Pontificia Universidad Católica del Perú - FCI

XieXieLucas Notebook - Froz/Phibrain/Ands

November 9, 2017

Contents

1 Chinese Remainder Theorem

```
/**
 * Chinese remainder theorem.
 * Find z such that z % x[i] = a[i] for all i.
 * */
long long crt(vector<long long> &a, vector<long long> &x) {
  long long z = 0;
  long long n = 1;
  for (int i = 0; i < x.size(); ++i)
    n *= x[i];

  for (int i = 0; i < a.size(); ++i) {
    long long tmp = (a[i] * (n / x[i])) % n;
    tmp = (tmp * mod_inv(n / x[i], x[i])) % n;
    z = (z + tmp) % n;
  }

  return (z + n) % n;
}
```

2 Cribas

```
// Criba en O(n)
// p[i] indica el valor del primo i-esimo
// A[i] indica que el menor factor primo de i
//                    es el primo A[i] - esimo
#define MAXN 100000

int A[MAXN + 1], p[MAXN + 1], pc = 0;

void sieve()
{
    for(int i=2; i<=MAXN; i++){
        if(!A[i]) p[A[i] = ++pc] = i;
        for(int j=1; j<=A[i] && i*p[j]<=MAXN; j++)
            A[i*p[j]] = j;
    }
}

//Criba para phi
int phi[MAX];
```

```
void CribaEuler(){
	REP(i,0, MAX) primo[i] = 1, phi[i] = 1;
	primo[0] = primo[1] = false;
	REP(i,2,MAX){
		if(primo[i]){
			phi[i] = i - 1;
			for(int j = i+i; j < MAX; j += i){
				primo[j] = false;
				int pot = 1, aux = j/i;
				while( aux % i == 0 ){
					aux /= i, pot *= i;
				}
				phi[j] *= (i-1)*pot ;
			}
		}
	}
}
```

3 Euler Totient

```
//Euler Totient
//Finds all k <= n where gcd(k,n) == 1

int phi(int n){
	int res = n;
	for(int p = 2; p*p <= n; p++){
		if(n % p == 0){
			while(n % p == 0)
				n/=p;
			res -= res/p;
		}
	}
	if (n > 1) res -= res/n;
	return res;
}
```

4 Inverso Modular

```
/** Inverso Modular **/
#define MAX 100
#define MOD 1000000009

long long inverso[MAX];

void inv(){
	inverso[1] = 1;
	REP(i,2,MAX) inverso[i] = ( (MOD-MOD/i) * inverso[MOD%i] ) % MOD;
}
```

5 Miller Rabin

```
//Miller-Rabin primality test

ll pow(ll a, ll b, ll c){
    ll ans = 1;
    while(b){
        if(b&1) ans = (1LL*ans*a)%c;
        a = (1LL*a*a)%c;
        b >>=1;
    }
    return ans;
}

bool miller(ll p, ll it = 10){
    if(p<2) return 0;
    if(p!=2 && (p&1) == 0) return 0;
    ll s=p-1;
    while((s&1) == 0) s>>=1;
    while(it--){
        ll a = rand()%(p-1)+1, temp = s;
        ll mod = pow(a,temp,p);
        while(temp!= p-1 && mod!=1 && mod!=p-1){
            mod = (1LL*mod*mod)%p;
            temp<<=1;
        }
        if(mod!=p-1 && (temp&1) == 0) return 0;
    }
    return 1;
}
```

6 Number Theory

```
// Encuentra el menor positivo de la forma ax+ b , my+ n
// ( x,y enteros no necesariamente postivos)
// Si son positivos, hallar los coeficientes y sumar lo
// que falta para que de positivo

// d: mcd(a,b) ( d > 0 )
// x,y enteros tales que a*x + b*y = d
// las demas soluciones son ( x + (b/d)t , y - (a/d)t )
void gcdextend(ll a, ll b, ll &x, ll &y, ll &d){
   if( b == 0){
      if(a>0) x = 1, y = 0 , d = a ;
      else x = -1 , y = 0 , d = -a ;
      return ;
   }
   gcdextend(b,a%b,x,y,d) ;
   ll x1 = y , y1 = x - (a/b)*y ;
   x = x1 , y = y1 ;
}


// menor positivo que es u modulo modPos
inline ll ADDTOPOSITIVE(ll u, ll modPOS){
   if(modPOS < 0) modPOS = -modPOS ;
   if(u >= 0) return u%modPOS;
   u = -u ;
   if(u%modPOS == 0) return 0;
   return modPOS*((u/modPOS)+1) - u ;
}

// Encuentra el menor positivo que es
// de la forma ax + b , my + n
// los demas son de la forma ans + ((a/d)*m)*t
// retorna -1 si no hay solucion ( mcd(a,m) no divide a n - b )
inline ll FINDmenorLCS(ll a,ll b,ll m,ll n){
    //Cuidado con el caso a = 0 , m = 0 ,
         //porque es solo verificar b = n

    a = abs(a) ; m = abs(m) ;
    if( a == 0 ) {
       swap(a,m);
       swap(b,n) ;
    }
    if( m == 0 ){
       if( (n-b) % a == 0) return n ;
       else return -1 ;
    }
    ll x , y , d ;
    gcdextend(a,m,x,y,d) ;
    if((n-b)%d != 0) return -1 ;
    ll temp = a*x*((n-b)/d) + b ;
    temp = ADDTOPOSITIVE(temp,a*(m/d)) ;
    return temp ;
}

//Finds partition of n (number of ways to obtain n as a sum of positive
    numbers)

int partition(int n) {
  int[] dp = new int[n + 1];
  dp[0] = 1;
  for (int i = 1; i <= n; i++) {
    for (int j = 1, r = 1; i - (3 * j * j - j) / 2 >= 0; j++, r *= -1) {
      dp[i] += dp[i - (3 * j * j - j) / 2] * r;
      if (i - (3 * j * j + j) / 2 >= 0) {
        dp[i] += dp[i - (3 * j * j + j) / 2] * r;
      }
    }
  }
  return dp[n];
}
```

7 Pollard Rho

```
#define MAXL (50000>>5)+1
#define GET(x) (mark[x>>5]>>(x&31)&1)
#define SET(x) (mark[x>>5] |= 1<<(x&31))

int mark[MAXL];
int P[50000], Pt = 0;
void sieve() {
   register int i, j, k;
   SET(1);
   int n = 46340;
   for (i = 2; i <= n; i++) {
```

```
        if (!GET(i)) {
            for (k = n/i, j = i*k; k >= i; k--, j -= i)
                SET(j);
            P[Pt++] = i;
        }
    }
}
ll mul(unsigned ll a, unsigned ll b, unsigned ll mod) {
    ll ret = 0;
    for (a %= mod, b %= mod; b != 0; b >>= 1, a <<= 1, a = a >= mod ? a -
         mod : a) {
        if (b&1) {
            ret += a;
            if (ret >= mod)   ret -= mod;
        }
    }
    return ret;
}
void exgcd(ll x, ll y, ll &g, ll &a, ll &b) {
    if (y == 0)
        g = x, a = 1, b = 0;
    else
        exgcd(y, x%y, g, b, a), b -= (x/y) * a;
}

ll inverse(ll x, ll p) {
    ll g, b, r;
    exgcd(x, p, g, r, b);
    if (g < 0) r = -r;
    return (r%p + p)%p;
}

ll mpow(ll x, ll y, ll mod) { // mod < 2^32
    ll ret = 1;
    while (y) {
        if (y&1)
            ret = (ret * x)%mod;
        y >>= 1, x = (x * x)%mod;
    }
    return ret % mod;
}

ll mpow2(ll x, ll y, ll mod) {
    ll ret = 1;
    while (y) {
        if (y&1)
            ret = mul(ret, x, mod);
        y >>= 1, x = mul(x, x, mod);
    }
    return ret % mod;
}

int isPrime(ll p) { // implements by miller-babin
    if (p < 2 || !(p&1))     return 0;
    if (p == 2)                   return 1;
    ll q = p-1, a, t;
    int k = 0, b = 0;
    while (!(q&1))    q >>= 1, k++;
    for (int it = 0; it < 2; it++) {
        a = rand()%(p-4) + 2;
        t = mpow2(a, q, p);
        b = (t == 1) || (t == p-1);
        for (int i = 1; i < k && !b; i++) {
            t = mul(t, t, p);
            if (t == p-1)
                b = 1;
        }
        if (b == 0)
            return 0;
    }
    return 1;
}

ll pollard_rho(ll n, ll c) {
    ll x = 2, y = 2, i = 1, k = 2, d;
    while (true) {
        x = (mul(x, x, n) + c);
        if (x >= n)    x -= n;
        d = __gcd(x - y, n);
        if (d > 1) return d;
        if (++i == k) y = x, k <<= 1;
    }
    return n;
}

void factorize(int n, vector<ll> &f) {
    for (int i = 0; i < Pt && P[i]*P[i] <= n; i++) {
        if (n%P[i] == 0) {
                while (n%P[i] == 0)
                    f.push_back(P[i]), n /= P[i];
```

```
        }
    }
    if (n != 1) f.push_back(n);
}

void llfactorize(ll n, vector<ll> &f) {
    if (n == 1)
        return ;
    if (n < 1e+9) {
        factorize(n, f);
        return ;
    }
    if (isPrime(n)) {
        f.push_back(n);
        return ;
    }

    ll d = n;
    for (int i = 2; d == n; i++)
        d = pollard_rho(n, i);
    llfactorize(d, f);
    llfactorize(n/d, f);
}

vector<ll> f;
map<ll, int> r;

int main() {
    sieve();
    ll n;
    scanf("%lld", &n);
    llfactorize(n, f);
    for (auto &x : f) r[x]++;
    ll last;
    for (auto it = r.begin(); it != r.end(); it++) {
        if (it != r.begin()) printf(" ");
        last = it -> first;
        printf("%lld", last);
        if (it->second > 1){
            for( int i = 0 ; i < it->second - 1 ; ++i ) printf(" %lld",last
                ) ;
        }
    }
    return 0;
}
```

8 Simplex Method

```
// Two-phase simplex algorithm for solving linear programs:
//    maximize    c^T x
//    subject to  Ax <= b
//                x >= 0
// INPUT: A -- an m x n matrix
//       b -- an m-dimensional vector
//       c -- an n-dimensional vector
//       x -- a vector where the optimal solution will be stored
// OUTPUT: value of the optimal solution (infinity if unbounded
//        above, nan if infeasible)
// To use this code, create an LPSolver object with A, b, and c as
// arguments. Then, call Solve(x).

typedef long double DOUBLE;
typedef vector<DOUBLE> VD;
typedef vector<VD> VVD;
typedef vector<int> VI;

const DOUBLE EPS = 1e-9;

struct LPSolver {
  int m, n;
  VI B, N;
  VVD D;

        LPSolver(const VVD &A, const VD &b, const VD &c) :
                m(b.size()), n(c.size()), N(n+1), B(m), D(m+2, VD(n+2)) {
                for (int i = 0; i < m; i++) for (int j = 0; j < n; j++)
                        D[i][j] = A[i][j];
                for (int i = 0; i < m; i++)
                        B[i] = n+i; D[i][n] = -1; D[i][n+1] = b[i];
                for (int j = 0; j < n; j++)
                        N[j] = j; D[m][j] = -c[j];
                N[n] = -1; D[m+1][n] = 1;
        }

        void Pivot(int r, int s) {
                for (int i = 0; i < m+2; i++) if (i != r)
                        for (int j = 0; j < n+2; j++) if (j != s)
                                D[i][j] -= D[r][j] * D[i][s] / D[r][s];
                for (int j = 0; j < n+2; j++) if (j != s)
                        D[r][j] /= D[r][s];
```

```
        for (int i = 0; i < m+2; i++) if (i != r)
                D[i][s] /= -D[r][s];
        D[r][s] = 1.0 / D[r][s];
        swap(B[r], N[s]);
}

bool Simplex(int phase) {
        int x = phase == 1 ? m+1 : m;
        while (true) {
                int s = -1;
                for (int j = 0; j <= n; j++) {
                        if (phase == 2 && N[j] == -1) continue;
                        if (s == -1 || D[x][j] < D[x][s] ||
                                D[x][j] == D[x][s] && N[j] < N[s]) s
                                    = j;
                }
                if (D[x][s] >= -EPS) return true;
                int r = -1;
                for (int i = 0; i < m; i++) {
                        if (D[i][s] <= 0) continue;
                        if (r == -1 ||
                                D[i][n+1] / D[i][s] < D[r][n+1] / D[r
                                    ][s] ||
                        D[i][n+1] / D[i][s] == D[r][n+1] / D[r][s
                            ] &&
                                B[i] < B[r]) r = i;
}
                if (r == -1) return false;
                Pivot(r, s);
        }
}

DOUBLE Solve(VD &x) {
        int r = 0;
        for (int i = 1; i < m; i++) if (D[i][n+1] < D[r][n+1]) r =
            i;
        if (D[r][n+1] <= -EPS) {
                Pivot(r, n);
                if (!Simplex(1) || D[m+1][n+1] < -EPS)
                        return -numeric_limits<DOUBLE>::infinity();
                for (int i = 0; i < m; i++) if (B[i] == -1) {
                        int s = -1;
                        for (int j = 0; j <= n; j++)
                                if (s == -1 || D[i][j] < D[i][s] ||
                                        D[i][j] == D[i][s] && N[j] < N
                                            [s]) s = j;
                                Pivot(i, s);
                        }
                }
                if (!Simplex(2)) return numeric_limits<DOUBLE>::infinity();
                x = VD(n);
                for (int i = 0; i < m; i++) if (B[i] < n)
                        x[B[i]] = D[i][n+1];
                return D[m][n+1];
        }
};

int main() {

  const int m = 4;
  const int n = 3;
  DOUBLE _A[m][n] = {
    { 6, -1, 0 },
    { -1, -5, 0 },
    { 1, 5, 1 },
    { -1, -5, -1 }
  };
  DOUBLE _b[m] = { 10, -4, 5, -5 };
  DOUBLE _c[n] = { 1, -1, 0 };

  VVD A(m);
  VD b(_b, _b + m);
  VD c(_c, _c + n);
  for (int i = 0; i < m; i++) A[i] = VD(_A[i], _A[i] + n);

  LPSolver solver(A, b, c);
  VD x;
  DOUBLE value = solver.Solve(x);

  cerr << "VALUE: "<< value << endl;
  cerr << "SOLUTION:";
  for (size_t i = 0; i < x.size(); i++) cerr << " " << x[i];
  cerr << endl;
  return 0;
}
```

9 Teorema de Lucas

```
ll comb[105][105];

//Devuleve la comb(n,k) % m para n,k grandes y m pequeno

ll lucas( ll n , ll k , ll m ){
	//Se puede precalcular la combinatoria afuera
	REP(i,0,52) REP(j,0,52){
		if( j == 0 ) comb[i][0] = 1;
		else if( j > i ) comb[i][j] = 0;
		else comb[i][j] = ( comb[i-1][j] + comb[i-1][j-1] ) % m;
	}

	ll ans = 1, x, y;

	while( n ){
		x = n % m, y = k % m;
		ans = ( ans * comb[x][y] ) % m;
		n /= m, k /= m;
	}
	return ans;
}
```