template

写在前面

基础模版

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
#define OPFI(x) freopen(#x".in", "r", stdin);\
                freopen(#x".out", "w", stdout)
#define REP(i, a, b) for(int i=(a); i<=(b); ++i)
#define REPd(i, a, b) for(int i=(a); i>=(b); --i)
inline ll rd(){
    ll r=0, k=1; char c;
    while(!isdigit(c=getchar())) if(c=='-') k=-k;
    while(isdigit(c)) r=r*10+c-'0', c=getchar();
    return r*k;
}
int main(){
    return 0;
}
```

vimrc

```vim
syntax on
set ts=4
set expandtab
set autoindent
set cindent
set shiftwidth=4
set nu
set softtabstop=4
set smartindent
set showmatch
set ruler
set mouse=a
inoremap <F1> <esc>:w<CR>
inoremap <F5> <esc>:below term<CR>
nmap <F1> :w<CR>
nmap <F5> :below term<CR>
colo habamax
set title
set shell=powershell
set wim=list
set backspace=indent,eol,start
set nocompatible
```

数据结构

zkw 线段树

单点修 区间查

```
ll s[N<<2], a[N];
int M;

ll f(ll x, ll y){
    return x+y; // 改这
}

void build(){
    for(M=1; M<=n+1; M<<=1);
    REP(i, 1, n) s[i+M]=a[i];
    REPd(i, M-1, 1) s[i]=f(s[2*i], s[2*i+1]);
}

ll qrange(int l, int r, ll init){ // 根据 f 传 init
    ll res=init;
    for(l=l+M-1, r=r+M+1; l^r^1; l>>=1, r>>=1){
        if(~l&1) res=f(res, s[l^1]);
        if(r&1) res=f(res, s[r^1]);
    }
    return res;
}

void edit(int x, ll v){
    for(s[x+=M]=v, x>>=1; x; x>>=1){
        s[x]=f(s[2*x], s[2*x+1]);
    }
}

ll qpoint(int x){
    return s[x+M];
}
```

珂朵莉树

```
struct node{
    int l, r;
    mutable int v;
    bool operator<(const node& rhs) const { return l<rhs.l; }
};

set<node> odt;
typedef set<node>::iterator iter;

iter split(ll p){
    iter tmp=odt.lower_bound((node){p, 0, 0});
    if(tmp!=odt.end()&&tmp->l==p) return tmp;
    --tmp;
    int tl=tmp->l, tr=tmp->r, tv=tmp->v;
    odt.erase(tmp);
    odt.insert((node){tl, p-1, tv});
    return odt.insert((node){p, tr, tv}).first;
}

// 【修改 & 查询】注意 split 顺序
// iter itr=split(r+1), itl=split(l);
```

数学

快速幂

```
const ll MOD=998244353; // 改模数

ll qpow(ll a, ll x){
    ll res=1;
    a%=MOD;
    while(x){
        if(x&1) res=res*a%MOD;
        a=a*a%MOD, x>>=1;
    }
    return res;
}

ll inv(ll x){ return qpow(x, MOD-2); } // 模数为质数时
```

高斯消元

```
const int N=110;
ll n;
```

```
double a[N][N], b[N];
void work(){
    n=rd();
    REP(i, 1, n){
        REP(j, 1, n) a[i][j]=rd();
        b[i]=rd();
    }
    REP(i, 1, n){
        int t=i;
        REP(j, i+1, n) if(abs(a[j][i])>1e-7&&(abs(a[t][i])>abs(a[j]
[i])||abs(a[t][i])<1e-7)) t=j;
        REP(j, i, n) swap(a[t][j], a[i][j]);
        if(abs(a[i][i])<1e-7){
            puts("No Solution");
            return 0;
        }
        swap(b[t], b[i]);
        double e=a[i][i];
        REP(j, i, n) a[i][j]/=e;
        b[i]/=e;
        REP(j, i+1, n){
            double d=a[j][i];
            REP(k, i, n) a[j][k]-=d*a[i][k];
            b[j]-=d*b[i];
        }
    }
    REPd(i, n, 1) REP(j, 1, i-1) b[j]-=a[j][i]*b[i], a[j][i]=0;
    // REP(i, 1, n) printf("%.2f\n", b[i]);
    // b[1...n] 保存 Ax=b 的解
}
```

图论

倍增

```
void dfs(int x, int fa){
    pa[x][0]=fa; dep[x]=dep[fa]+1;
    REP(i, 1, SP) pa[x][i]=pa[pa[x][i-1]][i-1];
    for(int& v:g[x]) if(v!=fa){
        dfs(v, x);
    }
}

int lca(int x, int y){
    if (dep[x]<dep[y]) swap(x, y);
    int t=dep[x]-dep[y];
    REP(i, 0, SP) if(t&(1<<i)) x=pa[x][i];
    REPd(i, SP-1, -1){
        int xx=pa[x][i], yy=pa[y][i];
        if (xx!=yy) x=xx, y=yy;
    }
    return x==y?x:pa[x][0];
}
```

网络流

不是我写的，但是看着还好

最大流

其中 `ll` 是我改的，不敢保证有没有漏改，但是过了洛谷模版题

```
constexpr ll INF=LLONG_MAX/2;

struct E {
    int to; ll cp;
    E(int to, ll cp): to(to), cp(cp) {}
};

struct Dinic {
    static const int M = 1E5 * 5;
    int m, s, t;
    vector<E> edges;
    vector<int> G[M];
    int d[M];
    int cur[M];

    void init(int n, int s, int t) {
        this->s = s; this->t = t;
        for (int i = 0; i <= n; i++) G[i].clear();
        edges.clear(); m = 0;
    }

    void addedge(int u, int v, ll cap) {
        edges.emplace_back(v, cap);
        edges.emplace_back(u, 0);
        G[u].push_back(m++);
        G[v].push_back(m++);
    }

    bool BFS() {
        memset(d, 0, sizeof d);
        queue<int> Q;
        Q.push(s); d[s] = 1;
        while (!Q.empty()) {
            int x = Q.front(); Q.pop();
            for (int& i: G[x]) {
                E &e = edges[i];
                if (!d[e.to] && e.cp > 0) {
                    d[e.to] = d[x] + 1;
                    Q.push(e.to);
                }
            }
        }
        return d[t];
    }

    ll DFS(int u, ll cp) {
        if (u == t || !cp) return cp;
        ll tmp = cp, f;
        for (int& i = cur[u]; i < G[u].size(); i++) {
            E& e = edges[G[u][i]];
            if (d[u] + 1 == d[e.to]) {
                f = DFS(e.to, min(cp, e.cp));
                e.cp -= f;
                edges[G[u][i] ^ 1].cp += f;
                cp -= f;
                if (!cp) break;
            }
        }
        return tmp - cp;
    }

    ll go() {
        ll flow = 0;
        while (BFS()) {
            memset(cur, 0, sizeof cur);
            flow += DFS(s, INF);
        }
        return flow;
    }
} DC;
```

费用流

```
struct E {
    int from, to, cp, v;
    E() {}
    E(int f, int t, int cp, int v) : from(f), to(t), cp(cp), v(v) {}
};
```

```cpp
struct MCMF {
    int n, m, s, t;
    vector<E> edges;
    vector<int> G[M];
    bool inq[M];
    int d[M], p[M], a[M];

    void init(int _n, int _s, int _t) {
        n = _n; s = _s; t = _t;
        FOR (i, 0, n + 1) G[i].clear();
        edges.clear(); m = 0;
    }

    void addedge(int from, int to, int cap, int cost) {
        edges.emplace_back(from, to, cap, cost);
        edges.emplace_back(to, from, 0, -cost);
        G[from].push_back(m++);
        G[to].push_back(m++);
    }

    bool BellmanFord(int &flow, int &cost) {
        FOR (i, 0, n + 1) d[i] = INF;
        memset(inq, 0, sizeof inq);
        d[s] = 0, a[s] = INF, inq[s] = true;
        queue<int> Q; Q.push(s);
        while (!Q.empty()) {
            int u = Q.front(); Q.pop();
            inq[u] = false;
            for (int& idx: G[u]) {
                E &e = edges[idx];
                if (e.cp && d[e.to] > d[u] + e.v) {
                    d[e.to] = d[u] + e.v;
                    p[e.to] = idx;
                    a[e.to] = min(a[u], e.cp);
                    if (!inq[e.to]) {
                        Q.push(e.to);
                        inq[e.to] = true;
                    }
                }
            }
        }
        if (d[t] == INF) return false;
        flow += a[t];
        cost += a[t] * d[t];
        int u = t;
        while (u != s) {
            edges[p[u]].cp -= a[t];
            edges[p[u] ^ 1].cp += a[t];
            u = edges[p[u]].from;
        }
        return true;
    }

    int go() {
        int flow = 0, cost = 0;
        while (BellmanFord(flow, cost));
        return cost;
    }
} MM;
```