制高精度数2*P*-1的位数。  
　　第2-11行：十进制高精度数2*P*-1的最后500位数字。（每行输出50位，共输出10行，不足500位时高位补0）  
　　不必验证2*P*-1与P是否为素数。

样例输入

1279

样例输出

386  
00000000000000000000000000000000000000000000000000  
00000000000000000000000000000000000000000000000000  
00000000000000104079321946643990819252403273640855  
38615262247266704805319112350403608059673360298012  
23944173232418484242161395428100779138356624832346  
49081399066056773207629241295093892203457731833496  
61583550472959420547689811211693677147548478866962  
50138443826029173234888531116082853841658502825560  
46662248318909188018470682222031405210266984354887  
32958028878050869736186900714720710555703168729087

锦囊1

二分，高精度计算。

锦囊2

使用数组来保存答案的最后500位，实现乘法运算。计算幂时使用二分，则计算a^b先算a^(floor(b/2))，再平方一下，根据需要看是不是再乘a。

本题的C++参考代码如下：

#include<iostream>

#include<vector>

#include<cmath>

#include<iomanip>

using std::cin;

using std::cout;

using std::endl;

using std::vector;

using std::setw;

using std::setfill;

vector<int> Multiply(vector<int> ivec1,vector<int> ivec2)

{

vector<int> ivec(125,0);

for(int i=0;i!=125;++i)

{

int n=0;

for(int j=0;j!=125-i;++j)

{

int tmp=ivec[i+j]+ivec1[j]\*ivec2[i]+n;

ivec[i+j]=tmp%10000;

n=tmp/10000;

}

}

return ivec;

}

int main()

{

int p=0;

cin>>p;

cout<<(int)(p\*log10(2.0))+1<<endl;

vector<int> Boolvec;

while(p)

{

Boolvec.push\_back(p%2);

p/=2;

}

vector<int> ivec;

vector<int> Result;

ivec.push\_back(2);

Result.push\_back(1);

for(int i=1;i!=125;++i)

{

ivec.push\_back(0);

Result.push\_back(0);

}

for(int i=0;i<(int)Boolvec.size();++i)

{

if(Boolvec[i])

Result=Multiply(Result,ivec);

ivec=Multiply(ivec,ivec);

}

Result[0]--;

for(int i=124;i>=0;--i)

{

if(i%25==12)

{

cout<<setfill('0')<<setw(2)<<Result[i]/100;

cout<<endl;

cout<<setfill('0')<<setw(2)<<Result[i]%100;

}

else

{

if(i%25==0)

{

cout<<setfill('0')<<setw(4)<<Result[i];

cout<<endl;

}

else

cout<<setfill('0')<<setw(4)<<Result[i];

}

}

return 0;

}

本题的C参考代码如下：

#include <stdio.h>

#include <string.h>

#include <math.h>

#define LEN 125 //每数组元素存放十进制的4位，因此数组最多只要125个元素即可

//Multiply函数功能是计算高精度乘法a\*b，结果的末500位放在a中

void Multiply(int \*a,int \*b)

{

int i,j;

int nCarry; //存放进位

int nTmp;

int c[LEN]; //存放结果的末500位

memset(c,0,sizeof(int)\*LEN);

for(i=0;i<LEN;i++)

{

nCarry=0;

for(j=0;j<LEN-i;j++)

{

nTmp=c[i+j]+a[j]\*b[i]+nCarry;

c[i+j]=nTmp%10000;

nCarry=nTmp/10000;

}

}

memcpy(a,c,LEN\*sizeof(int));

}

int main()

{

int i,p;

int anPow[LEN]; //存放不断增长的2的次幂

int aResult[LEN]; //存放最终结果的末500位

scanf("%d",&p);

printf("%d\n",(int)(p\*log10(2))+1);

//下面将2的次幂初始化为2^(2^0)(a^b表示a的b次方),最终结果初始化为1

anPow[0]=2;

aResult[0]=1;

for(i=1;i<LEN;i++)

{

anPow[i]=0;

aResult[i]=0;

}

//下面计算2的p次方

while(p>0) //p=0则说明p中的有效位都用过了，不需要再计算下去

{

if(p&1) //判断此时p中最低位是否为1

Multiply(aResult,anPow);

p>>=1;

Multiply(anPow,anPow);

}

aResult[0]--; //2的p次方算出后减1

//输出结果

for(i=LEN-1;i>=0;i--)

{

if(i%25==12)

printf("%02d\n%02d",aResult[i]/100,aResult[i]%100);

else

{

printf("%04d",aResult[i]);

if(i%25==0)

printf("\n");

}

}

return 0;

}

本题的Java参考代码如下：

import java.io.\*;

import java.math.BigInteger;

public class Main {

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int p = Integer.parseInt(br.readLine());

String result;

BigInteger a = new BigInteger("2");

result = pow(a, p).toString();

int count = result.length();

for (int b = 0; b < 500 - count; b++)

result = 0 + result;

System.out.println((int) (p \* Math.log10(2)) + 1);

for (int b = 0; b < 10; b++)

System.out.println(result.substring(b \* 50, (b + 1) \* 50));

}

public static BigInteger pow(BigInteger a, int b) {

BigInteger x = new BigInteger("1");

String s;

int len;

while (b > 0) {

if (b % 2 == 1) {

x = x.multiply(a);

s = x.toString();

len = s.length();

if (len > 500) {

x = new BigInteger(s.substring(len - 500, len));

}

}

a = a.pow(2);

s = a.toString();

len = s.length();

if (len > 500) {

a = new BigInteger(s.substring(len - 500, len));

}

b /= 2;

}

return x.subtract(new BigInteger("1"));

}

}