

Grimoire's Standard Code Library^{*}

Shanghai Jiao Tong University

Dated: 2017 年 9 月 19 日

^{*} <https://github.com/kzoacn/Grimoire>

目录

1	代数	5
1.1	$O(n^2 \log n)$ 求线性递推数列第 n 项	5
1.2	闪电数论变换与魔力 CRT	6
1.3	多项式求逆	7
1.4	多项式除法	8
1.5	多项式取指数取对数	9
2	数论	11
2.1	大整数相乘取模	11
2.2	线段下整点	11
2.3	中国剩余定理	11
3	图论	13
3.1	一般图匹配	13
3.2	无向图最小割	15
3.3	最大带权带花树	16
3.4	闪电二分图匹配	20
4	字符串	23
4.1	manacher	23
5	技巧	25
5.1	无敌的读入优化	25
5.2	真正释放 STL 内存	26

Chapter 1

代数

$O(n^2 \log n)$ 求线性递推数列第 n 项

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

```
1 void linear_recurrence(long long n, int m, int a[], int c[], int p) {
2     long long v[M] = {1 % p}, u[M << 1], msk = !!n;
3     for(long long i(n); i > 1; i >= 1) {
4         msk <= 1;
5     }
6     for(long long x(0); msk; msk >= 1, x <= 1) {
7         fill_n(u, m < 1, 0);
8         int b(!!(n & msk));
9         x |= b;
10        if(x < m) {
11            u[x] = 1 % p;
12        }else {
13            for(int i(0); i < m; i++) {
14                for(int j(0), t(i + b); j < m; j++, t++) {
15                    u[t] = (u[t] + v[i] * v[j]) % p;
16                }
17            }
18            for(int i((m < 1) - 1); i >= m; i--) {
19                for(int j(0), t(i - m); j < m; j++, t++) {
20                    u[t] = (u[t] + c[j] * u[i]) % p;
21                }
22            }
23        }
24        copy(u, u + m, v);
25    }
26    //a[n] = v[0] * a[0] + v[1] * a[1] + ... + v[m - 1] * a[m - 1].
27    for(int i(m); i < 2 * m; i++) {
28        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 memcpy(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] -> 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 $a * b \equiv 1 \pmod{x^n}$

```

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         memcpy(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    memcpy(tB, b, 0, m);
22    if (m < M) meminit(tB, m, M);
23    memcpy(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 $b \equiv e^a \pmod{x^n}$ or $b \equiv \ln a \pmod{x^n}$

```

1 void getDiff(int *a, int *b, int n) { // 多项式取微分
2     for (int i = 0; i + 1 < n; i++) {
3         b[i] = 1ll * (i + 1) * a[i + 1] % MOD;
4     }
5     b[n - 1] = 0;
6 }
7 void getInt(int *a, int *b, int n) { // 多项式取积分, 积分常数为 0
8     static int inv[MAXN];
9     inv[1] = 1;
10    for (int i = 2; i < n; i++) {
11        inv[i] = 1ll * (MOD - MOD / i) * inv[MOD % i] % MOD;
12    }
13    b[0] = 0;
14    for (int i = 1; i < n; i++) {
15        b[i] = 1ll * a[i - 1] * inv[i] % MOD;
16    }
17 }
18 void getLn(int *a, int *b, int n) {
19     static int inv[MAXN], d[MAXN];
20     int M = 1;
21     for (; M <= 2 * (n - 1); M <= 1);
22     getInv(a, inv, n);
23     getDiff(a, d, n);
24     meminit(d, n, M);
25     meminit(inv, n, M);
26     DFT(d, M, 0); DFT(inv, M, 0);
27     for (int i = 0; i < M; i++) {
28         d[i] = 1ll * d[i] * inv[i] % MOD;
29     }
30     DFT(d, M, 1);
31     getInt(d, b, n);
32 }
33 void getExp(int *a, int *b, int n) {
34     static int ln[MAXN], tmp[MAXN];
35     b[0] = 1;
36     for (int c = 2, M = 1; c < (n < 1); c <= 1) {
37         for (; M <= 2 * (c - 1); M <= 1);
38         int bound = std::min(c, n);
39         memcpy(tmp, a, 0, bound);
40         meminit(tmp, bound, M);
41         meminit(b, c, M);
42         getLn(b, ln, c);
43         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] = 111 * b[i] * (111 - 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

```
1 // x 与 y 须非负
2 long long mult(long long x, long long y, long long MODN) {
3     long long t = (x * y - (long long)((long double)x / MODN * y + 1e-3) * MODN) % MODN;
4     return t < 0 ? t + MODN : t;
5 }
```

线段下整点

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

```
1 LL solve(LL n, LL a, LL b, LL m){
2     if(b==0) return n*(a/m);
3     if(a>=m) return n*(a/m)+solve(n, a%m, b, m);
4     if(b>=m) return (n-1)*n/2*(b/m)+solve(n, a, b%m, m);
5     return solve((a+b*n)/m, (a+b*n)%m, m, b);
6 }
```

中国剩余定理

first is remainder, second is module

```
1 inline void fix(LL &x, LL y) {
2     x = (x % y + y) % y;
3 }
4 bool solve(int n, std::pair<LL, LL> a[],
5             std::pair<LL, LL> &ans) {
6     ans = std::make_pair(1, 1);
7     for (int i = 0; i < n; ++i) {
8         LL num, y;
9         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

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 }

```

无向图最小割

```

1  /*
2   * Stoer Wagner 全局最小割  $O(V^3)$ 
3   * lbase, 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]))slack[x]=u;
25    }
26    inline void set_slack(int x){
27        slack[x]=0;
28        for(int u=1;u<=n;++u)
29            if(g[u][x].w>0&&st[u]!=x&&S[st[u]]==0)update_slack(u,x);
30    }
31    void q_push(int x){
32        if(x<=n)q.push(x);

```



```

33     else for(size_t i=0;i<flower[x].size();i++)q_push(flower[x][i]);
34 }
35 inline void set_st(int x,int b){
36     st[x]=b;
37     if(x>n)for(size_t i=0;i<flower[x].size();++i)
38         set_st(flower[x][i],b);
39 }
40 inline int get_pr(int b,int xr){
41     int pr=find(flower[b].begin(),flower[b].end(),xr)-flower[b].begin();
42     if(pr%2==1){
43         reverse(flower[b].begin()+1,flower[b].end());
44         return (int)flower[b].size()-pr;
45     }else return pr;
46 }
47 inline void set_match(int u,int v){
48     match[u]=g[u][v].v;
49     if(u>n){
50         edge e=g[u][v];
51         int xr=flower_from[u][e.u],pr=get_pr(u,xr);
52         for(int i=0;i<pr;++i)set_match(flower[u][i],flower[u][i^1]);
53         set_match(xr,v);
54         rotate(flower[u].begin(),flower[u].begin()+pr,flower[u].end());
55     }
56 }
57 inline void augment(int u,int v){
58     for(;;){
59         int xnv=st[match[u]];
60         set_match(u,v);
61         if(!xnv)return;
62         set_match(xnv,st[pa[xnv]]);
63         u=st[pa[xnv]],v=xnv;
64     }
65 }
66 inline int get_lca(int u,int v){
67     static int t=0;
68     for(++t;u||v;swap(u,v)){
69         if(u==0)continue;
70         if(vis[u]==t)return u;
71         vis[u]=t;
72         u=st[match[u]];
73         if(u)u=st[pa[u]];
74     }
75     return 0;
76 }
77 inline void add_blossom(int u,int lca,int v){
78     int b=n+1;

```

```

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

```

```

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

```

```

171     }
172     q=queue<int>();
173     for(int x=1;x<=n_x;++x)
174         if(st[x]==x&&slack[x]&&st[slack[x]]!=x&&e_delta(g[slack[x]][x])==0)
175             if(on_found_edge(g[slack[x]][x]))return true;
176     for(int b=n+1;b<=n_x;++b)
177         if(st[b]==b&&S[b]==1&&lab[b]==0)expand_blossom(b);
178     }
179     return false;
180 }
181 inline pair<long long,int> solve(){
182     memset(match+1,0,sizeof(int)*n);
183     n_x=n;
184     int n_matches=0;
185     long long tot_weight=0;
186     for(int u=0;u<=n;++u)st[u]=u,flower[u].clear();
187     int w_max=0;
188     for(int u=1;u<=n;++u)
189         for(int v=1;v<=n;++v){
190             flower_from[u][v]=(u==v?u:0);
191             w_max=max(w_max,g[u][v].w);
192         }
193     for(int u=1;u<=n;++u)lab[u]=w_max;
194     while(matching())++n_matches;
195     for(int u=1;u<=n;++u)
196         if(match[u]&&match[u]<u)
197             tot_weight+=g[u][match[u]].w;
198     return make_pair(tot_weight,n_matches);
199 }
200 inline void init(){
201     for(int u=1;u<=n;++u)
202         for(int v=1;v<=n;++v)
203             g[u][v]=edge(u,v,0);
204 }
205 };

```

闪电二分图匹配

```

1 int matchx[N], matchy[N], level[N];
2 vector<int> edge[N];
3 bool dfs(int x) {
4     for (int i = 0; i < (int)edge[x].size(); ++i) {
5         int y = edge[x][i];
6         int w = matchy[y];
7         if (w == -1 || level[x] + 1 == level[w] && dfs(w)) {

```

```
8         matchx[x] = y;
9         matchy[y] = x;
10        return true;
11    }
12 }
13 level[x] = -1;
14 return false;
15 }
16 int solve() {
17     memset(matchx, -1, sizeof(*matchx) * n);
18     memset(matchy, -1, sizeof(*matchy) * m);
19     for (int ans = 0; ; ) {
20         std::vector<int> q;
21         for (int i = 0; i < n; ++i) {
22             if (matchx[i] == -1) {
23                 level[i] = 0;
24                 q.push_back(i);
25             } else {
26                 level[i] = -1;
27             }
28         }
29         for (int head = 0; head < (int)q.size(); ++head) {
30             int x = q[head];
31             for (int i = 0; i < (int)edge[x].size(); ++i) {
32                 int y = edge[x][i];
33                 int w = matchy[y];
34                 if (w != -1 && level[w] < 0) {
35                     level[w] = level[x] + 1;
36                     q.push_back(w);
37                 }
38             }
39         }
40         int delta = 0;
41         for (int i = 0; i < n; ++i) {
42             if (matchx[i] == -1 && dfs(i)) {
43                 delta++;
44             }
45         }
46         if (delta == 0) {
47             return ans;
48         } else {
49             ans += delta;
50         }
51     }
52 }
```


Chapter 4

字符串

manacher

```
1 //prime is the origin string(0-base)
2 //-10,-1,-20 are added to s
3 //length of s is exactly 2 * l + 3
4 inline void manacher(char prime[]) {
5     int l = strlen(prime), n = 0;
6     s[n++] = -10;
7     s[n++] = -1;
8     for (int i = 0; i < l; ++i) {
9         s[n++] = prime[i];
10        s[n++] = -1;
11    }
12    s[n++] = -20; f[0] = 1;
13    int mx = 0, id = 0;
14    for (int i = 1; i + 1 < n; ++i) {
15        f[i] = i > mx ? 1 : min(f[id * 2 - i], mx - i + 1);
16        while (s[i + f[i]] == s[i - f[i]]) ++f[i];
17        if (i + f[i] - 1 > mx) {
18            mx = i + f[i] - 1;
19            id = i;
20        }
21    }
22 }
```


Chapter 5

技巧

无敌的读入优化

```
1 // getchar() 读入优化 << 关同步 cin << 此优化
2 // 用 isdigit() 会小幅变慢
3 // 返回 false 表示读到文件尾
4 namespace Reader {
5     const int L = (1 << 15) + 5;
6     char buffer[L], *S, *T;
7     __inline bool getchar(char &ch) {
8         if (S == T) {
9             T = (S = buffer) + fread(buffer, 1, L, stdin);
10            if (S == T) {
11                ch = EOF;
12                return false;
13            }
14        }
15        ch = *S++;
16        return true;
17    }
18    __inline bool getint(int &x) {
19        char ch; bool neg = 0;
20        for (; getchar(ch) && (ch < '0' || ch > '9'); ) neg ^= ch == '-';
21        if (ch == EOF) return false;
22        x = ch - '0';
23        for (; getchar(ch), ch >= '0' && ch <= '9'; )
24            x = x * 10 + ch - '0';
25        if (neg) x = -x;
26        return true;
27    }
28 }
```

真正释放 STL 内存

```
1 template <typename T>
2 __inline void clear(T& container) {
3     container.clear(); // 或者删除了一堆元素
4     T(container).swap(container);
5 }
```