Number Theory Cheat Sheet *vici@nenu*

```cpp
//gcd & lcm
int gcd(int a, int b) { return b ? gcd(b, a % b) : a;}
int lcm(int a, int b) { return a / gcd(a, b) * b; }
double fgcd(double a, double b) {
  if(b > -eps && b < eps) return a;
  else return fgcd( b, fmod(a, b) );
}
//extgcd
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;
}
//b/a%c
int inv(int a, int b, int c) {
  int x, y;
  ext_gcd(a, c, x, y);
  return (1LL * x * b % c + c) % c;
}
//inv_init
void inv_init() {
  inv[1] = 1;
  for (int i = 2; i < maxn; ++i)
    inv[i] = inv[mod % i] * (mod - mod / i) % mod;
}
//sieve primes
//mark[i]: the minimum factor of i (when prime, mark[i]=i)
int pri[maxn], mark[maxn], cnt;
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;
    }
  }
}
//sieve the number of divisors (O(nlogn))
int nod[maxn];
void sieve_nod() {
  for (int i = 1; i < maxn; ++i)
    for (int j = i; j < maxn; j += i)
      ++nod[j];
}
//sieve the number of divisors (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;
      }
    }
  }
}
//sieve phi
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);
      }
    }
  }
}
//powMod
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;
}
//powMod 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;
}
//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;
}
//Pollard-Rho
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;
}
//find factors
int facs[maxf];
int find_fac(int n) {
  int cnt = 0;
  for(int i = 2; i * i <= n; i += 2) {
    while (!(n % i))
      n /= i, facs[cnt++] = i;
    if (i == 2) --i;
  }
  if (n > 1) facs[cnt++] = n;
  return cnt;
}
//find factors plus (sieve() first & (n < maxn))
int facs[maxf];
int find_fac(int n) {
  int cnt = 0;
  while (mark[n] != 1)
    facs[cnt++] = mark[n], n /= mark[n];
  return cnt;
}
//phi
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;
}
//phi plus (sieve() first & (n < maxn))
int phi(int n) {
  int ret = n, t;
  while ((t = mark[n]) != 1) {
    ret = ret / t * (t - 1);
    while (mark[n] == t) n /= mark[n];
  }
  return ret;
}
//the number of divisors (from 1 to n)
int d_func(int n) {
  int ret = 1, t;
  for (int i = 2; i * i <= n; i += (i == 2) ? 1 : 2) {
    if (!(n % i)) {
      t = 1;
      while (!(n % i)) ++t, n /= i;
      ret *= t;
    }
  }
  return n > 1 ? ret << 1 : ret;
}
//the sum of all divisors (from 1 to n)
int ds_func(int n) {
  int ret = 1, m = n, t;
  for (int i = 2; i * i <= n; i += 2) {
    if (!(n % i)) {
      t = i * i, n /= i;
      while (!(n % i)) t *= i, n /= i;
      ret *= (t - 1) / (i - 1);
    }
    if (i == 2) --i;
  }
  return n > 1 ? ret * (n + 1) : ret;
}
```

$a \perp b \Rightarrow a^{\varphi(b)} \equiv 1 \pmod{m}$; $a^n \equiv a^{\varphi(m)+n\%\varphi(m)} \pmod{m}$;

$\sum_{d|n} \varphi(d) = \sum_{d|n} \varphi\left(\frac{n}{d}\right) = n$; $\left(2^a-1, 2^b-1\right) = 2^{(a,b)} - 1$;

$\pi(x) \sim \frac{x}{\log x}$; $p_n \sim n \log n$; if $p^a \parallel n!$, $a = [\frac{n}{p}] + [\frac{n}{p^2}] + [\frac{n}{p^3}] + \ldots$;

exp of p in $n!$ is $\sum_{i \geq 1} \left[\frac{n}{p^i}\right]$; p is prime $\Rightarrow \frac{b}{a} \equiv b \cdot a^{p-2} \pmod{p}$;

$\sum_{m \perp n, m<n} m = \frac{n\varphi(n)}{2}$; $r = ord_n(a), ord_n(a^u) = \frac{r}{\gcd(r,u)}$;

$a \perp n \Rightarrow a^i \equiv a^j \pmod{n} \Leftrightarrow i \equiv j \pmod{ord_n(a)}$;