Number Theory Algorithms

vici

Northeast Normal University

March 14, 2013

最大公约数

递归

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

最大公约数

迭代

```
int gcd(int a, int b) {
  while (b) {
    a %= b;
    swap(a, b);
  }
  return a;
}
```

最大公约数

STL

```
#include <algorithm>
...



__gcd(a, b);
```

最大公约数

浮点数最大公约数

```
#define eps 1e-8

double fgcd(double a, double b) {
  if (b > -eps && b < eps) {
    return a;
  } else {
    return fgcd( b, fmod(a, b) );
  }
}
```

最大公约数

扩展欧几里得

```
int ext_gcd(int a, int b, int &x, int &y) {
  int t, ret;
  if (!b) {
    x = 1, y = 0;
    return a;
  }
  ret = ext_gcd(b, a % b, x, y);
  t = x, x = y, y = t - a / b * y;
  return ret;
}
```

最大公约数

Example

给出整数a, b，以及四种操作：

- 将一个数$+x$
- 将一个数$-x$
- 将一个数$+y$
- 将一个数$-y$

现在想让a变成b，是否可以实现？若可以求出最小操作次数。

筛法及其应用

处理器Intel Core i5-2430M @ 2.40Ghz

筛$1 \sim 10000000$的素数（总数：664579）

筛素数算法比较

方法	用时
朴素暴力	8515ms
暴力+优化	1694ms
朴素的Eratosthenes筛	339ms
优化的Eratosthenes筛	202ms
均摊的Eratosthenes筛	139ms
均摊的Eratosthenes筛+常数优化	79ms

筛法及其应用

埃拉托色尼筛

```
int const maxn = 1e7;
int pri[maxn], mark[maxn], cnt;
//mark[i]: the minimum factor of i (when prime, mark[i]=i)
void sieve() {
  cnt = 0, mark[0] = mark[1] = 1;
  for (int i = 2; i < maxn; ++i) {
    if (!mark[i]) pri[cnt++] = mark[i] = i;
    for (int j = 0; pri[j] * i < maxn; ++j) {
      mark[i * pri[j]] = pri[j];
      if (i % pri[j] == 0) break;
    }
  }
}
```

筛法及其应用

线性求约数个数

```
int nod[maxn];
void sieve_nod() {
  for (int i = 1; i < maxn; ++i)
    for (int j = i; j < maxn; j += i)
      ++nod[j];
}
```

线性求约数个数O(n)

```
int pri[maxn], e[maxn], divs[maxn], cnt;
void sieve_nod() {
  cnt = 0, divs[0] = divs[1] = 1;
  for (int i = 2; i < maxn; ++i) {
    if (!divs[i]) divs[i] = 2, e[i] = 1, pri[cnt++] = i;
    for (int j = 0; i * pri[j] < maxn; ++j) {
      int k = i * pri[j];
      if (i % pri[j] == 0) {
        e[k] = e[i] + 1, divs[k] = divs[i] / (e[i] + 1) * (e[i] + 2);
        break;
      } else {
        e[k] = 1, divs[k] = divs[i] << 1;
      }
    }
  }
}
```

筛法及其应用

线性求phi(O(log(n))

```
int pri[maxn], phi[maxn], cnt;
void sieve_phi() {
    cnt = 0;
    for (int i = 1; i < maxn; ++i) phi[i] = i;
    for (int i = 2; i < maxn; ++i) {
        if (phi[i] == i) {
            pri[cnt++] = i;
            for (int j = i; j < maxn; j += i) {
                phi[j] = phi[j] / i * (i - 1);
            }
        }
    }
}
```

筛法及其应用

线性求phi(O(n))

```
int pri[maxn], phi[maxn], cnt;
void sieve_phi() {
  cnt = 0, phi[1] = 1;
  for (int i = 2; i < maxn; ++i) {
    if (!phi[i]) pri[cnt++] = i, phi[i] = i - 1;
    for (int j = 0; pri[j] * i < maxn; ++j) {
      if (!(i % pri[j])) {
        phi[i * pri[j]] = phi[i] * pri[j];
        break;
      } else {
        phi[i * pri[j]] = phi[i] * (pri[j] - 1);
      }
    }
  }
}
```

筛法及其应用

Example (筛10^{12}之后的100000个素数)

由素数密度函数$\pi(x) \sim \frac{x}{\log x}$，对于$10^{12}$之后的1e5个素数，只需筛300万左右即可

定义start = 1e12, end = 1e12 + 3e6;

- 先筛出$1 \sim \sqrt{end}$的素数
- 用这些素数对[start, end]之间的数进行Eratosthenes筛
- 扫描[start, end]的mark数组，得到结果

Example (筛10^{12}之后的100000个素数)

由素数密度函数$\pi(x) \sim \frac{x}{\log x}$，对于$10^{12}$之后的1e5个素数，只需筛300万左右即可

定义start = 1e12, end = 1e12 + 3e6;

- 先筛出$1 \sim \sqrt{end}$的素数
- 用这些素数对[start, end]之间的数进行Eratosthenes筛
- 扫描[start, end]的mark数组，得到结果

或者用java BigInteger的*nextProbablePrime()*。

筛法及其应用

筛10^{12}之后的100000个素数

```
typedef long long ll;
int const maxn = 4e6; ll const start = 1e12, end = start + 3e6;
ll pl[maxn]; int pnt; bool markl[maxn];
void sieve_large() {
    sieve();
    ll pos;
    for (int i = 0; i < cnt; ++i) {
        if (start % pri[i] == 0) pos = start;
        else pos = start - start % pri[i] + pri[i];
        for (; pos <= end; pos += pri[i]) markl[pos - start] = true;
    }
    pnt = 0;
    for (int i = 0; i <= end - start; ++i) if (!markl[i])
        pl[pnt++] = start + i;
}
```

快速幂取模

快速幂

```
long long powMod(long long a, long long b, long long c) {
  long long ret = 1 % c;
  for (; b; a = a * a % c, b >>= 1)
    if (b & 1) ret = ret * a % c;
  return ret;
}
```

快速幂取模

Example (求$a^{b^c} \mod p$, (p是素数))

快速幂取模

Example (求$a^{b^c} \mod p$，(p是素数))

由欧拉定理$a^{\varphi(n)} \equiv 1 \pmod{n}$，$\gcd(a, n) = 1$，所以$a^{b^c} \bmod p = a^{(b^c \bmod \varphi(p))} \bmod p$
通过快速幂可解。

快速幂取模

快速幂plus

```
long long mulMod(long long a, long long b, long long c) {
  long long ret = 0;
  for (; b; a = (a << 1) % c, b >>= 1)
    if (b & 1) ret = (ret + a) % c;
  return ret;
}
long long powMod(long long a, long long b, long long c) {
  long long ret = 1 % c;
  for (; b; a = mulMod(a, a, c), b >>= 1)
    if (b & 1) ret = mulMod(ret, a, c);
  return ret;
}
```

素数判定法

Miller-Rabin

```
bool suspect(long long a, int s, long long d, long long n) {
  long long x = powMod(a, d, n);
  if (x == 1) return true;
  for (int r = 0; r < s; ++r) {
    if (x == n - 1) return true;
    x = mulMod(x, x, n);
  }
  return false;
}
int const test[] = {2,3,5,7,11,13,17,19,23,-1}; // for n < 10^16
bool isPrime(long long n) {
  if (n <= 1 || (n > 2 && n % 2 == 0)) return false;
  long long d = n - 1, s = 0;
  while (d % 2 == 0) ++s, d /= 2;
  for (int i = 0; test[i] < n && ~test[i]; ++i)
    if (!suspect(test[i], s, d, n)) return false;
  return true;
}
```

素数判定法

Pollard-Rho找到一个n的约数

```
long long pollard_rho(long long n, long long c) {
  long long d, x = rand() % n, y = x;
  for (long long i = 1, k = 2; ; ++i) {
    x = (mulMod(x, x, n) + c) % n;
    d = gcd(y - x, n);
    if (d > 1 && d < n) return d;
    if (x == y) return n;
    if (i == k) y = x, k <<= 1;
  }
  return 0;
}
```

欧拉函数

Definition

$\varphi(n) = p_1^{a_1-1} p_2^{a_2-1} ... p_s^{a_s-1}(p_1-1)(p_2-1)...(p_s-1)$

$\varphi(n) = n \cdot \frac{(p_1-1)}{p_1} \cdot \frac{(p_2-1)}{p_2} \cdot ... \cdot \frac{(p_s-1)}{p_s}$

欧拉函数

欧拉函数

```
int phi(int n) {
  int ret = n;
  for (int i = 2; i * i <= n; i += (i == 2) ? 1 : 2) {
    if (!(n % i)) {
      ret = ret / i * (i - 1);
      while (n % i == 0) n /= i;
    }
  }
  if (n > 1) ret = ret / n * (n - 1);
  return ret;
}
```