

Turtle Fan Club (Letao Zhang, Haoming Yu, Lian Huang) 的 icpc 模板

EuphoricStar & gdf_yhm & dieselhuang

2025 年 12 月 23 日

ver: 1.0.0

目录

1 杂项	4
1.1 fastio	4
1.2 取模优化	4
1.3 i24	4
2 dp	5
2.1 决策单调性	5
2.1.1 SMAWK	5
2.1.2 LARSCH	6
3 ds	7
3.1 线段树	7
3.1.1 zkw	7
3.1.2 楼房重建	7
3.1.3 beats	7
3.1.4 合并分裂	7
3.1.5 KTT	8
3.2 平衡树	8
3.3 莫队	8
3.4 分块	8
3.5 手写 STL	8
3.5.1 bitset	8
3.5.2 哈希表	8
3.5.3 deque	9
4 graph	10
4.1 树	10
4.1.1 毛毛虫剖分	10
4.1.2 点分治	12
4.1.3 动态 dp	12
4.1.4 prufer 序	12
4.1.5 并查集启发式分裂	12
4.1.6 Maintaining mst with online edge insertions —no LCT needed	13
4.2 连通性	14
4.2.1 边双	14
4.2.2 点双	14
4.2.3 双极定向	15
4.2.4 广义串并联图	16
4.3 流	17
4.3.1 预留推进	17
4.3.2 原始对偶	17
4.3.3 最小割树	18
4.3.4 一般图最大匹配	18
4.4 杂项	18
4.4.1 欧拉回路	18
4.4.2 四元环计数	18

5 geometry	19
6 math	23
6.1 篩	23
6.1.1 PN 篩	23
6.2 矩阵	25
6.2.1 高斯消元	25
6.2.2 高斯消元（模 2）	25
6.2.3 矩阵求逆	26
6.2.4 行列式	26
6.2.5 特征多项式	27
6.3 poly	27
6.3.1 fft	27
6.3.2 ntt	27
6.3.3 mtt	28
6.3.4 ni ln exp	28
6.4 另一份 poly	30
6.4.1 多点求值	35
6.5 集合幂级数	35
6.5.1 FWT	36
6.5.2 子集卷积	36
6.5.3 多项式复合集合幂级数	36
6.6 杂项	36
6.6.1 插值	36
7 string	37
7.1 Manacher	37
7.2 Z 函数	37
7.3 Runs	37
7.4 后缀自动机	40

1 杂项

1.1 fastio

```
static char buf[1000000],*p1=buf,*p2=buf;
#define getchar() p1==p2&&(p2=(p1=buf)+fread(buf,1,1000000,stdin),p1==p2)?EOF:*p1++
inline int read(){int x=0,f=1;char c=getchar();while(c<'0'||c>'9'){if(c=='-')f=-1;c=getchar();}
while(c>='0'&&c<='9'){x=(x<<3)+(x<<1)+c-48;c=getchar();}return x*f;}
inline void write(int x){static char buf[20];static int len=-1;if(x<0)putchar('-'),x=-x;do buf[++len]=x%10,x/=10;while(x);while(len>=0)putchar(buf[len--]+48);}
```

1.2 取模优化

1.3 i24

2 dp

2.1 决策单调性

2.1.1 SMAWK

求 $n \times m$ 矩阵每行最小值。矩阵满足对于任意 $x < y$, x 行最小值位置 $\leq y$ 行最小值位置。

cf 的交互格式，没有实际用过。

$4(n + m)$ 次。

```

int n,m,a[10][10];
map<pii,int> mp;
int pre[maxn],suf[maxn];
vector<int> reduce(vector<int> x,vector<int> y){
    int n=x.size(),m=y.size();
    for(int i=0;i<m-1;i++)pre[y[i+1]]=y[i],suf[y[i]]=y[i+1];
    pre[y[0]]=0,suf[0]=y[0],suf[y[m-1]]=y[m-1];
    auto del=[&](int u){suf[pre[u]]=suf[u],pre[suf[u]]=pre[u];};
    for(int i=0,j=y[0],t=n+1;t<=m;){
        if(ask(x[i],j)>ask(x[i],suf[j])){
            del(j);t++;
            if(i)i--,j=pre[j];
            else j=suf[j];
        }
        else{
            if(i==n-1)del(suf[j]),t++;
            else i++,j=suf[j];
        }
    }
    y.clear();
    for(int i=0,j=suf[0];i<n;i++,j=suf[j])y.pb(j);
    return y;
}
int p[maxn];
void smawk(vector<int> x,vector<int> y){
    y=reduce(x,y);
    if(x.size()==1){
        p[x[0]]=y[0];
        return ;
    }
    vector<int> z;
    for(int i=0;i<x.size();i++)if(!(i&1))z.pb(x[i]);
    smawk(z,y);
    for(int i=1;i<x.size();i+=2){
        int l=lower_bound(y.begin(),y.end(),p[x[i-1]])-y.begin();
        int r=(i==x.size()-1?y.size()-1:lower_bound(y.begin(),y.end(),p[x[i+1]])-y.begin());
        p[x[i]]=y[l];
        for(int j=l+1;j<=r;j++)if(ask(x[i],y[j])<ask(x[i],p[x[i]]))p[x[i]]=y[j];
    }
}
void work(){
    n=read();m=read();
    vector<int> x(n),y(m);
    for(int i=1;i<=n;i++)x[i-1]=i;
    for(int i=1;i<=m;i++)y[i-1]=i;
    smawk(x,y);
    int ans=inf;for(int i=1;i<=n;i++)ans=min(ans,ask(i,p[i]));
}

```

```

    printf("! %d\n",ans);fflush(stdout);
}

```

2.1.2 LARSCH

基于魔改的分治，可以在线， $O(n \log n)$ ，支持类莫队计算贡献，常数小，码量小。

```

struct ds{
    int l,r,ans;
    ds(){l=1,r=0;}
    ll que(int ql,int qr){
        while(r<qr)r++;
        while(l>ql)l--;
        while(r>qr)r--;
        while(l<ql)l++;
        return ans;
    }
}a[2];
void upd(int j,int i,int op){
    int nw=dp[j-1]+a[op].que(j,i)+w;
    if(nw<dp[i])dp[i]=nw,p[i]=j;
}
void sovle(int l,int r){
    if(l==r)return ;
    int mid=l+r>>1;
    for(int i=p[l];i<=p[r];i++)upd(i,mid,0);
    sovle(l,mid);
    for(int i=l;i<=mid;i++)upd(i,r,1);
    sovle(mid+1,r);
}

```

3 ds

3.1 线段树

3.1.1 zkw

3.1.2 楼房重建

```

int mn[maxn<<2], ans[maxn<<2];
int query(int nd, int l, int r, int w){
    if(w<mn[nd]) return (r-l+1)*w;
    if(l==r) return mn[nd];
    if(mn[l]<=w) return query(ls,l,mid,w)+ans[rs];
    else return (mid-l+1)*w+query(rs,mid+1,r,w);
}
void up(int nd, int l, int r){
    mn[nd]=min(mn[ls],mn[rs]);
    ans[rs]=query(rs,mid+1,r,mn[ls]);
}
void build(int nd, int l, int r){
    if(l==r){
        mn[nd]=a[l];
        return ;
    }
    build(ls,l,mid),build(rs,mid+1,r);
    up(nd,l,r);
}
int query(int nd, int l, int r, int ql, int qr, int &w){
    if(l>=ql&&r<=qr){
        int res=query(nd,l,r,w);
        w=min(w,mn[nd]);
        return res;
    }
    int res=0;
    if(ql<=mid)res+=query(ls,l,mid,ql,qr,w);
    if(qr>mid)res+=query(rs,mid+1,r,ql,qr,w);
    return res;
}

```

3.1.3 beats

3.1.4 合并分裂

```

int merge(int u, int v, int l, int r){
    if(!u||!v) return u|v;
    if(l==r){tree[u]+=tree[v];clr(v);return u;}
    lc[u]=merge(lc[u],lc[v],l,mid);
    rc[u]=merge(rc[u],rc[v],mid+1,r);
    tree[u]=tree[lc[u]]+tree[rc[u]];clr(v);
    return u;
}
int split(int nd, int l, int r, ll k){
    if(!nd) return 0;
    int u=newnode();
    if(k>tree[ls]) rc[u]=split(rs,mid+1,r,k-tree[ls]);

```

```

    else rc[u]=rs,rs=0;
    if(k<tree[ls])lc[u]=split(ls,l,mid,k);
    tree[nd]=tree[ls]+tree[rs],tree[u]=tree[lc[u]]+tree[rc[u]];
    return u;
}

```

3.1.5 KTT

3.2 平衡树

3.3 莫队

3.4 分块

3.5 手写 STL

3.5.1 bitset

```

#define ull unsigned long long
ull pw[65];
struct bs{
    vector<ull> a;
    int len,n;
    void init(int _n){
        n=_n,len=(n+63)/64;a.resize(len+1,0);
    }
    void set0(int x){a[x>>6]&=~pw[x&63];}
    void set1(int x){a[x>>6]|=pw[x&63];}
    bool operator[](int x){return (a[x>>6]>>(x&63))&1;}
    bs operator<<(int x)const{
        bs res;res.init(n);
        int y=x>>6,z=x&63;
        ull lst=0;
        for(int i=0;i+y<res.len;i++){
            res.a[i+y]=lst|(a[i]<<z);
            if(z)lst=a[i]>>(64ll-z);
        }
        return res;
    }
}f;

```

3.5.2 哈希表

```

struct hsh_table{
    int head[maxn],tot;
    struct nd{
        int nxt;ull key;int val;
    }e[maxn];
    inline int hsh(ull u){return u%maxn;}
    inline int &operator[](ull key){
        int u=hsh(key);
        for(int i=head[u];i;i=e[i].nxt){
            if(e[i].key==key) return e[i].val;
        }
    }
}

```

```

        e[++tot]={head[u],key,0};head[u]=tot;
        return e[tot].val;
    }
}mp;

```

3.5.3 deque

```

vector<int> st,ed;
void push_front(int x){st.pb(x);}
void push_back(int x){ed.pb(x);}
void rebuild(){

}
void rebuildfront(){
    int pos=(ed.size()+1)/2;
    for(int i=0;i<pos;i++)st.pb(ed[i]);
    reverse(st.begin(),st.end());
    reverse(ed.begin(),ed.end());
    for(int i=1;i<=pos;i++)ed.pop_back();
    reverse(ed.begin(),ed.end());
    rebuild();
}
void rebuildback(){
    int pos=(st.size()+1)/2;
    for(int i=0;i<pos;i++)ed.pb(st[i]);
    reverse(ed.begin(),ed.end());
    reverse(st.begin(),st.end());
    for(int i=1;i<=pos;i++)st.pop_back();
    reverse(st.begin(),st.end());
    rebuild();
}
int front(){
    if(!st.size())rebuildfront();
    return st.back();
}
int back(){
    if(!ed.size())rebuildback();
    return ed.back();
}
void pop_front(){
    if(!st.size())rebuildfront();
    st.pop_back();
}
void pop_back(){
    if(!ed.size())rebuildback();
    ed.pop_back();
}
int size(){
    return st.size()+ed.size();
}

```

4 graph

4.1 树

4.1.1 毛毛虫剖分

```

int n,q,k=3;
vector<int> e[maxn];
int siz[maxn],son[maxn],fa[maxn];
int dfn[maxn],st[18][maxn],tim;
void dfs(int u){
    siz[u]=1;
    st[0][dfn[u]=++tim]=fa[u];
    vector<int> tmp;
    for(int v:e[u])if(v!=fa[u]){
        fa[v]=u;dfs(v);siz[u]+=siz[v];
        tmp.pb(v);
    }
    e[u]=tmp;
    sort(e[u].begin(),e[u].end(),[&](int u,int v){return siz[u]>siz[v];});
    if(e[u].size())son[u]=e[u][0];
}
int id[maxn],idx;
void downid(int u,int d){
    if(!d){
        if(!id[u])id[u]=++idx;
        return ;
    }
    for(int v:e[u])downid(v,d-1);
}
void dfsid(int u){
    vector<int> path;
    for(int x=u;x;x=son[x])path.pb(x);
    for(int i=0;i<=k;i++){
        for(int u:path)downid(u,i);
    }
    reverse(path.begin(),path.end());
    for(int u:path){
        for(int v:e[u])if(v!=son[u])dfsid(v);
    }
}
void merge(vector<pii> &u,vector<pii> v){
    for(auto p:v)u.pb(p);
}
void reinit(vector<pii> &u){
    sort(u.begin(),u.end());
    vector<pii> nw;
    for(auto [l,r]:u){
        if(nw.size()&&nw.back().se+1==l)nw.back().se=r;
        else nw.pb({l,r});
    }
    u=nw;
}
vector<pii> sub[maxn],kson[maxn][maxk],bro[maxn][maxk];
void dfsup(int u){
    sub[u]={{id[u],id[u]}},kson[u][0]={{id[u],id[u]}};
    for(int v:e[u]){

```

```

dfsup(v);
merge(sub[u],sub[v]);
for(int i=0;i<=k;i++)bro[v][i]=kson[u][i];
for(int i=1;i<=k;i++)merge(kson[u][i],kson[v][i-1]),reinit(kson[u][i]);
}
if(e[u].size()){
vector<pii> tmp[maxk];
for(int ii=e[u].size()-1;~ii;ii--){
int v=e[u][ii];
for(int i=1;i<=k;i++)merge(bro[v][i],tmp[i]),reinit(bro[v][i]);
for(int i=1;i<=k;i++)merge(tmp[i],kson[v][i-1]),reinit(tmp[i]);
}
reinit(sub[u]);
}
vector<pii> kans[maxn][maxk],kdis[maxn][maxk];
void dfsdw(int u){
for(int i=0;i<=k;i++){
kans[u][i]=kans[fa[u]][i];
merge(kans[u][i],kson[u][i]);
reinit(kans[u][i]);
}
for(int i=0;i<=k;i++){
for(int j=0;j<=i;j++)merge(kdis[u][i],kson[u][j]);
for(int j=1,x=u;j<=k&&x==fa[x];j++){
for(int k=0;k<=i-j;k++)merge(kdis[u][i],bro[x][k]);
}
reinit(kdis[u][i]);
}
for(int v:e[u])dfsdw(v);
}
vector<pii> getsub(int u){return sub[u];}
vector<pii> gettp(int u,int tp,int k){
vector<pii> a=kans[u][k],b=kans[tp][k],nw;
for(int i=0,l=0;i<a.size();i++){
while(l<b.size()&&b[l].se<=a[i].fi)l++;
int r=l;while(r<b.size()&&b[r].se<=a[i].se)r++;
if(l==r)nw.pb(a[i]);
else{
int lst=a[i].fi;
for(int j=l;j<r;j++){
if(lst<b[j].fi)nw.pb({lst,b[j].fi-1});
lst=b[j].se+1;
}
if(lst<=a[i].se)nw.pb({lst,a[i].se});
}
l=r;
}
reinit(nw);
return nw;
}
vector<pii> getpath(int u,int v,int k){
int tp=lca(u,v);
vector<pii> a=kdis[tp][k];
merge(a,gettp(u,tp,k));
merge(a,gettp(v,tp,k));
reinit(a);
}

```

```

        return a;
    }
void work(){
    dfs(1);
    lca_init();
    dfsid(1);
    dfsup(1);
    dfsdw(1);
}

```

4.1.2 点分治

4.1.3 动态 dp

4.1.4 prufer 序

4.1.5 并查集启发式分裂

```

set<int> e[maxn],id[maxn<<1];
int fa[maxn],idx;
ll del[maxn*3];
struct node{
    int u,fa;
    set<int>::iterator it;
};
void add(int u,int v){
    e[u].insert(v),e[v].insert(u);
    int uu=fa[u],vv=fa[v];
    sum+=1ll*id[uu].size()*id[vv].size();
    if(id[uu].size()<id[vv].size())swap(u,v),swap(uu,vv);
    for(int i:id[vv])fa[i]=uu,id[uu].insert(i);id[vv].clear();
}
void del(int u,int v){
    e[u].erase(v),e[v].erase(u);
    vector<int> pos[2];
    queue<node> que[2];
    pos[0].pb(u),pos[1].pb(v);
    if(e[u].size())que[0].push({u,0,e[u].begin()});
    if(e[v].size())que[1].push({v,0,e[v].begin()});
    while(que[0].size()&&que[1].size()){
        int o=pos[1].size()-pos[0].size();
        auto [u,fa,it]=que[o].front();que[o].pop();
        int v=(*it);
        if(v!=fa){
            pos[o].pb(v);
            if(e[v].size())que[o].push({v,u,e[v].begin()});
        }
        it++;
        if(it==e[u].end())continue;
        if((*it)==fa)it++;
        if(it==e[u].end())continue;
        que[o].push({u,fa,it});
    }
    if(!que[0].size()&&(que[1].size()||pos[0].size()<pos[1].size())){
        swap(u,v),swap(pos[0],pos[1]);
    }
    del[qq]+=1ll*pos[1].size()*(id[fa[u]].size()-pos[1].size());
}

```

```

    ++idx;
    for(int i:pos[1])id[fa[u]].erase(i),id[idx].insert(i),fa[i]=idx;
}

```

4.1.6 Maintaining mst with online edge insertions —no LCT needed

比 lct 快 10 倍。

```

mt19937 rnd(time(0));
struct weightdsu{
    int fa[maxn],rd[maxn];pii val[maxn];
    int siz[maxn];
    weightdsu(int n){
        for(int i=1;i<=n;i++)fa[i]=i,rd[i]=rnd(),val[i]={inf,inf};
    }
    int fd(int u,pii w={inf-1,inf}){
        while(val[u]<=w){
            while(val[fa[u]]<=val[u]){
                // siz[fa[u]]-=siz[u];
                fa[u]=fa[fa[u]];
            }
            u=fa[u];
        }
        return u;
    }
    void del(int u){
        // if(fa[u]==u)return ;
        // del(fa[u]);siz[fa[u]]-=siz[u];
    }
    int ins(int u,pii w={inf-1,inf}){
        while(val[u]<=w){
            // siz[fa[u]]+=siz[u];
            u=fa[u];
        }
        return u;
    }
    void merge(int u,int v,pii w){
        del(u),del(v);
        while(u!=v){
            u=ins(u,w),v=ins(v,w);
            if(rd[u]<rd[v])swap(u,v);
            swap(fa[u],v),swap(val[u],w);
        }
        ins(u);
    }
    pii max_path(int u,int v){
        int uu=fd(u),vv=fd(v);
        if(uu!=vv)return {inf,-1};
        if(val[u]>val[v])swap(u,v);
        while(fa[u]!=v){
            u=fa[u];
            if(val[u]>val[v])swap(u,v);
        }
        return val[u];
    }
    pii del_path(int u,int v){
        int uu=fd(u),vv=fd(v);

```

```

if(uu!=vv) return {inf,-1};
if(val[u]>val[v]) swap(u,v);
while(fa[u]!=v){
    u=fa[u];
    if(val[u]>val[v]) swap(u,v);
}
v=u;
while(fa[v]!=v){
    // siz[fa[v]]-=siz[v];
    v=fa[v];
}
fa[u]=u;
pii res={inf,inf};swap(res,val[u]);
return res;
}
pii add_edge(int u,int v,pii w){
    pii res=del_path(u,v);
    if(res<=w)swap(res,w);
    merge(u,v,w);
    return res;
}
};


```

4.2 连通性

4.2.1 边双

```

int dfn[maxn],lw[maxn],idx;
int st[maxn],tp;
vector<int> g[maxn];
int scct;
bool vis[maxn];
void tar(int u,int fl){
    dfn[u]=lw[u]=++idx;st[++tp]=u;
    for(int i=head[u];i;i=e[i].nxt){
        int v=e[i].to;
        if(i==(fl^1))continue;
        if(!dfn[v]){
            tar(v,i);
            lw[u]=min(lw[u],lw[v]);
        }
        else lw[u]=min(lw[u],dfn[v]);
    }
    if(lw[u]==dfn[u]){
        g[scct].push_back(st[tp]);
        while(st[tp--]!=u){
            g[scct].push_back(st[tp]);
        }
    }
}


```

4.2.2 点双

```

vector<int> e[maxn],g[maxn];
int dfn[maxn],idx,lw[maxn];


```

```

int st[maxn], tp;
void tar(int u){
    dfn[u]=lw[u]=++idx; st[++tp]=u;
    for(int v:e[u]){
        if(!dfn[v]){
            tar(v);
            lw[u]=min(lw[u], lw[v]);
            if(lw[v]>=dfn[u]){
                g[++num].push_back(st[tp]);
                while(st[tp--]!=v){
                    g[num].push_back(st[tp]);
                }
                g[num].push_back(u);
            }
        }
        else lw[u]=min(lw[u], dfn[v]);
    }
}

```

4.2.3 双极定向

```

int n,m,s,t;
pii g[maxn];
int lw[maxn], dfn[maxn], idx, fa[maxn];
vector<int> id;
bool vis[maxn];
bool dfs(int u){
    dfn[u]=lw[u]=++idx; vis[u]=1;
    bool fl=u==t;
    for(int i=head[u]; i; i=e[i].nxt){
        int v=e[i].to;
        if(!vis[v]){
            fa[v]=u; fl|=dfs(v);
            lw[u]=min(lw[u], lw[v]);
        }
        else lw[u]=min(lw[u], dfn[v]);
    }
    if(fl)id.pb(u);
    return fl;
}
queue<int> q;
int d[maxn];
vector<int> a[maxn];
int st[maxn], tp, rnk[maxn];
void dfs1(int u){
    if(vis[u])return ; vis[u]=1;
    st[++tp]=u;
    for(int v:a[dfn[u]])dfs1(v);
}
void work(){
    n=read(); m=read(); s=read(); t=read();
    for(int i=1; i<=n; i++) head[i]=0; tot=0;
    for(int i=1; i<=m; i++){
        int u=read(), v=read();
        add(u, v), add(v, u);
        g[i]={u, v};
    }
}

```

```

}
idx=0;id.clear();
for(int i=1;i<=n;i++)vis[i]=0;
fa[s]=0;dfs(s);
for(int i=1;i<=n;i++)d[i]=0;
for(int i:id)d[i]++;
for(int i=1;i<=n;i++)d[fa[i]]++;
for(int i=1;i<=n;i++)if(!d[i])q.push(i);
for(int i=1;i<=n;i++)a[i].clear();
while(!q.empty()){
    int u=q.front();q.pop();
    a[lw[u]].pb(u),a[dfn[fa[u]]].pb(u);
    d[fa[u]]--;
    if(!d[fa[u]])q.push(fa[u]);
}
tp=0;
for(int i=1;i<=n;i++)vis[i]=0;
while(id.size())dfs1(id.back()),id.pop_back();
if(st[1]!=s||st[tp]!=t){puts("No");return ;}
check();
}

```

4.2.4 广义串并联图

```

map<int,int> mp[maxn];
int d[maxn];
queue<int> q;
void add(int u,int v,int w){
    if(mp[u].find(v)!=mp[u].end())ans=max(ans,mp[u][v]+w);
    else mp[u][v]=-inf,d[u]++;
    mp[u][v]=max(mp[u][v],w);
}
void work(){
    n=read();m=read();
    for(int i=1;i<=m;i++){
        int u=read(),v=read(),w=read();
        add(u,v,w),add(v,u,w);
    }
    for(int i=1;i<=n;i++)if(d[i]<=2)q.push(i);
    while(!q.empty()){
        int u=q.front();q.pop();
        if(!d[u])continue;
        else if(d[u]==1){
            int v=(*mp[u].begin()).fi;
            mp[u].erase(v),mp[v].erase(u),d[u]--,d[v]--;
            if(d[v]<=2)q.push(v);
        }
        else if(d[u]==2){
            int v1=(*mp[u].begin()).fi,v2=(--mp[u].end()).fi;
            int w1=(*mp[u].begin()).se,w2=(--mp[u].end()).se;
            add(v1,v2,w1+w2),add(v2,v1,w1+w2);
            mp[u].erase(v1),mp[u].erase(v2),mp[v1].erase(u),mp[v2].erase(u),d[u]==2,d[v1]--,d[v2]--
                ];
            if(d[v1]<=2)q.push(v1);
            if(d[v2]<=2)q.push(v2);
        }
    }
}

```

```
    }
    printf("%lld\n",ans);
}
```

4.3 流

4.3.1 预留推进

4.3.2 原始对偶

```

int h[maxn]; bool vis[maxn];
void spfa(){
    queue<int> q;
    for(int i=1;i<=n;i++) h[i]=inf, vis[i]=0;
    h[s]=0, vis[s]=1, q.push(s);
    while(!q.empty()){
        int u=q.front(); q.pop(); vis[u]=0;
        for(int i=head[u];i;i=e[i].nxt){
            int v=e[i].to;
            if(e[i].w&&h[v]>h[u]+e[i].c){
                h[v]=h[u]+e[i].c;
                if(!vis[v]) vis[v]=1, q.push(v);
            }
        }
    }
}
int dis[maxn], pre[maxn], id[maxn];
bool dij(){
    priority_queue<pii> q;
    for(int i=1;i<=n;i++) dis[i]=inf, vis[i]=0;
    dis[s]=0; q.push({0,s});
    while(!q.empty()){
        int u=q.top().se; q.pop();
        if(vis[u]) continue; vis[u]=1;
        for(int i=head[u];i;i=e[i].nxt){
            int v=e[i].to, val=e[i].c+h[u]-h[v];
            if(e[i].w&&dis[v]>dis[u]+val){
                dis[v]=dis[u]+val, pre[v]=u, id[v]=i;
                q.push({-dis[v],v});
            }
        }
    }
    return dis[t]!=inf;
}
void work(){
    n=read(); m=read(); s=read(); t=read();
    for(int i=1;i<=m;i++){
        int u=read(), v=read(), w=read(), c=read();
        add(u,v,w,c);
    }
    spfa();
    while(dij()){
        for(int i=1;i<=n;i++) h[i]+=dis[i];
        int mn=inf;
        for(int u=t;u!=s;u=pre[u]) mn=min(mn, e[id[u]].w);
        flow+=mn;
    }
}

```

```

        for(int u=t;u!=s;u=pre[u]){
            e[id[u]].w-=mn,e[id[u]^1].w+=mn;
            ans+=e[id[u]].c*mn;
        }
    }
    printf("%lld %lld\n",flow,ans);
}

```

4.3.3 最小割树

4.3.4 一般图最大匹配

4.4 杂项

4.4.1 欧拉回路

4.4.2 四元环计数

```

vector<int> e[maxn],g[maxn];
int d[maxn],cnt[maxn],ans;
void work(){
    n=read();m=read();
    for(int i=1;i<=m;i++){
        int u=read(),v=read();
        e[u].push_back(v),e[v].push_back(u);
        d[u]++,d[v]++;
    }
    for(int u=1;u<=n;u++){
        for(int v:e[u]){
            if(d[u]>d[v]||(d[u]==d[v]&&u>v))g[u].push_back(v);
        }
    }
    for(int i=1;i<=n;i++){
        for(int j:g[i]){
            for(int k:e[j])if(d[i]>d[k]||(d[i]==d[k]&&i>k))ans+=cnt[k]++;
        }
        for(int j:g[i]){
            for(int k:e[j])cnt[k]=0;
        }
    }
    printf("%lld\n",ans);
}

```

5 geometry

```

struct node {
    ll x, y;
    node(ll _x = 0, ll _y = 0) : x(_x), y(_y) {}

    inline ll len() {
        return x * x + y * y;
    }

    inline ll db dis() {
        return sqrtl(x * x + y * y);
    }

    // 极角排序用 (从 x 轴负半轴 (不含) 开始进行逆时针排序)
    inline int reg() {
        if (x < 0 && y < 0) {
            return 1;
        } else if (x == 0 && y < 0) {
            return 2;
        } else if (x > 0 && y < 0) {
            return 3;
        } else if (x >= 0 && y == 0) {
            return 4;
        } else if (x > 0 && y > 0) {
            return 5;
        } else if (x == 0 && y > 0) {
            return 6;
        } else if (x < 0 && y > 0) {
            return 7;
        } else {
            return 8;
        }
    }
};

typedef vector<node> conv;

inline node operator + (const node &a, const node &b) {
    return node(a.x + b.x, a.y + b.y);
}

inline node operator - (const node &a, const node &b) {
    return node(a.x - b.x, a.y - b.y);
}

inline bool operator < (const node &a, const node &b) {
    return a.x < b.x || (a.x == b.x && a.y < b.y);
}

inline bool operator == (const node &a, const node &b) {
    return a.x == b.x && a.y == b.y;
}

inline ll operator * (const node &a, const node &b) {
    return a.x * b.y - a.y * b.x;
}

```

```

inline ll dot(const node &a, const node &b) {
    return a.x * b.x + a.y * b.y;
}

// 检查是否为凸包
inline bool checkcon(conv a) {
    int n = (int)a.size(), p = 0;
    if ((a[1] - a[0]) * (a[2] - a[0]) < 0) {
        reverse(a.begin(), a.end());
    }
    for (int i = 1; i < n; ++i) {
        if (a[i] < a[p]) {
            p = i;
        }
    }
    rotate(a.begin(), a.begin() + p, a.end());
    for (int i = 1; i <= n - 2; ++i) {
        if ((a[i] - a[0]) * (a[i + 1] - a[i]) <= 0) {
            return 0;
        }
    }
    for (int i = 0; i < n; ++i) {
        int j = (i == n - 1 ? 0 : i + 1);
        int k = (j == n - 1 ? 0 : j + 1);
        if ((a[j] - a[i]) * (a[k] - a[j]) <= 0) {
            return 0;
        }
    }
    return 1;
}

// 建凸包
inline conv makecon(conv a) {
    int n = (int)a.size();
    sort(a.begin(), a.end());
    vector<int> stk(n + 1);
    int top = 0;
    for (int i = 0; i < n; ++i) {
        while (top >= 2 && (a[stk[top]] - a[stk[top - 1]]) * (a[i] - a[stk[top - 1]]) <= 0) {
            --top;
        }
        stk[++top] = i;
    }
    conv b;
    for (int i = 1; i < top; ++i) {
        b.pb(a[stk[i]]);
    }
    top = 0;
    for (int i = n - 1; ~i; --i) {
        while (top >= 2 && (a[stk[top]] - a[stk[top - 1]]) * (a[i] - a[stk[top - 1]]) <= 0) {
            --top;
        }
        stk[++top] = i;
    }
    for (int i = 1; i < top; ++i) {
        b.pb(a[stk[i]]);
    }
}

```

```

    }
    return b;
}

// 凸包直径
inline ll diam(conv a) {
    int n = (int)a.size(), j = 2;
    if (n == 2) {
        return (a[0] - a[1]).len();
    }
    ll ans = 0;
    for (int i = 0; i < n; ++i) {
        ans = max(ans, (a[i] - a[(i + 1) % n]).len());
        while ((a[(i + 1) % n] - a[i]) * (a[j] - a[(i + 1) % n]) < (a[(i + 1) % n] - a[i]) * (a[(j + 1) % n] - a[(i + 1) % n])) {
            j = (j + 1) % n;
        }
        ans = max({ans, (a[i] - a[j]).len(), (a[(i + 1) % n] - a[j]).len()});
    }
    return ans;
}

// P 到 AB 所在直线距离
inline ldb pointdis(node a, node b, node p) {
    if (dot(p - a, b - a) <= 0) {
        return (p - a).dis();
    } else if (dot(p - b, a - b) <= 0) {
        return (p - b).dis();
    } else {
        return abs((p - a) * (p - b)) / (a - b).dis();
    }
}

// P 是否在线段 AB 上
inline bool onseg(node a, node b, node p) {
    if ((p - a) * (p - b) != 0) {
        return 0;
    } else {
        return dot(p - a, p - b) <= 0;
    }
}

// 线段 AB 和线段 CD 是否相交（含端点）
inline bool seginter(node a, node b, node c, node d) {
    auto sgn = [&](ll x) -> int {
        return x > 0 ? 1 : (x < 0 ? -1 : 0);
    };
    int o1 = sgn((b - a) * (c - a)), o2 = sgn((b - a) * (d - a)), o3 = sgn((d - c) * (a - c)), o4 = sgn((d - c) * (b - c));
    return ((o1 * o2 < 0 && o3 * o4 < 0) || onseg(a, b, c) || onseg(a, b, d) || onseg(c, d, a) || onseg(c, d, b));
}

// 线段 AB 到线段 CD 距离
inline ldb segdis(node a, node b, node c, node d) {
    return seginter(a, b, c, d) ? 0 : min({pointdis(a, b, c), pointdis(a, b, d), pointdis(c, d, a), pointdis(c, d, b)});
}

```

}

6 math

6.1 筛

6.1.1 PN 筛

定义

Powerful Number (以下简称 PN) 筛类似于杜教筛，或者说是杜教筛的一个扩展，可以拿来求一些积性函数的前缀和。

要求：

- 存在一个函数 g 满足：
 - g 是积性函数。
 - g 易求前缀和。
 - 对于质数 p , $g(p) = f(p)$ 。

假设现在要求积性函数 f 的前缀和 $F(n) = \sum_{i=1}^n f(i)$ 。

Powerful Number

定义：对于正整数 n , 记 n 的质因数分解为 $n = \prod_{i=1}^m p_i^{e_i}$. n 是 PN 当且仅当 $\forall 1 \leq i \leq m, e_i > 1$.

性质 1：所有 PN 都可以表示成 a^2b^3 的形式。

证明：若 e_i 是偶数，则将 $p_i^{e_i}$ 合并进 a^2 里；若 e_i 为奇数，则先将 p_i^3 合并进 b^3 里，再将 $p_i^{e_i-3}$ 合并进 a^2 里。

性质 2： n 以内的 PN 至多有 $O(\sqrt{n})$ 个。

证明：考虑枚举 a , 再考虑满足条件的 b 的个数，有 PN 的个数约等于

$$\int_1^{\sqrt{n}} \sqrt[n]{\frac{1}{x^2}} dx = O(\sqrt{n})$$

那么如何求出 n 以内所有的 PN 呢？线性筛找出 \sqrt{n} 内的所有素数，再 DFS 搜索各素数的指数组即可。由于 n 以内的 PN 至多有 $O(\sqrt{n})$ 个，所以至多搜索 $O(\sqrt{n})$ 次。

PN 筛

首先，构造出一个易求前缀和的积性函数 g ，且满足对于素数 p , $g(p) = f(p)$ 。记 $G(n) = \sum_{i=1}^n g(i)$ 。

然后，构造函数 $h = f/g$, 这里的 $/$ 表示狄利克雷卷积除法。根据狄利克雷卷积的性质可以得知 h 也为积性函数，因此 $h(1) = 1$ 。 $f = g * h$, 这里 $*$ 表示狄利克雷卷积。

对于素数 p , $f(p) = g(1)h(p) + g(p)h(1) - h(p) + g(p) \Rightarrow h(p) = 0$ 。根据 $h(p) = 0$ 和 h 是积性函数可以推出对于非 PN 的数 n 有 $h(n) = 0$, 即 h 仅在 PN 处取有效值。

现在，根据 $f = g * h$ 有

$$\begin{aligned} F(n) &= \sum_{i=1}^n f(i) \\ &= \sum_{i=1}^n \sum_{d|i} h(d)g\left(\frac{i}{d}\right) \\ &= \sum_{d=1}^n \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} h(d)g(i) \\ &= \sum_{d=1}^n h(d) \sum_{i=1}^{\lfloor \frac{n}{d} \rfloor} g(i) \\ &= \sum_{d=1}^n h(d)G\left(\left\lfloor \frac{n}{d} \right\rfloor\right) \\ &= \sum_{d \in \text{PN}} h(d)G\left(\left\lfloor \frac{n}{d} \right\rfloor\right) \end{aligned}$$

$O(\sqrt{n})$ 找出所有 PN，计算出所有 h 的有效值。对于 h 有效值的计算，只需要计算出所有 $h(p^e)$ 处的值，就可以根据 h 为积性函数推出 h 的所有有效值。现在对于每一个有效值 d , 计算 $h(d)G\left(\left\lfloor \frac{n}{d} \right\rfloor\right)$ 并累加即可得到 $F(n)$ 。

下面考虑计算 $h(p^e)$ ，一共有两种方法：一种是直接推出 $h(p^e)$ 仅与 p, e 有关的计算公式，再根据公式计算 $h(p^e)$ ；另一种是根据 $f = g * h$ 有 $f(p^e) = \sum_{i=0}^e g(p^i)h(p^{e-i})$ ，移项可得 $h(p^e) = f(p^e) - \sum_{i=1}^e g(p^i)h(p^{e-i})$ ，现在就可以枚举素数 p 再枚举指数 e 求解出所有 $h(p^e)$ 。

过程

1. 构造 g
2. 构造快速计算 G 的方法
3. 计算 $h(p^e)$
4. 搜索 PN，过程中累加答案
5. 得到结果

对于第 3 步，可以直接根据公式计算，可以使用枚举法预处理打表，也可以搜索到了再临时推。

```

ll n, pr[maxn / 5], tot, pw[999], ans;
bool vis[maxn];
int phi[maxn], a[maxn];
vector<ll> h[100100];

inline void init() {
    phi[1] = 1;
    for (int i = 2; i <= N; ++i) {
        if (!vis[i]) {
            pr[++tot] = i;
            phi[i] = i - 1;
        }
        for (int j = 1; j <= tot && i * pr[j] <= N; ++j) {
            vis[i * pr[j]] = 1;
            if (i % pr[j] == 0) {
                phi[i * pr[j]] = phi[i] * pr[j];
                break;
            }
        }
        phi[i * pr[j]] = phi[i] * (pr[j] - 1);
    }
}

```

```

        }
    }
    for (int i = 1; i <= N / 2; ++i) {
        a[i] = (a[i - 1] + phi[i * 2]) % mod;
    }
    for (int i = 1; i <= N; ++i) {
        phi[i] = (phi[i] + phi[i - 1]) % mod;
    }
}

unordered_map<ll, ll> mp, M;

ll dfs(ll n) {
    if (n <= N) {
        return phi[n];
    }
    if (mp.find(n) != mp.end()) {
        return mp[n];
    }
    ll ans = (n % mod) * ((n + 1) % mod) % mod * inv2 % mod;
    for (ll i = 2, j; i <= n; i = j + 1) {
        j = n / (n / i);
        ans = (ans - (j - i + 1) % mod * dfs(n / i) % mod) % mod;
    }
    return mp[n] = ans;
}

ll calc(ll n) {
    if (n <= N / 2) {
        return a[n];
    }
    if (M.find(n) != M.end()) {
        return M[n];
    }
    return M[n] = (dfs(n) + calc(n / 2)) % mod;
}

inline ll G(ll n) {
    return (dfs(n) + calc(n / 2) * 2 % mod) % mod;
}

void dfs(int d, ll x, ll y) {
    if (pr[d] * pr[d] > n / x) {
        ans = (ans + y * G(n / x) % mod) % mod;
        return;
    }
    int k = 0;
    while (x <= n) {
        if (k != 1) {
            dfs(d + 1, x, y * h[d][k] % mod);
        }
        x *= pr[d];
        ++k;
    }
}

void solve() {

```

```

scanf("%lld", &n);
for (int i = 1; i <= tot && pr[i] * pr[i] <= n; ++i) {
    int cnt = 0;
    ll x = n;
    while (x) {
        ++cnt;
        x /= pr[i];
    }
    pw[0] = 1;
    for (int j = 1; j <= cnt + 3; ++j) {
        pw[j] = pw[j - 1] * pr[i] % mod;
    }
    ll p = 1;
    for (int j = 0; p <= n; ++j, p *= pr[i]) {
        if (j == 0) {
            h[i].pb(1);
        } else if (j == 1) {
            h[i].pb(0);
        } else {
            ll t = pr[i] ^ j;
            for (int k = 0; k < j; ++k) {
                ll g = pw[j - k - 1] * (pr[i] - 1) % mod;
                if (pr[i] == 2) {
                    g = g * 3 % mod;
                }
                t = (t - g * h[i][k] % mod + mod) % mod;
            }
            h[i].pb(t);
        }
    }
}
dfs(1, 1, 1);
printf("%lld\n", ans);
}

```

6.2 矩阵

6.2.1 高斯消元

6.2.2 高斯消元（模 2）

```

const int maxn = 5050;

int n, m, b[maxn], ans[maxn];
bitset<maxn> a[maxn];

void solve() {
    n = read();
    m = read();
    for (int i = 1; i <= m; ++i) {
        for (int j = 1; j <= n + 1; ++j) {
            int x = read();
            if (x) {
                a[i].set(j);
            }
        }
    }
}

```

```

}
mems(b, -1);
for (int i = 1, r = 1; i <= n; ++i) {
    int p = -1;
    for (int j = r; j <= m; ++j) {
        if (a[j].test(i)) {
            p = j;
            break;
        }
    }
    if (p == -1) {
        continue;
    }
    if (p != r) {
        swap(a[p], a[r]);
    }
    for (int j = 1; j <= m; ++j) {
        if (j != r && a[j].test(i)) {
            a[j] ^= a[r];
        }
    }
    ++r;
}
for (int i = 1; i <= m; ++i) {
    if (a[i].any()) {
        ans[a[i]._Find_first()] = a[i].test(n + 1);
    }
}
for (int i = 1; i <= n; ++i) {
    pc('0' + ans[i]);
    pc(" \n"[i == n]);
}
}
}

```

6.2.3 矩阵求逆

6.2.4 行列式

```

inline int det() {
    int ans = 1;
    for (int i = 1; i <= n; ++i) {
        for (int j = i; j <= n; ++j) {
            if (a[j][i]) {
                if (j != i) {
                    swap(a[j], a[i]);
                    fix(ans = -ans);
                }
                break;
            }
        }
        if (!a[i][i]) {
            return 0;
        }
        ans = 1LL * ans * a[i][i] % mod;
        int t = qpow(a[i][i], mod - 2);
        for (int j = i; j <= n; ++j) {

```

```

        a[i][j] = 1LL * a[i][j] * t % mod;
    }
    for (int j = 1; j <= n; ++j) {
        if (i == j) {
            continue;
        }
        for (int k = i + 1; k <= n; ++k) {
            fix(a[j][k] -= 1LL * a[j][i] * a[i][k] % mod);
        }
    }
}
return ans;
}

```

6.2.5 特征多项式

6.3 poly

6.3.1 fft

```

struct cp{
    db a,b;
    cp(db u=0,db v=0){a=u,b=v;}
    cp operator+(const cp&tmp) const{return {a+tmp.a,b+tmp.b);}
    cp operator-(const cp&tmp) const{return {a-tmp.a,b-tmp.b);}
    cp operator*(const cp&tmp) const{return {a*tmp.a-b*tmp.b,a*tmp.b+b*tmp.a);}
};

const db pi=acos(-1);
int to[maxn<<3];
void fft(vector<cp> &a,int flag){
    int n=a.size();
    for(int i=0;i<n;i++) if(i<to[i]) swap(a[i],a[to[i]]);
    for(int l=2;l<=n;l<<=1){
        cp bas=cp(cos(2*pi/l),flag*sin(2*pi/l));
        int k=l>>1;
        for(int i=0;i<n;i+=l){
            cp mul=cp(1,0);
            for(int j=i;j<i+k;j++){
                cp val=mul*a[j+k];
                a[j+k]=a[j]-val,a[j]=a[j]+val;
                mul=mul*bas;
            }
        }
    }
    if(flag===-1){
        for(int i=0;i<n;i++) a[i].a/=n,a[i].b/=n;
    }
}

```

6.3.2 ntt

```

int gg=3,invg=ksm(gg);
int to[maxn<<3];
vector<int> ntt(vector<int> a,int flag){
    int n=a.size();

```

```

for(int i=0;i<n;i++)if(i<to[i])swap(a[i],a[to[i]]);
for(int len=2;len<=n;len<<=1){
    int bas=ksm(flag==1?gg:invg,(mod-1)/len),l=len>>1;
    for(int i=0;i<n;i+=len){
        int mul=1;
        for(int j=i;j<i+l;j++){
            int val=mul*a[j+1]%mod;
            inc(a[j+1]=a[j],mod-val);
            inc(a[j],val);
            mul=mul*bas%mod;
        }
    }
}
if(flag===-1){
    int inv=ksm(n);
    for(int i=0;i<n;i++)a[i]=a[i]*inv%mod;
}
return a;
}

vector<int> mul(vector<int> a,vector<int> b){
    int n=a.size()-1,m=b.size()-1,int k=1;
    while(k<n+m+1)k<<=1;
    vector<int> f(k),g(k);
    for(int i=0;i<=n;i++)f[i]=a[i];
    for(int i=0;i<=m;i++)g[i]=b[i];
    for(int i=0;i<k;i++)to[i]=(to[i>>1]>>1)|((i&1)?(k>>1):0);
    f=ntt(f,1),g=ntt(g,1);
    for(int i=0;i<k;i++)f[i]=f[i]*g[i]%mod;
    f=ntt(f,-1);f.resize(n+m+1);
    return f;
}

```

6.3.3 mtt

```

const int B=(1<<15)-1;
int calc(db x){return (long long)(x+0.5)%_mod;}
vector<int> mul(vector<int> a,vector<int> b){
    int n=a.size()-1,m=b.size()-1,k=1;
    while(k<n+m+1)k<<=1;
    for(int i=0;i<k;i++)to[i]=(to[i>>1]>>1)|((i&1)?(k>>1):0);
    vector<cp> f(k),g(k),h(k);
    for(int i=0;i<=n;i++)f[i]=cp(a[i]&B,0),g[i]=cp(a[i]>>15,0);
    for(int i=0;i<=m;i++)h[i]=cp(b[i]&B,b[i]>>15);
    fft(f,1),fft(g,1),fft(h,1);
    for(int i=0;i<k;i++)f[i]=f[i]*h[i],g[i]=g[i]*h[i];
    fft(f,-1),fft(g,-1);
    vector<int> ans(n+m+1);
    for(int i=0;i<=n+m;i++)ans[i]=(111*calc(f[i].a)+(111*(calc(f[i].b)+calc(g[i].a))<<1511)%_mod
        +(111*calc(g[i].b)<<3011)%_mod)%_mod;
    return ans;
}

```

6.3.4 ni ln exp

```

vector<int> f,g;

```

```

void cdqni(int l,int r){
    if(r-l+1<=64){
        for(int i=l;i<=r;i++){
            for(int j=1;j<i;j++)inc(g[i],111*g[j]*f[i-j]%mod);
            g[i]=111*(mod-g[i])*g[0]%mod;
        }
        return ;
    }
    if(l==r){g[1]=111*(mod-g[1])*g[0]%mod;return ;}
    int mid=l+r>>1;
    cdqni(l,mid);
    vector<int> ff(mid-l+1),gg(r-l+1);
    for(int i=1;i<=mid;i++)ff[i-1]=g[i];
    for(int i=0;i<=r-l;i++)gg[i]=f[i];
    ff=poly::mul(ff,gg);
    for(int i=mid+1;i<=r;i++)inc(g[i],ff[i-1]);
    cdqni(mid+1,r);
}

vector<int> ni(vector<int> a){
    int n=a.size()-1;
    f.resize(n+1),g.resize(n+1);
    for(int i=0;i<=n;i++)f[i]=a[i],g[i]=0;
    g[0]=ksm(f[0]);for(int i=1;i<=n;i++)inc(g[i],111*g[0]*f[i]%mod);
    cdqni(1,n);
    return g;
}

void cdqln(int l,int r){
    if(r-l+1<=64){
        for(int i=l;i<=r;i++){
            for(int j=1;j<i;j++)inc(g[i],111*g[j]*j%mod*f[i-j]%mod);
            g[i]=111*::ni[i]*(111*f[i]*i%mod-g[i]+mod)%mod;
        }
        return ;
    }
    if(l==r){g[1]=111*::ni[1]*(111*f[1]*1%mod-g[1]+mod)%mod;return ;}
    int mid=l+r>>1;
    cdqln(l,mid);
    vector<int> ff(mid-l+1),gg(r-l+1);
    for(int i=1;i<=mid;i++)ff[i-1]=111*g[i]*i%mod;
    for(int i=0;i<=r-l;i++)gg[i]=f[i];
    ff=poly::mul(ff,gg);
    for(int i=mid+1;i<=r;i++)inc(g[i],ff[i-1]);
    cdqln(mid+1,r);
}

vector<int> ln(vector<int> a){
    int n=a.size()-1;
    f.resize(n+1);g.resize(n+1);
    for(int i=0;i<=n;i++)f[i]=a[i],g[i]=0;
    f[0]=1,g[0]=0;cdqln(1,n);
    return g;
}

void cdqexp(int l,int r){
    if(r-l+1<=64){
        for(int i=l;i<=r;i++){
            for(int j=1;j<i;j++)inc(g[i],111*g[j]*f[i-j]%mod);
            g[i]=111*::ni[i]*g[i]%mod;
        }
    }
}

```

```

        return ;
    }
    if(l==r){g[1]=111*::ni[l]*g[1]%mod;return ;}
    int mid=l+r>>1;
    cdqexp(l,mid);
    vector<int> ff(mid-l+1),gg(r-l+1);
    for(int i=l;i<=mid;i++)ff[i-1]=g[i];
    for(int i=0;i<=r-l;i++)gg[i]=f[i];
    ff=poly::mul(ff,gg);
    for(int i=mid+1;i<=r;i++)inc(g[i],ff[i-1]);
    cdqexp(mid+1,r);
}
vector<int> exp(vector<int> a){
    int n=a.size()-1;
    f.resize(n+1);g.resize(n+1);
    for(int i=0;i<=n;i++)f[i]=11*a[i]*i%mod,g[i]=0;
    f[0]=0,g[0]=1;cdqexp(0,n);
    return g;
}

```

6.4 另一份 poly

```

const int maxn = (1 << 22) + 50;
const int mod = 998244353, G = 3;

inline void fix(int &x) {
    x += ((x >> 31) & mod);
}

inline int qpow(int b, int p) {
    int res = 1;
    while (p) {
        if (p & 1) {
            res = 1ULL * res * b % mod;
        }
        b = 1ULL * b * b % mod;
        p >>= 1;
    }
    return res;
}

typedef vector<int> poly;

int r[maxn];

inline void NTT(poly &a, int o) {
    int n = (int)a.size();
    for (int i = 0; i < n; ++i) {
        if (i < r[i]) {
            swap(a[i], a[r[i]]);
        }
    }
    static int pw[maxn];
    for (int k = 1; k < n; k <= 1) {
        int wn = qpow(o ? G : qpow(G, mod - 2), (mod - 1) / (k << 1));
        pw[0] = 1;

```

```

        for (int i = 1; i < k; ++i) {
            pw[i] = 1ULL * pw[i - 1] * wn % mod;
        }
        for (int i = 0; i < n; i += (k << 1)) {
            for (int j = 0; j < k; ++j) {
                int x = a[i + j], y = 1ULL * a[i + j + k] * pw[j] % mod;
                fix(a[i + j] = x + y - mod);
                fix(a[i + j + k] = x - y);
            }
        }
    }
    if (!o) {
        int inv = qpow(n, mod - 2);
        for (int i = 0; i < n; ++i) {
            a[i] = 1ULL * a[i] * inv % mod;
        }
    }
}

inline poly operator * (poly a, poly b) {
    NTT(a, 1);
    NTT(b, 1);
    int n = (int)a.size();
    for (int i = 0; i < n; ++i) {
        a[i] = 1ULL * a[i] * b[i] % mod;
    }
    NTT(a, 0);
    return a;
}

inline poly mul(poly a, poly b) {
    int n = (int)a.size() - 1, m = (int)b.size() - 1, k = 0;
    if (min(n, m) <= 100) {
        poly res(n + m + 1);
        for (int i = 0; i <= n + m; ++i) {
            ull x = 0;
            for (int j = max(i - m, 0); j <= i && j <= n; ++j) {
                x += 1ULL * a[j] * b[i - j];
                if (!(j & 15)) {
                    x %= mod;
                }
            }
            res[i] = x % mod;
        }
        return res;
    }
    while ((1 << k) <= n + m) {
        ++k;
    }
    for (int i = 1; i < (1 << k); ++i) {
        r[i] = (r[i >> 1] >> 1) | ((i & 1) << (k - 1));
    }
    poly A(1 << k), B(1 << k);
    for (int i = 0; i <= n; ++i) {
        A[i] = a[i];
    }
    for (int i = 0; i <= m; ++i) {

```

```

        B[i] = b[i];
    }
    poly res = A * B;
    res.resize(n + m + 1);
    return res;
}

poly inv(poly &a, int m) {
    if (m == 1) {
        return poly(1, qpow(a[0], mod - 2));
    }
    poly b = inv(a, m >> 1), c(m), res(m);
    for (int i = 0; i < m; ++i) {
        c[i] = a[i];
        if (i < (m >> 1)) {
            res[i] = b[i] * 2 % mod;
        }
    }
    c = mul(c, mul(b, b));
    for (int i = 0; i < m; ++i) {
        fix(res[i] -= c[i]);
    }
    return res;
}

inline poly inv(poly a) {
    int n = (int)a.size() - 1;
    int t = __lg(n + 1);
    if ((1 << t) < n + 1) {
        ++t;
    }
    poly b(1 << t);
    for (int i = 0; i <= n; ++i) {
        b[i] = a[i];
    }
    b = inv(b, 1 << t);
    b.resize(n + 1);
    return b;
}

inline poly der(poly a) {
    int n = (int)a.size() - 1;
    poly res(n);
    for (int i = 1; i <= n; ++i) {
        res[i - 1] = 1ULL * a[i] * i % mod;
    }
    return res;
}

inline poly itg(poly a) {
    int n = (int)a.size() - 1;
    poly res(n + 2), I(n + 2);
    I[1] = 1;
    for (int i = 2; i <= n + 1; ++i) {
        I[i] = 1ULL * (mod - mod / i) * I[mod % i] % mod;
    }
    for (int i = 1; i <= n + 1; ++i) {

```

```

        res[i] = 1ULL * a[i - 1] * I[i] % mod;
    }
    return res;
}

inline poly ln(poly a) {
    int n = (int)a.size() - 1;
    poly res = itg(mul(der(a), inv(a)));
    res.resize(n + 1);
    return res;
}

poly exp(poly &a, int m) {
    if (m == 1) {
        poly res(1, 1);
        return res;
    }
    poly b = exp(a, m >> 1);
    b.resize(m);
    poly c = ln(b), d(m);
    for (int i = 0; i < m; ++i) {
        fix(d[i] = a[i] - c[i]);
    }
    fix(d[0] += 1 - mod);
    b.resize(m >> 1);
    poly res = mul(b, d);
    res.resize(m);
    return res;
}

inline poly exp(poly a) {
    int n = (int)a.size() - 1;
    int t = __lg(n + 1);
    if ((1 << t) < n + 1) {
        ++t;
    }
    poly b(1 << t);
    for (int i = 0; i <= n; ++i) {
        b[i] = a[i];
    }
    b = exp(b, 1 << t);
    b.resize(n + 1);
    return b;
}

inline poly pmod(poly a, poly b) {
    int n = (int)a.size() - 1, m = (int)b.size() - 1;
    if (n < m) {
        return a;
    }
    poly c = a, d = b;
    reverse(c.begin(), c.end());
    c.resize(n - m + 1);
    reverse(d.begin(), d.end());
    d.resize(n - m + 1);
    poly q = mul(c, inv(d));
    q.resize(n - m + 1);
}

```

```

reverse(q.begin(), q.end());
q = mul(q, b);
poly res(m);
for (int i = 0; i < m; ++i) {
    fix(res[i] = a[i] - q[i]);
}
return res;
}

inline poly multipoint(poly a, poly b) {
    int n = (int)a.size() - 1, m = (int)b.size() - 1;
    vector<poly> F(1 << (_lg(m + 1) + 2));
    auto dfs = [&](auto &self, int rt, int l, int r) -> void {
        if (l == r) {
            F[rt] = poly(2);
            fix(F[rt][0] = -b[1]);
            F[rt][1] = 1;
            return;
        }
        int mid = (l + r) >> 1;
        self(self, rt << 1, l, mid);
        self(self, rt << 1 | 1, mid + 1, r);
        F[rt] = mul(F[rt << 1], F[rt << 1 | 1]);
    };
    dfs(dfs, 1, 0, m);
    for (int i = 1; i < (int)F.size(); ++i) {
        reverse(F[i].begin(), F[i].end());
    }
    reverse(a.begin(), a.end());
    F[1].resize(n);
    poly q = mul(a, inv(F[1])), ans(m + 1);
    q.resize(n);
    auto dfs2 = [&](auto &self, int rt, int l, int r, poly A) -> void {
        if ((int)A.size() > r - l + 1) {
            A.erase(A.begin(), A.end() - (r - l + 1));
        }
        if (l == r) {
            ans[l] = (a[n] + 1ULL * A[0] * b[l]) % mod;
            return;
        }
        int mid = (l + r) >> 1;
        poly a = mul(A, F[rt << 1 | 1]);
        a.resize(A.size());
        self(self, rt << 1, l, mid, a);
        a = mul(A, F[rt << 1]);
        a.resize(A.size());
        self(self, rt << 1 | 1, mid + 1, r, a);
    };
    dfs2(dfs2, 1, 0, m, q);
    return ans;
}

inline int recur(int m, poly a, poly b) {
    int n = (int)a.size();
    if (m < n) {
        return a[m];
    }
}

```

```

poly f(n + 1), g(2);
g[1] = f[n] = 1;
for (int i = 0; i < n; ++i) {
    fix(f[i] = -b[n - i]);
}
auto dfs = [&](auto &self, int k) -> poly {
    if (k < n) {
        poly res(k + 1);
        res[k] = 1;
        return res;
    }
    auto res = self(self, k >> 1);
    res = mul(res, res);
    if (k & 1) {
        res.insert(res.begin(), 0);
    }
    res = pmod(res, f);
    return res;
};
auto res = dfs(dfs, m);
int ans = 0;
for (int i = 0; i < n; ++i) {
    fix(ans += 1ULL * a[i] * res[i] % mod - mod);
}
return ans;
}

```

6.4.1 多点求值

6.5 集合幂级数

```

void fmt(int *a,int n,int w=1){
    for(int i=0;i<n;i++){
        for(int s=0;s<(1<<n);s++)if(s&(1<<i))(a[s]+=a[s^(1<<i)]*w)%=mod;
    }
}
int ff[maxn+1][1<<maxn],gg[maxn+1][1<<maxn],hh[1<<maxn],ni[maxn+1];
void xormul(int *a,int *b,int *c,int n){
    for(int i=0;i<=n;i++){
        for(int s=0;s<(1<<n);s++)ff[i][s]=gg[i][s]=0;
    }
    for(int s=0;s<(1<<n);s++)ff[_builtin_popcount(s)][s]=a[s];
    for(int s=0;s<(1<<n);s++)gg[_builtin_popcount(s)][s]=b[s];
    for(int i=0;i<=n;i++)fmt(ff[i],n,1);
    for(int i=0;i<=n;i++)fmt(gg[i],n,1);
    for(int s=0;s<(1<<n);s++){
        for(int i=0;i<=n;i++){
            hh[i]=0;
            for(int j=0;j<=i;j++) (hh[i]+=ff[j][s]*gg[i-j][s])%=mod;
        }
        for(int i=0;i<=n;i++)ff[i][s]=hh[i];
    }
/*ln
    for(int s=0;s<(1<<n);s++){
        for(int i=0;i<n;i++){
            hh[i]=ff[i+1][s]*(i+1)%mod;
        }
    }
*/
}

```

```
        for(int j=1;j<=i;j++) (hh[i]+=mod-ff[j][s]*hh[i-j]%mod)%=mod;
    }
    for(int i=1;i<=n;i++) ff[i][s]=hh[i-1]*ni[i]%mod;
}
*/
/*exp
for(int s=0;s<(1<<n);s++){
    for(int i=0;i<n;i++){
        hh[i]=ff[i+1][s]*(i+1)%mod;
        for(int j=1;j<=i;j++) (hh[i]+=ff[j][s]*j%mod*hh[i-j]%mod*ni[i-j+1])%=mod;
    }
    for(int i=1;i<=n;i++) ff[i][s]=hh[i-1]*ni[i]%mod;
}
*/
for(int i=0;i<=n;i++) fmt(ff[i],n,mod-1);
for(int s=0;s<(1<<n);s++) c[s]=ff[__builtin_popcount(s)][s];
}
```

6.5.1 FWT

6.5.2 子集卷积

6.5.3 多项式复合集合幂级数

6.6 杂项

6.6.1 插值

7 string

7.1 Manacher

```

for (int i = 1; i <= n; ++i) {
    s[i * 2 - 1] = '#';
    s[i * 2] = t[i];
}
s[n * 2 + 1] = '#';
int ans = 0;
for (int i = 1, mid = 0, r = 0; i <= n * 2 + 1; ++i) {
    if (i <= r) {
        f[i] = min(f[mid * 2 - i], r - i + 1);
    }
    while (i + f[i] <= n * 2 + 1 && i - f[i] >= 1 && s[i + f[i]] == s[i - f[i]]) {
        ++f[i];
    }
    if (i + f[i] - 1 > r) {
        mid = i;
        r = i + f[i] - 1;
    }
    ans = max(ans, f[i] - 1);
}

```

7.2 Z 函数

```

z[1] = n;
for (int i = 2, l = 0, r = 0; i <= n; ++i) {
    if (i <= r) {
        z[i] = min(z[i - l + 1], r - i + 1);
    }
    while (i + z[i] <= n && s[z[i] + 1] == s[i + z[i]]) {
        ++z[i];
    }
    if (i + z[i] - 1 > r) {
        l = i;
        r = i + z[i] - 1;
    }
}

```

7.3 Runs

```

const int maxn = 1000100;
const int logn = 22;

int n, m, sa[maxn], id[maxn], old[maxn << 1], h[maxn], cnt[maxn];
char s[maxn];
pii p[maxn];

struct SA {
    int f[logn][maxn], rk[maxn];
    char t[maxn];

    inline int qmin(int l, int r) {

```

```

    int k = __lg(r - l + 1);
    return min(f[k][l], f[k][r - (1 << k) + 1]);
}

inline int lcp(int x, int y) {
    if (x == y) {
        return n - x + 1;
    }
    if (t[x] != t[y]) {
        return 0;
    }
    if (t[x + 1] != t[y + 1]) {
        return 1;
    }
    if (t[x + 2] != t[y + 2]) {
        return 2;
    }
    x = rk[x];
    y = rk[y];
    if (x > y) {
        swap(x, y);
    }
    return qmin(x + 1, y);
}

inline void build() {
    int m = 127;
    for (int i = 1; i <= m; ++i) {
        cnt[i] = 0;
    }
    for (int i = 1; i <= n; ++i) {
        rk[i] = s[i];
        t[i] = s[i];
        ++cnt[rk[i]];
    }
    for (int i = 1; i <= m; ++i) {
        cnt[i] += cnt[i - 1];
    }
    for (int i = n; i; --i) {
        sa[cnt[rk[i]]--] = i;
    }
    for (int w = 1;; w <= 1) {
        int tot = 0;
        for (int i = n - w + 1; i <= n; ++i) {
            id[++tot] = i;
        }
        for (int i = 1; i <= n; ++i) {
            old[i] = rk[i];
            if (sa[i] > w) {
                id[++tot] = sa[i] - w;
            }
        }
        for (int i = 1; i <= m; ++i) {
            cnt[i] = 0;
        }
        for (int i = 1; i <= n; ++i) {
            ++cnt[rk[id[i]]];
        }
    }
}

```

```

    }
    for (int i = 1; i <= m; ++i) {
        cnt[i] += cnt[i - 1];
    }
    for (int i = n; i; --i) {
        sa[cnt[rk[id[i]]]--] = id[i];
    }
    int p = 0;
    for (int i = 1; i <= n; ++i) {
        if (old[sa[i]] == old[sa[i - 1]] && old[sa[i] + w] == old[sa[i - 1] + w]) {
            rk[sa[i]] = p;
        } else {
            rk[sa[i]] = ++p;
        }
    }
    if (p == n) {
        break;
    }
    m = p;
}
h[1] = 0;
for (int i = 1, k = 0; i <= n; ++i) {
    if (rk[i] == 1) {
        continue;
    }
    if (k) {
        --k;
    }
    while (i + k <= n && sa[rk[i] - 1] + k <= n && s[i + k] == s[sa[rk[i] - 1] + k]) {
        ++k;
    }
    h[rk[i]] = k;
}
for (int i = 1; i <= n; ++i) {
    f[0][i] = h[i];
}
for (int j = 1; (1 << j) <= n; ++j) {
    for (int i = 1; i + (1 << j) - 1 <= n; ++i) {
        f[j][i] = min(f[j - 1][i], f[j - 1][i + (1 << (j - 1))]);
    }
}
}
}

A, B;

inline int lcp(int x, int y) {
    return A.lcp(x, y);
}

inline int lcs(int x, int y) {
    return B.lcp(n - x + 1, n - y + 1);
}

struct node {
    int l, r, k;
    node(int a = 0, int b = 0, int c = 0) : l(a), r(b), k(c) {}
} a[maxn << 1], b[maxn << 1];
}

```

```

void runs() {
    n = reads(s + 1);
    A.build();
    reverse(s + 1, s + n + 1);
    B.build();
    int m = 0;
    for (int k = 1; k * 2 <= n; ++k) {
        int tot = 0;
        for (int i = k + 1; i + k - 1 <= n; i += k) {
            int l = max(i - k, i - lcs(i - 1, i + k - 1)), r = min(i - 1, i + lcp(i, i + k) - k);
            if (l <= r) {
                if (!tot) {
                    p[++tot] = mkp(l, r);
                } else {
                    if (p[tot].scd == l - 1) {
                        p[tot].scd = r;
                    } else {
                        p[++tot] = mkp(l, r);
                    }
                }
            }
        }
        for (int i = 1; i <= tot; ++i) {
            int l = p[i].fst, r = p[i].scd;
            a[++m] = node(l, r + k * 2 - 1, k);
        }
    }
    sort(a + 1, a + m + 1, [&](const node &a, const node &b) {
        return a.l < b.l || (a.l == b.l && (a.r < b.r || (a.r == b.r && a.k < b.k)));
    });
    int tot = 0;
    for (int i = 1; i <= m; ++i) {
        if (!tot || !(a[i].l == b[tot].l && a[i].r == b[tot].r)) {
            b[++tot] = a[i];
        }
    }
}

```

7.4 后缀自动机

```

struct SAM {
    int lst, tot, fa[maxn], ch[maxn][26], len[maxn];

    inline void init() {
        lst = tot = 1;
    }

    inline void insert(int k, int c) {
        int u = ++tot, p = lst;
        sz[u] = 1;
        lst = u;
        len[u] = k;
        for (; p && !ch[p][c]; p = fa[p]) {
            ch[p][c] = u;
        }
        if (!p) {

```

```
    fa[u] = 1;
    return;
}
int q = ch[p][c];
if (len[q] == len[p] + 1) {
    fa[u] = q;
    return;
}
int nq = ++tot;
len[nq] = len[p] + 1;
fa[nq] = fa[q];
memcpy(ch[nq], ch[q], sizeof(ch[q]));
fa[u] = fa[q] = nq;
for (; p && ch[p][c] == q; p = fa[p]) {
    ch[p][c] = nq;
}
}
} sam;
```