

# yhddd 的 icpc 模板

yhddd

2025 年 12 月 23 日

ver: 1.0.0

# 目录

<b>1</b>	<b>杂项</b>	<b>4</b>
1.1	fastio . . . . .	4
1.2	取模优化 . . . . .	4
1.3	i24 . . . . .	4
<b>2</b>	<b>dp</b>	<b>5</b>
2.1	决策单调性 . . . . .	5
2.1.1	SMAWK . . . . .	5
2.1.2	LARSCH . . . . .	6
<b>3</b>	<b>ds</b>	<b>7</b>
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	KT <sub>T</sub> . . . . .	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
<b>4</b>	<b>graph</b>	<b>10</b>
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
4.2	连通性 . . . . .	12
4.2.1	边双 . . . . .	12
4.2.2	点双 . . . . .	12
4.2.3	双极定向 . . . . .	13
4.2.4	广义串并联图 . . . . .	14
4.3	流 . . . . .	15
4.3.1	预留推进 . . . . .	15
4.3.2	原始对偶 . . . . .	15
4.3.3	最小割树 . . . . .	15
4.3.4	一般图最大匹配 . . . . .	15
4.4	杂项 . . . . .	15
4.4.1	欧拉回路 . . . . .	15
4.4.2	四元环计数 . . . . .	15
<b>5</b>	<b>geometry</b>	<b>16</b>

<b>6 math</b>	<b>19</b>
6.1 篩 . . . . .	19
6.2 矩阵 . . . . .	19
6.2.1 高斯消元 . . . . .	19
6.2.2 矩阵求逆 . . . . .	19
6.2.3 行列式 . . . . .	19
6.2.4 特征多项式 . . . . .	19
6.3 poly . . . . .	19
6.3.1 fft . . . . .	19
6.3.2 ntt . . . . .	19
6.3.3 mtt . . . . .	20
6.3.4 ni ln exp . . . . .	20
6.3.5 多点求值 . . . . .	22
6.4 集合幂级数 . . . . .	22
6.4.1 FWT . . . . .	22
6.4.2 子集卷积 . . . . .	22
6.4.3 多项式复合集合幂级数 . . . . .	22
6.5 杂项 . . . . .	22
6.5.1 插值 . . . . .	22
<b>7 string</b>	<b>23</b>

# 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

### 4.2 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

### 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

### 6.2 矩阵

#### 6.2.1 高斯消元

#### 6.2.2 矩阵求逆

#### 6.2.3 行列式

#### 6.2.4 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+1;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]=(1ll*calc(f[i].a)+(1ll*(calc(f[i].b)+calc(g[i].a))<<1511)%_mod
        +(1ll*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-1;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=1;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-1;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=1;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[1]*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=1;i<=mid;i++)ff[i-1]=g[i];
for(int i=0;i<=r-1;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]=1ll*a[i]*i%mod,g[i]=0;
    f[0]=0,g[0]=1;cdqexp(0,n);
    return g;
}
```

### 6.3.5 多点求值

## 6.4 集合幂级数

### 6.4.1 FWT

### 6.4.2 子集卷积

### 6.4.3 多项式复合集合幂级数

## 6.5 杂项

### 6.5.1 插值

**7 string**

???