Template

YuanyuanWu

September 18, 2018

目录

1 !

1.1 .vimrc

```
set nu ai ci si mouse=a ts=2 sts=2 sw=2
nmap<F2> : vs %<.in <CR>
nmap<F3> : !gedit % <CR>
nmap<F8> : !time ./%< < %<.in <CR>
nmap<F9> : :w <CR> :make %< <CR>
nmap<F5> : !./%< <CR>
nmap<F10> : :w <CR> :!g++ % -o %< -O2 -g -std=c++11 -Wall <CR>
```

1.2 Float

```
cout << setiosflags(ios::fixed);
cout << setprecision(3);
```

1.3 Head

```
#include<bits/stdc++.h>
using namespace std;
#define fi first
#define se second
#define mp make_pair
#define pb push_back
#define rep(i, a, b) for(int i=(a); i<(b); i++)
#define per(i, a, b) for(int i=(b)-1; i>=(a); i--)
#define sz(a) (int)a.size()
#define de(a) cout << #a << " = " << a << endl
#define dd(a) cout << #a << " = " << a << " "
#define all(a) a.begin(), a.end()
#define pw(x) (1ll<<(x))
#define endl "\n"
typedef long long ll;
typedef pair<int, int> pii;
typedef vector<int> vi;

int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(0);
    return 0;
}
```

2 DataStructure

2.1 1. Splay

```
/// init!!
struct Splay {
    #define ls son[u][0]
    #define rs son[u][1]
    static const int N=101010;
    int rt, L, w[N], fa[N], son[N][2], cnt[N], siz[N], rev[N];
    void init() {
        fill_n(w, L+1, 0);
        fill_n(fa, L+1, 0);
        fill_n(cnt, L+1, 0);
        fill_n(siz, L+1, 0);
        fill_n(rev, L+1, 0);
        fill_n(son[0], L+1, 0);
        fill_n(son[1], L+1, 0);
        L=rt=0;
    }
    void up(int u) {
        if(!u) return ;
        siz[u]=cnt[u];
        if(ls) siz[u]+=siz[ls];
        if(rs) siz[u]+=siz[rs];
    }
    void gao(int u) {
        if(!u) return ;
        rev[u]^=1;swap(ls, rs);
    }
    void down(int u) {
        if(!rev[u]) return ;
        rev[u]=0;gao(ls);gao(rs);
    }
    int id(int u) {
        return son[fa[u]][1]==u;
    }
    void rot(int x) {
        int y=fa[x], z=fa[y];
        int l=id(x), r=(l^1);
        fa[x]=z;
        if(z) son[z][id(y)]=x;
        son[y][l]=son[x][r];
        if(son[y][l]) fa[son[y][l]]=y;
        son[x][r]=y;
        fa[y]=x;
        up(y); up(x);
    }
    void splay(int x, int g=0) {
        while(fa[x]!=g) {
            int y=fa[x], z=fa[y];
            if(z!=g) (id(x)^id(y))?rot(x):rot(y);
            rot(x);
        }
        if(!g) rt=x;
    }
    void ins(int c) {
        if(!rt) {
            w[++L]=c;
            cnt[L]=siz[L]=1;
            rt=L;
            return ;
        }
        int u=rt, f=0;
```

```
    while(1) {
        if(c==w[u]) {
            ++cnt[u];
            up(u); up(f);
            splay(u);
            return ;
        }
        f=u;
        u=son[u][w[u]<c];
    }
    if(!u) {
        w[++L]=c;
        fa[L]=f;
        if(f) son[f][w[f]<c]=L;
        cnt[L]=siz[L]=1;
        up(f);
        splay(L);
        return ;
    }
}
// c in splay
// splay(u)
int rank(int c) {
    int u=rt, ans=0;
    while(1) {
        if(c<w[u]) {
            u=ls;
        } else if(c==w[u]) {
            if(ls) ans+=siz[ls];
            splay(u);
            return ans+1;
        } else {
            ans+=cnt[u];
            if(ls) ans+=siz[ls];
            u=rs;
        }
    }
}
// return w[u]
int mink(int k) {
    int u=rt;
    while(1) {
        if(siz[ls]>=k) {
            u=ls;
        } else {
            k-=siz[ls];
            if(cnt[u]>=k) {
                splay(u);
                return w[u];
            } else {
                k-=cnt[u];
                u=rs;
            }
        }
    }
}
```

```
}
// Next of rt
// 0 pre 1 next
// return u
int Next(int t) {
    int u=son[rt][t];
    while(son[u][t^1]) u=son[u][t^1];
    return u;
}
void del(int c) {
    rank(c);
    int u=rt;
    if(cnt[rt]>1) {
        --cnt[rt];
        up(rt);
        return ;
    }
    if(ls&&rs) {
        int pre=Next(0);
        int ne=Next(1);
        splay(pre);
        splay(ne, pre);
        son[ne][0]=0;
        up(ne);
        up(pre);
    } else if(ls) {
        rt=ls;
        fa[ls]=0;
    } else if(rs) {
        rt=rs;
        fa[rs]=0;
    } else {
        rt=0;
    }
}
};
```

2.2 2. Treap

```
// init!!
struct Treap {
    #define ls son[u][0]
    #define rs son[u][1]
    static const int N=101010;
    static const int inf=1e9+7;
    int rt, L, son[N][2], w[N], cnt[N], siz[N];
    ll r[N];
    void init() {
        fill_n(son[0], L+1, 0);
        fill_n(son[1], L+1, 0);
        fill_n(w, L+1, 0);
        fill_n(r, L+1, 0);
        fill_n(cnt, L+1, 0);
        fill_n(siz, L+1, 0);
        rt=L=0;
```

```cpp
    srand(time(0));
}
void up(int u) {
    if(!u) return ;
    siz[u]=cnt[u];
    if(ls) siz[u]+=siz[ls];
    if(rs) siz[u]+=siz[rs];
}
// 1 left 0 right
void rot(int &u, int t) {
    int v=son[u][t];
    son[u][t]=son[v][t^1];
    son[v][t^1]=u;
    up(u); up(v);
    u=v;
}
// return u w[u]=c
int ins(int &u, int c) {
    int po;
    if(!u) {
        u=++L;
        w[u]=c;
        r[u]=((1ll*rand()<<30)^(rand()));
        cnt[u]=siz[u]=1;
        po=u;
    } else if(w[u]==c) {
        ++cnt[u];
        po=u;
    } else {
        int &s=son[u][w[u]<c];
        po=ins(s, c);
        if(r[s]<r[u]) rot(u, w[u]<c);
    }
    up(u);
    return po;
}
void del(int &u, int c) {
    if(w[u]==c) {
        if(cnt[u]>1) {
            --cnt[u];
        } else {
            if(ls&&rs) {
                int t=r[ls]>r[rs];
                rot(u, t);
                del(son[u][t^1], c);
            } else {
                u=ls+rs;
            }
        }
    } else {
        del(son[u][w[u]<c], c);
    }
    up(u);
}
// c in treap
```

```cpp
int rank(int c) {
    int u=rt, ans=0;
    while(1) {
        if(c<w[u]) {
            u=ls;
        } else if(c==w[u]) {
            if(ls) ans+=siz[ls];
            return ans+1;
        } else {
            if(ls) ans+=siz[ls];
            ans+=cnt[u];
            u=rs;
        }
    }
}
// return w[u]
int mink(int k) {
    int u=rt;
    while(1) {
        if(siz[ls]>=k) {
            u=ls;
        } else {
            k-=siz[ls];
            if(cnt[u]>=k) {
                return w[u];
            } else {
                k-=cnt[u];
                u=rs;
            }
        }
    }
}
int Pre(int u, int c) {
    if(!u) return -inf;
    if(w[u]>=c) return Pre(ls, c);
    return max(w[u], Pre(rs, c));
}
int Next(int u, int c) {
    if(!u) return inf;
    if(w[u]<=c) return Next(rs, c);
    return min(w[u], Next(ls, c));
}
}T;
```

2.3   3. fhqTreap

```cpp
// init!!
// rt=merge()
struct fhqTreap {
    #define ls son[u][0]
    #define rs son[u][1]
    static const int N=101010;
    int rt, L;
    int w[N], son[N][2], siz[N];
    ll r[N];
```

```cpp
void init() {
    fill_n(w, L+1, 0);
    fill_n(r, L+1, 0);
    fill_n(siz, L+1, 0);
    fill_n(son[0], L+1, 0);
    fill_n(son[1], L+1, 0);
    rt=L=0;
    srand(time(0));
}
void up(int u) {
    if(!u) return ;
    siz[u]=1;
    if(ls) siz[u]+=siz[ls];
    if(rs) siz[u]+=siz[rs];
}
int newnode(int c) {
    w[++L]=c;
    siz[L]=1;
    r[L]=((1ll*rand()<<30)^rand());
    return L;
}
// c
void split(int u, int c, int &x, int &y) {
    if(!u) {
        x=y=0;
    } else {
        if(w[u]<=c) {
            x=u;
            split(rs, c, rs, y);
        } else {
            y=u;
            split(ls, c, x, ls);
        }
        up(u);
    }
}
// k
void split(int u, int k, int &x, int &y) {
    if(!u) {
        x = y = 0;
    } else {
        if(siz[ls] < k) {
            x = u;
            split(rs, k - siz[ls] - 1, rs, y);
        } else {
            y = u;
            split(ls, k, x, ls);
        }
        up(u);
    }
}
int merge(int x,int y) {
    if(x&&y) {
        if(r[x]<r[y]) {
            son[x][1]=merge(son[x][1], y);
            up(x);
            return x;
        } else {
            son[y][0]=merge(x, son[y][0]);
            up(y);
            return y;
        }
    } else {
        return x+y;
    }
}
void ins(int c) {
    int x, y;
    split(rt, c, x, y);
    rt=merge(x, merge(newnode(c), y));
}
void del(int c) {
    int x, y, z;
    split(rt, c-1, x, y);
    split(y, c, y, z);
    y=merge(son[y][0], son[y][1]);
    rt=merge(x, merge(y, z));
}
int rank(int c) {
    int x, y;
    split(rt, c-1, x, y);
    int res=siz[x]+1;
    rt=merge(x, y);
    return res;
}
int mink(int k) {
    int u=rt;
    while(1) {
        if(k<=siz[ls]) {
            u=ls;
        } else {
            k-=siz[ls];
            if(k==1) {
                return w[u];
            } else {
                --k;
                u=rs;
            }
        }
    }
}
int Pre(int c) {
    int x, y;
    split(rt, c-1, x, y);
    int u=x;
    while(rs) u=rs;
    rt=merge(x, y);
    return w[u];
}
int Next(int c) {
```

```
            int x, y;
            split(rt, c, x, y);
            int u=y;
            while(ls) u=ls;
            rt=merge(x, y);
            return w[u];
        }
};T;
```

2.4 4. PerTreap

```
// init!!
struct PerTreap {
#define ls son[u][0]
#define rs son[u][1]
    static const int N=500005;
    int L, tim;
    int rt[N], w[N*50], siz[N*50], son[N*50][2], r[N*50];
    void init() {
        fill_n(rt, tim+1, 0);
        fill_n(w, L+1, 0);
        fill_n(r, L+1, 0);
        fill_n(siz, L+1, 0);
        fill_n(son[0], L+1, 0);
        fill_n(son[1], L+1, 0);
        L=tim=0;
        srand(time(0));
    }
    void up(int u) {
        if(!u) return ;
        siz[u]=1;
        if(ls) siz[u]+=siz[ls];
        if(rs) siz[u]+=siz[rs];
    }
    int newnode(int c) {
        w[++L]=c;
        siz[L]=1;
        r[L]=rand();
        return L;
    }
    void copy(int &x, int u) {
        x=++L;
        w[x]=w[u];
        r[x]=r[u];
        siz[x]=siz[u];
        son[x][0]=son[u][0];
        son[x][1]=son[u][1];
    }
    void split(int u, int c, int &x, int &y) {
        if(!u) {
            x=y=0;
        } else {
            if(w[u]<=c) {
                copy(x, u);
                split(rs, c, son[x][1], y);
                up(x);
            } else {
                copy(y, u);
                split(ls, c, x, son[y][0]);
                up(y);
            }
        }
    }
    int merge(int x,int y) {
        if(x&&y) {
            int u;
            if(r[x]<r[y]) {
                copy(u, x);
                son[u][1]=merge(son[x][1], y);
            } else {
                copy(u, y);
                son[u][0]=merge(x, son[y][0]);
            }
            up(u);
            return u;
        } else {
            return x+y;
        }
    }
    void ins(int pre, int &now, int c) {
        int x, y;
        split(pre, c, x, y);
        now=merge(x, merge(newnode(c), y));
    }
    void del(int pre, int &now, int c) {
        int x, y, z;
        split(pre, c-1, x, y);
        split(y, c, y, z);
        if(!y) {
            now=pre;
            return ;
        }
        y=merge(son[y][0], son[y][1]);
        now=merge(x, merge(y, z));
    }
    int rank(int now, int c) {
        int x, y;
        split(now, c-1, x, y);
        int res=siz[x]+1;
        now=merge(x, y);
        return res;
    }
    int mink(int now, int k) {
        int u=now;
        while(1) {
            if(k<=siz[ls]) {
                u=ls;
            } else {
                k-=siz[ls];
                if(k==1) {
```

```
        return w[u];
    } else {
        --k;
        u=rs;
    }
        }
    }
}
int Pre(int now, int c) {
    int x, y;
    split(now, c-1, x, y);
    if(!x) return -2147483647;
    int u=x;
    while(rs) u=rs;
    now=merge(x, y);
    return w[u];
}
int Next(int now, int c) {
    int x, y;
    split(now, c, x, y);
    if(!y) return 2147483647;
    int u=y;
    while(ls) u=ls;
    now=merge(x, y);
    return w[u];
}
}T;
```

2.5 5. SegIntervalMin

```
// O(nlogn)
struct Seg {
#define ls rt << 1
#define rs ls | 1
    static const int N = ::N << 2;
    ll sum[N];
    int mi[N][2], cnt[N];
    void up(int rt) {
        sum[rt] = sum[ls] + sum[rs];
        rep(i, 0, 2) mi[rt][i] = min(mi[ls][i], mi[rs][i]);
        cnt[rt] = 0;
        rep(i, 0, 2) {
            if(mi[rt][0] == mi[ls | i][0]) cnt[rt] += cnt[ls | i];
            else mi[rt][1] = min(mi[rt][1], mi[ls | i][0]);
        }
    }
    void build(int l, int r, int rt) {
        if(l == r) {
            sum[rt] = mi[rt][0] = 1; //modify
            mi[rt][1] = inf;
            cnt[rt] = 1;
            return ;
        }
        int mid = l + r >> 1;
        build(l, mid, ls);
```

```
        build(mid + 1, r, rs);
        up(rt);
    }
    void gao(int rt, int c) {
        if(c <= mi[rt][0]) return ;
        sum[rt] += 1ll * cnt[rt] * (c - mi[rt][0]);
        mi[rt][0] = c;
    }
    void down(int rt) {
        gao(ls, mi[rt][0]);
        gao(rs, mi[rt][0]);
    }
    void upd(int L, int R, int c, int l, int r, int rt) {
        if(L > R) return ;
        if(L <= l && r <= R && c < mi[rt][1]) {
            gao(rt, c);
            return ;
        }
        int mid = l + r >> 1;
        down(rt);
        if(L <= mid) upd(L, R, c, l, mid, ls);
        if(R > mid) upd(L, R, c, mid + 1, r, rs);
        up(rt);
    }
}seg;
```

2.6 6. 2DSegTree

```
// 区域覆盖, 标记永久化, 标记单调
const int N=1010;
int n,m,q;
struct seg {
    int ma[N<<2], la[N<<2];
    void upd(int L,int R,int c,int l=0,int r=m,int rt=1) {
        ma[rt]=max(ma[rt], c);
        if(L<=l&&r<=R) {
            la[rt]=max(la[rt], c);
            return ;
        }
        int mid=l+r>>1;
        if(L<=mid) upd(L, R, c, l, mid, rt<<1);
        if(R>=mid+1) upd(L, R, c, mid+1, r, rt<<1|1);
    }
    int qry(int L,int R,int l=0,int r=m,int rt=1) {
        int ans=0;
        ans=max(ans, la[rt]);
        if(L<=l&&r<=R) {
            ans=max(ans, ma[rt]);
            return ans;
        }
        int mid=l+r>>1;
        if(L<=mid) ans=max(ans, qry(L, R, l, mid, rt<<1));
        if(R>=mid+1) ans=max(ans, qry(L, R, mid+1, r, rt<<1|1));
        return ans;
    }
}
```

```cpp
};
struct Seg {
    seg ma[N<<2], la[N<<2];
    void upd(int x1,int x2,int y1,int y2,int c,int l=0,int r=n,int rt=1) {
        ma[rt].upd(y1, y2, c);
        if(x1<=l&&r<=x2){
            la[rt].upd(y1, y2, c);
            return ;
        }
        int mid=l+r>>1;
        if(x1<=mid) upd(x1, x2, y1, y2, c, l, mid, rt<<1);
        if(x2>=mid+1) upd(x1, x2, y1, y2, c, mid+1, r, rt<<1|1);
    }
    int qry(int x1,int x2,int y1,int y2,int l=0,int r=n,int rt=1) {
        int ans=0;
        ans=max(ans, la[rt].qry(y1, y2));
        if(x1<=l&&r<=x2){
            ans=max(ans, ma[rt].qry(y1, y2));
            return ans;
        }
        int mid=l+r>>1;
        if(x1<=mid) ans=max(ans, qry(x1, x2, y1, y2, l, mid, rt<<1));
        if(x2>=mid+1) ans=max(ans, qry(x1, x2, y1, y2, mid+1, r, rt<<1|1));
        return ans;
    }
}T;
int main() {
    scanf("%d%d%d",&n,&m,&q);
    while(q--) {
        int d, s, h, x, y;scanf("%d%d%d%d%d",&d, &s,&h,&x,&y);
        int t=T.qry(x, x+d-1, y, y+s-1);
        T.upd(x, x+d-1, y, y+s-1, h+t);
    }
    printf("%d\n",T.qry(0, n, 0, m));
    return 0;
}
```

2.7   7. Fenwick

```cpp
// [1,n] , init!!
template<class T>
struct Fenwick{
    #define lb(x) ((x)&-(x))
    static const int N = 1000001;
    int n;T a[N];
    void ini(int _n){ fill_n(a+1,n=_n,0);}
    void Pre(){ for(int i=1,j=i+lb(i);i<=n;++i,j=i+lb(i)) if(j<=n) a[j]+=a[i];}
    void add(int x,T d){ for(;x<=n;x+=lb(x)) a[x]+=d;}
    T sum(int x){ T r=0;for(;x>=1;x^=lb(x)) r+=a[x];return r;}
};
```

2.8   8. ST

```cpp
const int N=101010;
int n,m,Max[N][22];
int main() {
    while(~scanf("%d%d",&n,&m)) {
        rep(i,1,n+1) scanf("%d",&Max[i][0]);
        for(int i=1;(1<<i)<=n;++i) {
            for(int j=1;j+(1<<i)-1<=n;++j)
                Max[j][i]=max(Max[j][i-1], Max[j+(1<<(i-1))][i-1]);
        }
        while(m--) {
            int l, r;scanf("%d%d",&l,&r);
            int _ = log2(r-l+1);
            printf("%d\n",max(Max[l][_], Max[r-(1<<_)+1][_]));
        }
    }
    return 0;
}
```

2.9   Rope

```cpp
#include <ext/rope>
using namespace __gnu_cxx;

//index : [0..sz(rp))
rope<char> rp;
rp.push_back(ch);            // 在末尾添加 ch
rp.erase(cur, len);          // 删除 cur 开始的 len 个字符
rp.insert(cur, 字符数组);     // 在 cur 处插入字符数组
rp.copy(cur, len, 字符数组);  // 复制 cur 处开始的 len 个字符到字符数组
rp.replace(cur, 字符数组);    // 删除 cur 处的字符, 换成字符数组
rp.substr(cur, len);         // 提取从 cur 处开始的 len 个字符
rp.at(cur);                  // 取第 cur 个字符
rp[cur];                     // 同上
rp[i] = rp[i - 1];           // 可持久化, O(1), 直接拷贝根节点

/*
* 一）翻转操作
*  1. 维护一正一反两个 rope
*  2. 翻转等价于交换两个子串
*
* 二）区间循环位移
*  1. 拆成多个子串连在一起
*
* 三）区间 a>b, b>c, c>d ... z>a
*  1. 维护 26 个 rope
*/
```

2.10   动态 k 大

```cpp
// zoj 2112 动态区间 k 大
const int N = 50505, M = 10101;
int n, m, a[N], rt[N<<1];
```

```cpp
vi V, add, sub;
inline int rk(int x) {
    return lower_bound(all(V), x) - V.begin();
}
struct Q {
    bool op;
    int a, b, k;
}q[M];
struct Seg {
    static const int N = 2500005;//((::N + 32 * ::M) * 16;
    int cntn, cnt[N], ls[N], rs[N];
    void init() {
        fill_n(rt+1, n, cntn = 0);
    }
    void upd(int pre, int &now, int p, int c, int l, int r) {
        now = ++cntn;
        cnt[now] = cnt[pre] + c;
        ls[now] = ls[pre];
        rs[now] = rs[pre];
        if(l == r) return ;
        int mid = l+r>>1;
        if(p<=mid) upd(ls[pre], ls[now], p, c, l, mid);
        else upd(rs[pre], rs[now], p, c, mid+1, r);
    }
    int qry(int L, int R, int k, int l, int r) {
        if(l == r) return l;
        int mid = l+r>>1;
        int lc = 0;
        if(lc>=k) {
            for(auto i : add) lc += cnt[ls[i]];
            for(auto i : sub) lc -= cnt[ls[i]];
            return qry(L, R, k, l, mid);
        } else {
            rep(i, 0, sz(add)) add[i] = ls[add[i]];
            rep(i, 0, sz(sub)) sub[i] = ls[sub[i]];
            rep(i, 0, sz(add)) add[i] = rs[add[i]];
            rep(i, 0, sz(sub)) sub[i] = rs[sub[i]];
            return qry(L, R, k-lc, mid+1, r);
        }
    }
}seg;
struct Fenwick {
#define lb(x) ((x)&(-x))
    void init() {
        fill_n(rt+1+n, n, 0);
    }
    void upd(int x, int p, int c) {
        for(; x<=n; x+=lb(x)) seg.upd(rt[x+n], rt[x+n], p, c, 0, sz(V)-1);
    }
    int qry(int l, int r, int k) {
        add.clear();sub.clear();
        add.pb(rt[r]);sub.pb(rt[l-1]);
        int x = r;
        for(; x; x^=lb(x)) add.pb(rt[n+x]);
        x = l-1;
        for(; x; x^=lb(x)) sub.pb(rt[n+x]);
        return seg.qry(1, r, k, 0, sz(V)-1);
    }
}fw;
int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(0);
    int T;
    cin >> T;
    while(T--) {
        ///
        cin >> n >> m;
        //init
        V.clear();
        seg.init();
        fw.init();
        ///read
        rep(i, 1, n+1) cin >> a[i], V.pb(a[i]);
        rep(i, 1, m+1) {
            string s;
            cin >> s >> q[i].a >> q[i].b;
            q[i].op = (s[0]=='Q');
            if(s[0]=='Q') {
                cin >> q[i].k;
            } else {
                V.pb(q[i].b);
            }
        }
        ///solve
        sort(all(V));
        V.erase(unique(all(V)), V.end());
        rep(i, 1, n+1) seg.upd(rt[i-1], rt[i], rk(a[i]), 1, 0, sz(V)-1);
        rep(i, 1, m+1) {
            if(q[i].op) {
                cout << V[fw.qry(q[i].a, q[i].b, q[i].k)] << endl;
            } else {
                int p = q[i].a, c = q[i].b;
                fw.upd(p, rk(a[p]), -1);
                fw.upd(p, rk(a[p] = c), 1);
            }
        }
    }
    return 0;
}
```

3 Game

3.1 Game

```cpp
// 威佐夫博弈
// * 两堆物品，个数 (n, m)(n <= m)，两人轮流从某一堆拿任意数量的物品或同时从两堆中取同样
多的物品，每次至少一个，不能操作的人败。
// * 必败态：(m - n) * (1 + sqrt5) / 2 == n
// 威佐夫博弈扩展
```

```cpp
// * 两堆物品，个数 (n, m)(n <= m) ，两人轮流从某一堆拿任意数量的物品或同时从两堆中取绝对
// 值 <=k 的物品，每次至少一个，不能操作的人败。
// * 必败态:
// *      d = k + 1, t^2 + (d − 2) * t − d = 0 −> 解出 t
// *      必败: (m − n) / d * t == n
// 博弈fib
// * 一堆石子，两人轮流取。先手不能任第一次取光，之后可以取的石子数介于 1 到对手刚刚取的石子数
// 的两倍之间（左闭右闭），不能操作的人败。
// * 必败态: 石子个数是 fib 数
```

4 Geo

4.1 2D

```cpp
/*
 * 欧拉定理: 平面图满足  V+F−E=2
 * 直线的一般式:  Ax+By+C=0
 * 点到直线的距离:  |Ax0+By0+C|/sqrt(A*A+B*B)
 */
#include<bits/stdc++.h>
using namespace std;
#define fi first
#define se second
#define pb push_back
#define mp make_pair
#define sz(a) (int)a.size()
#define de(x) cout << #x << " = " << x << endl;
#define rep(i,a,b) for(int i=(a);i<(b);++i)
#define x(a) a.x
#define y(a) a.y
typedef double db;
const db eps = 1e−8;
const db pi = acos(−1);

// 负数 −1 零 0 正数 1
int sign(db x) {
    return (x > eps) − (x < −eps);
}

struct P {
    db x,y;
    P() {}
    P(db x, db y) {
        this->x = x;
        this->y = y;
    }
    P operator + (const P &c) const {
        return P(x + c.x, y + c.y);
    }
    P operator − (const P &c) const {
        return P(x − c.x, y − c.y);
    }
    P operator * (const db &c) const {
        return P(x * c, y * c);
    }
    P operator / (const db &c) const {
        return P(x / c, y / c);
    }
    bool operator < (const P &c) const {
        int f = sign(x − c.x);
        return f ? f < 0 : sign(y − c.y) < 0;
    }
    bool operator == (const P &c) const {
        return !sign(x − c.x) && !sign(y − c.y);
    }
    bool operator != (const P &c) const {
        return !(*this == c);
    }
    bool operator > (const P &c) const {
        return !(*this == c) && !(*this < c);
    }
};

P read() {
    db x,y;scanf("%lf%lf", &x, &y);
    return P(x, y);
}
void print(P p) {
    printf("%f %f\n",x(p),y(p));
}
db abs(P a) {
    return sqrt(x(a) * x(a) + y(a) * y(a));
}
db norm(P a) {
    return x(a) * x(a) + y(a) * y(a);
}
db dot(P a, P b) {
    return x(a) * x(b) + y(a) * y(b);
}
db cross(P a, P b) {
    return x(a) * y(b) − x(b) * y(a);
}
// 两点距离的平方
db disq(P a, P b) {
    return norm(a − b);
}
// 两点距离
db dis(P a, P b) {
    return sqrt(norm(a − b));
}
// 向量 ab 与 x 轴的夹角，弧度，取值范围 (−pi, pi]
db ang(P a, P b) {
    return atan2(y(b)−y(a),x(b)−x(a));
}
// 向量 oa 与 ob 的夹角，弧度，取值范围 [0, pi]
db ang(P a, P o, P b) {
    return acos(dot(a − o, b − o) / abs(a − o) / abs(b − o));
}
// 向量逆时针旋转 rad（弧度）
```

```cpp
P rot(P a, db rad) {
    return P(x(a) * cos(rad) - y(a) * sin(rad), x(a) * sin(rad) + y(a) * cos(rad));
}
P rot(P a, P o, db rad) {
    return rot(a - o, rad) + o;
}
// 逆时针旋转 90 度
P rot90(P p) {
    return P(-y(p), x(p));
}
// 向量 p 在向量 v 方向上的投影 (点)
P proj(P p, P v) {
    return v * dot(p, v) / norm(v);
}
// 向量 ap 在向量 ab 方向上的投影 (点)
P proj(P p, P a, P b) {
    return proj(p - a, b - a) + a;
}
// p 点关于 ab 的对称点
P reflect(P p, P a, P b) {
    P o = proj(p, a, b);
    return o * 2 - p;
}
// 直线 pv 和 qw 的交点
P insLL(P p, P v, P q, P w) {
    P u = p - q;
    v = v - p;
    w = w - q;
    db t = cross(w, u) / cross(v, w);
    return p + v * t;
}
// 判断点是否在线段上 (不包括端点)
bool onS0(P p, P a, P b) {
    return sign(cross(p - a, b - a)) == 0 && sign(dot(p - a, p - b)) < 0;
}
// 判断点是否在线段上 (包括端点)
bool onS1(P p, P a, P b) {
    return sign(cross(p - a, b - a)) == 0 && sign(dot(p - a, p - b)) <= 0;
}
// 判断两直线是否相交
bool isLL(P a1, P a2, P b1, P b2) {
    return sign(cross(a2 - a1, b2 - b1)) != 0;
}
// 判断线段是否规范相交 (交点不在任一个端点上)
bool isSS0(P a1, P a2, P b1, P b2) {
    db c1 = cross(a2 - a1, b1 - a1),  c2 = cross(a2 - a1, b2 - a1),
       c3 = cross(b2 - b1, a1 - b1),  c4 = cross(b2 - b1, a2 - b1);
    return sign(c1) * sign(c2) < 0 && sign(c3) * sign(c4) < 0;
}
// 判断线段是否不规范相交
bool isSS1(P a1, P a2, P b1, P b2) {
    db c1 = cross(a2 - a1, b1 - a1),  c2 = cross(a2 - a1, b2 - a1),
       c3 = cross(b2 - b1, a1 - b1),  c4 = cross(b2 - b1, a2 - b1);
    return sign(max(x(a1), x(a2)) - min(x(b1), x(b2))) >= 0 &&
           sign(max(x(b1), x(b2)) - min(x(a1), x(a2))) >= 0 &&
           sign(max(y(a1), y(a2)) - min(y(b1), y(b2))) >= 0 &&
           sign(max(y(b1), y(b2)) - min(y(a1), y(a2))) >= 0 &&
           sign(c1) * sign(c2) <= 0 && sign(c3) * sign(c4) <= 0;
}
// 判断直线与线段是否相交 (端点也算)
bool isLS(P a1, P a2, P b1, P b2) {
    db c1 = cross(a2 - a1, b1 - a1),  c2 = cross(a2 - a1, b2 - a1);
    return sign(c1) * sign(c2) <= 0;
}
// 点到直线距离
db distoL(P p, P a, P b) {
    return fabs(cross(b - a, p - a)) / abs(b - a);
}
// 点到线段距离
db distoS(P p, P a, P b) {
    if(sign(dot(b - a, p - a)) < 0) return abs(p - a);
    if(sign(dot(a - b, p - b)) < 0) return abs(p - b);
    return distoL(p, a, b);
}
// 直线两点式转一般式
// 直线的一般式: Ax+By+C=0
void getLABC(P a, P b, db &A, db &B, db &C) {
    A = y(a) - y(b);
    B = x(b) - x(a);
    C = x(a) * y(b) - y(a) * x(b);
}
// 多边形面积
db areaP(P *p, int n) {
    db ans = 0; p[n] = p[0];
    rep(i, 0, n) ans += cross(p[i], p[i+1]);
    return fabs(ans) / 2;
}
// 判断点和多边形关系在上 -1 外 0 内 1
int Pinploy(P o, P *p, int n) {
    int res = 0;
    rep(i, 0, n) {
        P u = p[i], v = p[(i + 1) % n];
        if(onS1(o, u, v)) return -1;
        int k = sign(cross(v - u, o - u));
        int d1 = sign(y(u) - y(o));
        int d2 = sign(y(v) - y(o));
        if(k > 0 && d1 <= 0 && d2 > 0) ++res;
        if(k < 0 && d2 <= 0 && d1 > 0) --res;
    }
    return res != 0;
}
// 求凸包: 把给定点包围在内部的, 面积最小的凸多边形
// 复杂度: O(n) 加上排序: O(nlogn)
// 输入的点要先去重
// 如果不希望在凸包的边上有输入点, 把两个 <= 改成 <
int convexhull(P *p, int n, P *ch) {
    sort(p, p + n);
    int m = 0;
    rep(i, 0, n) {
        while(m > 1 && sign(cross(ch[m - 1] - ch[m - 2], p[i] - ch[m - 2])) <= 0) --m;
```

```cpp
        ch[m++] = p[i];
    }
    int k = m;
    for(int i = n − 2; i >= 0; −−i) {
        while(m > k && sign(cross(ch[m − 1] − ch[m − 2], p[i] − ch[m − 2])) <= 0) −−m;
        ch[m++] = p[i];
    }
    if(n > 1) −−m;
    return m;
}

struct C {
    P o;
    db r;
    C() {}
    C(P o, db r) : o(o), r(r) {}
    // 通过圆心角（弧度）求圆上坐标
    P point(db rad) {
        return P(o.x + cos(rad) * r, o.y + sin(rad) * r);
    }
};
// 判断、求线圆交点
bool isLC(C c, P a, P b, P &p1, P &p2) {
    db x = dot(a − c.o, b − a), y = norm(b − a),
    d = x * x − y * (norm(a − c.o) − c.r * c.r);
    if(sign(d) < 0) return 0; if(d < 0) d = 0;
    P q1 = a − (b − a) * (x / y),
    q2 = (b − a) * (sqrt(d) / y);
    p1 = q1− q2;
    p2 = q1 + q2;
    return 1;
}
// 判断两圆关系
// 相等 0 相离 1 外切 2 相交 3 内切 4 内含 5
int relCC(C c1, C c2) {
    P p1 = c1.o, p2 = c2.o;
    db r1 = c1.r, r2 = c2.r;
    db d = dis(p1, p2);
    if(sign(d) == 0 && sign(r1 − r2) == 0) return 0;
    int x = sign(d − r1 − r2), y = sign(d − fabs(r1 − r2));
    if(x == 0) return 2;
    if(y == 0) return 4;
    if(x > 0) return 1;
    if(y < 0) return 5;
    if(y > 0 && x < 0) return 3;
    return −1;
}
// 返回值表示是否有交点
// 求两圆交点
bool isCC(C c1, C c2, P &p1, P &p2) {
    db x = norm(c1.o − c2.o),
    y = ((c1.r * c1.r − c2.r * c2.r) / x + 1) / 2,
    d = c1.r * c1.r / x − y * y;
    if(sign(d) < 0) return 0; if(d < 0) d = 0;
    P q1 = (c2.o − c1.o) * y + c1.o,
        q2 = rot90((c2.o − c1.o) * sqrt(d));
    p1 = q1 − q2;
    p2 = q1 + q2;
    return 1;
}
// 求点圆切点
vector<P> tanCP(P p, C c, P &p1, P &p2) {
    db x = norm(p − c.o), d = x − c.r * c.r;
    vector<P> ans;
    if(sign(d) < 0) return ans; if(d < 0) d = 0;
    P q1 = (p − c.o) * (c.r * c.r / x),
    q2 = rot90((p − c.o) * (−c.r * c.r * sqrt(d) / x));
    p1 = c.o + q1 − q2;
    p2 = c.o + q1 + q2;
    ans.pb(p1);ans.pb(p2);
    return ans;
}
// 求圆圆切线
vector<pair<P, P> > tanCC(C c1, C c2) {
    vector<pair<P, P>> ans;
    if(!sign(c1.r − c2.r)) {
        P dir = c2.o − c1.o;
        dir = rot90(dir * (c1.r / abs(dir)));
        ans.pb(mp(c1.o + dir, c2.o + dir));
        ans.pb(mp(c1.o − dir, c2.o − dir));
    } else {
        P p = (c1.o * (−c2.r) + c2.o * c1.r) / (c1.r − c2.r);
        P t1,t2;
        vector<P> ps = tanCP(p, c1, t1, t2);
        vector<P> qs = tanCP(p, c2, t1, t2);
        for (int i = 0; i < sz(ps) && i < sz(qs); ++i) {
            if(!i || !(ps[i] == ps[i−1] && qs[i] == qs[i−1]))
                ans.pb(mp(ps[i],qs[i]));
        }
    }
    P p = (c1.o * c2.r + c2.o * c1.r) / (c1.r + c2.r);
    P t1,t2;
    vector<P> ps = tanCP(p, c1, t1, t2);
    vector<P> qs = tanCP(p, c2, t1, t2);
    for (int i = 0; i < sz(ps) && i < sz(qs); ++i) {
        if(!i || !(ps[i] == ps[i−1] && qs[i] == qs[i−1]))
            ans.pb(mp(ps[i],qs[i]));
    }
    return ans;
}
// 圆面积交
db areaCC(C c1, C c2) {
    db d = abs(c1.o − c2.o);
    if(sign(c1.r + c2.r − d) <= 0) return 0;
    if(sign(d − fabs(c1.r − c2.r)) <= 0) {
        db r = min(c1.r, c2.r);
        return r*r*pi;
    }
    db x = (d * d + c1.r * c1.r − c2.r * c2.r) / (2 * d);
    db t1 = acos(x / c1.r);
```

```cpp
    db t2 = acos((d - x) / c2.r);
    return c1.r * c1.r * t1 + c2.r * c2.r * t2 - d * c1.r * sin(t1);
}
// 圆三角形面积交
// 圆，半径: r 圆心: 原点
// 三角形: 圆心，p1，p2
db areaCT(db r, P p1, P p2) {
    P q1, q2, o = P(0, 0);
    C c = C(o, r);
    int f = isLC(c, p1, p2, q1, q2);
    if(!f) return r * r * ang(p1, o, p2) / 2;
    bool b1 = sign(abs(p1) - r) > 0;
    bool b2 = sign(abs(p2) - r) > 0;
    if(b1 && b2) {
        if(sign(dot(p1 - q1, p2 - q1)) <= 0 && sign(dot(p1 - q2, p2 - q2)) <= 0) {
            return (r * r * (ang(p1, o, p2) - ang(q1, o, q2))) + fabs(cross(q1, q2)) / 2;
        } else {
            return r * r * ang(p1, o, p2) / 2;
        }
    } else if(b1) {
        return (r * r * ang(p1, o, q1) ) + fabs(cross(q1, p2)) / 2;
    } else if(b2) {
        return (r * r * ang(q2, o, p2) ) + fabs(cross(p1, q2)) / 2;
    } else {
        return fabs(cross(p1, p2)) / 2;
    }
}
// 三角形内心
P inC(P A, P B, P C) {
    db a = abs(B - C);
    db b = abs(A - C);
    db c = abs(A - B);
    return (A * a + B * b + C * c) / (a + b + c);
}
// 三角形外心
P outC(P A, P B, P C) {
    P b = B - A, c = C - A;
    db db = norm(b), dc = norm(c), d = 2 * cross(b, c);
    return A - P(y(b) * dc - y(c) * db, x(c) * db - x(b) * dc) / d;
}
// 三角形垂心
P othroC(P A, P B, P C) {
    P ba = B - A, ca = C - A, bc = B - C;
    db Y = y(ba) * y(ca) * y(bc);
    db a = cross(ca, ba);
    db xx = (Y + x(ca) * y(ba) * x(B) - x(ba) * y(ca) * x(C)) / a;
    db yy = -x(ba) * (xx - x(C)) / y(ba) + y(ca);
    return P(xx, yy);
}
// 最小圆覆盖  O(n)
void Mincir(P *p, int n){
    random_shuffle(p, p+n);
    P cir = p[0]; db r= 0;
    for(int i = 1; i < n; i++){
        if(sign(dis(cir, p[i]) - r) <= 0) continue;
        cir = p[i], r = 0;
        for(int j = 0; j < i; j++){
            if(sign(dis(cir, p[j]) - r) <= 0) continue;
            cir = P ((x(p[i]) + x(p[j])) / 2, (y(p[i]) + y(p[j])) / 2);
            r = dis(cir, p[j]);
            for(int k = 0; k < j; k++){
                if(sign(dis(cir, p[k]) - r) <= 0) continue;
                cir = outC(p[i], p[j], p[k]);
                r = dis(cir, p[k]);
            }
        }
    }
    printf("%.2f %.2f\n", x(cir), y(cir), r);
}
// 半平面交未测试
const int N=450005;
struct Seg{
    P s, e;
    double r;
    void getr(){r = atan2(y(e)-y(s), x(e)-x(s));}
    bool operator < (const Seg& c)const {
        int d = sign(r - c.r);
        if (!d) return sign(cross(c.s - s, c.e -s)) > 0;
        return d < 0;
    }
}seg[N], Q[N];
int sz;
P insLL(Seg a, Seg b){return insLL(a.s,a.e,b.s,b.e);}
void add_seg(db xa, db ya, db xb, db yb){
    seg[sz].s=P(xa,ya);seg[sz].e=P(xb,yb);
    seg[sz].getr();sz++;
}
int hpi(P *p){
    sort(seg, seg+sz);
    int tmp=1;
    for(int i=1; i<sz; i++)
        if(sign(seg[i].r-seg[tmp-1].r))
            seg[tmp++]=seg[i];
    sz=tmp; Q[0]=seg[0];Q[1]=seg[1];
    int h=0,r=1;
    for(int i=2; i<sz; i++){
        while(h<r&&sign(cross(Q[h].e-seg[i].s,insLL(Q[r],Q[r-1])-seg[i].s,insLL(Q[r],Q[r-1])-seg[i].s))<=0) r--;
        while(h<r&&sign(cross(Q[r].e-seg[i].s,insLL(Q[h],Q[h+1])-seg[i].s,insLL(Q[h],Q[h+1])-seg[i].s))<=0) h++;
        Q[++r]=seg[i];
    }
    while(h<r&&sign(cross(Q[h].e-seg[i].s,insLL(Q[r],Q[r-1])-Q[h].s)<=0) r--;
    while(h<r&&sign(cross(Q[r].e-seg[i].s,insLL(Q[h],Q[h+1])-Q[r].s)<=0)h++;
    if(h+1>=r) return 0;
    int m=0;
    for(int i=h;i<r;i++)p[m++]=insLL(Q[i], Q[i+1]);
    if(r>h+1)p[m++]=insLL(Q[h], Q[r]);
    return m;
}
// 圆面积交 k
struct Event{
```

```cpp
P p;
db ang;
int delta;
Event ( ) {}
Event (P p = P(0, 0), db ang = 0, int delta = 0):p(p), ang(ang), delta(delta){}
bool operator <(const Event& c) const {return ang < c.ang;}
};
db sqr(db x) {return x*x;}
void addEvent(C a, C b, vector<Event> &evt, int&cnt){
    db d2=norm(a.o − b.o),
    dRatio=((a.r − b.r) * (a.r + b.r)/d2+1)/2,
    pRatio=sqrt(−(d2−sqr(a.r−b.r))*(d2−sqr(a.r+b.r))/(d2*d2*4));
    P d=b.o−a.o, p=rot(d, pi/2),
    q0=a.o+d*dRatio+p*pRatio,
    q1=a.o+d*dRatio−p*pRatio;
    db ang0 = ang(a.o, q0), ang1=ang(a.o, q1);
    evt.pb(Event(q1, ang1, 1));evt.pb(Event(q0, ang0, −1));
    cnt += ang1>ang0;
}
bool issame(C a, C b){return !sign(abs(a.o − b.o))&&!sign(a.r−b.r);}
bool overlap(C a, C b){return sign(a.r−b.r−abs(a.o−b.o))>=0;}
bool intersect(C a, C b){return sign(abs(a.o−b.o) − a.r − b.r) < 0;}
void solve(C *c, int n, db *ans){
    memset(ans, 0, sizeof(db) * (n+2));
    for(int i=0; i<n; i++){
        int cnt=1;
        vector<Event> evt;
        for(int j=0; j<i; j++) if(issame(c[i], c[j])) ++cnt;
        for(int j=0; j<n; j++)
            if(j != i && !issame(c[i], c[j]) && overlap(c[j], c[i]))
                cnt++;
        for(int j=0; j<n; j++)
            if(j!=i&&!overlap(c[j], c[i])&& !overlap(c[i], c[j])&&intersect(c[i], c[j]))
                addEvent(c[i], c[j], evt, cnt);
        if(!sz(evt))ans[cnt]+=pi*c[i].r*c[i].r;
        else{
            sort(evt.begin(), evt.end());
            evt.pb(evt.front());
            for(int j=0; j+1<sz(evt); j++){
                cnt+=evt[j].delta;
                ans[cnt]+=cross(evt[j].p, evt[j+1].p)/2;
                db ang=evt[j+1].ang−evt[j].ang;
                if(ang<0)ang+=pi*2;
                ans[cnt]+=ang*c[i].r*c[i].r/2−sin(ang)*c[i].r*c[i].r/2;
            }
        }
    }
}
```

4.2 MaxAreaTri

```cpp
void maxAreaTri(P *p, int n, P &a, P &b, P &c) {
    int i = 0, j = 1, k = 2;
    a = p[i], b = p[j], c = p[k];
    T res = area(a, b, c), cur = res, tmp;
    do {
        while(1) {
            while(cur <= (tmp = area(p[i], p[j], p[(k + 1) % n]))) (++k) %= n, cur = tmp;
            if(cur <= (tmp = area(p[i], p[(j + 1) % n], p[k]))) (++j) %= n, cur = tmp;
            else break;
        }
        if(cur > res) a = p[i], b = p[j], c = p[k], res = cur;
        (++i) %= n;
        if(i == j) (++j) %= n;
        if(j == k) (++k) %= n;
        cur = area(p[i], p[j], p[k]);
    } while(i);
}
```

5 Graph

5.1 1. DCC

```cpp
// key is cuts
// dcc is edges , i−>j , i(points) , j(bcc_block)
// st is stack
// _st is top of stack
// _ is number of dcc
// can handle isolate point and not connected graph and muti edge
// can handle self circle ?
namespace DCC{
    const int N = 202020;
    vi key , dcc[N];
    int dfn[N] , low[N] , st[N] , _st ,_;
    void dfs(int c,int dep,const vi g[]){
        int cc=0,out=1<dep;st[_st++]=c;
        dfn[c]=low[c]=dep;
        for(auto t:g[c])
            if(!dfn[t]){
                dfs(t,dep+1,g);
                low[c]=min(low[c], low[t]);
                if(low[t]>=dfn[c]){
                    if(++out==2) key.pb(c);
                    while(st[__st]!=t) dcc[st[_st]].pb(_);
                    dcc[c].pb(_);dcc[t].pb(_++);
                }
            } else if(dfn[t] != dfn[c] − 1 || cc++)
                low[c] = min(low[c] , dfn[t]);
    }
    int solve(int n,const vi g[]){// n is size of points
        fill_n(dfn,n,=0);
        fill_n(low,n,_st=0);
        fill_n(dcc,n,key=vi());
        rep(i,0,n) if(!dfn[i]) dfs(i,1,g);
        rep(i,0,n) if(sz(dcc[i]) == 0) dcc[i].pb(_++);
        return _;
    }
}
```

5.2 2. BCC

```cpp
// key contains the id of edges
// _ starts from 0
namespace BCC{
    const int N = 202020;
    vi key , bcc[N];
    int dfn[N] , low[N] , id[N] , st[N] , _st , _;
    void dfs(int c,int dep,vector<pii> g[]){
        int cc=0;st[_st++]=c;
        dfn[c]=low[c]=dep;
        for(auto e:g[c]){
            int t=e.fi;
            if(!dfn[t]){
                dfs(t,dep+1,g);
                low[c]=min(low[c],low[t]);
                if(low[t]>dfn[c]) key.pb(e.se);
            } else if(dfn[t] != dfn[c] - 1 || cc++)
                low[c] = min(low[c] , dfn[t]);
        }
        if(low[c]==dfn[c]){
            do{id[st[--_st]]=_;}while(st[_st]!=c);
            _++;
        }
    }
    int solve(int n,vector<pii> g[]){
        fill_n(dfn,n,_=0);
        fill_n(low,n,_st=0);
        fill_n(bcc,n,key=vi());
        rep(i,0,n) if(!dfn[i]) dfs(i,1,g);
        rep(i,0,n) for(auto j:g[i]) if(id[i]!=id[j.fi])
            bcc[id[i]].pb(id[j.fi]);
        return _;
    }
};
```

5.3 3. SCC

```cpp
// _ starts from 0
namespace SCC{
    const int N = 100050;
    int dfn[N],low[N],id[N],st[N],_st,_,cc;
    void dfs(int c,vi g[]){
        dfn[c]=low[c]=++cc;
        st[_st++]=c;
        for(auto t:g[c]){
            if(!dfn[t])
                dfs(t,g),low[c]=min(low[c],low[t]);
            else if(!id[t])
                low[c] =min(low[c],dfn[t]);
        }
        if(low[c]==dfn[c]){
            ++_;
            do{id[st[--_st]]=_;}while(st[_st]!=c);
        }
    }
    vi ng[N];
    int solve(int n,vi g[]){
        fill_n(dfn,n,cc=0);
        fill_n(low,n,_st=0);
        fill_n(id,n,_=0);
        rep(i,0,n) if(!dfn[i]) dfs(i,g);
        rep(i,0,n) --id[i];
        fill_n(ng,_,vi());
        rep(i,0,n) for(auto j:g[i]) if(id[i]!=id[j]) ng[id[i]].pb(id[j]);
        return _;
    }
}
```

5.4 4. MaxMatch

```cpp
namespace MaxMatch{
    const int N = 5050;
    int link[N],vis[N];
    int dfs(int c,vi g[]){
        for(auto t : g[c])
            if(!vis[t]){
                vis[t] = true;
                if(link[t]==-1||dfs(link[t],g))
                    return link[t]=c,1;
            }
        return 0;
    }
    int solve(int n,int m,vi g[]){
        fill_n(link,m,-1);
        int ret=0;
        rep(i,0,n){
            memset(vis,0,m*sizeof(int));
            ret += dfs(i,g);
        }
        return ret;
    }
}
```

5.5 5. KM

```cpp
/*
 * 输入保证左边点数 <= 右边点数
 */
// init!!, id starts from 0
template<class T>
struct KM {
    static const int N = 505;
    static const T inf = ~0U>>2;
    int n, m, left[N], pre[N], used[N];
    T g[N][N], Lx[N], Ly[N], slack[N];
    void init(int _n, int _m) {
        n =_n , m =_m;
        rep(i,0,n) rep(j,0,m) g[i][j] = -inf;
```

```
}
void go(int now) {
  rep(i,0,m+1) used[i]=0,slack[i]=inf;
  left[m] = now;int u,v;
  for(u=m;~left[u];u=v){
    used[u] = 1;
    T d = inf;
    rep(i,0,m) if(!used[i]){
      T tmp = Lx[left[u]] + Ly[i] − g[left[u]][i];
      if(tmp < slack[i]) slack[i] = tmp, pre[i] = u;
      if(slack[i] < d) d = slack[v=i];
    }
    rep(i,0,m+1) if(used[i]) Lx[left[i]] −= d , Ly[i] += d;
      else slack[i] −= d;
  }
  for(;u!=m;left[u]=left[pre[u]],u=pre[u]);
}
T run() {
  fill_n(Lx,n,0);fill_n(Ly,m,0);
  fill_n(left,m,−1);
  rep(i,0,n) go(i);
  T ans = 0;
  rep(i,0,n) ans += Lx[i];
  rep(i,0,m) ans += Ly[i];
  return ans;
}
};
```

5.6 6. 生成树计数与欧拉回路方案数

```
// d[][]:
//    i!=j  d[i][j]=0
//    i==j  d[i][j]=out_deg(i)
// b[][]:
//    from i to j has b[i][j] directed edges
// a[][] = d[][] − b[][]

// 无向图生成树个数: a[][] 任何一个 n−1 阶主子式的绝对值
// 有向图以 i 为根的生成树个数: a[][] 去掉第 i 行第 i 列的行列式的绝对值

int det(int n) { // det(a[1..n−1][1..n−1])
  int ans=1;
  rep(i, 1, n) {
    rep(j, i+1, n) while(a[j][i]) {
      int t = a[i][i] / a[j][i];
      rep(k, i, n) a[i][k] = sub(a[i][k], mul1(a[j][k], t));
      rep(k, i, n) swap(a[i][k], a[j][k]);
      ans = P − ans;
    }
    if(a[i][i] == 0) return 0;
    ans = mul(ans, a[i][i]);
  }
  return ans;
}
```

```
// 有向图要记得判断每个点的出度入度是否相等
// 无向图需要转换成有向图
// tw(G): 以 w 为根的生成树个数
// ec(G) = tw(G) * pi((deg[V] − 1)!)
// ans = ec(G) * deg[w]; 如果求的不是本质不同的, 就还需要这个
// 本质相同:     1231341  1341231
// 本质不同:     1231341  1312341
```

5.7 7. Dijkstra

```
int n, dis[N];
void Dijkstra(int st) {
  priority_queue<pii> q;
  fill_n(dis + 1, n, inf);
  dis[st] = 0;
  q.push(mp(0, st));
  while(!q.empty()) {
    pii u = q.top();q.pop();
    if(dis[u.se] != −u.fi) continue;
    for(auto v : g[u.se]) {
      if(dis[v.fi] > dis[u.se] + v.se) {
        dis[v.fi] = dis[u.se] + v.se;
        q.push(mp(−dis[v.fi], v.fi));
      }
    }
  }
}
```

6 Math

6.1 BerlekampMassey

```
// O(len^2)
vi BM(vi s) {
  vi C(1, 1), B(1, 1);
  int L = 0, m = 1, b = 1;
  rep(n, 0, sz(s)) {
    ll d = 0;
    rep(i, 0, L+1) (d += 1ll * C[i] * s[n−i]) %= P;
    if(d == 0) ++m;
    else {
      vi T = C;
      ll c = P − d * kpow(b, P − 2) % P;
      while(sz(C) < sz(B) + m) C.pb(0);
      rep(i, 0, sz(B)) C[i + m] = add(C[i + m], mul(c, B[i]));
      if(2 * L <= n) {
        L = n + 1 − L, B = T, b = d, m = 1;
      } else {
        ++m;
      }
    }
  }
  reverse(all(C));
```

```
rep(i, 0, sz(C)) c[i] = P - c[i];
return vi(c.begin(), c.end() - 1);
}
```

6.2 Fib

```
// sum(fib[1..n]) + 1=fib[n + 2]
// gcd(fib[n], fib[m]) = fib[gcd(n, m)]
```

6.3 GaussDB

```
namespace GaussDB{
static const int N=210;
double mat[N][N];//增广矩阵
double x[N];//解集
bool free_x[N];//标记是否是不确定的变元
const double eps = 1e-7;
int Gauss(int equ, int var){
    int k;
    int max_r, col;
    int free_index, free_num;
    memset(free_x, 1, sizeof(free_x));
    memset(x, 0, sizeof(x));
    for(k=col=0; k<equ&&col<var; ++k, ++col){
        max_r=k;
        rep(i, k+1, equ)
            if(fabs(mat[i][col])-mat[max_r][col]>eps) max_r=i;
        if(max_r!=k)
            rep(j, k, var+1)swap(mat[max_r][j], mat[k][j]);
        if(fabs(mat[k][col])<eps){--k;continue;}
        rep(i, k+1, equ){
            if(fabs(mat[i][col])<=eps) continue;
            double tmp=mat[i][col]/mat[k][col];
            rep(j, col, var+1)
                mat[i][j]-=mat[k][j]*tmp;
        }
    }
    rep(i, k, equ)
        if(fabs(mat[i][var])>eps) return 0;//无解
    if(k<var){
        for(int i=k-1; i>=0; --i){
            free_num=0;
            rep(j, 0, var){
                if(fabs(mat[i][j])>eps&&free_x[j]){
                    free_num+=1;
                    free_index=j;
                }
            }
            if(free_num>1) continue;
            double tmp=mat[i][var];
            rep(j, 0, var){
                if(j!=free_index&&fabs(mat[i][j])>eps)
                    tmp-=mat[i][j]*x[j];
            }
            free_x[free_index]=0;
            x[free_index]=tmp/mat[i][free_index];
        }
        return var-k;//自由变元个数
    }
    for(int i=var-1; i>=0; --i){
        double tmp=mat[i][var];
        rep(j, i+1, var){
            if(fabs(mat[i][j])>eps)
                tmp-=x[j]*mat[i][j];
        }
        x[i]=tmp/mat[i][i];
    }
    return 1;
}
}
```

6.4 GaussInt

```
namespace Gauss{
static const int N=210;
int a[510][N];
int kpow(int a, int b){
    int r=1;
    while(b>0){
        if(b&1)r=r*a%P;
        a=a*a%P;
        b>>=1;
    }
    return r;
}
int solve(int n, int m){//n=equ, m=var 同 Gaussxor
    int i=0, x=0;
    for(; i<n&&x<m; i++, x++){
        int r=i;
        while(r<n&&!a[r][x])r++;
        if(r>=n){
            i--;
            continue;
        }
        if(r!=i)
            rep(j, 0, m+1)swap(a[r][j], a[i][j]);
        int inv=kpow(a[i][x], P-2);
        for(int k=m; k>=x; k--)a[i][k]=a[i][k]*inv%P;
        rep(j, 0, n)
            if(i!=j&&a[j][x])
                for(int k=m; k>=x; k--)
                    a[j][k]=(a[j][k]-a[i][k]*a[j][x]%P+P)%P;
    }
    rep(k, i, n)if(a[k][m])return -1;
    return m-i;
}
void out(int n, int m){
    rep(i, 0, n){
```

```
        rep(j, 0, m)cout<<a[i][j]<<' ';
        cout<<endl;
    }
};
```

6.5 GaussXor

```
//对 2 取模的 01 方程组
namespace Gause{
    //有 equ 个方程，var 个变元。增广矩阵行数为 equ 列数为，[0..var]
    static const int N=310;
    int equ,var;
    bitset<N> a[N]; //增广矩阵 modif
    int x[N]; //解集
    int free_x[N];//用来存储自由变元（多解枚举自由变元可以使用）
    int free_num;//自由变元的个数
    //返回值为 -1 表示无解，为 0 是唯一解，否则返回自由变元个数
    int Gauss(){
        int max_r,col,k;// k 为增广矩阵的秩
        free_num = 0;
        for(k=0, col=0; k<equ&&col<var; k++, col++){
            max_r = k;
            for(int i=k+1; i<equ; i++){
                if(abs(a[i][col])>abs(a[max_r][col]))
                    max_r=i;
            }
            if(a[max_r][col]==0){
                k--;
                free_x[free_num++]=col;//这个是自由变元
                continue;
            }
            if(max_r!=k){
                swap(a[k],a[max_r]);
            }
            for(int i=k+1; i<equ; i++){
                if(a[i][col]!=0)
                    a[i]^=a[k];
            }
        }
        for(int i=k; i<equ; i++)
            if(a[i][col]!=0)
                return -1;//无解
        if(k<var) return var-k;//自由变元个数
        //唯一解，回代
        for(int i=var-1; i>=0; i--){
            x[i]=a[i][var];
            for(int j=i+1; j<var; j++)
                x[i]^=(a[i][j]&&x[j]);
        }
        return 0;
    }
}
```

6.6 LinearBasis

```
struct Base{
    ll a[63];
    Base() {memset(a,0,sizeof(a));}
    void ins(ll x){
        for(int i=62;~i;--i) {
            if(x>>i&1) {
                if(a[i]) x^=a[i];
                else{ a[i]=x; break; }
            }
        }
    }
};
```

6.7 LinearRecursion

```
// a_{m} = \sum_{j=0}^{-m1}a_{j}*c_{j} O(m^21gn)
int linear_recurrence(ll n, int m, vi a, vi c) {
    vector<ll> v(m, 0), u(m<<1, 0);
    v[0] = 1;
    for(ll x = 0, w = n ? 1ll<<(63 - __builtin_clzll(n)) : 0; w; w >>= 1, x <<= 1) {
        fill(all(u), 0);
        int b = !!(n & w); if(b) x++;
        if(x < m) u[x] = 1;
        else {
            rep(i, 0, m) rep(j, 0, m) (u[i + b + j] += v[i] * v[j]) %= P;
            per(i, m, 2*m) rep(j, 0, m) (u[i - m + j] += c[j] * u[i]) %= P;
        }
        copy(u.begin(), u.begin() + m, v.begin());
    }
    ll ans = 0;
    rep(i, 0, m) (ans += v[i] * a[i]) %= P;
    return ans;
}
```

6.8 Polya

```
/*
* Burnside's lemma
* 首先列出所有可能的染色方案，然后找出每个置换下保持不变的方案（不动点）数。
* 等价类数目：所有置换的不动点数的平均值。
* Polya enumeration theorem
* 一个循环的颜色需相同
*/
```

6.9 Prepare

```
rep(i, 0, n) {
    for(int j = (1 << n) - 1; ~j; --j) if(!(j >> i & 1))
        upd(s[j], s[j] | (1 << i));
    }
}

//
int __builtin_ffs (unsigned int x)
int __builtin_ffsl (unsigned long)
int __builtin_ffsll (unsigned long long)
Returns one plus the index of the least significant 1-bit of x, or if x is zero, returns
zero.

//
int __builtin_clz (unsigned int x)
Returns the number of leading 0-bits in x, starting at the most significant bit position
. If x is 0, the result is undefined.

//
int __builtin_ctz (unsigned int x)
Returns the number of trailing 0-bits in x, starting at the least significant bit
position. If x is 0, the result is undefined.

//
int __builtin_popcount (unsigned int x)
Returns the number of 1-bits in x.

//
int __builtin_parity (unsigned int x)
Returns the parity of x, i.e. the number of 1-bits in x modulo 2.
```

7.2　Bitset

```
// Base
b.any();       // has 1 ?
b.none();      // all 0 ?
b.count();     // cnt of 1

b.set();       // all to 1
b.reset();     // all to 0
b.flip();      // all = 0 <-> 1

b.set(p);      // b[p] = 1
b.test(p);     // b[p] is 1
b.reset(p);    // b[p] = 0
b.flip(p);     // b[p] = 0 <-> 1

// Black tech
// __builtin_ctz in bitst
b._Find_first();

// travel all 1
for (int i = b._Find_first(); i < sz(b); i = b._Find_next(i));
```

```
// 模数不是素数，需要做除法
(a / b) % P = a % (P * b) / b

// 矩阵乘法
// 没有交换律，有结合律。
// 左乘向量取行，右乘取列。

// inv O(n)
inv[1] = 1;
rep(i, 2, N) inv[i] = mul(inv[P%i], P - P/i);

// p O(n)
vi p;
bool vis[N];
for(int i = 2; i < N; ++i) {
    if(!vis[i]) p.pb(i);
    for(int j = 0; j < sz(p) && i * p[j] < N; ++j) {
        vis[i * p[j]] = 1;
        if(i % p[j] == 0) break;
    }
}

// phi O(n)
int cntp, p[N], phi[N], vis[N];
phi[1]=1;
rep(i,2,N) {
    if(!vis[i]) p[cntp++]=i, phi[i]=i-1;
    for(int j=0;j<cntp&&p[j]*i<N;++j) {
        vis[p[j]*i]=1;
        if(i%p[j]==0) {
            phi[p[j]*i]=phi[i]*p[j]%P;
            break;
        } else {
            phi[p[j]*i]=phi[i]*(p[j]-1)%P;
        }
    }
}
```

7　Others

7.1　BitOperation

```
// 枚举子集
for(int i = x; i; (--i) & x) {
}
// 统计子集的答案
rep(i, 0, n) {
    rep(j, 0, 1 << n) if(j >> i & 1) {
        upd(s[j], s[j] ^ (1 << i));
    }
}
// 统计超集的答案
```

```
struct Trie{
    static const int N = 101010 , M = 26;
    int ne[N][M] , fail[N] , fa[N] , rt , L;
    void ini(){ fill_n(ne[fail[0] = N-1],M,0);L = 0;rt = newnode(); }
    int newnode(){ fill_n(ne[L],M,0); return L++; }
    void add(char *s){
        int p = rt;
        for(int i=0;s[i];++i){
            int c = s[i] - 'a';// modify
            if(!ne[p][c]) ne[p][c] = newnode() , fa[L-1] = p;
            p = ne[p][c];
        }
    }
    void Build(){
        vi v;v.pb(rt);
        rep(i,0,sz(v)){
            int c = v[i];
            rep(i,0,M) ne[c][i] ?
                v.pb(ne[c][i]) , fail[ne[c][i]] = ne[fail[c]][i] :
                ne[c][i] = ne[fail[c]][i];
        }
    }
};
```

8.4   4. DoublingArray

```
namespace Doubling{
    static const int N = 101010;
    // sa[0-n]: 排名第i的后缀是以i sa[i] 开头
    // h[1-n]:S[sa[i-1]] 与 S[sa[i]] 的最长公共前缀长度为 h[i]
    int t[N] , wa[N] , wb[N] , sa[N] , h[N];
    void sort(int *x,int *y,int n,int m){
        rep(i,0,m) t[i] = 0;
        rep(i,0,n) t[x[y[i]]]++;
        rep(i,1,m) t[i] += t[i-1];
        per(i,0,n) sa[--t[x[y[i]]]] = y[i];
    }
    bool cmp(int *x,int a,int b,int d){
        return x[a] == x[b] && x[a+d] == x[b+d];
    }
    void da(int *s,int n,int m){
        int *x=wa, *y=wb;
        rep(i,0,n) x[i] = s[i] , y[i] = i;
        sort(x , y , n , m);
        for(int j=1,p=1;p<n;m=p,j<<=1){
            p = 0;rep(i,n-j,n) y[p++] = i;
            rep(i,0,n) if(sa[i] >= j) y[p++] = sa[i] - j;
            sort(x , y , n , m);
            swap(x , y);p = 1;x[sa[0]] = 0;
            rep(i,1,n) x[sa[i]] = cmp(y,sa[i],sa[i-1],j)?p-1:p++;
        }
    }
    void cal_h(int *s,int n,int *rk){
        int j,k=0;
        for(int i=1;i<=n;++i) rk[sa[i]] = i;
```

8   String

8.1   1. StringHash

```
// id starts from 1
const int mod=1e9+7;
ull base[N],ha[N];
char s[N];
void init() {
    base[0]=1;
    rep(i,1,N) base[i]=base[i-1]*mod;
}
void Hash() {
    int len=strlen(s+1);
    ha[0]=0;
    rep(i,1,len+1) ha[i]=ha[i-1]*mod+s[i];
}
ull getHa(int l,int r) {
    return ha[r]-ha[l-1]*base[r-l+1];
}
```

8.2   2. Kmp

```
/*
t:  a  b  a
nt:-1 -1  0
s:  a  b  a  c  a  b  a
ns: 0  1  2 -1  0  1  2
*/
void kmp(char *s,int *ns,char *t,int *nt){
    int lens = strlen(s);
    int lent = strlen(t);
    nt[0] = -1;
    for(int i=0,j=-1;i<lens;++i){
        while(j >= 0 && s[i] != t[j + 1]) j = nt[j];
        if(s[i] == t[j + 1]) ++j;
        ns[i] = j;
        if(j + 1 == lent) j = nt[j];
    }
}
void KMP(){
    scanf("%s%s",s,t);
    kmp(t+1,nt+1,t,nt);
    kmp(s,ns,t,nt);
}
```

8.3   3. ACAutomaton

```
/*
* [0,L) , N-1 is virtual , 0 is rt
* init!!
* addation: end[] end[c]|=end[fail[c]]
*/
```

```cpp
        for(int i=0;i<n;h[rk[i++]] = k)
            for(k&&--k, j=sa[rk[i]-1];s[i+k]==s[j+k];++k);
    }
};
// rank[0~n-1]; 以 i 开头的后缀排名 rank[i]
struct DA{ // [0,n] , in[n] = 0 , n load
    static const int N = 101010;
    int p[18][N] , rk[N] , in[N] , Log[N] , n;
    void Build(){
        Doubling::da(in,n+1,300);
        Doubling::cal_h(in,n,rk);
        Log[0] = -1;for(int i=1;i<=n;++i) Log[i] = Log[i-1] + (i==(i&(-i)));
        for(int i=1;i<=n;++i) p[0][i] = Doubling::h[i];
        for(int j=1;1<<j<=n;++j){
            int lim = n+1-(1<<j);
            for(int i=1;i<=lim;++i)
                p[j][i] = min(p[j-1][i] , p[j-1][i+(1<<j>>1)]);
        }
    }
    // 某两个后缀的最长公共前缀
    int lcp(int a,int b){
        a = rk[a] , b = rk[b];
        if(a > b) swap(a , b);++a;
        int t = Log[b-a+1];
        return min(p[t][a] , p[t][b-(1<<t)+1]);
    }
};
```

8.5   5. SAIS

```cpp
/**
 * Ensure that str[n] is the unique lexicographically smallest character in str.
 * time complexity: O(n)
 */
namespace SA {
    const static int N = 100000 + 10;
    int sa[N], rk[N], ht[N], s[N<<1], t[N<<1], p[N], cnt[N], cur[N];
#define pushS(x) sa[cur[s[x]]--] = x
#define pushL(x) sa[cur[s[x]]++] = x
#define inducedSort(v) std::fill_n(sa, n, -1); std::fill_n(cnt, m, 0); \
    for (int i = 0; i < n; i++) cnt[s[i]]++; \
    for (int i = 1; i < m; i++) cnt[i] += cnt[i-1]; \
    for (int i = 0; i < m; i++) cur[i] = cnt[i-1]; \
    for (int i = n-1; ~i; i--) pushS(v[i]); \
    for (int i = 1; i < m; i++) cur[i] = cnt[i-1]; \
    for (int i = 0; i < n; i++) if (sa[i] > 0 && t[sa[i]-1]) pushL(sa[i]-1); \
    for (int i = 0; i < m; i++) cur[i] = cnt[i]-1; \
    for (int i = n-1; ~i; i--) if (sa[i] > 0 && !t[sa[i]-1]) pushS(sa[i]-1)
    void sais(int n, int m, int *s, int *t, int *p) {
        int n1 = t[n-1] = 0, ch = rk[0] = -1, *s1 = s+n;
        for (int i = n-2; ~i; i--) t[i] = s[i] == s[i+1] ? t[i+1] : s[i] > s[i+1];
        for (int i = 1; i < n; i++) rk[i] = t[i-1] && !t[i] ? (p[n1] = i, n1++) : -1;
        inducedSort(p);
        for (int i = 0, x, y; i < n; i++) if (~(x = rk[sa[i]])) {
            if (ch < 1 || p[x+1] - p[x] != p[y+1] - p[y]) ch++;
            else for (int j = p[x], k = p[y]; j <= p[x+1]; j++, k++)
                if ((s[j]<<1|t[j]) != (s[k]<<1|t[k])) {ch++; break;}
            s1[y = x] = ch;
        }
        if (ch+1 < n1) sais(n1, ch+1, s1, t+n, p+n1);
        else for (int i = 0; i < n1; i++) sa[s1[i]] = i;
        for (int i = 0; i < n1; i++) s1[i] = p[sa[i]];
        inducedSort(s1);
    }
    template<typename T>
    int mapCharToInt(int n, const T *str) {
        int m = *max_element(str, str+n);
        std::fill_n(rk, m+1, 0);
        for (int i = 0; i < n; i++) rk[str[i]] = 1;
        for (int i = 0; i < m; i++) rk[i+1] += rk[i];
        for (int i = 0; i < n; i++) s[i] = rk[str[i]] - 1;
        return rk[m];
    }
    template<typename T>
    void suffixArray(int n, const T *str) {
        int m = mapCharToInt(++n, str);
        sais(n, m, s, t, p);
        for (int i = 0; i < n; i++) rk[sa[i]] = i;
        for (int i = 0, h = ht[0] = 0; i < n-1; i++) {
            int j = sa[rk[i]-1];
            while (i+h < n && j+h < n && s[i+h] == s[j+h]) h++;
            if (ht[rk[i]] = h) h--;
        }
    }
};
// 出现 i 次的子串有 c[i] 个
namespace S {
    int top;
    int sta[N<<1], cnt[N<<1];
    ll c[N];
    inline int gao(int k) {
        int cc = 0;
        while(top && sta[top] > k) {
            cc += cnt[top];
            cnt[top] = 0;
            c[cc] += sta[top] - max(k, sta[top-1]);
            --top;
        }
        return cc;
    }
    inline void push(int x, int y) {
        if(!top || sta[top] != x) sta[++top] = x;
        cnt[top] += y;
    }
    inline void build(int n) {
        top = 0;
        fill_n(c+1, n, 0);
        rep(i, 1, n+1) {
            int lcp = SA::ht[i], cc = gao(lcp);
            push(lcp, cc);
```

```cpp
        push(n - SA::sa[i], 1);
    gao(0);
}
};
```

8.6 6. SuffixAutomaton

```cpp
// [0,L] , 0 is virtual , 1 is rt , init!!
struct SAM{
    static const int N = 101010 , M = 26;
    int par[N] , l[N] , ne[N][M];
    int rt , last , L;
    void add(int c){
        int p = last , np = ++L;
        fill(ne[np] , ne[np] + M , 0);
        l[np] = l[p] + 1;
        last = np;
        while(p && !ne[p][c]) ne[p][c] = np , p = par[p];
        if(!p) par[np] = rt;
        else{
            int q = ne[p][c];
            if(l[q] == l[p] + 1) par[np] = q;
            else{
                int nq = ++L;
                l[nq] = l[p] + 1;
                copy(ne[q] , ne[q] + M , ne[nq]);
                par[nq] = par[q];
                par[q] = par[np] = nq;
                while(p && ne[p][c] == q) ne[p][c] = nq , p = par[p];
            }
        }
    }
    void ini(){
        rt = last = L = 1;
        fill(ne[rt] , ne[rt] + M , 0);
        l[0] = -1;
    }
};
```

8.7 7. Manacher

```cpp
/*
 * length of pa is two size of str
 * pa[i<<1] : odd string 整个回文长度为 2*pa[i<<1]-1
 * pa[i<<1|1] : even string 整个回文长度为 2*pa[i<<1]
 * N>2*n
 */
void Manacher(char *s,int n,int *pa){
    pa[0] = 1;
    for(int i=1,j=0;i<(n<<1)-1;++i){
        int p = i >> 1, q = i - p, r = ((j + 1)>>1) + pa[j] - 1;
        pa[i] = r < q ? 0 : min(r - q + 1, pa[(j<<1) - i]);
        while(0 <= p - pa[i] && q + pa[i] < n && s[p - pa[i]] == s[q + pa[i]])
            pa[i]++;
        if(q + pa[i] - 1 > r) j = i;
    }
}
```

8.8 8. PalindromicTree

```cpp
// [0,p) , 0(even) and 1(odd) is virtual , init!!
struct Palindromic_Tree {
    static const int N = 101010 , M = 26;
    int ne[N][M] , fail[N] , len[N] , S[N] , last , n , p;
    int newnode(int l){
        fill(ne[p] , ne[p] + M , 0);
        len[p] = l;
        return p++;
    }
    void ini(){
        p = 0;newnode(0);newnode(-1);
        S[n = last = 0] = -1;
        fail[0] = 1;
    }
    int get_fail(int x){
        while(S[n - len[x] - 1] != S[n]) x = fail[x];
        return x;
    }
    void add(int c){
        S[++n] = c;
        int cur = get_fail(last);
        if(!ne[cur][c]){
            int now = newnode(len[cur] + 2);
            fail[now] = ne[get_fail(fail[cur])][c];
            ne[cur][c] = now;
        }
        last = ne[cur][c];
    }
};
```

9 Tree

9.1 Centroid

```cpp
// id starts from 1
namespace Centriod {
    const int N = 101010;
    bool vis[N];
    int sz[N];
    void dfssz(int c,int fa,int Sz,int &rt){
        sz[c] = 1;
        for(auto t : g[c]) if(!vis[t]&&t!=fa) dfssz(t,c,Sz,rt) , sz[c]+=sz[t];
        if(!rt && sz[c]*2>Sz) rt=c;
    }
    void dfs(int c){
        int rt=0;dfssz(c,0,0,rt);dfssz(c,0,sz[c],rt=0);
```

```cpp
    // cal something
    vis[rt] = true;
    for(auto t : g[rt]) if(!vis[t]) dfs(t);
  }
};
```

9.2 HeavyChain

```cpp
// id starts with 1
struct HeavyChain{
  static const int N = 100005, inf = ~0U>>1;
  int sz[N], wson[N], top[N], dep[N], id[N], _, par[N], who[N];
  void dfs(int c, int fa, vi g[]){
    sz[c] = 1;
    par[c] = fa;
    dep[c] = dep[fa] + 1;
    int &s = wson[c] = top[c] = 0;
    for(auto t : g[c]) if(t != fa) {
      dfs(t, c, g);
      sz[c] += sz[t];
      if(sz[t] >= sz[s]) s = t;
    }
  }
  void dfs2(int c, int fa, vi g[]){
    id[c] = ++_;
    who[_] = c;
    int s = wson[c];
    if(!top[c]) top[c] = c;
    if(s) top[s] = top[c], dfs2(s, c, g);
    for(auto t : g[c]) if(t != fa && t != s) dfs2(t, c, g);
  }
  void Query(int a, int b){// info in points
    int fa = top[a], fb = top[b];
    while(fa != fb){
      if(dep[fa] < dep[fb]) swap(a, b), swap(fa, fb);
      // Cal id[fa] .. id[a]
      a = par[fa]; fa = top[a];
    }
    if(dep[a] < dep[b]) swap(a, b);
    // Cal id[b] .. id[a]
  }
  void Build(vi g[]){
    dfs(1, 0, g);
    _=0;
    dfs2(1, 0, g);
  }
}hc;
```