算法竞赛中的数学问题——基础知识

东北育才学校　张听海

February 5, 2018

1. 快速幂
   - 快速幂
2. 素数筛法
3. 两个重要定理
4. 欧几里得算法
5. 三分法

1 快速幂
- 快速幂

2 素数筛法

3 两个重要定理

4 欧几里得算法

5 三分法

快速幂

```
int ans=1;
while(n){
    if(n&1) ans=ans*x%p;
    n>>=1;
    x=x*x%p;
}
```

快速幂

```cpp
int ans=1;
while(n){
    if(n&1) ans=ans*x%p;
    n>>=1;
    x=x*x%p;
}
```

时间复杂度: $O(\log n)$.

1. 快速幂
2. 素数筛法
   - 朴素的素数判断
   - Eratosthenes 筛法
   - 线性筛法
3. 两个重要定理
4. 欧几里得算法
5. 三分法

1 快速幂

2 素数筛法

- 朴素的素数判断
- Eratosthenes 筛法
- 线性筛法

3 两个重要定理

4 欧几里得算法

5 三分法

朴素的素数判断

```
for(int i=2; i<=n; i++){
    bool flag=1;
    for(int j=2; j<=sqrt(i+0.5); j++)
        if(i%j==0) flag=0;
    ...
}
```

朴素的素数判断

```
for(int i=2; i<=n; i++){
    bool flag=1;
    for(int j=2; j<=sqrt(i+0.5); j++)
        if(i%j==0) flag=0;
    ...
}
```

时间复杂度：$O\left(n\sqrt{n}\right)$.

1. 快速幂
2. 素数筛法
   - 朴素的素数判断
   - Eratosthenes 筛法
   - 线性筛法
3. 两个重要定理
4. 欧几里得算法
5. 三分法

Eratosthenes 筛法

```
int m=sqrt(n+0.5);
memset(vst,0,sizeof(vst));
for(int i=2; i<=m; i++)
    if(!vst[i])     //!vst[i]表示i为素数.
        for(int j=i*i; j<=n; j+=i) vst[j]=1;
```

Eratosthenes 筛法

```
int m=sqrt(n+0.5);
memset(vst,0,sizeof(vst));
for(int i=2; i<=m; i++)
    if(!vst[i])     //!vst[i]表示i为素数.
        for(int j=i*i; j<=n; j+=i) vst[j]=1;
```

特点：从质因子的角度考虑，显然质因子的数目比正整数的数目少. 这样就大大降低了时间复杂度.

Eratosthenes 筛法

```
int m=sqrt(n+0.5);
memset(vst,0,sizeof(vst));
for(int i=2; i<=m; i++)
    if(!vst[i])     //!vst[i]表示i为素数.
        for(int j=i*i; j<=n; j+=i) vst[j]=1;
```

特点：从质因子的角度考虑，显然质因子的数目比正整数的数目少. 这样就大大降低了时间复杂度.

时间复杂度：$O(n\log\log n)$.

1 快速幂

2 素数筛法
- 朴素的素数判断
- Eratosthenes 筛法
- 线性筛法

3 两个重要定理

4 欧几里得算法

5 三分法

线性筛法

```
bool check[MAXN]={0};
int prime[MAXN]={0};
int tot=0;
for(int i=2; i<=N; i++){
    if(!check[i]) prime[tot++]=i;
    for(int j=0; j<tot&&i*prime[j]<=N; j++){
        check[i*prime[j]]=1;
        if(i%prime[j]==0) break;    //关键在于理解此句.
    }
}
```

线性筛法

```
bool check[MAXN]={0};
int prime[MAXN]={0};
int tot=0;
for(int i=2; i<=N; i++){
    if(!check[i]) prime[tot++]=i;
    for(int j=0; j<tot&&i*prime[j]<=N; j++){
        check[i*prime[j]]=1;
        if(i%prime[j]==0) break;    //关键在于理解此句.
    }
}
```

如果 $p_j \mid i_0$，那么对于 p_j 后的任意 p_k，$p_j \mid i_0p_k$，我们不妨让 i_0p_k 在 $i=\frac{i_0p_k}{p_j}(>i_0)$ 时被 ip_j 筛掉. 这样就避免了重复筛掉同一个正整数. 事实上，这样做之后，每个正整数 i 都会被它的最小质因子筛掉.

线性筛法拓展——求欧拉函数

```
bool check[MAXN]={0};
int prime[MAXN]={0};
int tot=0;
int phi[MAXN]={0};
phi[1]=1;     //不要忘记.
for(int i=2; i<=N; i++){
    if(!check[i]){
        prime[tot++]=i;
        phi[i]=i-1;
    }
    for(int j=0; j<tot&&i*prime[j]<=N; j++){
        check[i*prime[j]]=1;
        if(i%prime[j]==0){
            phi[i*prime[j]]=phi[i]*prime[j];
            break;
        }
        else phi[i*prime[j]]=phi[i]*(prime[j]-1);
    }
}
```

1 快速幂

2 素数筛法

3 两个重要定理
- 费马小定理
- 欧拉定理

4 欧几里得算法

5 三分法

1. 快速幂
2. 素数筛法
3. 两个重要定理
   - 费马小定理
   - 欧拉定理
4. 欧几里得算法
5. 三分法

费马小定理

已知正整数 a 和质数 p，且 $\gcd(a,p)=1$，则 $a^{p-1}\equiv 1 \pmod{p}$.

费马小定理

已知正整数 a 和质数 p，且 $\gcd(a,p)=1$，则 $a^{p-1}\equiv 1 \pmod p$.

应用：结合快速幂求乘法逆元，即 a 模 p 的乘法逆元为 a^{p-2}.

1. 快速幂
2. 素数筛法
3. 两个重要定理
   - 费马小定理
   - 欧拉定理
4. 欧几里得算法
5. 三分法

欧拉定理

若正整数 a,n 互质，则 $a^{\varphi(n)} \equiv 1 \pmod{n}$.

欧拉定理

若正整数 a,n 互质，则 $a^{\varphi(n)} \equiv 1 \pmod{n}$.

广义欧拉定理：对于任何整数 x 和 $n \geqslant \varphi(k)$，则

$$x^n \equiv x^{n\%\varphi(k)+\varphi(k)} \pmod{k}.$$

1 快速幂

2 素数筛法

3 两个重要定理

4 欧几里得算法
- 普通的欧几里得算法
- 拓展欧几里得算法

5 三分法

普通的欧几里得算法

```
int gcd(int a,int b){
    if(b==0) return a;
    else return gcd(b,a%b);
}
```

1 快速幂

2 素数筛法

3 两个重要定理

4 欧几里得算法
- 普通的欧几里得算法
- 拓展欧几里得算法

5 三分法

拓展欧几里得算法

裴蜀定理：若正整数 a,b 满足 $\gcd(a,b)=d$，则必存在整数 x,y 满足 $ax+by=d$.

拓展欧几里得算法

裴蜀定理: 若正整数 a,b 满足 $\gcd(a,b)=d$, 则必存在整数 x,y 满足 $ax+by=d$.

特别地, 正整数 a,b 互质的充要条件为方程

$$ax+by=1$$

有整数解 (x,y).

拓展欧几里得算法

```
void gcd(int a,int b,int& d,int& x,int& y){
    if(b==0) {d=a,x=1,y=0;}
    else{
        gcd(b,a%b,d,y,x);
        y-=x*(a/b);     //关键在于理解此句.
    }
}
```

拓展欧几里得算法

```
void gcd(int a,int b,int& d,int& x,int& y){
    if(b==0) {d=a,x=1,y=0;}
    else{
        gcd(b,a%b,d,y,x);
        y-=x*(a/b);     //关键在于理解此句.
    }
}
```

已知正整数 a, b 满足 $\gcd(a, b) = d$, 且整数 x_0, y_0 满足 $ax_0 + by_0 = d$.

拓展欧几里得算法

```
void gcd(int a,int b,int& d,int& x,int& y){
    if(b==0) {d=a,x=1,y=0;}
    else{
        gcd(b,a%b,d,y,x);
        y-=x*(a/b);     //关键在于理解此句.
    }
}
```

已知正整数 a,b 满足 $\gcd(a,b)=d$，且整数 x_0,y_0 满足 $ax_0+by_0=d$. 设 $a/b=p, a\%b=r$，这里 "/" 按计算机中的整除理解. 则有 $(bp+r)x_0+by_0=d$，整理得 $rx_0+b(px_0+y_0)=d$.

拓展欧几里得算法

```
void gcd(int a,int b,int& d,int& x,int& y){
    if(b==0) {d=a,x=1,y=0;}
    else{
        gcd(b,a%b,d,y,x);
        y-=x*(a/b);     //关键在于理解此句.
    }
}
```

已知正整数 a,b 满足 $\gcd(a,b)=d$, 且整数 x_0,y_0 满足 $ax_0+by_0=d$. 设 $a/b=p, a\%b=r$, 这里 “/” 按计算机中的整除理解. 则有 $(bp+r)x_0+by_0=d$, 整理得 $rx_0+b(px_0+y_0)=d$.

当从下一层递归中回到上一层时, 传回的参数为 $x=x_0, y=px_0+y_0$. 而在上一层递归中我们想得到 $x=x_0, y=y_0$, 因此需要 `y-=x*(a/b)`.

应用：求乘法逆元

(1) 乘法逆元：若 $ab \equiv 1 \pmod{p}$，则 a, b 互为模 p 的乘法逆元.

应用：求乘法逆元

(1) 乘法逆元：若 $ab \equiv 1 \pmod{p}$，则 a, b 互为模 p 的乘法逆元.

(2) 在扩展欧几里得算法中取 $b = p$，得到的 x 即为 a 模 p 的乘法逆元.

1 快速幂

2 素数筛法

3 两个重要定理

4 欧几里得算法

5 三分法

- 三分法

三分法

```
double f(double x);
double solve(double l, double r){
    double x1=l+(r-l)/3;
    double x2=r-(r-l)/3;
    if(fabs(x1-x2)<0.0001) return f(x1);
    if(f(x1)>f(x2)) return solve(l,x2);
    else return solve(x1,r);
}
```

谢谢大家

谢谢大家.