习题1

* 1. **用于计算gcd(m,n)的欧几里得算法**

1.1.1.算法描述

辗转相除法，又名欧几里得算法（Euclidean algorithm），是求最大公约数(greater common divisor)的一种，通常做法是：用较小的数去除较大的数，用第二余数再去除第一余数，最终我们可以得到最终的余数为0以及最大公约数。

1.1.2.伪代码

Euclid(m,n)

//使用Euclid计算gcd(m,n)

//输入:两个不全为0的非负整数m,n

//输出:m,n的最大公约数

while n≠0 do

r ← m mod n

m← n

n ← r

return m

1.1.3.算法实现

int Euclid(int m,int n){

int r;

while(n!=0){

r=m%n;

m=n;

n=r;

}

return m;

}

* 1. **用于计算gcd(m,n)的枚举算法**

1.2.1.算法描述

枚举算法，是求最大公约数的一种，通常做法是：从1到自己本身进行遍历，如果说能够被整除，那么将这个数进行返回。

1.2.2.伪代码

[enumeration](D:/%E6%9C%89%E9%81%93%E8%AF%8D%E5%85%B8/Dict/8.5.3.0/resultui/html/index.html" \l "/javascript:;)(m,n)

//使用 [enumeration](D:/%E6%9C%89%E9%81%93%E8%AF%8D%E5%85%B8/Dict/8.5.3.0/resultui/html/index.html" \l "/javascript:;)计算gcd(m,n)

//输入:两个不全为0的非负整数m,n

//输出:m,n的最大公约数

for i 1 to n by incr 1 do

if ((n mod i == 0) and (m mod i == 0) ) then

ans = n

end if

return ans

1.2.3.算法实现

int  [enumeration](D:/%E6%9C%89%E9%81%93%E8%AF%8D%E5%85%B8/Dict/8.5.3.0/resultui/html/index.html" \l "/javascript:;)(int m,int n){

int res=1;

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

if(m%i==0&&n%i==0) res=i;

return res;

}

* 1. **实现****Eratosthenes筛选法**

1.3.1.算法描述

埃拉托色尼筛选法(sieve of Eratosthenes) ，是用来筛选素数(Prime)的一种方法，通常做法是：新建一个布尔类型的数组，从1到该数字的平方根(root)进行遍历，将自己本身的倍数变为1，那么，剩余为0的数字将是素数

1.3.2.伪代码

Eratos(n)

//使用sieve of Eratosthenes打印素数表

//输入：给定数字的最大区间

//输出：小于该数字的自然数的所有素数(从小到大)

np[n+1]

for i 1 to n+1 incr 1 do

while i\*j<=n do

Np[i\*j]=1

for i 2 to n+1 incr 1 do

if np[i]==0 then

print i

end if

1.3.3.算法实现

void Eratos(int n){

int np[n+1]={0};

for(int i=2;i\*i<=n;i++)

for(int j=2;j\*i<=n;j++)

np[j\*i]=1;

for(int i=2;i<n+1;i++)

if(np[i]==0) cout<<i<<" ";

}

* 1. **试验小结**

|  |  |  |
| --- | --- | --- |
| **算法** | **时间复杂度** | **空间复杂度** |
| 欧几里得算法 | O(logn) | O(1) |
| 枚举算法 | O(n) | O(1) |
| 埃拉托色尼筛选法 | O(nlogn) | O(1) |

表1-1

关于三种算法，时间空间复杂度分析如上表1-1，算法课第一节课我们学习了欧几里得和枚举两种可计算gcd的算法，然而，我们欧几里得算法仍然可以简化代码，简化为递归进行求解gcd，这样实现，时间复杂度并不会提高，而空间复杂度会提高。埃氏筛法和传统素数求解不一样，传统素数求解需要O(n^2)的时间复杂度，这种筛法大大提高了求解素数的效率。