Grimoire's Standard Code Library*

Shanghai Jiao Tong University

Dated: 2017 年 10 月 12 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* n*th 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 $a * b \equiv 1(\mod x^n)$

```
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);
    }
}
```

快速沃尔什变换

```
void FWT(LL a[],int n,int ty){
    for(int d=1;d<n;d<<=1){
        for(int m=(d<<1),i=0;i<n;i+=m){
            if(ty==1){
                for(int j=0;j<d;j++){
                    LL x=a[i+j],y=a[i+j+d];
                    a[i+j]=x+y;
                    a[i+j+d]=x-y;
                    //xor:a[i+j]=x+y,a[i+j+d]=x-y;
                    //and:a[i+j]=x+y;
                    //or:a[i+j+d]=x+y;
                }
            }else{
                for(int j=0;j<d;j++){
                    LL x=a[i+j],y=a[i+j+d];
                    a[i+j]=(x+y)/2;
                    a[i+j+d]=(x-y)/2;
                    //xor:a[i+j]=(x+y)/2,a[i+j+d]=(x-y)/2;
                    //and:a[i+j]=x-y;
                    //or:a[i+j+d]=y-x;
                }
            }
        }
    }
}
    FWT(a,1<<n,1);
    FWT(b,1<<n,1);
    for(int i=0;i<(1<<n);i++)
        c[i]=a[i]*b[i];
    FWT(c,1<<n,-1);
```

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);
```

```
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

```
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

```
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];
```

```
        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;
        }
    }
```

```
}
```

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;
}
```

一般最大权匹配

```
//maximum weight blossom,  change g[u][v].w to INF - g[u][v].w when minimum weight blossom is needed
//type of ans is long long
//replace all int to long long if weight of edge is long long

struct WeightGraph {
    static const int INF = INT_MAX;
    static const int MAXN = 400;
    struct edge{
        int u, v, w;
        edge() {}
        edge(int u, int v, int w): u(u), v(v), w(w) {}
    };
    int n, n_x;
    edge g[MAXN * 2 + 1][MAXN * 2 + 1];
    int lab[MAXN * 2 + 1];
    int match[MAXN * 2 + 1], slack[MAXN * 2 + 1], st[MAXN * 2 + 1], pa[MAXN * 2 + 1];
    int flower_from[MAXN * 2 + 1][MAXN+1], S[MAXN * 2 + 1], vis[MAXN * 2 + 1];
    vector<int> flower[MAXN * 2 + 1];
    queue<int> q;
    inline int e_delta(const edge &e){ // does not work inside blossoms
        return lab[e.u] + lab[e.v] - g[e.u][e.v].w * 2;
    }
    inline void update_slack(int u, int x){
        if(!slack[x] || e_delta(g[u][x]) < e_delta(g[slack[x]][x]))
            slack[x] = u;
    }
    inline void set_slack(int x){
        slack[x] = 0;
        for(int u = 1;u <= n; ++u)
            if(g[u][x].w > 0 && st[u] != x && S[st[u]] == 0)
                update_slack(u, x);
    }
    void q_push(int x){
        if(x <= n)q.push(x);
        else for(size_t i = 0;i < flower[x].size(); i++)
            q_push(flower[x][i]);
    }
    inline void set_st(int x, int b){
        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;
```

```
        match[b] = match[lca];
        flower[b].clear();
        flower[b].push_back(lca);
        for(int x = u, y; x != lca; x = st[pa[y]]) {
            flower[b].push_back(x),
            flower[b].push_back(y = st[match[x]]),
            q_push(y);
        }
        reverse(flower[b].begin() + 1, flower[b].end());
        for(int x = v, y; x != lca; x = st[pa[y]]) {
            flower[b].push_back(x),
            flower[b].push_back(y = st[match[x]]),
            q_push(y);
        }
        set_st(b, b);
        for(int x = 1; x <= n_x; ++x) g[b][x].w = g[x][b].w = 0;
        for(int x = 1; x <= n; ++x) flower_from[b][x] = 0;
        for(size_t i = 0 ; i < flower[b].size(); ++i){
            int xs = flower[b][i];
            for(int x = 1; x <= n_x; ++x)
                if(g[b][x].w == 0 || e_delta(g[xs][x]) < e_delta(g[b][x]))
                    g[b][x] = g[xs][x], g[x][b] = g[x][xs];
            for(int x = 1;x <= n; ++x)
                if(flower_from[xs][x]) flower_from[b][x] = xs;
        }
        set_slack(b);
    }
    inline void expand_blossom(int b){ // S[b]  ==  1
        for(size_t i = 0; i < flower[b].size(); ++i)
            set_st(flower[b][i], flower[b][i]);
        int xr = flower_from[b][g[b][pa[b]].u], pr = get_pr(b, xr);
        for(int i = 0; i < pr; i += 2){
            int xs = flower[b][i], xns = flower[b][i + 1];
            pa[xs] = g[xns][xs].u;
            S[xs] = 1, S[xns] = 0;
            slack[xs] = 0, set_slack(xns);
            q_push(xns);
        }
        S[xr] = 1, pa[xr] = pa[b];
        for(size_t i = pr + 1;i < flower[b].size(); ++i){
            int xs = flower[b][i];
            S[xs] = -1, set_slack(xs);
        }
        st[b] = 0;
    }
    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)
```

```
                if(st[b] == b){
                    if(S[st[b]] == 0) lab[b] += d * 2;
                    else if(S[st[b]] == 1) lab[b] -= d * 2;
                }
            q=queue<int>();
            for(int x = 1; x <= n_x; ++x)
                if(st[x] == x && slack[x] && st[slack[x]] != x && e_delta(g[slack[x]][x]) == 0)
                    if(on_found_edge(g[slack[x]][x]))return true;
            for(int b = n + 1; b <= n_x; ++b)
                if(st[b] == b && S[b] == 1 && lab[b] == 0)expand_blossom(b);
        }
        return false;
    }
    inline pair<long long, int> solve(){
        memset(match + 1, 0, sizeof(int) * n);
        n_x = n;
        int n_matches = 0;
        long long tot_weight = 0;
        for(int u = 0; u <= n; ++u) st[u] = u, flower[u].clear();
        int w_max = 0;
        for(int u = 1; u <= n; ++u)
            for(int v = 1; v <= n; ++v){
                flower_from[u][v] = (u == v ? u : 0);
                w_max = max(w_max, g[u][v].w);
            }
        for(int u = 1; u <= n; ++u) lab[u] = w_max;
        while(matching()) ++n_matches;
        for(int u = 1; u <= n; ++u)
            if(match[u] && match[u] < u)
                tot_weight += g[u][match[u]].w;
        return make_pair(tot_weight, n_matches);
    }
    inline void init(){
        for(int u = 1; u <= n; ++u)
            for(int v = 1; v <= n; ++v)
                g[u][v]=edge(u, v, 0);
    }
};
```

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]);

        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;
}
```

最大带权带花树

```
//maximum weight blossom,  change g[u][v].w to INF - g[u][v].w when minimum weight blossom
  ↪is needed
//type of ans is long long
//replace all int to long long if weight of edge is long long

struct WeightGraph {
```

```
    static const int INF = INT_MAX;
    static const int MAXN = 400;
    struct edge{
        int u, v, w;
        edge() {}
        edge(int u, int v, int w): u(u), v(v), w(w) {}
    };
    int n, n_x;
    edge g[MAXN * 2 + 1][MAXN * 2 + 1];
    int lab[MAXN * 2 + 1];
    int match[MAXN * 2 + 1], slack[MAXN * 2 + 1], st[MAXN * 2 + 1], pa[MAXN * 2 + 1];
    int flower_from[MAXN * 2 + 1][MAXN+1], S[MAXN * 2 + 1], vis[MAXN * 2 + 1];
    vector<int> flower[MAXN * 2 + 1];
    queue<int> q;
    inline int e_delta(const edge &e){ // does not work inside blossoms
        return lab[e.u] + lab[e.v] - g[e.u][e.v].w * 2;
    }
    inline void update_slack(int u, int x){
        if(!slack[x] || e_delta(g[u][x]) < e_delta(g[slack[x]][x]))
            slack[x] = u;
    }
    inline void set_slack(int x){
        slack[x] = 0;
        for(int u = 1;u <= n; ++u)
            if(g[u][x].w > 0 && st[u] != x && S[st[u]] == 0)
                update_slack(u, x);
    }
    void q_push(int x){
        if(x <= n)q.push(x);
        else for(size_t i = 0;i < flower[x].size(); i++)
            q_push(flower[x][i]);
    }
    inline void set_st(int x, int b){
        st[x]=b;
        if(x > n) for(size_t i = 0;i < flower[x].size(); ++i)
                    set_st(flower[x][i], b);
    }
    inline int get_pr(int b, int xr){
        int pr = find(flower[b].begin(), flower[b].end(), xr) - flower[b].begin();
        if(pr % 2 == 1){
            reverse(flower[b].begin() + 1, flower[b].end());
            return (int)flower[b].size() - pr;
        } else return pr;
    }
    inline void set_match(int u, int v){
        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         }
```

```
        return false;
    }
    inline pair<long long, int> solve(){
        memset(match + 1, 0, sizeof(int) * n);
        n_x = n;
        int n_matches = 0;
        long long tot_weight = 0;
        for(int u = 0; u <= n; ++u) st[u] = u, flower[u].clear();
        int w_max = 0;
        for(int u = 1; u <= n; ++u)
            for(int v = 1; v <= n; ++v){
                flower_from[u][v] = (u == v ? u : 0);
                w_max = max(w_max, g[u][v].w);
            }
        for(int u = 1; u <= n; ++u) lab[u] = w_max;
        while(matching()) ++n_matches;
        for(int u = 1; u <= n; ++u)
            if(match[u] && match[u] < u)
                tot_weight += g[u][match[u]].w;
        return make_pair(tot_weight, n_matches);
    }
    inline void init(){
        for(int u = 1; u <= n; ++u)
            for(int v = 1; v <= n; ++v)
                g[u][v]=edge(u, v, 0);
    }
};
```

必经点 dominator-tree

```
//solve(s, n, raw_g): s is the root and base accords to base of raw_g
//idom[x] will be x if x does not have a dominator,and will be -1 if x is not reachable from
  ↪ s.

struct dominator_tree {
    int base, dfn[N], sdom[N], idom[N], id[N], f[N], fa[N], smin[N], stamp;
    Graph *g;
    void predfs(int u) {
        id[dfn[u] = stamp++] = u;
        for (int i = g -> adj[u]; ~i; i = g -> nxt[i]) {
            int v = g -> v[i];
            if (dfn[v] < 0) {
                f[v] = u;
                predfs(v);
            }
```

```
        }
    }
    int getfa(int u) {
        if (fa[u] == u) return u;
        int ret = getfa(fa[u]);
        if (dfn[sdom[smin[fa[u]]]] < dfn[sdom[smin[u]]])
            smin[u] = smin[fa[u]];
        return fa[u] = ret;
    }
    void solve (int s, int n, Graph *raw_graph) {
        g = raw_graph;
        base = g -> base;
        memset(dfn + base, -1, sizeof(*dfn) * n);
        memset(idom + base, -1, sizeof(*idom) * n);
        static Graph pred, tmp;
        pred.init(base, n);
        for (int i = 0; i < n; ++i) {
            for (int p = g -> adj[i + base]; ~p; p = g -> nxt[p])
                pred.ins(g -> v[p], i + base);
        }
        stamp = 0; tmp.init(base, n); predfs(s);
        for (int i = 0; i < stamp; ++i) {
            fa[id[i]] = smin[id[i]] = id[i];
        }
        for (int o = stamp - 1; o >= 0; --o) {
            int x = id[o];
            if (o) {
                sdom[x] = f[x];
                for (int i = pred.adj[x]; ~i; i = pred.nxt[i]) {
                    int p = pred.v[i];
                    if (dfn[p] < 0) continue;
                    if (dfn[p] > dfn[x]) {
                        getfa(p);
                        p = sdom[smin[p]];
                    }
                    if (dfn[sdom[x]] > dfn[p]) sdom[x] = p;
                }
                tmp.ins(sdom[x], x);
            }
            while (~tmp.adj[x]) {
                int y = tmp.v[tmp.adj[x]];
                tmp.adj[x] = tmp.nxt[tmp.adj[x]];
                getfa(y);
                if (x != sdom[smin[y]]) idom[y] = smin[y];
                else idom[y] = x;
            }
```

```
            for (int i = g -> adj[x]; ~i; i = g -> nxt[i])
                if (f[g -> v[i]] == x) fa[g -> v[i]] = x;
        }
        idom[s] = s;
        for (int i = 1; i < stamp; ++i) {
            int x = id[i];
            if (idom[x] != sdom[x]) idom[x] = idom[idom[x]];
        }
    }
};
```

欧拉回路

```
//从一个奇度点 dfs，sqn 即为回路/路径
//first 存点，second 存边的编号，正反边编号一致
//清空 cur、used 数组
void getCycle(int u)
{
    for(int &i=cur[u]; i < (int)adj[u].size(); ++ i) {
        int id = adj[u][i].second;
        if (used[id]) continue;
        used[id] = true;
        getCycle(adj[u][i].first);
    }
    sqn.push_back(u);
}
```

朱刘最小树形图

```
struct D_MT {
    struct Edge {
        int u, v, w;
        inline Edge() {}
        inline Edge(int _u, int _v, int _w):u(_u), v(_v), w(_w) {
        }
    };
    int nn, mm, n, m, vis[maxn], pre[maxn], id[maxn], in[maxn];
    Edge edges[maxn], bac[maxn];
    void init(int _n) {
        n = _n;
        m = 0;
    }
    void AddEdge(int u, int v, int w) {
        edges[m++] = Edge(u, v, w);
```

```
    }
    int work(int root) {
        int ret = 0;
        while(true) {
            for (int i = 0; i < n; i++) in[i]=inf + 1;
            for (int i = 0; i < m; i++) {
                int u = edges[i].u, v = edges[i].v;
                if(edges[i].w < in[v] && u != v){
                    in[v] = edges[i].w;
                    pre[v] = u;
                }
            }
            for (int i = 0; i < n; i++) {
                if(i == root) continue;
                if(in[i] == inf + 1) return inf;
            }
            int cnt = 0;
            for (int i = 0; i < n; i++) {
                id[i] = -1;
                vis[i] = -1;
            }
            in[root] = 0;
            for (int i = 0; i < n; i++) {
                ret += in[i];
                int v = i;
                while (vis[v] != i&& id[v] == -1 && v != root ){
                    vis[v] = i;
                    v = pre[v];
                }
                if (v != root && id[v] == -1) {
                    for (int u = pre[v]; u != v; u = pre[u]) id[u] = cnt;
                    id[v] = cnt++;
                }
            }
            if (!cnt) break;
            for (int i=0; i<n; i++)
                if (id[i] == -1) id[i] = cnt++;
            for (int i = 0; i < m; i++){
                int u = edges[i].u, v = edges[i].v;
                edges[i].v = id[v];
                edges[i].u = id[u];
                if(id[u] != id[v]) edges[i].w -= in[v];
            }
            n = cnt;
            root = id[root];
        }
```

```
62          return ret;
63      }
64  } MT;
```

Chapter 4

数据结构

LCT

```
struct LCT{
    struct node{
        bool rev;
        int mx,val;
        node *f,*c[2];
        bool d(){return this==f->c[1];}
        bool rt(){return !f||(f->c[0]!=this&&f->c[1]!=this);}
        void sets(node *x,int d){pd();if(x)x->f=this;c[d]=x;rz();}
        void makerv(){rev^=1;swap(c[0],c[1]);}
        void pd(){
            if(rev){
                if(c[0])c[0]->makerv();
                if(c[1])c[1]->makerv();
                rev=0;
            }
        }
        void rz(){
            mx=val;
            if(c[0])mx=max(mx,c[0]->mx);
            if(c[1])mx=max(mx,c[1]->mx);
        }
    }nd[int(1e4)+1];
    void rot(node *x){
        node *y=x->f;if(!y->rt())y->f->pd();
        y->pd();x->pd();bool d=x->d();
        y->sets(x->c[!d],d);
        if(y->rt())x->f=y->f;
        else y->f->sets(x,y->d());
        x->sets(y,!d);
    }
    void splay(node *x){
```

```
        while(!x->rt())
            if(x->f->rt())rot(x);
            else if(x->d()==x->f->d())rot(x->f),rot(x);
            else rot(x),rot(x);
    }
    node* access(node *x){
        node *y=0;
        for(;x;x=x->f){
            splay(x);
            x->sets(y,1);y=x;
        }return y;
    }
    void makert(node *x){
        access(x)->makerv();
        splay(x);
    }
    void link(node *x,node *y){
        makert(x);
        x->f=y;
        access(x);
    }
    void cut(node *x,node *y){
        makert(x);access(y);splay(y);
        y->c[0]=x->f=0;
        y->rz();
    }
    void link(int x,int y){link(nd+x,nd+y);}
    void cut(int x,int y){cut(nd+x,nd+y);}
}T;
```

Kd-tree

```
int n;
LL norm(const LL &x) {
    //    For manhattan distance
       //return std::abs(x);
    //    For euclid distance
    return x * x;
}

struct P{
    int a[2],val;
    int id;
    int& operator[](int s){return a[s];}
    const int& operator[](int s)const{return a[s];}
```

```
14
15      LL dis(const P &b)const{
16          LL ans=0;
17          for (int i = 0; i < 2; ++i) {
18              ans += norm(a[i] - b[i]);
19          }
20          return ans;
21      }
22  }p[maxn];
23
24  bool operator==(const P &a,const P &b){
25      for(int i=0;i<DIM;i++)
26          if(a[i]!=b[i])
27              return false;
28      return true;
29  }
30  bool byVal(P a,P b){
31      return a.val!=b.val ? a.val<b.val : a.id<b.id;
32  }
33
34  struct Rec{
35      int mn[DIM],mx[DIM];
36      Rec(){}
37      Rec(const P &p){
38          for(int i=0;i<DIM;i++){
39              mn[i]=mx[i]=p[i];
40          }
41      }
42      void add(const P &p){
43          for(int i=0;i<DIM;i++){
44              mn[i]=min(p[i],mn[i]);
45              mx[i]=max(p[i],mx[i]);
46          }
47      }
48
49      LL dis(const P &p) {
50          LL ans = 0;
51          for (int i = 0; i < 2; ++i) {
52              //    For minimum distance
53              ans += norm(min(max(p[i], mn[i]), mx[i]) - p[i]);
54              //    For maximum distance
55              //ans += std::max(norm(max[i] - p[i]), norm(min[i] - p[i]));
56          }
57          return ans;
58      }
59  };
```

```
inline Rec operator+(const Rec &ls,const Rec &rs){
    static Rec rec;
    for(int i=0;i<DIM;i++){
        rec.mn[i]=min(ls.mn[i],rs.mn[i]);
        rec.mx[i]=max(ls.mx[i],rs.mx[i]);
    }
    return rec;
}
struct node{
    Rec rec;
    P sep;
    int sum,siz;
    node *c[2];
    node *rz(){
        sum=sep.val;
        rec=Rec(sep);
        siz=1;
        if(c[0]){
            sum+=c[0]->sum;
            rec=rec+c[0]->rec;
            siz+=c[0]->siz;
        }
        if(c[1]){
            sum+=c[1]->sum;
            rec=rec+c[1]->rec;
            siz+=c[1]->siz;
        }
        return this;
    }
    node(){sum=0;siz=1;c[0]=c[1]=0;}
}*root,*re,pool[maxn],*cur=pool;
node *sta[maxn];
P tmp[maxn];
int D,si;
bool cmp(const P &A,const P &B){

    if(!(A[D]==B[D]))
        return A[D]<B[D];
    return A.id<B.id;
}
int top;
node *newnode(){
    if(si)return sta[si--];
    return cur++;
}
node* build(P *p,int l,int r,int d){
```

```
106     int mid=(l+r)>>1;D=d;
107     nth_element(p+l,p+mid,p+r+1,cmp);
108     node *t=newnode();
109     t->sep=p[mid];
110     if(l<=mid-1)
111         t->c[0]=build(p,l,mid-1,d^1);
112     if(mid+1<=r)
113         t->c[1]=build(p,mid+1,r,d^1);
114     return t->rz();
115 }
116 void dfs(node *&t){
117     if(t->c[0])dfs(t->c[0]);
118     tmp[++top]=t->sep;
119     if(t->c[1])dfs(t->c[1]);
120     sta[++si]=t;*t=node();
121     //delete t;
122 }
123 node* rebuild(node *&t){
124     if(!t)return 0;
125     top=0;dfs(t);
126     return build(tmp,1,top,0);
127 }
128 #define siz(x) (x?x->siz:0)
129 void Add(node *&t,const P &p,int d=0){
130     D=d;
131     if(!t){
132         t=newnode();
133         t->sep=p;t->rz();
134         return;
135     }
136     if(t->sep==p){
137         t->sep.val+=p.val;
138         t->rz();
139         return;
140     }
141     if(p[D]<t->sep[D])
142         Add(t->c[0],p,d^1);
143     else
144         Add(t->c[1],p,d^1);
145 
146     t->rz();
147 
148     if(max(siz(t->c[0]),siz(t->c[1]))>0.7*t->siz)
149         re=t;
150 }
151 int ans;
```

```
152
153 bool Out(const Rec &a,const Rec &b){
154     for(int i=0;i<DIM;i++){
155         int l=max(a.mn[i],b.mn[i]);
156         int r=min(a.mx[i],b.mx[i]);
157         if(l>r)
158             return true;
159     }
160     return false;
161 }
162 bool In(const Rec &a,const Rec &b){
163     for(int i=0;i<DIM;i++){
164         if(a.mn[i]<b.mn[i])
165             return false;
166         if(a.mx[i]>b.mx[i])
167             return false;
168     }
169     return true;
170 }
171
172 bool In(const P &a,const Rec &b){
173     for(int i=0;i<DIM;i++){
174         if(!(b.mn[i]<=a[i]&&a[i]<=b.mx[i]))
175             return false;
176     }
177     return true;
178 }
179
180 void Q(node *t,const Rec &R){
181     if(Out(t->rec,R))return ;
182     if(In(t->rec,R)){
183         ans+=t->sum;
184         return;
185     }
186     if(In(t->sep,R))
187         ans+=t->sep.val;
188     if(t->c[0])
189         Q(t->c[0],R);
190     if(t->c[1])
191         Q(t->c[1],R);
192 }
193
194 priority_queue<pair<long long, int> > kNN;
195 void query(node *t, const P &p, int k, int d = 0) {
196     D=d;
197     if (!t || ((int)kNN.size() == k && t->rec.dis(p) > kNN.top().first)) {
```

```
        return;
    }
    kNN.push(make_pair(t->sep.dis(p), t->sep.id));
    if ((int)kNN.size() > k) {
        kNN.pop();
    }
    if (cmp(p, t->sep)) {
        query(t->c[0], p, k, d ^ 1);
        query(t->c[1], p, k, d ^ 1);
    } else {
        query(t->c[1], p, k, d ^ 1);
        query(t->c[0], p, k, d ^ 1);
    }
}
```

虚树

```
int a[maxn*2],sta[maxn*2];
int top=0,k;
void build(){
    top=0;
    sort(a,a+k,bydfn);
    k=unique(a,a+k)-a;
    sta[top++]=1;_n=k;
    for(int i=0;i<k;i++){
        int LCA=lca(a[i],sta[top-1]);
        while(dep[LCA]<dep[sta[top-1]]){
            if(dep[LCA]>=dep[sta[top-2]]){
                add_edge(LCA,sta[--top]);
                if(sta[top-1]!=LCA)sta[top++]=LCA;
                break;
            }add_edge(sta[top-2],sta[top-1]);top--;
        }if(sta[top-1]!=a[i])sta[top++]=a[i];
    }
    while(top>1)
        add_edge(sta[top-2],sta[top-1]),top--;
    for(int i=0;i<k;i++)inr[a[i]]=1;
}
```

树状数组上二分第 k 大

```
int find(int k){
    int cnt=0,ans=0;
    for(int i=22;i>=0;i--){
```

```
        ans+=(1<<i);
        if(ans>n || cnt+d[ans]>=k)ans-=(1<<i);
        else cnt+=d[ans];
    }
    return ans+1;
}
```

Treap

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=1e5+5;
#define sz(x) (x?x->siz:0)
struct Treap{
    struct node{
        int key,val;
        int siz,s;
        node *c[2];
        node(int v=0){
            val=v;
            key=rand();
            siz=1,s=1;
            c[0]=c[1]=0;
        }
        void rz(){siz=s;if(c[0])siz+=c[0]->siz;if(c[1])siz+=c[1]->siz;}
    }pool[maxn],*cur,*root;
    Treap(){cur=pool;}
    node* newnode(int val){return *cur=node(val),cur++;}
    void rot(node *&t,int d){
        if(!t->c[d])t=t->c[!d];
        else{
            node *p=t->c[d];t->c[d]=p->c[!d];
            p->c[!d]=t;t->rz();p->rz();t=p;
        }
    }
    void insert(node *&t,int x){
        if(!t){t=newnode(x);return;}
        if(t->val==x){t->s++;t->siz++;return;}
        insert(t->c[x>t->val],x);
        if(t->key<t->c[x>t->val]->key)
            rot(t,x>t->val);
        else t->rz();
    }
    void del(node *&t,int x){
        if(!t)return;
```

```
37          if(t->val==x){
38              if(t->s>1){t->s--;t->siz--;return;}
39              if(!t->c[0]||!t->c[1]){
40                  if(!t->c[0])t=t->c[1];
41                  else t=t->c[0];
42                  return;
43              }
44              int d=t->c[0]->key<t->c[1]->key;
45              rot(t,d);
46              del(t,x);
47              return;
48          }
49          del(t->c[x>t->val],x);
50          t->rz();
51      }
52      int pre(node *t,int x){
53          if(!t)return INT_MIN;
54          int ans=pre(t->c[x>t->val],x);
55          if(t->val<x)ans=max(ans,t->val);
56          return ans;
57      }
58      int nxt(node *t,int x){
59          if(!t)return INT_MAX;
60          int ans=nxt(t->c[x>=t->val],x);
61          if(t->val>x)ans=min(ans,t->val);
62          return ans;
63      }
64      int rank(node *t,int x){
65          if(!t)return 0;
66          if(t->val==x)return sz(t->c[0]);
67          if(t->val<x)return sz(t->c[0])+t->s+rank(t->c[1],x);
68          if(t->val>x)return rank(t->c[0],x);
69      }
70      int kth(node *t,int x){
71          if(sz(t->c[0])>=x)return kth(t->c[0],x);
72          if(sz(t->c[0])+t->s>=x)return t->val;
73          return kth(t->c[1],x-t->s-sz(t->c[0]));
74      }
75      void deb(node *t){
76          if(!t)return;
77          deb(t->c[0]);
78          printf("%d ",t->val);
79          deb(t->c[1]);
80      }
81      void insert(int x){insert(root,x);}
82      void del(int x){del(root,x);}
```

```
    int pre(int x){return pre(root,x);}
    int nxt(int x){return nxt(root,x);}
    int rank(int x){return rank(root,x);}
    int kth(int x){return kth(root,x);}
    void deb(){deb(root);puts("");}
}T;
```

FHQ-Treap

```
#include<bits/stdc++.h>
using namespace std;
typedef long long LL;
const int maxn=1e5+5;
int in(){
    int r=0,f=1;char c=getchar();
    while(!isdigit(c))f=c=='-'?-1:f,c=getchar();
    while(isdigit(c))r=r*10+c-'0',c=getchar();
    return r*f;
}
int n,m;
#define sz(x) (x?x->siz:0)
struct node{
    int siz,key;
    LL val,sum;
    LL mu,a,d;
    node *c[2],*f;
    void split(int ned,node *&p,node *&q);
    node* rz(){
        sum=val;siz=1;
        if(c[0])sum+=c[0]->sum,siz+=c[0]->siz;
        if(c[1])sum+=c[1]->sum,siz+=c[1]->siz;
        return this;
    }
    void make(LL _mu,LL _a,LL _d){
        sum=sum*_mu+_a*siz+_d*siz*(siz-1)/2;
        val=val*_mu+_a+_d*sz(c[0]);
        mu*=_mu;a=a*_mu+_a;d=d*_mu+_d;
    }
    void pd(){
        if(mu==1&&a==0&&d==0)return;
        if(c[0])c[0]->make(mu,a,d);
        if(c[1])c[1]->make(mu,a+d+d*sz(c[0]),d);
        mu=1;a=d=0;
    }
    node(){mu=1;}
```

```
}nd[maxn*2],*root;
node *merge(node *p,node *q){
    if(!p||!q)return p?p->rz():(q?q->rz():0);
    p->pd();q->pd();
    if(p->key<q->key){
        p->c[1]=merge(p->c[1],q);
        return p->rz();
    }else{
        q->c[0]=merge(p,q->c[0]);
        return q->rz();
    }
}
void node::split(int ned,node *&p,node *&q){
    if(!ned){p=0;q=this;return;}
    if(ned==siz){p=this;q=0;return;}
    pd();
    if(sz(c[0])>=ned){
        c[0]->split(ned,p,q);c[0]=0;rz();
        q=merge(q,this);
    }else{
        c[1]->split(ned-sz(c[0])-1,p,q);c[1]=0;rz();
        p=merge(this,p);
    }
}
int tot;
void C(int l,int r,int v){
    node *p,*q,*x,*y;
    root->split(l-1,p,q);
    q->split(r-l+1,x,y);
    x->make(0,v,0);x->pd();
    root=merge(p,merge(x,y));
}
void A(int l,int r,int d){
    node *p,*q,*x,*y;
    root->split(l-1,p,q);
    q->split(r-l+1,x,y);
    x->make(1,d,d);x->pd();
    root=merge(p,merge(x,y));
}
void I(int ps,int v){
    node *p,*q;
    root->split(ps-1,p,q);
    node *x=nd+(++tot);
    x->key=rand();x->val=v;x->rz();
    root=merge(merge(p,x),q);
}
```

```
LL Q(int l,int r){
    node *p,*q,*x,*y;
    root->split(l-1,p,q);
    q->split(r-l+1,x,y);
    LL ans=x->sum;
    root=merge(p,merge(x,y));
    return ans;
}
int main(){
//  freopen("bzoj3188.in","r",stdin);
    n=in();m=in();
    for(int i=1;i<=n;i++){
        nd[i].val=in();
        nd[i].key=rand();
        nd[i].rz();
        root=merge(root,nd+i);
    }tot=n;
    while(m--){
        int ty=in();
        int l,r;
        if(ty==1){
            l=in();r=in();
            C(l,r,in());
        }else if(ty==2){
            l=in();r=in();
            A(l,r,in());
        }else if(ty==3){
            int ps=in();
            I(ps,in());
        }else if(ty==4){
            l=in();r=in();
            printf("%lld\n",Q(l,r));
        }
    }
    return 0;
}
```

真-FHQTreap

```
const int mo=1e9+7;
int rnd(){
    static int x=1;
    return x=(x*23333+233);
}
int rnd(int n){
```

```
    int x=rnd();
    if(x<0)x=-x;
    return x%n+1;
}
struct node{
    int siz,key;
    int val;
    LL sum;
    node *c[2];
    node* rz(){
        sum=val;siz=1;
        if(c[0])sum+=c[0]->sum,siz+=c[0]->siz;
        if(c[1])sum+=c[1]->sum,siz+=c[1]->siz;
        return this;
    }
    node(){}
    node(int v){
        siz=1;key=rnd();
        val=v;sum=v;
        c[0]=c[1]=0;
    }

}pool[maxn*8],*root,*cur=pool,*old_root,*stop;
node *newnode(int v=0){
    *cur=node(v);
    return cur++;
}
node *old_merge(node *p,node *q){
    if(!p&&!q)return 0;
    node *u=0;
    if(!p||!q)return u=p?p->rz():(q?q->rz():0);
    if(rnd(sz(p)+sz(q))<sz(p)){
        u=p;
        u->c[1]=old_merge(u->c[1],q);
    }else{
        u=q;
        u->c[0]=old_merge(p,u->c[0]);
    }
    return u->rz();
}
node *merge(node *p,node *q){
    if(!p&&!q)return 0;
    node *u=newnode();
    if(!p||!q)return u=p?p->rz():(q?q->rz():0);
    if(rnd(sz(p)+sz(q))<sz(p)){
        *u=*p;
```

```
        u->c[1]=merge(u->c[1],q);
    }else{
        *u=*q;
        u->c[0]=merge(p,u->c[0]);
    }
    return u->rz();
}
node *split(node *u,int l,int r){
    if(l>r||!u)return 0;
    node *x=0;
    if(l==1&&r==sz(u)){
        x=newnode();
        *x=*u;
        return x->rz();
    }
    int lsz=sz(u->c[0]);
    if(r<=lsz)
        return split(u->c[0],l,r);
    if(l>lsz+1)
        return split(u->c[1],l-lsz-1,r-lsz-1);
    x=newnode();
    *x=*u;
    x->c[0]=split(u->c[0],l,lsz);
    x->c[1]=split(u->c[1],1,r-lsz-1);
    return x->rz();
}
```

莫队上树

```
bool operator<(qes a,qes b){
    if(dfn[a.x]/B!=dfn[b.x]/B)return dfn[a.x]/B<dfn[b.x]/B;
    if(dfn[a.y]/B!=dfn[b.y]/B)return dfn[a.y]/B<dfn[b.y]/B;
    if(a.tm/B!=b.tm/B)return a.tm/B<b.tm/B;
    return a.tm<b.tm;
}
void vxor(int x){
    if(vis[x])ans-=(LL)W[cnt[col[x]]]*V[col[x]],cnt[col[x]]--;
    else cnt[col[x]]++,ans+=(LL)W[cnt[col[x]]]*V[col[x]];
    vis[x]^=1;
}
void change(int x,int y){
    if(vis[x]){
        vxor(x);col[x]=y;vxor(x);
    }else col[x]=y;
}
```

```
void TimeMachine(int tar){//XD
    for(int i=now+1;i<=tar;i++)change(C[i].x,C[i].y);
    for(int i=now;i>tar;i--)change(C[i].x,C[i].pre);
    now=tar;
}
void vxor(int x,int y){
    while(x!=y)if(dep[x]>dep[y])vxor(x),x=fa[x];
    else vxor(y),y=fa[y];
}
    for(int i=1;i<=q;i++){
        int ty=getint(),x=getint(),y=getint();
        if(ty&&dfn[x]>dfn[y])swap(x,y);
        if(ty==0) C[++Csize]=(oper){x,y,pre[x],i},pre[x]=y;
        else Q[Qsize+1]=(qes){x,y,Qsize+1,Csize},Qsize++;
    }sort(Q+1,Q+1+Qsize);
    int u=Q[1].x,v=Q[1].y;
    TimeMachine(Q[1].tm);
    vxor(Q[1].x,Q[1].y);
    int LCA=lca(Q[1].x,Q[1].y);
    vxor(LCA);anss[Q[1].id]=ans;vxor(LCA);
    for(int i=2;i<=Qsize;i++){
        TimeMachine(Q[i].tm);
        vxor(Q[i-1].x,Q[i].x);
        vxor(Q[i-1].y,Q[i].y);
        int LCA=lca(Q[i].x,Q[i].y);
        vxor(LCA);
        anss[Q[i].id]=ans;
        vxor(LCA);
    }
```

Chapter 5

字符串

Manacher

```
//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;
        }
    }
}
```

指针版回文自动机

```
/*
 * Palindrome Automaton - pointer version
 * PAMPAMPAM? PAMPAMPAM!
 */

```

```
namespace PAM {
    struct Node *pool_pointer;
    struct Node {
        Node *fail, *to[26];
        int cnt, len;

        Node() {}
        Node(int len): len(len) {
            memset(to, 0, sizeof(to));
            fail = 0;
            cnt = 0;
        }

        void *operator new (size_t) {
            return pool_pointer++;
        }
    } pool[100005], *root[2], *last;
    int pam_len, str[100005];

    void init() {
        pool_pointer = pool;
        root[0] = new Node(0);
        root[1] = new Node(-1);
        root[0]->fail = root[1]->fail = root[1];
        str[pam_len = 0] = -1; // different from all characters
        last = root[0];
    }

    void extend(char ch) {
        static Node *p, *np, *q;

        int x = str[++pam_len] = ch - 'a';

        p = last;
        while (str[pam_len - p->len - 1] != x)
            p = p->fail;
        if (!p->to[x]) {
            np = new Node(p->len + 2), q = p->fail;
            while (str[pam_len - q->len - 1] != x) q = q->fail;
            np->fail = q->to[x] ? q->to[x] : root[0];
            p->to[x] = np;
        }
        last = p->to[x];
        ++last->cnt;
    }
}
```

数组版后缀自动机

```cpp
/*
 * Suffix Automaton - array version
 * SAMSAMSAM? SAMSAMSAM!
 */

namespace SAM {
    int to[100005 << 1][26], parent[100005 << 1], step[100005 << 1], tot;
    int root, np;
    int sam_len;

    int newnode(int STEP = 0) {
        ++tot;
        memset(to[tot], 0, sizeof to[tot]);
        parent[tot] = 0;
        step[tot] = STEP;
        return tot;
    }

    void init() {
        tot = 0;
        root = np = newnode(sam_len = 0);
    }

    void extend(char ch) {
        int x = ch - 'a';
        int last = np; np = newnode(++sam_len);
        for (; last && !to[last][x]; last = parent[last])
            to[last][x] = np;
        if (!last) parent[np] = root;
        else {
            int q = to[last][x];
            if (step[q] == step[last] + 1) parent[np] = q;
            else {
                nq = newnode(step[last] + 1);
                memcpy(to[nq], to[q], sizeof to[q]);
                parent[nq] = parent[q];
                parent[q] = parent[np] = nq;
                for (; last && to[last][x] == q; last = parent[last])
                    to[last][x] = nq;
            }
        }
    }
}
```

指针版后缀自动机

```
/*
 * Suffix Automaton - pointer version
 * SAMSAMSAM? SAMSAMSAM!
 */

namespace SAM {
    struct Node *pool_pointer;
    struct Node {
        Node *to[26], *parent;
        int step;

        Node(int STEP = 0): step(STEP) {
            memset(to, 0, sizeof to);
            parent = 0;
            step = 0;
        }

        void *operator new (size_t) {
            return pool_pointer++;
        }
    } pool[100005 << 1], *root, *np;
    int sam_len;

    void init() {
        pool_pointer = pool;
        root = np = new Node(sam_len = 0);
    }

    void extend(char ch) {
        static Node *last, *q, *nq;

        int x = ch - 'a';
        last = np; np = new Node(++sam_len);
        for (; last && !last->to[x]; last = last->parent)
            last->to[x] = np;
        if (!last) np->parent = root;
        else {
            q = last->to[x];
            if (q->step == last->step + 1) np->parent = q;
            else {
                nq = new Node(*q);
                nq->step = last->step + 1;
                q->parent = np->parent = nq;
                for (; last && last->to[x] == q; last = last->parent)
```

```
                    last->to[x] = nq;
                }
            }
        }
```

广义后缀自动机

```
/*
 * EX Suffix Automaton - pointer version
 * SAMSAMSAM? SAMSAMSAM!
 */

namespace SAM {
    struct Node *pool_pointer;
    struct Node {
        Node *parent, *to[26];
        int step;

        Node(int step = 0): step(step) {
            memset(to, 0, sizeof to);
            parent = 0;
        }

        void *operator new (size_t) {
            return pool_pointer++;
        }
    } pool[100005 * 10 << 1], *root, *np;
    int sam_len, now_len;

    void init() {
        sam_len = now_len = 0;
        pool_pointer = pool;
        root = new Node();
    }

    void new_str() { // a new string start
        now_len = 0;
        np = root;
    }

    void extend(char ch) {
        static Node *last, *q, *nq;

        int x = ch - 'a';
```

```
        if (np->to[x]) {
            np = np->to[x];
            ++now_len;
        }
        else {
            last = np; np = new Node(++now_len);
            for (; last && !last->to[x]; last = last->parent)
                last->to[x] = np;
            if (!last) np->parent = root;
            else {
                q = last->to[x];
                if (q->step == last->step + 1) np->parent = q;
                else {
                    nq = new Node(*q);
                    nq->step = last->step + 1;
                    q->parent = np->parent = nq;
                    for (; last && last->to[x] == q; last = last->parent)
                        last->to[x] = nq;
                }
            }
        }

        sam_len = std::max(sam_len, now_len);
    }
}
```

后缀数组

```
const int maxl=1e5+1e4+5;
const int maxn=maxl*2;
int a[maxn],x[maxn],y[maxn],c[maxn],sa[maxn],rank[maxn],height[maxn];
void calc_sa(int n){
    int m=alphabet,k=1;
    memset(c,0,sizeof(*c)*(m+1));
    for(int i=1;i<=n;i++)c[x[i]=a[i]]++;
    for(int i=1;i<=m;i++)c[i]+=c[i-1];
    for(int i=1;i<=n;i++)sa[c[x[i]]--]=i;
    for(;k<=n;k<<=1){
        int tot=k;
        for(int i=n-k+1;i<=n;i++)y[i-n+k]=i;
        for(int i=1;i<=n;i++)
            if(sa[i]>k)y[++tot]=sa[i]-k;
        memset(c,0,sizeof(*c)*(m+1));
        for(int i=1;i<=n;i++)c[x[i]]++;
        for(int i=1;i<=m;i++)c[i]+=c[i-1];
```

```
        for(int i=n;i>=1;i--)sa[c[x[y[i]]]--]=y[i];
        for(int i=1;i<=n;i++)y[i]=x[i];
        tot=1;x[sa[1]]=1;
        for(int i=2;i<=n;i++){
            if(max(sa[i],sa[i-1])+k>n||y[sa[i]]!=y[sa[i-1]]||y[sa[i]+k]!=y[sa[i-1]+k])
                ++tot;
            x[sa[i]]=tot;
        }
        if(tot==n)break;else m=tot;
    }
}
void calc_height(int n){
    for(int i=1;i<=n;i++)rank[sa[i]]=i;
    for(int i=1;i<=n;i++){
        height[rank[i]]=max(0,height[rank[i-1]]-1);
        if(rank[i]==1)continue;
        int j=sa[rank[i]-1];
        while(max(i,j)+height[rank[i]]<=n&&a[i+height[rank[i]]]==a[j+height[rank[i]]])
            ++height[rank[i]];
    }
}
```

最小表示法

```
int solve(char *text, int length) {//0-base , 多解答案为起点最小
    int i = 0, j = 1, delta = 0;
    while (i < length && j < length && delta < length) {
        char tokeni = text[(i + delta) % length];
        char tokenj = text[(j + delta) % length];
        if (tokeni == tokenj) {
            delta++;
        } else {
            if (tokeni > tokenj) {
                i += delta + 1;
            } else {
                j += delta + 1;
            }
            if (i == j) {
                j++;
            }
            delta = 0;
        }
    }
    return std::min(i, j);
}
```

Chapter 6

计算几何

点类

```
int sgn(double x){return (x>eps)-(x<-eps);}
int sgn(double a,double b){return sgn(a-b);}
double sqr(double x){return x*x;}
struct P{
    double x,y;
    P(){}
    P(double x,double y):x(x),y(y){}
    double len2(){
        return sqr(x)+sqr(y);
    }
    double len(){
        return sqrt(len2());
    }
    void print(){
        printf("(%.3f,%.3f)\n",x,y);
    }
    P turn90(){return P(-y,x);}
    P norm(){return P(x/len(),y/len());}
};
bool operator==(P a,P b){
    return !sgn(a.x-b.x) and !sgn(a.y-b.y);
}
P operator+(P a,P b){
    return P(a.x+b.x,a.y+b.y);
}
P operator-(P a,P b){
    return P(a.x-b.x,a.y-b.y);
}
P operator*(P a,double b){
    return P(a.x*b,a.y*b);
}
```

```
P operator/(P a,double b){
    return P(a.x/b,a.y/b);
}
double operator^(P a,P b){
    return a.x*b.x + a.y*b.y;
}
double operator*(P a,P b){
    return a.x*b.y - a.y*b.x;
}
double det(P a,P b,P c){
    return (b-a)*(c-a);
}
double dis(P a,P b){
    return (b-a).len();
}
double Area(vector<P>poly){
    double ans=0;
    for(int i=1;i<poly.size();i++)
        ans+=(poly[i]-poly[0])*(poly[(i+1)%poly.size()]-poly[0]);
    return fabs(ans)/2;
}
struct L{
    P a,b;
    L(){}
    L(P a,P b):a(a),b(b){}
    P v(){return b-a;}
};
bool onLine(P p,L l){
    return sgn((l.a-p)*(l.b-p))==0;
}
bool onSeg(P p,L s){
    return onLine(p,s) and sgn((s.b-s.a)^(p-s.a))>=0 and sgn((s.a-s.b)^(p-s.b))>=0;
}
bool parallel(L l1,L l2){
    return sgn(l1.v()*l2.v())==0;
}
P intersect(L l1,L l2){
    double s1=det(l1.a,l1.b,l2.a);
    double s2=det(l1.a,l1.b,l2.b);
    return (l2.a*s2-l2.b*s1)/(s2-s1);
}
P project(P p,L l){
    return l.a+l.v()*((p-l.a)^l.v())/l.v().len2();
}
double dis(P p,L l){
    return fabs((p-l.a)*l.v())/l.v().len();
```

```
}
int dir(P p,L l){
    int t=sgn((p-l.b)*(l.b-l.a));
    if(t<0)return -1;
    if(t>0)return 1;
    return 0;
}
bool segIntersect(L l1,L l2){//strictly
    if(dir(l2.a,l1)*dir(l2.b,l1)<0&&dir(l1.a,l2)*dir(l1.b,l2)<0)
        return true;
    return false;
}
bool in_tri(P pt,P *p){
    if((p[1]-p[0])*(p[2]-p[0])<0)
        reverse(p,p+3);
    for(int i=0;i<3;i++){
        if(!onLeft(pt,L(p[i],p[(i+1)%3])))
            return false;
    }
    return true;
}
```

圆基础

```
struct C{
    P o;
    double r;
    C(){}
    C(P _o,double _r):o(_o),r(_r){}
};
// 求圆与直线的交点
//turn90() P(-y,x)
double fix(double x){return x>=0?x:0;}
bool intersect(C a, L l, P &p1, P &p2) {
    double x = ((l.a - a.o)^ (l.b - l.a)),
        y = (l.b - l.a).len2(),
        d = x * x - y * ((l.a - a.o).len2() - a.r * a.r);
    if (sgn(d) < 0) return false;
    d = max(d, 0.0);
    P p = l.a - ((l.b - l.a) * (x / y)), delta = (l.b - l.a) * (sqrt(d) / y);
    p1 = p + delta, p2 = p - delta;
    return true;
}
// 求圆与圆的交点，注意调用前要先判定重圆
bool intersect(C a, C b, P &p1, P &p2) {
```

```
    double s1 = (a.o - b.o).len();
    if (sgn(s1 - a.r - b.r) > 0 || sgn(s1 - fabs(a.r - b.r)) < 0) return false;
    double s2 = (a.r * a.r - b.r * b.r) / s1;
    double aa = (s1 + s2) * 0.5, bb = (s1 - s2) * 0.5;
    P o = (b.o - a.o) * (aa / (aa + bb)) + a.o;
    P delta = (b.o - a.o).norm().turn90() * sqrt(fix(a.r * a.r - aa * aa));
    p1 = o + delta, p2 = o - delta;
    return true;
}
// 求点到圆的切点，按关于点的顺时针方向返回两个点
bool tang(const C &c, const P &p0, P &p1, P &p2) {
    double x = (p0 - c.o).len2(), d = x - c.r * c.r;
    if (d < eps) return false; // 点在圆上认为没有切点
    P p = (p0 - c.o) * (c.r * c.r / x);
    P delta = ((p0 - c.o) * (-c.r * sqrt(d) / x)).turn90();
    p1 = c.o + p + delta;
    p2 = c.o + p - delta;
    return true;
}
// 求圆到圆的外共切线，按关于 c1.o 的顺时针方向返回两条线
vector<L> extan(const C &c1, const C &c2) {
    vector<L> ret;
    if (sgn(c1.r - c2.r) == 0) {
        P dir = c2.o - c1.o;
        dir = (dir * (c1.r / dir.len())).turn90();
        ret.push_back(L(c1.o + dir, c2.o + dir));
        ret.push_back(L(c1.o - dir, c2.o - dir));
    } else {
        P p = (c1.o * -c2.r + c2.o * c1.r) / (c1.r - c2.r);
        P p1, p2, q1, q2;
        if (tang(c1, p, p1, p2) && tang(c2, p, q1, q2)) {
            if (c1.r < c2.r) swap(p1, p2), swap(q1, q2);
            ret.push_back(L(p1, q1));
            ret.push_back(L(p2, q2));
        }
    }
    return ret;
}
// 求圆到圆的内共切线，按关于 c1.o 的顺时针方向返回两条线
vector<L> intan(const C &c1, const C &c2) {
    vector<L> ret;
    P p = (c1.o * c2.r + c2.o * c1.r) / (c1.r + c2.r);
    P p1, p2, q1, q2;
    if (tang(c1, p, p1, p2) && tang(c2, p, q1, q2)) { // 两圆相切认为没有切线
        ret.push_back(L(p1, q1));
        ret.push_back(L(p2, q2));
```

```
    }
    return ret;
}
```

点在多边形内

```
bool InPoly(P p,vector<P>poly){
    int cnt=0;
    for(int i=0;i<poly.size();i++){
        P a=poly[i],b=poly[(i+1)%poly.size()];
        if(OnLine(p,L(a,b)))
            return false;
        int x=sgn(det(a,p,b));
        int y=sgn(a.y-p.y);
        int z=sgn(b.y-p.y);
        cnt+=(x>0&&y<=0&&z>0);
        cnt-=(x<0&&z<=0&&y>0);
    }
    return cnt;
}
```

二维最小覆盖圆

```
struct line{
    point p,v;
};
point Rev(point v){return point(-v.y,v.x);}
point operator*(line A,line B){
    point u=B.p-A.p;
    double t=(B.v*u)/(B.v*A.v);
    return A.p+A.v*t;
}
point get(point a,point b){
    return (a+b)/2;
}
point get(point a,point b,point c){
    if(a==b)return get(a,c);
    if(a==c)return get(a,b);
    if(b==c)return get(a,b);
    line ABO=(line){(a+b)/2,Rev(a-b)};
    line BCO=(line){(c+b)/2,Rev(b-c)};
    return ABO*BCO;
}
int main(){
```

```
    scanf("%d",&n);
    for(int i=1;i<=n;i++)scanf("%lf%lf",&p[i].x,&p[i].y);
    random_shuffle(p+1,p+1+n);
    O=p[1];r=0;
    for(int i=2;i<=n;i++){
        if(dis(p[i],O)<r+1e-6)continue;
        O=get(p[1],p[i]);r=dis(O,p[i]);
        for(int j=1;j<i;j++){
            if(dis(p[j],O)<r+1e-6)continue;
            O=get(p[i],p[j]);r=dis(O,p[i]);
            for(int k=1;k<j;k++){
                if(dis(p[k],O)<r+1e-6)continue;
                O=get(p[i],p[j],p[k]);r=dis(O,p[i]);
            }
        }
    }printf("%.2lf %.2lf %.2lf\n",O.x,O.y,r);
    return 0;
}s
```

半平面交

```
struct P{
    int quad() const { return sgn(y) == 1 || (sgn(y) == 0 && sgn(x) >= 0);}
};
struct L{
    bool onLeft(const P &p) const { return sgn((b - a)*( p - a)) > 0; }
    L push() const{ // push out eps
        const double eps = 1e-10;
        P delta = (b - a).turn90().norm() * eps;
        return L(a - delta, b - delta);
    }
};
bool sameDir(const L &l0, const L &l1) {
    return parallel(l0, l1) && sgn((l0.b - l0.a)^(l1.b - l1.a)) == 1;
}
bool operator < (const P &a, const P &b) {
    if (a.quad() != b.quad())
        return a.quad() < b.quad();
    else
        return sgn((a*b)) > 0;
}
bool operator < (const L &l0, const L &l1) {
    if (sameDir(l0, l1))
        return l1.onLeft(l0.a);
    else
```

```
        return (l0.b - l0.a) < (l1.b - l1.a);
}
bool check(const L &u, const L &v, const L &w) {
    return w.onLeft(intersect(u, v));
}
vector<P> intersection(vector<L> &l) {
    sort(l.begin(), l.end());
    deque<L> q;
    for (int i = 0; i < (int)l.size(); ++i) {
        if (i && sameDir(l[i], l[i - 1])) {
            continue;
        }
        while (q.size() > 1
            && !check(q[q.size() - 2], q[q.size() - 1], l[i]))
                q.pop_back();
        while (q.size() > 1
            && !check(q[1], q[0], l[i]))
                q.pop_front();
        q.push_back(l[i]);
    }
    while (q.size() > 2
        && !check(q[q.size() - 2], q[q.size() - 1], q[0]))
            q.pop_back();
    while (q.size() > 2
        && !check(q[1], q[0], q[q.size() - 1]))
            q.pop_front();
    vector<P> ret;
    for (int i = 0; i < (int)q.size(); ++i)
    ret.push_back(intersect(q[i], q[(i + 1) % q.size()]));
    return ret;
}
```

求凸包

```
vector<P> convex(vector<P>p){
    sort(p.begin(),p.end());
    vector<P>ans,S;
    for(int i=0;i<p.size();i++){
        while(S.size()>=2
                && sgn(det(S[S.size()-2],S.back(),p[i]))<=0)
                    S.pop_back();
        S.push_back(p[i]);
    }//dw
    ans=S;
    S.clear();
```

```
    for(int i=(int)p.size()-1;i>=0;i--){
        while(S.size()>=2
                && sgn(det(S[S.size()-2],S.back(),p[i]))<=0)
                    S.pop_back();
        S.push_back(p[i]);
    }//up
    for(int i=1;i+1<S.size();i++)
        ans.push_back(S[i]);
    return ans;
}
```

凸包游戏

```
/*
   给定凸包，log n 内完成各种询问，具体操作有 :
   1. 判定一个点是否在凸包内
   2. 询问凸包外的点到凸包的两个切点
   3. 询问一个向量关于凸包的切点
   4. 询问一条直线和凸包的交点
   INF 为坐标范围，需要定义点类大于号
   改成实数只需修改 sign 函数，以及把 long long 改为 double 即可
   构造函数时传入凸包要求无重点，面积非空，以及 pair(x,y) 的最小点放在第一个
*/
const int INF = 1000000000;
struct Convex
{
    int n;
    vector<Point> a, upper, lower;
    Convex(vector<Point> _a) : a(_a) {
        n = a.size();
        int ptr = 0;
        for(int i = 1; i < n; ++ i) if (a[ptr] < a[i]) ptr = i;
        for(int i = 0; i <= ptr; ++ i) lower.push_back(a[i]);
        for(int i = ptr; i < n; ++ i) upper.push_back(a[i]);
        upper.push_back(a[0]);
    }
    int sign(long long x) { return x < 0 ? -1 : x > 0; }
    pair<long long, int> get_tangent(vector<Point> &convex, Point vec) {
        int l = 0, r = (int)convex.size() - 2;
        for( ; l + 1 < r; ) {
            int mid = (l + r) / 2;
            if (sign((convex[mid + 1] - convex[mid]).det(vec)) > 0) r = mid;
            else l = mid;
        }
        return max(make_pair(vec.det(convex[r]), r)
```

```
            , make_pair(vec.det(convex[0]), 0));
    }
    void update_tangent(const Point &p, int id, int &i0, int &i1) {
        if ((a[i0] - p).det(a[id] - p) > 0) i0 = id;
        if ((a[i1] - p).det(a[id] - p) < 0) i1 = id;
    }
    void binary_search(int l, int r, Point p, int &i0, int &i1) {
        if (l == r) return;
        update_tangent(p, l % n, i0, i1);
        int sl = sign((a[l % n] - p).det(a[(l + 1) % n] - p));
        for( ; l + 1 < r; ) {
            int mid = (l + r) / 2;
            int smid = sign((a[mid % n] - p).det(a[(mid + 1) % n] - p));
            if (smid == sl) l = mid;
            else r = mid;
        }
        update_tangent(p, r % n, i0, i1);
    }
    int binary_search(Point u, Point v, int l, int r) {
        int sl = sign((v - u).det(a[l % n] - u));
        for( ; l + 1 < r; ) {
            int mid = (l + r) / 2;
            int smid = sign((v - u).det(a[mid % n] - u));
            if (smid == sl) l = mid;
            else r = mid;
        }
        return l % n;
    }
    // 判定点是否在凸包内，在边界返回 true
    bool contain(Point p) {
        if (p.x < lower[0].x || p.x > lower.back().x) return false;
        int id = lower_bound(lower.begin(), lower.end()
            , Point(p.x, -INF)) - lower.begin();
        if (lower[id].x == p.x) {
            if (lower[id].y > p.y) return false;
        } else if ((lower[id - 1] - p).det(lower[id] - p) < 0) return false;
        id = lower_bound(upper.begin(), upper.end(), Point(p.x, INF)
            , greater<Point>()) - upper.begin();
        if (upper[id].x == p.x) {
            if (upper[id].y < p.y) return false;
        } else if ((upper[id - 1] - p).det(upper[id] - p) < 0) return false;
        return true;
    }
    // 求点 p 关于凸包的两个切点，如果在凸包外则有序返回编号
    // 共线的多个切点返回任意一个，否则返回 false
    bool get_tangent(Point p, int &i0, int &i1) {
```

```
        if (contain(p)) return false;
        i0 = i1 = 0;
        int id = lower_bound(lower.begin(), lower.end(), p) - lower.begin();
        binary_search(0, id, p, i0, i1);
        binary_search(id, (int)lower.size(), p, i0, i1);
        id = lower_bound(upper.begin(), upper.end(), p
            , greater<Point>()) - upper.begin();
        binary_search((int)lower.size() - 1, (int)lower.size() - 1 + id, p, i0, i1);
        binary_search((int)lower.size() - 1 + id
            , (int)lower.size() - 1 + (int)upper.size(), p, i0, i1);
        return true;
    }
    // 求凸包上和向量 vec 叉积最大的点，返回编号，共线的多个切点返回任意一个
    int get_tangent(Point vec) {
        pair<long long, int> ret = get_tangent(upper, vec);
        ret.second = (ret.second + (int)lower.size() - 1) % n;
        ret = max(ret, get_tangent(lower, vec));
        return ret.second;
    }
    // 求凸包和直线 u,v 的交点，如果无严格相交返回 false.
    //如果有则是和 (i,next(i)) 的交点，两个点无序，交在点上不确定返回前后两条线段其中之一
    bool get_intersection(Point u, Point v, int &i0, int &i1) {
        int p0 = get_tangent(u - v), p1 = get_tangent(v - u);
        if (sign((v - u).det(a[p0] - u)) * sign((v - u).det(a[p1] - u)) < 0) {
            if (p0 > p1) swap(p0, p1);
            i0 = binary_search(u, v, p0, p1);
            i1 = binary_search(u, v, p1, p0 + n);
            return true;
        } else {
            return false;
        }
    }
};
```

平面最近点

```
/*
   给定凸包，log n 内完成各种询问，具体操作有 :
   1. 判定一个点是否在凸包内
   2. 询问凸包外的点到凸包的两个切点
   3. 询问一个向量关于凸包的切点
   4. 询问一条直线和凸包的交点
   INF 为坐标范围，需要定义点类大于号
   改成实数只需修改 sign 函数，以及把 long long 改为 double 即可
   构造函数时传入凸包要求无重点，面积非空，以及 pair(x,y) 的最小点放在第一个
```

```
*/
const int INF = 1000000000;
struct Convex
{
    int n;
    vector<Point> a, upper, lower;
    Convex(vector<Point> _a) : a(_a) {
        n = a.size();
        int ptr = 0;
        for(int i = 1; i < n; ++ i) if (a[ptr] < a[i]) ptr = i;
        for(int i = 0; i <= ptr; ++ i) lower.push_back(a[i]);
        for(int i = ptr; i < n; ++ i) upper.push_back(a[i]);
        upper.push_back(a[0]);
    }
    int sign(long long x) { return x < 0 ? -1 : x > 0; }
    pair<long long, int> get_tangent(vector<Point> &convex, Point vec) {
        int l = 0, r = (int)convex.size() - 2;
        for( ; l + 1 < r; ) {
            int mid = (l + r) / 2;
            if (sign((convex[mid + 1] - convex[mid]).det(vec)) > 0) r = mid;
            else l = mid;
        }
        return max(make_pair(vec.det(convex[r]), r)
            , make_pair(vec.det(convex[0]), 0));
    }
    void update_tangent(const Point &p, int id, int &i0, int &i1) {
        if ((a[i0] - p).det(a[id] - p) > 0) i0 = id;
        if ((a[i1] - p).det(a[id] - p) < 0) i1 = id;
    }
    void binary_search(int l, int r, Point p, int &i0, int &i1) {
        if (l == r) return;
        update_tangent(p, l % n, i0, i1);
        int sl = sign((a[l % n] - p).det(a[(l + 1) % n] - p));
        for( ; l + 1 < r; ) {
            int mid = (l + r) / 2;
            int smid = sign((a[mid % n] - p).det(a[(mid + 1) % n] - p));
            if (smid == sl) l = mid;
            else r = mid;
        }
        update_tangent(p, r % n, i0, i1);
    }
    int binary_search(Point u, Point v, int l, int r) {
        int sl = sign((v - u).det(a[l % n] - u));
        for( ; l + 1 < r; ) {
            int mid = (l + r) / 2;
            int smid = sign((v - u).det(a[mid % n] - u));
```

```
            if (smid == sl) l = mid;
            else r = mid;
        }
        return l % n;
    }
    // 判定点是否在凸包内，在边界返回 true
    bool contain(Point p) {
        if (p.x < lower[0].x || p.x > lower.back().x) return false;
        int id = lower_bound(lower.begin(), lower.end()
            , Point(p.x, -INF)) - lower.begin();
        if (lower[id].x == p.x) {
            if (lower[id].y > p.y) return false;
        } else if ((lower[id - 1] - p).det(lower[id] - p) < 0) return false;
        id = lower_bound(upper.begin(), upper.end(), Point(p.x, INF)
            , greater<Point>()) - upper.begin();
        if (upper[id].x == p.x) {
            if (upper[id].y < p.y) return false;
        } else if ((upper[id - 1] - p).det(upper[id] - p) < 0) return false;
        return true;
    }
    // 求点 p 关于凸包的两个切点，如果在凸包外则有序返回编号
    // 共线的多个切点返回任意一个，否则返回 false
    bool get_tangent(Point p, int &i0, int &i1) {
        if (contain(p)) return false;
        i0 = i1 = 0;
        int id = lower_bound(lower.begin(), lower.end(), p) - lower.begin();
        binary_search(0, id, p, i0, i1);
        binary_search(id, (int)lower.size(), p, i0, i1);
        id = lower_bound(upper.begin(), upper.end(), p
            , greater<Point>()) - upper.begin();
        binary_search((int)lower.size() - 1, (int)lower.size() - 1 + id, p, i0, i1);
        binary_search((int)lower.size() - 1 + id
            , (int)lower.size() - 1 + (int)upper.size(), p, i0, i1);
        return true;
    }
    // 求凸包上和向量 vec 叉积最大的点，返回编号，共线的多个切点返回任意一个
    int get_tangent(Point vec) {
        pair<long long, int> ret = get_tangent(upper, vec);
        ret.second = (ret.second + (int)lower.size() - 1) % n;
        ret = max(ret, get_tangent(lower, vec));
        return ret.second;
    }
    // 求凸包和直线 u,v 的交点，如果无严格相交返回 false.
    //如果有则是和 (i,next(i)) 的交点，两个点无序，交在点上不确定返回前后两条线段其中之一
    bool get_intersection(Point u, Point v, int &i0, int &i1) {
        int p0 = get_tangent(u - v), p1 = get_tangent(v - u);
```

```
102         if (sign((v - u).det(a[p0] - u)) * sign((v - u).det(a[p1] - u)) < 0) {
103             if (p0 > p1) swap(p0, p1);
104             i0 = binary_search(u, v, p0, p1);
105             i1 = binary_search(u, v, p1, p0 + n);
106             return true;
107         } else {
108             return false;
109         }
110     }
111 };
```

Chapter 7

技巧

无敌的读入优化

```
// 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

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

梅森旋转算法

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

蔡勒公式

```cpp
int solve(int year, int month, int day) {
    int answer;
    if (month == 1 || month == 2) {
        month += 12;
        year--;
    }
    if ((year < 1752) || (year == 1752 && month < 9) ||
        (year == 1752 && month == 9 && day < 3)) {
        answer = (day + 2 * month + 3 * (month + 1) / 5 + year + year / 4 + 5) % 7;
    } else {
        answer = (day + 2 * month + 3 * (month + 1) / 5 + year + year / 4
               - year / 100 + year / 400) % 7;
    }
    return answer;
}
```

开栈

```cpp
register char *_sp __asm__("rsp");
int main() {
    const int size = 400 << 20;//400MB
    static char *sys, *mine(new char[size] + size - 4096);
    sys = _sp; _sp = mine; _main(); _sp = sys;
}
```

size 为 k 的子集

```cpp
void solve(int n, int k) {
    for (int comb = (1 << k) - 1; comb < (1 << n); ) {
        // ...
        int x = comb & -comb, y = comb + x;
        comb = (((comb & ~y) / x) >> 1) | y;
    }
}
```

32-bit/64-bit 随机素数

32-bit	64-bit
73550053	1249292846855685773
148898719	1701750434419805569
189560747	3605499878424114901
459874703	5648316673387803781
1202316001	6125342570814357977
1431183547	6215155308775851301
1438011109	6294606778040623451
1538762023	6347330550446020547
1557944263	7429632924303725207
1981315913	8524720079480389849

NTT 素数及其原根

Prime	Primitive root
1053818881	7
1051721729	6
1045430273	3
1012924417	5
1007681537	3