Grimoire'l Standard Code Library [*]

Shanghai Jiao Tong University

Dated: 2017 年 9 月 19 日

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 $n+m-1th$ element

```cpp
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;
```

```
29        for(int j(0); j < m; j++) {
30            a[i] = (a[i] + (long long)c[j] * a[i + j - m]) % p;
31        }
32    }
33    for(int j(0); j < m; j++) {
34        b[j] = 0;
35        for(int i(0); i < m; i++) {
36            b[j] = (b[j] + v[i] * a[i + j]) % p;
37        }
38    }
39    for(int j(0); j < m; j++) {
40        a[j] = b[j];
41    }
42 }
```

闪电数论变换与魔力 CRT

```
1  #define meminit(A, l, r) memset(A + (l), 0, sizeof(*A) * ((r) - (l)))
2  #define memcopy(B, A, l, r) memcpy(B, A + (l), sizeof(*A) * ((r) - (l)))
3  void DFT(int *a, int n, int f) { //f=1 逆 DFT
4      for (register int i = 0, j = 0; i < n; i++) {
5          if (i > j) std::swap(a[i], a[j]);
6          for (register int t = n >> 1; (j ^= t) < t; t >>= 1);
7      }
8      for (register int i = 2; i <= n; i <<= 1) {
9          static int exp[MAXN];
10         exp[0] = 1; exp[1] = fpm(PRT, (MOD - 1) / i, MOD);
11         if (f == 1) exp[1] = fpm(exp[1], MOD - 2, MOD);
12         for (register int k = 2; k < (i >> 1); k++) {
13             exp[k] = 1ll * exp[k - 1] * exp[1] % MOD;
14         }
15         for (register int j = 0; j < n; j += i) {
16             for (register int k = 0; k < (i >> 1); k++) {
17                 register int &pA = a[j + k], &pB = a[j + k + (i >> 1)];
18                 register long long B = 1ll * pB * exp[k];
19                 pB = (pA - B) % MOD;
20                 pA = (pA + B) % MOD;
21             }
22         }
23     }
24     if (f == 1) {
25         register int rev = fpm(n, MOD - 2, MOD);
26         for (register int i = 0; i < n; i++) {
27             a[i] = 1ll * a[i] * rev % MOD;
28             if (a[i] < 0) { a[i] += MOD; }
```

```
29              }
30          }
31 }
32 // 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
33 // 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
34 int CRT(int *a) {
35      static int x[3];
36      for (int i = 0; i < 3; i++) {
37          x[i] = a[i];
38          for (int j = 0; j < i; j++) {
39              int t = (x[i] - x[j] + FFT[i] -> MOD) % FFT[i] -> MOD;
40              if (t < 0) t += FFT[i] -> MOD;
41              x[i] = 1LL * t * inv[j][i] % FFT[i] -> MOD;
42          }
43      }
44      int sum = 1, ret = x[0] % MOD;
45      for (int i = 1; i < 3; i ++) {
46          sum = 1LL * sum * FFT[i - 1] -> MOD % MOD;
47          ret += 1LL * x[i] * sum % MOD;
48          if(ret >= MOD) ret -= MOD;
49      }
50      return ret;
51 }
52 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
53      for (int j = 0; j < 3; j++)
54          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 $a * b \equiv 1(\mod x^n)$

```
1  void getInv(int *a, int *b, int n) {
2      static int tmp[MAXN];
3      b[0] = fpm(a[0], MOD - 2, MOD);
4      for (int c = 2, M = 1; c < (n << 1); c <<= 1) {
5          for (; M <= 3 * (c - 1); M <<= 1);
6          meminit(b, c, M);
7          meminit(tmp, c, M);
8          memcopy(tmp, a, 0, c);
9          DFT(tmp, M, 0);
10         DFT(b, M, 0);
11         for (int i = 0; i < M; i++) {
12             b[i] = 1ll * b[i] * (2ll - 1ll * tmp[i] * b[i] % MOD + MOD) % MOD;
13         }
14         DFT(b, M, 1);
15         meminit(b, c, M);
```

```
16        }
17  }
```

多项式除法

d is quotient and r is remainder

```
1  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
2      static int M, tA[MAXN], tB[MAXN], inv[MAXN], tD[MAXN];
3      for (; n > 0 && a[n - 1] == 0; n--);
4      for (; m > 0 && b[m - 1] == 0; m--);
5      for (int i = 0; i < n; i++) tA[i] = a[n - i - 1];
6      for (int i = 0; i < m; i++) tB[i] = b[m - i - 1];
7      for (M = 1; M <= n - m + 1; M <<= 1);
8      if (m < M) meminit(tB, m, M);
9      getInv(tB, inv, M);
10     for (M = 1; M <= 2 * (n - m + 1); M <<= 1);
11     meminit(inv, n - m + 1, M);
12     meminit(tA, n - m + 1, M);
13     DFT(inv, M, 0);
14     DFT(tA, M, 0);
15     for (int i = 0; i < M; i++) {
16         d[i] = 1ll * inv[i] * tA[i] % MOD;
17     }
18     DFT(d, M, 1);
19     std::reverse(d, d + n - m + 1);
20     for (M = 1; M <= n; M <<= 1);
21     memcopy(tB, b, 0, m);
22     if (m < M) meminit(tB, m, M);
23     memcopy(tD, d, 0, n - m + 1);
24     meminit(tD, n - m + 1, M);
25     DFT(tD, M, 0);
26     DFT(tB, M, 0);
27     for (int i = 0; i < M; i++) {
28         r[i] = 1ll * tD[i] * tB[i] % MOD;
29     }
30     DFT(r, M, 1);
31     meminit(r, n, M);
32     for (int i = 0; i < n; i++) {
33         r[i] = (a[i] - r[i] + MOD) % MOD;
34     }
35  }
```

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);
```

```
44        DFT(b, M, 0);
45        DFT(tmp, M, 0);
46        DFT(ln, M, 0);
47        for (int i = 0; i < M; i++) {
48            b[i] = 1ll * b[i] * (1ll - ln[i] + tmp[i] + MOD) % MOD;
49        }
50        DFT(b, M, 1);
51        meminit(b, c, M);
52    }
53 }
```

Chapter 2

数论

大整数相乘取模

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

```cpp
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

```cpp
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);
```

```
10        LL divisor = std::__gcd(ans.second, a[i].second);
11        if ((a[i].first - ans.first) % divisor) {
12            return false;
13        }
14        num *= (a[i].first - ans.first) / divisor;
15        fix(num, a[i].second);
16        ans.first += ans.second * num;
17        ans.second *= a[i].second / divisor;
18        fix(ans.first, ans.second);
19    }
20    return true;
21 }
```

Chapter 3

图论

图论基础

```cpp
struct Graph {  // Remember to call .init()!
    int e, nxt[M], v[M], adj[N], n;
    bool base;
    __inline void init(bool _base, int _n = 0) {
        assert(n < N);
        n = _n; base = _base;
        e = 0; memset(adj + base, -1, sizeof(*adj) * n);
    }
    __inline int new_node() {
        adj[n + base] = -1;
        assert(n + base + 1 < N);
        return n++ + base;
    }
    __inline void ins(int u0, int v0) {  // directional
        assert(u0 < n + base && v0 < n + base);
        v[e] = v0; nxt[e] = adj[u0]; adj[u0] = e++;
        assert(e < M);
    }
    __inline void bi_ins(int u0, int v0) {  // bi-directional
        ins(u0, v0); ins(v0, u0);
    }
};
```

闪电二分图匹配

```cpp
int matchx[N], matchy[N], level[N];
vector<int> edge[N];
bool dfs(int x) {
    for (int i = 0; i < (int)edge[x].size(); ++i) {
        int y = edge[x][i];
```

```cpp
        int w = matchy[y];
        if (w == -1 || level[x] + 1 == level[w] && dfs(w)) {
            matchx[x] = y;
            matchy[y] = x;
            return true;
        }
    }
    level[x] = -1;
    return false;
}
int solve() {
    memset(matchx, -1, sizeof(*matchx) * n);
    memset(matchy, -1, sizeof(*matchy) * m);
    for (int ans = 0; ; ) {
        std::vector<int> q;
        for (int i = 0; i < n; ++i) {
            if (matchx[i] == -1) {
                level[i] = 0;
                q.push_back(i);
            } else {
                level[i] = -1;
            }
        }
        for (int head = 0; head < (int)q.size(); ++head) {
            int x = q[head];
            for (int i = 0; i < (int)edge[x].size(); ++i) {
                int y = edge[x][i];
                int w = matchy[y];
                if (w != -1 && level[w] < 0) {
                    level[w] = level[x] + 1;
                    q.push_back(w);
                }
            }
        }
        int delta = 0;
        for (int i = 0; i < n; ++i) {
            if (matchx[i] == -1 && dfs(i)) {
                delta++;
            }
        }
        if (delta == 0) {
            return ans;
        } else {
            ans += delta;
        }
    }
```

```
52 }
```

一般图匹配

```
1  // 0-base, match[u] is linked to u
2  vector<int> lnk[MAXN];
3  int match[MAXN], Queue[MAXN], pred[MAXN], base[MAXN], head, tail, sta, fin, nbase;
4  bool inQ[MAXN], inB[MAXN];
5  inline void push(int u) {
6      Queue[tail++] = u; inQ[u] = 1;
7  }
8  inline int pop() {
9      return Queue[head++];
10 }
11 inline int FindCA(int u, int v) {
12     static bool inP[MAXN];
13     fill(inP, inP + n, false);
14     while (1) {
15         u = base[u]; inP[u] = 1;
16         if(u == sta) break;
17         u = pred[match[u]];
18     }
19     while (1) {
20         v = base[v];
21         if (inP[v]) break;
22         v = pred[match[v]];
23     }
24     return v;
25 }
26 inline void RT(int u) {
27     int v;
28     while (base[u] != nbase) {
29         v = match[u];
30         inB[base[u]] = inB[base[v]] = 1;
31         u = pred[v];
32         if (base[u] != nbase) pred[u] = v;
33     }
34 }
35 inline void BC(int u, int v) {
36     nbase = FindCA(u, v);
37     fill(inB, inB + n, 0);
38     RT(u); RT(v);
39     if (base[u] != nbase) pred[u] = v;
40     if (base[v] != nbase) pred[v] = u;
41     for (int i = 0; i < n; ++i)
```

```
42              if (inB[base[i]]) {
43                  base[i] = nbase;
44                  if (!inQ[i]) push(i);
45              }
46  }
47  bool FindAP(int u) {
48      bool found = false;
49      for (int i = 0; i < n; ++i) {
50          pred[i] = -1; base[i] = i; inQ[i] = 0;
51      }
52      sta = u; fin = -1; head = tail = 0; push(sta);
53      while (head < tail) {
54          int u = pop();
55          for (int i = (int)lnk[u].size() - 1; i >= 0; --i) {
56              int v = lnk[u][i];
57              if (base[u] != base[v] && match[u] != v) {
58                  if (v == sta || match[v] >= 0 && pred[match[v]] >= 0) BC(u, v);
59                  else if (pred[v] == -1) {
60                      pred[v] = u;
61                      if (match[v] >= 0) push(match[v]);
62                      else {
63                          fin = v;
64                          return true;
65                      }
66                  }
67              }
68          }
69      }
70      return found;
71  }
72  inline void AP() {
73      int u = fin, v, w;
74      while (u >= 0) {
75          v = pred[u]; w = match[v];
76          match[v] = u; match[u] = v;
77          u = w;
78      }
79  }
80  inline int FindMax() {
81      for (int i = 0; i < n; ++i) match[i] = -1;
82      for (int i = 0; i < n; ++i)
83          if (match[i] == -1 && FindAP(i)) AP();
84      int ans = 0;
85      for (int i = 0; i < n; ++i) {
86          ans += (match[i] != -1);
87      }
```

```
88     return ans;
89 }
```

一般最大权匹配

```
1  //maximum weight blossom,  change g[u][v].w to INF - g[u][v].w when minimum weight blossom
       ↪ is needed
2  //type of ans is long long
3  //replace all int to long long if weight of edge is long long
4
5  struct WeightGraph {
6      static const int INF = INT_MAX;
7      static const int MAXN = 400;
8      struct edge{
9          int u, v, w;
10         edge() {}
11         edge(int u, int v, int w): u(u), v(v), w(w) {}
12     };
13     int n, n_x;
14     edge g[MAXN * 2 + 1][MAXN * 2 + 1];
15     int lab[MAXN * 2 + 1];
16     int match[MAXN * 2 + 1], slack[MAXN * 2 + 1], st[MAXN * 2 + 1], pa[MAXN * 2 + 1];
17     int flower_from[MAXN * 2 + 1][MAXN+1], S[MAXN * 2 + 1], vis[MAXN * 2 + 1];
18     vector<int> flower[MAXN * 2 + 1];
19     queue<int> q;
20     inline int e_delta(const edge &e){ // does not work inside blossoms
21         return lab[e.u] + lab[e.v] - g[e.u][e.v].w * 2;
22     }
23     inline void update_slack(int u, int x){
24         if(!slack[x] || e_delta(g[u][x]) < e_delta(g[slack[x]][x]))
25             slack[x] = u;
26     }
27     inline void set_slack(int x){
28         slack[x] = 0;
29         for(int u = 1;u <= n; ++u)
30             if(g[u][x].w > 0 && st[u] != x && S[st[u]] == 0)
31                 update_slack(u, x);
32     }
33     void q_push(int x){
34         if(x <= n)q.push(x);
35         else for(size_t i = 0;i < flower[x].size(); i++)
36             q_push(flower[x][i]);
37     }
38     inline void set_st(int x, int b){
39         st[x]=b;
```

```
40          if(x > n) for(size_t i = 0;i < flower[x].size(); ++i)
41                  set_st(flower[x][i], b);
42      }
43      inline int get_pr(int b, int xr){
44          int pr = find(flower[b].begin(), flower[b].end(), xr) - flower[b].begin();
45          if(pr % 2 == 1){
46              reverse(flower[b].begin() + 1, flower[b].end());
47              return (int)flower[b].size() - pr;
48          } else return pr;
49      }
50      inline void set_match(int u, int v){
51          match[u]=g[u][v].v;
52          if(u > n){
53              edge e=g[u][v];
54              int xr = flower_from[u][e.u], pr=get_pr(u, xr);
55              for(int i = 0;i < pr; ++i)
56                  set_match(flower[u][i], flower[u][i ^ 1]);
57              set_match(xr, v);
58              rotate(flower[u].begin(), flower[u].begin()+pr, flower[u].end());
59          }
60      }
61      inline void augment(int u, int v){
62          for(; ; ){
63              int xnv=st[match[u]];
64              set_match(u, v);
65              if(!xnv)return;
66              set_match(xnv, st[pa[xnv]]);
67              u=st[pa[xnv]], v=xnv;
68          }
69      }
70      inline int get_lca(int u, int v){
71          static int t=0;
72          for(++t; u || v; swap(u, v)){
73              if(u == 0)continue;
74              if(vis[u] == t)return u;
75              vis[u] = t;
76              u = st[match[u]];
77              if(u) u = st[pa[u]];
78          }
79          return 0;
80      }
81      inline void add_blossom(int u, int lca, int v){
82          int b = n + 1;
83          while(b <= n_x && st[b]) ++b;
84          if(b > n_x) ++n_x;
85          lab[b] = 0, S[b] = 0;
```

```
86          match[b] = match[lca];
87          flower[b].clear();
88          flower[b].push_back(lca);
89          for(int x = u, y; x != lca; x = st[pa[y]]) {
90              flower[b].push_back(x),
91              flower[b].push_back(y = st[match[x]]),
92              q_push(y);
93          }
94          reverse(flower[b].begin() + 1, flower[b].end());
95          for(int x = v, y; x != lca; x = st[pa[y]]) {
96              flower[b].push_back(x),
97              flower[b].push_back(y = st[match[x]]),
98              q_push(y);
99          }
100         set_st(b, b);
101         for(int x = 1; x <= n_x; ++x) g[b][x].w = g[x][b].w = 0;
102         for(int x = 1; x <= n; ++x) flower_from[b][x] = 0;
103         for(size_t i = 0 ; i < flower[b].size(); ++i){
104             int xs = flower[b][i];
105             for(int x = 1; x <= n_x; ++x)
106                 if(g[b][x].w == 0 || e_delta(g[xs][x]) < e_delta(g[b][x]))
107                     g[b][x] = g[xs][x], g[x][b] = g[x][xs];
108             for(int x = 1;x <= n; ++x)
109                 if(flower_from[xs][x]) flower_from[b][x] = xs;
110         }
111         set_slack(b);
112     }
113     inline void expand_blossom(int b){ // S[b]  ==  1
114         for(size_t i = 0; i < flower[b].size(); ++i)
115             set_st(flower[b][i], flower[b][i]);
116         int xr = flower_from[b][g[b][pa[b]].u], pr = get_pr(b, xr);
117         for(int i = 0; i < pr; i += 2){
118             int xs = flower[b][i], xns = flower[b][i + 1];
119             pa[xs] = g[xns][xs].u;
120             S[xs] = 1, S[xns] = 0;
121             slack[xs] = 0, set_slack(xns);
122             q_push(xns);
123         }
124         S[xr] = 1, pa[xr] = pa[b];
125         for(size_t i = pr + 1;i < flower[b].size(); ++i){
126             int xs = flower[b][i];
127             S[xs] = -1, set_slack(xs);
128         }
129         st[b] = 0;
130     }
131     inline bool on_found_edge(const edge &e){
```

```
132             int u = st[e.u], v = st[e.v];
133             if(S[v] == -1){
134                 pa[v] = e.u, S[v] = 1;
135                 int nu = st[match[v]];
136                 slack[v] = slack[nu] = 0;
137                 S[nu] = 0, q_push(nu);
138             }else if(S[v] == 0){
139                 int lca = get_lca(u, v);
140                 if(!lca) return augment(u, v), augment(v, u), true;
141                 else add_blossom(u, lca, v);
142             }
143             return false;
144         }
145     inline bool matching(){
146         memset(S + 1, -1, sizeof(int) * n_x);
147         memset(slack + 1, 0, sizeof(int) * n_x);
148         q = queue<int>();
149         for(int x = 1;x <= n_x; ++x)
150             if(st[x] == x && !match[x]) pa[x]=0, S[x]=0, q_push(x);
151         if(q.empty())return false;
152         for(;;){
153             while(q.size()){
154                 int u = q.front();q.pop();
155                 if(S[st[u]] == 1)continue;
156                 for(int v = 1;v <= n; ++v)
157                     if(g[u][v].w > 0 && st[u] != st[v]){
158                         if(e_delta(g[u][v]) == 0){
159                             if(on_found_edge(g[u][v]))return true;
160                         }else update_slack(u, st[v]);
161                     }
162             }
163             int d = INF;
164             for(int b = n + 1; b <= n_x;++b)
165                 if(st[b] == b && S[b] == 1)d = min(d, lab[b]/2);
166             for(int x = 1; x <= n_x; ++x)
167                 if(st[x] == x && slack[x]){
168                     if(S[x] == -1)d = min(d, e_delta(g[slack[x]][x]));
169                     else if(S[x] == 0)d = min(d, e_delta(g[slack[x]][x])/2);
170                 }
171             for(int u = 1; u <= n; ++u){
172                 if(S[st[u]] == 0){
173                     if(lab[u] <= d)return 0;
174                     lab[u] -= d;
175                 }else if(S[st[u]] == 1)lab[u] += d;
176             }
177             for(int b = n+1; b <= n_x; ++b)
```

```
178              if(st[b] == b){
179                  if(S[st[b]] == 0) lab[b] += d * 2;
180                  else if(S[st[b]] == 1) lab[b] -= d * 2;
181              }
182          q=queue<int>();
183          for(int x = 1; x <= n_x; ++x)
184              if(st[x] == x && slack[x] && st[slack[x]] != x && e_delta(g[slack[x]][x]) ==
    ↪ 0)
185                  if(on_found_edge(g[slack[x]][x]))return true;
186          for(int b = n + 1; b <= n_x; ++b)
187              if(st[b] == b && S[b] == 1 && lab[b] == 0)expand_blossom(b);
188          }
189          return false;
190      }
191      inline pair<long long, int> solve(){
192          memset(match + 1, 0, sizeof(int) * n);
193          n_x = n;
194          int n_matches = 0;
195          long long tot_weight = 0;
196          for(int u = 0; u <= n; ++u) st[u] = u, flower[u].clear();
197          int w_max = 0;
198          for(int u = 1; u <= n; ++u)
199              for(int v = 1; v <= n; ++v){
200                  flower_from[u][v] = (u == v ? u : 0);
201                  w_max = max(w_max, g[u][v].w);
202              }
203          for(int u = 1; u <= n; ++u) lab[u] = w_max;
204          while(matching()) ++n_matches;
205          for(int u = 1; u <= n; ++u)
206              if(match[u] && match[u] < u)
207                  tot_weight += g[u][match[u]].w;
208          return make_pair(tot_weight, n_matches);
209      }
210      inline void init(){
211          for(int u = 1; u <= n; ++u)
212              for(int v = 1; v <= n; ++v)
213                  g[u][v]=edge(u, v, 0);
214      }
215  };
```

无向图最小割

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

```
4   * 返回值为全局最小割
5   */
6
7   int StoerWagner() {
8       static int v[MAXN], wage[MAXN];
9       static bool vis[MAXN];
10
11      for (int i = 1; i <= n; ++i) v[i] = i;
12
13      int res = INF;
14
15      for (int nn = n; nn > 1; --nn) {
16          memset(vis, 0, sizeof(bool) * (nn + 1));
17          memset(wage, 0, sizeof(int) * (nn + 1));
18
19          int pre, last = 1; // vis[1] = 1;
20
21          for (int i = 1; i < nn; ++i) {
22              pre = last; last = 0;
23              for (int j = 2; j <= nn; ++j) if (!vis[j]) {
24                  wage[j] += edge[v[pre]][v[j]];
25                  if (!last || wage[j] > wage[last]) last = j;
26              }
27              vis[last] = 1;
28          }
29
30          res = std::min(res, wage[last]);
31
32          for (int i = 1; i <= nn; ++i) {
33              edge[v[i]][v[pre]] += edge[v[last]][v[i]];
34              edge[v[pre]][v[i]] += edge[v[last]][v[i]];
35          }
36          v[last] = v[nn];
37      }
38      return res;
39  }
```

最大带权带花树

```
1   //maximum weight blossom,  change g[u][v].w to INF - g[u][v].w when minimum weight blossom
        ↪ is needed
2   //type of ans is long long
3   //replace all int to long long if weight of edge is long long
4
5   struct WeightGraph {
```

```
6    static const int INF = INT_MAX;
7    static const int MAXN = 400;
8    struct edge{
9        int u, v, w;
10       edge() {}
11       edge(int u, int v, int w): u(u), v(v), w(w) {}
12   };
13   int n, n_x;
14   edge g[MAXN * 2 + 1][MAXN * 2 + 1];
15   int lab[MAXN * 2 + 1];
16   int match[MAXN * 2 + 1], slack[MAXN * 2 + 1], st[MAXN * 2 + 1], pa[MAXN * 2 + 1];
17   int flower_from[MAXN * 2 + 1][MAXN+1], S[MAXN * 2 + 1], vis[MAXN * 2 + 1];
18   vector<int> flower[MAXN * 2 + 1];
19   queue<int> q;
20   inline int e_delta(const edge &e){ // does not work inside blossoms
21       return lab[e.u] + lab[e.v] - g[e.u][e.v].w * 2;
22   }
23   inline void update_slack(int u, int x){
24       if(!slack[x] || e_delta(g[u][x]) < e_delta(g[slack[x]][x]))
25           slack[x] = u;
26   }
27   inline void set_slack(int x){
28       slack[x] = 0;
29       for(int u = 1;u <= n; ++u)
30           if(g[u][x].w > 0 && st[u] != x && S[st[u]] == 0)
31               update_slack(u, x);
32   }
33   void q_push(int x){
34       if(x <= n)q.push(x);
35       else for(size_t i = 0;i < flower[x].size(); i++)
36           q_push(flower[x][i]);
37   }
38   inline void set_st(int x, int b){
39       st[x]=b;
40       if(x > n) for(size_t i = 0;i < flower[x].size(); ++i)
41               set_st(flower[x][i], b);
42   }
43   inline int get_pr(int b, int xr){
44       int pr = find(flower[b].begin(), flower[b].end(), xr) - flower[b].begin();
45       if(pr % 2 == 1){
46           reverse(flower[b].begin() + 1, flower[b].end());
47           return (int)flower[b].size() - pr;
48       } else return pr;
49   }
50   inline void set_match(int u, int v){
51       match[u]=g[u][v].v;
```

```
52          if(u > n){
53              edge e=g[u][v];
54              int xr = flower_from[u][e.u], pr=get_pr(u, xr);
55              for(int i = 0;i < pr; ++i)
56                  set_match(flower[u][i], flower[u][i ^ 1]);
57              set_match(xr, v);
58              rotate(flower[u].begin(), flower[u].begin()+pr, flower[u].end());
59          }
60      }
61      inline void augment(int u, int v){
62          for(; ; ){
63              int xnv=st[match[u]];
64              set_match(u, v);
65              if(!xnv)return;
66              set_match(xnv, st[pa[xnv]]);
67              u=st[pa[xnv]], v=xnv;
68          }
69      }
70      inline int get_lca(int u, int v){
71          static int t=0;
72          for(++t; u || v; swap(u, v)){
73              if(u == 0)continue;
74              if(vis[u] == t)return u;
75              vis[u] = t;
76              u = st[match[u]];
77              if(u) u = st[pa[u]];
78          }
79          return 0;
80      }
81      inline void add_blossom(int u, int lca, int v){
82          int b = n + 1;
83          while(b <= n_x && st[b]) ++b;
84          if(b > n_x) ++n_x;
85          lab[b] = 0, S[b] = 0;
86          match[b] = match[lca];
87          flower[b].clear();
88          flower[b].push_back(lca);
89          for(int x = u, y; x != lca; x = st[pa[y]]) {
90              flower[b].push_back(x),
91              flower[b].push_back(y = st[match[x]]),
92              q_push(y);
93          }
94          reverse(flower[b].begin() + 1, flower[b].end());
95          for(int x = v, y; x != lca; x = st[pa[y]]) {
96              flower[b].push_back(x),
97              flower[b].push_back(y = st[match[x]]),
```

```
98          q_push(y);
99        }
100       set_st(b, b);
101       for(int x = 1; x <= n_x; ++x) g[b][x].w = g[x][b].w = 0;
102       for(int x = 1; x <= n; ++x) flower_from[b][x] = 0;
103       for(size_t i = 0 ; i < flower[b].size(); ++i){
104           int xs = flower[b][i];
105           for(int x = 1; x <= n_x; ++x)
106               if(g[b][x].w == 0 || e_delta(g[xs][x]) < e_delta(g[b][x]))
107                   g[b][x] = g[xs][x], g[x][b] = g[x][xs];
108           for(int x = 1;x <= n; ++x)
109               if(flower_from[xs][x]) flower_from[b][x] = xs;
110       }
111       set_slack(b);
112     }
113     inline void expand_blossom(int b){ // S[b]  ==  1
114       for(size_t i = 0; i < flower[b].size(); ++i)
115           set_st(flower[b][i], flower[b][i]);
116       int xr = flower_from[b][g[b][pa[b]].u], pr = get_pr(b, xr);
117       for(int i = 0; i < pr; i += 2){
118           int xs = flower[b][i], xns = flower[b][i + 1];
119           pa[xs] = g[xns][xs].u;
120           S[xs] = 1, S[xns] = 0;
121           slack[xs] = 0, set_slack(xns);
122           q_push(xns);
123       }
124       S[xr] = 1, pa[xr] = pa[b];
125       for(size_t i = pr + 1;i < flower[b].size(); ++i){
126           int xs = flower[b][i];
127           S[xs] = -1, set_slack(xs);
128       }
129       st[b] = 0;
130     }
131     inline bool on_found_edge(const edge &e){
132       int u = st[e.u], v = st[e.v];
133       if(S[v] == -1){
134           pa[v] = e.u, S[v] = 1;
135           int nu = st[match[v]];
136           slack[v] = slack[nu] = 0;
137           S[nu] = 0, q_push(nu);
138       }else if(S[v] == 0){
139           int lca = get_lca(u, v);
140           if(!lca) return augment(u, v), augment(v, u), true;
141           else add_blossom(u, lca, v);
142       }
143       return false;
```

```
144         }
145     inline bool matching(){
146         memset(S + 1, -1, sizeof(int) * n_x);
147         memset(slack + 1, 0, sizeof(int) * n_x);
148         q = queue<int>();
149         for(int x = 1;x <= n_x; ++x)
150             if(st[x] == x && !match[x]) pa[x]=0, S[x]=0, q_push(x);
151         if(q.empty())return false;
152         for(;;){
153             while(q.size()){
154                 int u = q.front();q.pop();
155                 if(S[st[u]] == 1)continue;
156                 for(int v = 1;v <= n; ++v)
157                     if(g[u][v].w > 0 && st[u] != st[v]){
158                         if(e_delta(g[u][v]) == 0){
159                             if(on_found_edge(g[u][v]))return true;
160                         }else update_slack(u, st[v]);
161                     }
162             }
163             int d = INF;
164             for(int b = n + 1; b <= n_x;++b)
165                 if(st[b] == b && S[b] == 1)d = min(d, lab[b]/2);
166             for(int x = 1; x <= n_x; ++x)
167                 if(st[x] == x && slack[x]){
168                     if(S[x] == -1)d = min(d, e_delta(g[slack[x]][x]));
169                     else if(S[x] == 0)d = min(d, e_delta(g[slack[x]][x])/2);
170                 }
171             for(int u = 1; u <= n; ++u){
172                 if(S[st[u]] == 0){
173                     if(lab[u] <= d)return 0;
174                     lab[u] -= d;
175                 }else if(S[st[u]] == 1)lab[u] += d;
176             }
177             for(int b = n+1; b <= n_x; ++b)
178                 if(st[b] == b){
179                     if(S[st[b]] == 0) lab[b] += d * 2;
180                     else if(S[st[b]] == 1) lab[b] -= d * 2;
181                 }
182             q=queue<int>();
183             for(int x = 1; x <= n_x; ++x)
184                 if(st[x] == x && slack[x] && st[slack[x]] != x && e_delta(g[slack[x]][x]) ==
    ↪ 0)
185                     if(on_found_edge(g[slack[x]][x]))return true;
186             for(int b = n + 1; b <= n_x; ++b)
187                 if(st[b] == b && S[b] == 1 && lab[b] == 0)expand_blossom(b);
188         }
```

```
189        return false;
190    }
191    inline pair<long long, int> solve(){
192        memset(match + 1, 0, sizeof(int) * n);
193        n_x = n;
194        int n_matches = 0;
195        long long tot_weight = 0;
196        for(int u = 0; u <= n; ++u) st[u] = u, flower[u].clear();
197        int w_max = 0;
198        for(int u = 1; u <= n; ++u)
199            for(int v = 1; v <= n; ++v){
200                flower_from[u][v] = (u == v ? u : 0);
201                w_max = max(w_max, g[u][v].w);
202            }
203        for(int u = 1; u <= n; ++u) lab[u] = w_max;
204        while(matching()) ++n_matches;
205        for(int u = 1; u <= n; ++u)
206            if(match[u] && match[u] < u)
207                tot_weight += g[u][match[u]].w;
208        return make_pair(tot_weight, n_matches);
209    }
210    inline void init(){
211        for(int u = 1; u <= n; ++u)
212            for(int v = 1; v <= n; ++v)
213                g[u][v]=edge(u, v, 0);
214    }
215 };
```

必经点 dominator tree

```
1  //solve(s, n, raw_g): s is the root and base accords to base of raw_g
2  //idom[x] will be x if x does not have a dominator,and will be -1 if x is not reachable from
   ↪ s.
3
4  struct dominator_tree {
5      int base, dfn[N], sdom[N], idom[N], id[N], f[N], fa[N], smin[N], stamp;
6      Graph *g;
7      void predfs(int u) {
8          id[dfn[u] = stamp++] = u;
9          for (int i = g -> adj[u]; ~i; i = g -> nxt[i]) {
10             int v = g -> v[i];
11             if (dfn[v] < 0) {
12                 f[v] = u;
13                 predfs(v);
14             }
```

```
15          }
16      }
17      int getfa(int u) {
18          if (fa[u] == u) return u;
19          int ret = getfa(fa[u]);
20          if (dfn[sdom[smin[fa[u]]]] < dfn[sdom[smin[u]]])
21              smin[u] = smin[fa[u]];
22          return fa[u] = ret;
23      }
24      void solve (int s, int n, Graph *raw_graph) {
25          g = raw_graph;
26          base = g -> base;
27          memset(dfn + base, -1, sizeof(*dfn) * n);
28          memset(idom + base, -1, sizeof(*idom) * n);
29          static Graph pred, tmp;
30          pred.init(base, n);
31          for (int i = 0; i < n; ++i) {
32              for (int p = g -> adj[i + base]; ~p; p = g -> nxt[p])
33                  pred.ins(g -> v[p], i + base);
34          }
35          stamp = 0; tmp.init(base, n); predfs(s);
36          for (int i = 0; i < stamp; ++i) {
37              fa[id[i]] = smin[id[i]] = id[i];
38          }
39          for (int o = stamp - 1; o >= 0; --o) {
40              int x = id[o];
41              if (o) {
42                  sdom[x] = f[x];
43                  for (int i = pred.adj[x]; ~i; i = pred.nxt[i]) {
44                      int p = pred.v[i];
45                      if (dfn[p] < 0) continue;
46                      if (dfn[p] > dfn[x]) {
47                          getfa(p);
48                          p = sdom[smin[p]];
49                      }
50                      if (dfn[sdom[x]] > dfn[p]) sdom[x] = p;
51                  }
52                  tmp.ins(sdom[x], x);
53              }
54              while (~tmp.adj[x]) {
55                  int y = tmp.v[tmp.adj[x]];
56                  tmp.adj[x] = tmp.nxt[tmp.adj[x]];
57                  getfa(y);
58                  if (x != sdom[smin[y]]) idom[y] = smin[y];
59                  else idom[y] = x;
60              }
```

```
61              for (int i = g -> adj[x]; ~i; i = g -> nxt[i])
62                  if (f[g -> v[i]] == x) fa[g -> v[i]] = x;
63          }
64          idom[s] = s;
65          for (int i = 1; i < stamp; ++i) {
66              int x = id[i];
67              if (idom[x] != sdom[x]) idom[x] = idom[idom[x]];
68          }
69      }
70 };
```

Chapter 4

字符串

Manacher

```cpp
//prime is the origin string(0-base)
//-10,-1,-20 are added to s
//length of s is exactly 2 * l + 3
inline void manacher(char prime[]) {
    int l = strlen(prime), n = 0;
    s[n++] = -10;
    s[n++] = -1;
    for (int i = 0; i < l; ++i) {
        s[n++] = prime[i];
        s[n++] = -1;
    }
    s[n++] = -20; f[0] = 1;
    int mx = 0, id = 0;
    for (int i = 1; i + 1 < n; ++i) {
        f[i] = i > mx ? 1 : min(f[id * 2 - i], mx - i + 1);
        while (s[i + f[i]] == s[i - f[i]]) ++f[i];
        if (i + f[i] - 1 > mx) {
            mx = i + f[i] - 1;
            id = i;
        }
    }
}
```

Chapter 5

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);
}
```