Grimoire's Standard Code Library*

Shanghai Jiao Tong University

Dated: 2017 年 8 月 11 日

*https://github.com/kzoacn/Grimoire

目录

Chapter 1

代数

$O(n^2 \log n)$ 求线性递推数列第 n 项

Given $a_0, a_1, \cdots, a_{m-1}$
$a_n = c_0 * a_{n-m} + \cdots + c_{m-1} * a_0$
a_0 *is the nth element*, $\cdots$, a_{m-1} *is the* $n+m-1$*th element*

```
void linear_recurrence(long long n, int m, int a[], int c[], int p) {
    long long v[M] = {1 % p}, u[M << 1], msk = !!n;
    for(long long i(n); i > 1; i >>= 1) {
        msk <<= 1;
    }
    for(long long x(0); msk; msk >>= 1, x <<= 1) {
        fill_n(u, m << 1, 0);
        int b(!!(n & msk));
        x |= b;
        if(x < m) {
            u[x] = 1 % p;
        }else {
            for(int i(0); i < m; i++) {
                for(int j(0), t(i + b); j < m; j++, t++) {
                    u[t] = (u[t] + v[i] * v[j]) % p;
                }
            }
            for(int i((m << 1) - 1); i >= m; i--) {
                for(int j(0), t(i - m); j < m; j++, t++) {
                    u[t] = (u[t] + c[j] * u[i]) % p;
                }
            }
        }
        copy(u, u + m, v);
    }
    //a[n] = v[0] * a[0] + v[1] * a[1] + ... + v[m - 1] * a[m - 1].
    for(int i(m); i < 2 * m; i++) {
        a[i] = 0;
```

```
        for(int j(0); j < m; j++) {
            a[i] = (a[i] + (long long)c[j] * a[i + j - m]) % p;
        }
    }
    for(int j(0); j < m; j++) {
        b[j] = 0;
        for(int i(0); i < m; i++) {
            b[j] = (b[j] + v[i] * a[i + j]) % p;
        }
    }
    for(int j(0); j < m; j++) {
        a[j] = b[j];
    }
}
```

闪电数论变换与魔力 CRT

```
#define meminit(A, l, r) memset(A + (l), 0, sizeof(*A) * ((r) - (l)))
#define memcopy(B, A, l, r) memcpy(B, A + (l), sizeof(*A) * ((r) - (l)))
void DFT(int *a, int n, int f) { //f=1 逆 DFT
    for (register int i = 0, j = 0; i < n; i++) {
        if (i > j) std::swap(a[i], a[j]);
        for (register int t = n >> 1; (j ^= t) < t; t >>= 1);
    }
    for (register int i = 2; i <= n; i <<= 1) {
        static int exp[MAXN];
        exp[0] = 1; exp[1] = fpm(PRT, (MOD - 1) / i, MOD);
        if (f == 1) exp[1] = fpm(exp[1], MOD - 2, MOD);
        for (register int k = 2; k < (i >> 1); k++) {
            exp[k] = 1ll * exp[k - 1] * exp[1] % MOD;
        }
        for (register int j = 0; j < n; j += i) {
            for (register int k = 0; k < (i >> 1); k++) {
                register int &pA = a[j + k], &pB = a[j + k + (i >> 1)];
                register long long B = 1ll * pB * exp[k];
                pB = (pA - B) % MOD;
                pA = (pA + B) % MOD;
            }
        }
    }
    if (f == 1) {
        register int rev = fpm(n, MOD - 2, MOD);
        for (register int i = 0; i < n; i++) {
            a[i] = 1ll * a[i] * rev % MOD;
            if (a[i] < 0) { a[i] += MOD; }
```

```
        }
    }
}
// 在不写高精度的情况下合并 FFT 所得结果对 MOD 取模后的答案
// 值得注意的是，这个东西不能最后再合并，而是应该每做一次多项式乘法就 CRT 一次
int CRT(int *a) {
    static int x[3];
    for (int i = 0; i < 3; i++) {
        x[i] = a[i];
        for (int j = 0; j < i; j++) {
            int t = (x[i] - x[j] + FFT[i] -> MOD) % FFT[i] -> MOD;
            if (t < 0) t += FFT[i] -> MOD;
            x[i] = 1LL * t * inv[j][i] % FFT[i] -> MOD;
        }
    }
    int sum = 1, ret = x[0] % MOD;
    for (int i = 1; i < 3; i ++) {
        sum = 1LL * sum * FFT[i - 1] -> MOD % MOD;
        ret += 1LL * x[i] * sum % MOD;
        if(ret >= MOD) ret -= MOD;
    }
    return ret;
}
for (int i = 0; i < 3; i++) // inv 数组的预处理过程，inverse(x, p) 表示求 x 在 p 下逆元
    for (int j = 0; j < 3; j++)
        inv[i][j] = inverse(FFT[i] -> MOD, FFT[j] -> MOD);
```

多项式求逆

Given polynomial a and n, b is the polynomial such that $a * b \equiv 1 \pmod{x^n}$

```
void getInv(int *a, int *b, int n) {
    static int tmp[MAXN];
    b[0] = fpm(a[0], MOD - 2, MOD);
    for (int c = 2, M = 1; c < (n << 1); c <<= 1) {
        for (; M <= 3 * (c - 1); M <<= 1);
        meminit(b, c, M);
        meminit(tmp, c, M);
        memcopy(tmp, a, 0, c);
        DFT(tmp, M, 0);
        DFT(b, M, 0);
        for (int i = 0; i < M; i++) {
            b[i] = 1ll * b[i] * (2ll - 1ll * tmp[i] * b[i] % MOD + MOD) % MOD;
        }
        DFT(b, M, 1);
        meminit(b, c, M);
```

```
    }
}
```

多项式除法

d is quotient and r is remainder

```
void divide(int n, int m, int *a, int *b, int *d, int *r) { // n、m 分别为多项式 A（被除数）
↪ 和 B（除数）的指数 + 1
    static int M, tA[MAXN], tB[MAXN], inv[MAXN], tD[MAXN];
    for (; n > 0 && a[n - 1] == 0; n--);
    for (; m > 0 && b[m - 1] == 0; m--);
    for (int i = 0; i < n; i++) tA[i] = a[n - i - 1];
    for (int i = 0; i < m; i++) tB[i] = b[m - i - 1];
    for (M = 1; M <= n - m + 1; M <<= 1);
    if (m < M) meminit(tB, m, M);
    getInv(tB, inv, M);
    for (M = 1; M <= 2 * (n - m + 1); M <<= 1);
    meminit(inv, n - m + 1, M);
    meminit(tA, n - m + 1, M);
    DFT(inv, M, 0);
    DFT(tA, M, 0);
    for (int i = 0; i < M; i++) {
        d[i] = 1ll * inv[i] * tA[i] % MOD;
    }
    DFT(d, M, 1);
    std::reverse(d, d + n - m + 1);
    for (M = 1; M <= n; M <<= 1);
    memcopy(tB, b, 0, m);
    if (m < M) meminit(tB, m, M);
    memcopy(tD, d, 0, n - m + 1);
    meminit(tD, n - m + 1, M);
    DFT(tD, M, 0);
    DFT(tB, M, 0);
    for (int i = 0; i < M; i++) {
        r[i] = 1ll * tD[i] * tB[i] % MOD;
    }
    DFT(r, M, 1);
    meminit(r, n, M);
    for (int i = 0; i < n; i++) {
        r[i] = (a[i] - r[i] + MOD) % MOD;
    }
}
```

多项式取指数取对数

Given polynomial a and n, b is the polynomial such that $b \equiv e^a (\mod x^n)$ or $b \equiv \ln a (\mod x^n)$

```
void getDiff(int *a, int *b, int n) { // 多项式取微分
    for (int i = 0; i + 1 < n; i++) {
        b[i] = 1ll * (i + 1) * a[i + 1] % MOD;
    }
    b[n - 1] = 0;
}
void getInt(int *a, int *b, int n) { // 多项式取积分，积分常数为 0
    static int inv[MAXN];
    inv[1] = 1;
    for (int i = 2; i < n; i++) {
        inv[i] = 1ll * (MOD - MOD / i) * inv[MOD % i] % MOD;
    }
    b[0] = 0;
    for (int i = 1; i < n; i++) {
        b[i] = 1ll * a[i - 1] * inv[i] % MOD;
    }
}
void getLn(int *a, int *b, int n) {
    static int inv[MAXN], d[MAXN];
    int M = 1;
    for (; M <= 2 * (n - 1); M <<= 1);
    getInv(a, inv, n);
    getDiff(a, d, n);
    meminit(d, n, M);
    meminit(inv, n, M);
    DFT(d, M, 0); DFT(inv, M, 0);
    for (int i = 0; i < M; i++) {
        d[i] = 1ll * d[i] * inv[i] % MOD;
    }
    DFT(d, M, 1);
    getInt(d, b, n);
}
void getExp(int *a, int *b, int n) {
    static int ln[MAXN], tmp[MAXN];
    b[0] = 1;
    for (int c = 2, M = 1; c < (n << 1); c <<= 1) {
        for (; M <= 2 * (c - 1); M <<= 1);
        int bound = std::min(c, n);
        memcopy(tmp, a, 0, bound);
        meminit(tmp, bound, M);
        meminit(b, c, M);
        getLn(b, ln, c);
        meminit(ln, c, M);
```

```
        DFT(b, M, 0);
        DFT(tmp, M, 0);
        DFT(ln, M, 0);
        for (int i = 0; i < M; i++) {
            b[i] = 1ll * b[i] * (1ll - ln[i] + tmp[i] + MOD) % MOD;
        }
        DFT(b, M, 1);
        meminit(b, c, M);
    }
}
```

Chapter 2

数论

大整数相乘取模

```
// x 与 y 须非负
long long mult(long long x, long long y, long long MODN) {
    long long t = (x * y - (long long)((long double)x / MODN * y + 1e-3) * MODN) % MODN;
    return t < 0 ? t + MODN : t;
}
```

线段下整点

solve for $\sum_{i=0}^{n-1} \lfloor \frac{a+bi}{m} \rfloor$, $n, m, a, b > 0$

```
LL solve(LL n,LL a,LL b,LL m){
    if(b==0) return n*(a/m);
    if(a>=m) return n*(a/m)+solve(n,a%m,b,m);
    if(b>=m) return (n-1)*n/2*(b/m)+solve(n,a,b%m,m);
    return solve((a+b*n)/m,(a+b*n)%m,m,b);
}
```

中国剩余定理

first is remainder, second is module

```
inline void fix(LL &x, LL y) {
    x = (x % y + y) % y;
}
bool solve(int n, std::pair<LL, LL> a[],
                  std::pair<LL, LL> &ans) {
    ans = std::make_pair(1, 1);
    for (int i = 0; i < n; ++i) {
        LL num, y;
        euclid(ans.second, a[i].second, num, y);
```

```
        LL divisor = std::__gcd(ans.second, a[i].second);
        if ((a[i].first - ans.first) % divisor) {
            return false;
        }
        num *= (a[i].first - ans.first) / divisor;
        fix(num, a[i].second);
        ans.first += ans.second * num;
        ans.second *= a[i].second / divisor;
        fix(ans.first, ans.second);
    }
    return true;
}
```

Chapter 3

图论

一般图匹配

```
// 0-base, match[u] is linked to u
vector<int> lnk[MAXN];
int match[MAXN], Queue[MAXN], pred[MAXN], base[MAXN], head, tail, sta, fin, nbase;
bool inQ[MAXN], inB[MAXN];
inline void push(int u) {
    Queue[tail++] = u; inQ[u] = 1;
}
inline int pop() {
    return Queue[head++];
}
inline int FindCA(int u, int v) {
    static bool inP[MAXN];
    fill(inP, inP + n, false);
    while (1) {
        u = base[u]; inP[u] = 1;
        if(u == sta) break;
        u = pred[match[u]];
    }
    while (1) {
        v = base[v];
        if (inP[v]) break;
        v = pred[match[v]];
    }
    return v;
}
inline void RT(int u) {
    int v;
    while (base[u] != nbase) {
        v = match[u];
        inB[base[u]] = inB[base[v]] = 1;
        u = pred[v];
```

```
        if (base[u] != nbase) pred[u] = v;
    }
}
inline void BC(int u, int v) {
    nbase = FindCA(u, v);
    fill(inB, inB + n, 0);
    RT(u); RT(v);
    if (base[u] != nbase) pred[u] = v;
    if (base[v] != nbase) pred[v] = u;
    for (int i = 0; i < n; ++i)
        if (inB[base[i]]) {
            base[i] = nbase;
            if (!inQ[i]) push(i);
        }
}
bool FindAP(int u) {
    bool found = false;
    for (int i = 0; i < n; ++i) {
        pred[i] = -1; base[i] = i; inQ[i] = 0;
    }
    sta = u; fin = -1; head = tail = 0; push(sta);
    while (head < tail) {
        int u = pop();
        for (int i = (int)lnk[u].size() - 1; i >= 0; --i) {
            int v = lnk[u][i];
            if (base[u] != base[v] && match[u] != v) {
                if (v == sta || match[v] >= 0 && pred[match[v]] >= 0) BC(u, v);
                else if (pred[v] == -1) {
                    pred[v] = u;
                    if (match[v] >= 0) push(match[v]);
                    else {
                        fin = v;
                        return true;
                    }
                }
            }
        }
    }
    return found;
}
inline void AP() {
    int u = fin, v, w;
    while (u >= 0) {
        v = pred[u]; w = match[v];
        match[v] = u; match[u] = v;
        u = w;
```

```
    }
}
inline int FindMax() {
    for (int i = 0; i < n; ++i) match[i] = -1;
    for (int i = 0; i < n; ++i)
        if (match[i] == -1 && FindAP(i)) AP();
    int ans = 0;
    for (int i = 0; i < n; ++i) {
        ans += (match[i] != -1);
    }
    return ans;
}
```

无向图最小割

```
/*
 * Stoer Wagner 全局最小割 O(V ^ 3)
 * 1base, 点数 n, 邻接矩阵 edge[MAXN][MAXN]
 * 返回值为全局最小割
 */

int StoerWagner() {
    static int v[MAXN], wage[MAXN];
    static bool vis[MAXN];

    for (int i = 1; i <= n; ++i) v[i] = i;

    int res = INF;

    for (int nn = n; nn > 1; --nn) {
        memset(vis, 0, sizeof(bool) * (nn + 1));
        memset(wage, 0, sizeof(int) * (nn + 1));

        int pre, last = 1; // vis[1] = 1;

        for (int i = 1; i < nn; ++i) {
            pre = last; last = 0;
            for (int j = 2; j <= nn; ++j) if (!vis[j]) {
                wage[j] += edge[v[pre]][v[j]];
                if (!last || wage[j] > wage[last]) last = j;
            }
            vis[last] = 1;
        }

        res = std::min(res, wage[last]);
```

```cpp

        for (int i = 1; i <= nn; ++i) {
            edge[v[i]][v[pre]] += edge[v[last]][v[i]];
            edge[v[pre]][v[i]] += edge[v[last]][v[i]];
        }
        v[last] = v[nn];
    }
    return res;
}
```

Chapter 4

技巧

无敌的读入优化

```cpp
// getchar() 读入优化 << 关同步 cin << 此优化
// 用 isdigit() 会小幅变慢
// 返回 false 表示读到文件尾
namespace Reader {
    const int L = (1 << 15) + 5;
    char buffer[L], *S, *T;
    __inline bool getchar(char &ch) {
        if (S == T) {
            T = (S = buffer) + fread(buffer, 1, L, stdin);
            if (S == T) {
                ch = EOF;
                return false;
            }
        }
        ch = *S++;
        return true;
    }
    __inline bool getint(int &x) {
        char ch; bool neg = 0;
        for (; getchar(ch) && (ch < '0' || ch > '9'); ) neg ^= ch == '-';
        if (ch == EOF) return false;
        x = ch - '0';
        for (; getchar(ch), ch >= '0' && ch <= '9'; )
            x = x * 10 + ch - '0';
        if (neg) x = -x;
        return true;
    }
}
```

真正释放 STL 内存

```
template <typename T>
__inline void clear(T& container) {
    container.clear(); // 或者删除了一堆元素
    T(container).swap(container);
}
```