ACM Templates

初始化 / Front Matters

```cpp
#include<bits/stdc++.h>
using namespace std;

typedef long long ll;
typedef unsigned int ui;
typedef unsigned long long ull;
typedef pair<int,int> pint;
typedef long double ld;
typedef __int128 i128;

const int N=299213;
const int INF=0x3f3f3f3f;
const ll INF_LL=0x3f3f3f3f3f3f3f3f;
const int primeTable[]={929292929, 299213, 19260817, 991145149};
```

图论 / Graph Theroy

```cpp
namespace _TwoSAT{
    // 两倍空间
    vector<int> a[N*2];
    stack<int> s; bool mark[N*2];

    // x==valX || y==valY
    void AddClause(int x,bool vx,int y,bool vy){
        x=x*2+vx, y=y*2+vy;
        a[x^1].push_back(y);
        a[y^1].push_back(x);
    }

    void Init(){
        for(int i=0;i<N*2;i++)
            a[i].clear();
        s=stack<int>();
        memset(mark,0,sizeof(mark));
    }

    bool DFS(int u){
        if(mark[u^1])return 0;
        if(mark[u])return 1;
```

```
        s.push(u); mark[u]=1;
        for(int i=0;i<a[u].size();i++)
            if(!DFS(a[u][i]))return 0;
        return 1;
    }

    // 下标从0开始
    bool Solve(int n){
        for(int i=0;i<n*2;i++)
            if(!mark[i]&&!mark[i^1])
                if(!DFS(i)){
                    while(s.top()!=i)
                        mark[s.top()]=0, s.pop();
                    mark[i]=0, s.pop();
                    if(!DFS(i^1))return 0;
                }
        return 1;
    }
};
```

网络流 / Network Flow

```
namespace _MaxFlow{
    struct Dinic{
        struct Edge{int v,res;};
        vector<Edge> edg;
        vector<int> a[N*2];
        int st,ed;

        void AddEdge(int u,int v,int cap){
            edg.push_back((Edge){v,cap});
            edg.push_back((Edge){u,0});
            int siz=edg.size();
            a[u].push_back(siz-2);
            a[v].push_back(siz-1);
        }

        int dep[N*2];
        bool BFS(){
            memset(dep,-1,sizeof(dep));
            dep[st]=0;
            queue<int> q; q.push(st);

            while(!q.empty()){
                int u=q.front(); q.pop();
                for(int i=0;i<a[u].size();i++){
                    Edge& e=edg[a[u][i]];
                    if(dep[e.v]==-1&&e.res>0){
                        q.push(e.v), dep[e.v]=dep[u]+1;
```

```
                    }
                }
            }

            return dep[ed]!=-1;
        }

        int cur[N*2];
        int DFS(int u,int minF){
            if(u==ed||minF==0)return minF;

            int tmpF,sumF=0;
            for(int& i=cur[u];i<a[u].size();i++){
                Edge& e=edg[a[u][i]];
                if( dep[e.v]==dep[u]+1 &&
(tmpF=DFS(e.v,min(e.res,minF)))>0 ){
                    e.res-=tmpF; edg[a[u][i]^1].res+=tmpF;
                    sumF+=tmpF; minF-=tmpF;
                }
                if(minF==0)break;
            }

            return sumF;
        }

        int MaxFlow(){
            int ret=0;
            while(BFS()){
                memset(cur,0,sizeof(cur));
                ret+=DFS(st,INF);
            }
            return ret;
        }
    };
};

namespace _MaxFlowMinCost{
    struct Dinic{
        struct Edge{int v,w,res;};
        vector<Edge> edg;
        vector<int> a[N*2];
        int st,ed;

        void AddEdge(int u,int v,int w,int cap){
            edg.push_back((Edge){v,w,cap});
            edg.push_back((Edge){u,-w,0});
            int siz=edg.size();
            a[u].push_back(siz-2);
            a[v].push_back(siz-1);
        }

```

```
78          int dis[N*2],pa[N*2],det[N*2];
79          bool SPFA(){
80              static int inQ[N*2];
81              memset(inQ,0,sizeof(inQ));
82              memset(dis,0x3f,sizeof(dis));
83              deque<int> q;q.push_back(st);
84              dis[st]=0, inQ[st]=1, det[st]=INF, pa[st]=-1;
85
86              while(!q.empty()){
87                  int u=q.front();q.pop_front();inQ[u]=0;
88                  for(int i=0;i<a[u].size();i++){
89                      Edge &e=edg[a[u][i]];
90                      if(e.res>0 && dis[e.v]>dis[u]+e.w){
91                          dis[e.v]=dis[u]+e.w;
92                          det[e.v]=min(det[u],e.res);
93                          pa[e.v]=a[u][i];
94                          if(!inQ[e.v]){
95                              if(!q.empty() &&
    dis[q.front()]>=dis[e.v])q.push_front(e.v);
96                              else q.push_back(e.v);
97                              inQ[e.v]=1;
98                          }
99                      }
100                 }
101             }
102
103             return dis[ed]!=INF;
104         }
105
106         void Augment(int &w){
107             w+=dis[ed]*det[ed];
108             int u=ed;
109             while(u!=st){
110                 edg[pa[u]].res-=det[ed];
111                 edg[pa[u]^1].res+=det[ed];
112                 u=edg[pa[u]^1].v;
113             }
114         }
115
116         int MaxFlowMinCost(){
117             int ret=0;
118             while(SPFA())Augment(ret);
119             return ret;
120         }
121     };
122 };
123
124 namespace _Hungary{
125     int g[N][N];
126
127     bool vst[N]; int lnk[N];
```

```
128
129     bool DFS(int u,int n){
130         for(int v=1;v<=n;v++)
131             if(g[u][v] && !vst[v]){
132                 vst[v]=1;
133                 if(!lnk[v] || DFS(lnk[v],n)){
134                     lnk[v]=u;
135                     return 1;
136                 }
137             }
138
139         return 0;
140     }
141
142     int Match(int n){
143         int ans=0;
144         for(int i=1;i<=n;i++){
145             memset(vst,0,sizeof(vst));
146             if(DFS(i,n))ans++;
147         }
148         return ans;
149     }
150 };
151
152 // DEBUG:
153 namespace _MinVertexCover{
154     int lnk[N];
155     bool inS[N],inT[N];
156
157     bool DFS(int u,int n){
158         inS[u]=1;
159         for(auto v:a[u]){
160             if(inT[v])continue;
161             inT[v]=1;
162             if(!lnk[v]||DFS(lnk[v],n)){
163                 lnk[v]=u;
164                 return 1;
165             }
166         }
167         return 0;
168     }
169
170     void Hungary(int n){
171         static bool isMatch[N];
172         for(int i=1;i<=n;i++){
173             fill(inT+1,inT+n+1,0);
174             if(DFS(i,n))isMatch[i]=1;
175         }
176
177         fill(inS+1,inS+n+1,0);
178         fill(inT+1,inT+n+1,0);
```

```
179
180         vector<int> ans;
181         for(int i=1;i<=n;i++)
182             if(!isMatch[i])DFS(i,n);
183         for(int i=1;i<=n;i++)
184             if(__builtin_parity(b[i])==1 && inS[i])
185                 ans.push_back(b[i]);
186         for(int i=1;i<=n;i++)
187             if(__builtin_parity(b[i])==0 && !inT[i])
188                 ans.push_back(b[i]);
189 
190         cout<<ans.size()<<endl;
191         for(auto i:ans)
192             cout<<i<<' ';
193     }
194 };
195 
196 namespace _KM{
197     int g[N][N];
198 
199     int lx[N],ly[N],lnk[N],slack[N];
200     bool inS[N],inT[N];
201 
202     bool DFS(int u,int n){
203         inS[u]=1;
204         for(int v=1;v<=n;v++){
205             if(inT[v])continue;
206             int w=lx[u]+ly[v]-g[u][v];
207             if(w==0){
208                 inT[v]=1;
209                 if(!lnk[v] || DFS(lnk[v],n)){
210                     lnk[v]=u;
211                     return 1;
212                 }
213             }else slack[v]=min(slack[v],w);
214         }
215         return 0;
216     }
217 
218     void Update(int n){
219         int det=INF;
220         for(int i=1;i<=n;i++)
221             if(!inT[i])
222                 det=min(det,slack[i]);
223         for(int i=1;i<=n;i++){
224             if(inS[i])lx[i]-=det;
225             if(inT[i])ly[i]+=det;
226             else slack[i]-=det;
227         }
228     }
229 
```

```
    // n是两侧点的最大值，从1标号
    void KM(int n){
        for(int i=1;i<=n;i++){
            lx[i]=-INF;
            for(int j=1;j<=n;j++)
                lx[i]=max(lx[i],g[i][j]);
        }
        for(int i=1;i<=n;i++){
            memset(slack,0x3f,sizeof(slack));
            while(1){
                memset(inS,0,sizeof(inS));
                memset(inT,0,sizeof(inT));
                if(DFS(i,n))break;
                else Update(n);
            }
        }
    }
};
```

字符串 / String Algorithm

```
// FIXME: 修改成不用string的版本
namespace _KMP{
    int Init(int f[],string s){
        f[0]=f[1]=0;
        for(int i=1;i<s.length();i++) {
            int j=f[i];
            while(j&&s[i]!=s[j])j=f[j];
            f[i+1] = (s[i]==s[j])?j+1:0;
        }
    }
    int Query(string s,string t,int f[]){
        int cnt=0,j=0;
        for(int i=0;i<s.length();i++){
            while(j&&s[i]!=t[j])j=f[j];
            if(s[i]==t[j])j++;
            if(j==t.length())cnt++;
        }
        return cnt;
    }
};

namespace _AhoCorasick{
    int ch[N][C],f[N],pre[N];       // f=fail, pre=pre_end
    bool isEnd[N]; int idx;

    void Init(){
        memset(ch,0,sizeof(ch));
        memset(f,0,sizeof(f));
```

```
        memset(pre,0,sizeof(pre));
        memset(isEnd,0,sizeof(isEnd));
        idx=0;
    }

    void Build(char s[],int n){
        int o=0;
        for(int i=0;i<n;i++){
            int c=s[i];
            if(ch[o][c])o=ch[o][c];
            else o=ch[o][c]=++idx;
        }
        isEnd[o]=1;
    }

    void GetFail(){
        queue<int> q;
        for(int i=0;i<C;i++)
            if(ch[0][i])q.push(ch[0][i]);

        while(!q.empty()){
            int h=q.front(); q.pop();
            for(int i=0;i<C;i++){
                int &u=ch[h][i], j=f[h];
                if(!u){
                    u=ch[j][i];
                    continue;
                }
                q.push(u);
                while(j&&!ch[j][i])j=f[j];
                f[u]=ch[j][i];
                pre[u] = isEnd[f[u]]?f[u]:pre[f[u]];
            }
        }
    }

    // 查询的时候如果isEnd=1，则要不断向上找pre
};

// FIXME: 没用过，需要看看会不会出锅
namespace _Manacher{
    ll Manacher(char t[],int n){
        static char s[2*N];
        static int cnt[2*N],f[2*N];

        for(int i=0;i<=n*2;i++)
            cnt[i]=f[i]=0;

        for(int i=1;i<=n;i++){
            s[i*2-1]=t[i-1];
            s[i*2]=1;
```

```
        }
        s[0]=2,s[n*2]=3;

        int cur=f[0]=0,idx=0;
        for(int i=1;i<2*n;i++){
            int& j=f[i]; j=0;
            if(cur-i>=0&&2*idx-i>=0)j=min(f[2*idx-i],cur-i);
            while(s[i-j-1]==s[i+j+1])j++;
            if(i+j>cur)cur=i+j,idx=i;
            //ans=max(ans,(j*2+1)/2);

            cnt[max(0,i-j)]++;
            cnt[i+1]--;
        }

        ll ret=0;
        for(int i=1;i<=2*n;i++){
            cnt[i]+=cnt[i-1];
            if(i&1)ret+=cnt[i];
        }

        return ret;
    }
};

// SAM空间开两倍
namespace _SAM{
    int ch[N*2][C],pa[N*2],len[N*2],siz[N*2];
    int idx=1,pre=1;

    void Insert(int x){
        int p=pre,np=++idx;pre=np;
        siz[np]=1; len[np]=len[p]+1;
        for(;p&&ch[p][x]==0;p=pa[p])ch[p][x]=np;

        if(p==0)pa[np]=1;
        else{
            int q=ch[p][x];
            if(len[q]==len[p]+1)pa[np]=q;
            else{
                int nq=++idx; len[nq]=len[p]+1;
                memcpy(ch[nq],ch[q],sizeof(ch[q]));
                pa[nq]=pa[q]; pa[q]=pa[np]=nq;
                for(;p&&ch[p][x]==q;p=pa[p])ch[p][x]=nq;
            }
        }
    }

    // 本质不同子串个数 = sum_[i=1..n] len[i]-len[pa[i]]
    // PAM的pa只可能在前面，所以不需要拓扑
    // SAM里len越小越接近根节点，但是idx会越大，所以要拓扑一下
```

```
    int tmp[N*2],topo[N*2];
    void Build(){
        for(int i=1;i<=idx;i++)tmp[len[i]]++;
        for(int i=1;i<=idx;i++)tmp[i]+=tmp[i-1];
        for(int i=1;i<=idx;i++)topo[tmp[len[i]]--]=i;
        for(int i=idx;i>1;i--){
            int v=topo[i], u=pa[v];
            siz[u]+=siz[v];
        }
    }

    int Init(char s[],int n){
        for(int i=1;i<=n;i++)
            _SAM::Insert(s[i]);
        _SAM::Build();
    }
};

namespace _PAM{
    int ch[N][C],pa[N]={1},len[N]={0,-1},siz[N];
    int idx=1,pre=0;

    void Insert(char s[],int pos){
        int p=pre, x=s[pos]-'a';
        for(;s[pos-len[p]-1]!=s[pos];)p=pa[p];
        if(ch[p][x]==0){
            int q=pa[p], np=++idx;
            len[np]=len[p]+2;
            for(;s[pos-len[q]-1]!=s[pos];)q=pa[q];
            pa[np]=ch[q][x]; ch[p][x]=np;
        }
        pre=ch[p][x]; siz[pre]++;
    }

    // 一个节点就是一个本质不同的回文串
    // 本质不同回文子串个数 = idx-1（去除两个根节点）
    ll Build(){
        ll ans=0;
        for(int i=idx;i>1;i--){
            siz[pa[i]]+=siz[i];
            ans=max(ans,1LL*siz[i]*len[i]);
        }
        return ans;
    }

    // 从1开始编号，默认s范围为[a,z]
    int Init(char s[],int n){
        for(int i=1;i<=n;i++)
            _PAM::Insert(s,i);
        printf("%lld",_PAM::Build());
    }
```

```
};

// DEBUG:
namespace SA{
    //a \in [0,n)
    //$a_n$ = min(0)
    //1 \leq a_i< m
    struct SuffixArray{
        int sa[N],hei[N],rnk[N];

        void Init(int *a,int n){
            InitSa(a,n);
            InitHeight(a,n);
            for(int i=0;i<n;i++){
                sa[i]=sa[i+1];
                hei[i]=hei[i+1];
                rnk[i]--;
            }
        }

        inline bool Cmp(int *a,int x,int y,int l){
            return a[x]==a[y]&&a[x+l]==a[y+l];
        }

        void InitSa(int *a,int n){
            int m=26;
            static int tmpX[N],tmpY[N],s[N];
            int *x=tmpX,*y=tmpY;

            a[n]=0;
            for(int i=0;i<m;i++)s[i]=0;
            for(int i=0;i<=n;i++)s[x[i]=a[i]]++;
            for(int i=1;i<m;i++)s[i]+=s[i-1];
            for(int i=n;i>=0;i--)sa[--s[x[i]]]=i;

            for(int i=1,p=1;p<=n;i<<=1,m=p){
                p=0;
                for(int j=n-i+1;j<=n;j++)y[p++]=j;
                for(int j=0;j<=n;j++)if(sa[j]>=i)y[p++]=sa[j]-i;
                for(int j=0;j<m;j++)s[j]=0;
                for(int j=0;j<=n;j++)s[x[y[j]]]++;
                for(int j=1;j<m;j++)s[j]+=s[j-1];
                for(int j=n;j>=0;j--)sa[--s[x[y[j]]]]=y[j];
                swap(x,y);
                p=1,x[sa[0]]=0;
                for(int j=1;j<=n;j++)x[sa[j]]=Cmp(y,sa[j-1],sa[j],i)?p-
1:p++;
            }
        }

        void InitHeight(int *a,int n){
```

```
            for(int i=1;i<=n;i++)rnk[sa[i]]=i;
            for(int i=0,j,k=0;i<n;hei[rnk[i++]]=k)
                for(k?k--:0,j=sa[rnk[i]-1];a[i+k]==a[j+k];k++);
        }
    };
};
```

FFT / Fast Fourier Transformation

```
struct Complex{
    double x,y;
    Complex(double _x=0,double _y=0){
        x=_x; y=_y;
    }
    Complex operator + (Complex a){
        return Complex(x+a.x,y+a.y);
    }
    Complex operator - (Complex a){
        return Complex(x-a.x,y-a.y);
    }
    Complex operator * (Complex a){
        return Complex(x*a.x-y*a.y,y*a.x+x*a.y);
    }
    Complex operator ~ (){
        return Complex(x,-y);
    }
};

namespace _FFT{
    // M需要开到比N大的2^n的两倍
    const int M=(1<<(__lg(N-1)+1))*2+5;
    const double PI=acos(-1.0);

    Complex rot[M];

    void FFT(Complex w[],int n,int op){
        static int r[M];
        for(int i=0;i<n;i++)
            r[i]=(r[i>>1]>>1)|((i&1)?n>>1:0);
        for(int i=0;i<n;i++)
            if(i<r[i])swap(w[i],w[r[i]]);

        for(int len=2;len<=n;len<<=1){
            int sub=len>>1;
            for(int l=0;l<n;l+=len){
                for(int i=l;i<l+sub;i++){
                    Complex &r=rot[sub+i-l];
                    Complex x=w[i];
                    Complex y=(Complex){r.x,op*r.y}*w[i+sub];
```

```
                    w[i]=x+y; w[i+sub]=x-y;
                }
            }
        }
    }

    // 无共轭优化
    // n是最高次项次数而不是长度
    // FIXME: 修改成长度而不是最高次项
    // TODO: 测试能不能正常运行
    void Cal(int f[],int g[],int n,int ans[]){
        static Complex a[N],b[N];

        int len=1;
        for(;len<=(n<<1);len<<=1);
        for(int i=0;i<len;i++)
            a[i].x=f[i], b[i].x=g[i];

        for(int i=1;i<len;i<<=1)
            for(int j=0;j<i;j++)
                rot[i+j]=Complex(cos(PI*j/i),sin(PI*j/i));

        FFT(a,len,1); FFT(b,len,1);
        for(int i=0;i<len;i++)
            a[i]=a[i]*b[i];
        FFT(a,len,-1);

        for(int i=0;i<len;i++)
            ans[i]=round(a[i].x/len);
    }

    // 有共轭优化
    // n是最高次项次数而不是长度
    // FIXME: 修改成长度而不是最高次项
    // TODO: 测试能不能正常运行
    void Cal_Conj(int f[],int g[],int n,int ans[]){
        static Complex a[N];

        int len=1;
        for(;len<=(n<<1);len<<=1);
        for(int i=0;i<len;i++)
            a[i].x=f[i], a[i].y=g[i];

        for(int i=1;i<len;i<<=1)
            for(int j=0;j<i;j++)
                rot[i+j]=Complex(cos(PI*j/i),sin(PI*j/i));

        FFT(a,len,1);
        for(int i=0;i<len;i++)
            a[i]=a[i]*a[i];
        FFT(a,len,-1);
```

```
92
93          for(int i=0;i<len;i++)
94              ans[i]=round(a[i].y/2/len);
95      }
96  };
97
98  // FIXME: 没有预处理rot
99  // NOTE: 中途可能会变成负数，最后需要模一下
100 namespace _NTT{
101     const int MOD=998244353, G=3;
102
103     ll QPow(ll bas,int t){
104         ll ret=1;
105         for(;t;t>>=1,bas=bas*bas%MOD)
106             if(t&1)ret=ret*bas%MOD;
107         return ret;
108     }
109
110     ll Inv(ll x){
111         return QPow(x,MOD-2);
112     }
113
114     void NTT(int w[],int n,int op){
115         static int r[N];
116
117         for(int i=0;i<n;i++)
118             r[i]=(r[i>>1]>>1)|((i&1)?n>>1:0);
119         for(int i=0;i<n;i++)
120             if(i<r[i])swap(w[i],w[r[i]]);
121
122         for(int len=2;len<=n;len<<=1){
123             int sub=len>>1;
124             ll det=QPow(G,MOD-1+op*(MOD-1)/len);
125             for(int l=0;l<n;l+=len){
126                 ll rot=1;
127                 for(int i=l;i<l+sub;i++){
128                     ll x=w[i], y=rot*w[i+sub]%MOD;
129                     w[i]=(x+y)%MOD;
130                     w[i+sub]=(x-y)%MOD;      //maybe minus
131                     rot=rot*det%MOD;
132                 }
133             }
134         }
135
136         if(op==1)return;
137         ll inv=Inv(n);
138         for(int i=0;i<n;i++)
139             w[i]=inv*w[i]%MOD;
140     }
141 };
142
```

```cpp
// 7次FFT
namespace _MTT_7{
    using namespace _FFT;

    void MTT(int f[],int g[],int n,int ans[]){
        static const int D=(1<<15);
        static const int MOD=998244353;
        static Complex a[M],b[M],c[M],d[M];

        memset(a,0,sizeof(a)); memset(b,0,sizeof(b));
        memset(c,0,sizeof(c)); memset(d,0,sizeof(d));

        int len=1;
        for(;len<=(n<<1);len<<=1);

        for(int i=0;i<=n;i++){
            a[i].x=f[i]/D; b[i].x=f[i]%D;
            c[i].x=g[i]/D; d[i].x=g[i]%D;
        }

        for(int i=1;i<len;i<<=1)
            for(int j=0;j<i;j++)
                rot[i+j]=Complex(cos(PI*j/i),sin(PI*j/i));

        FFT(a,len,1); FFT(b,len,1);
        FFT(c,len,1); FFT(d,len,1);

        for(int i=0;i<len;i++){
            Complex _a=a[i], _b=b[i], _c=c[i], _d=d[i];
            a[i]=_a*_c;
            b[i]=_a*_d+_b*_c;
            c[i]=_b*_d;
        }

        FFT(a,len,-1); FFT(b,len,-1); FFT(c,len,-1);

        for(int i=0;i<len;i++){
            ll w=0;
            w += (ll)round(a[i].x/len)%MOD*D%MOD*D%MOD;
            w += (ll)round(b[i].x/len)%MOD*D%MOD;
            w += (ll)round(c[i].x/len)%MOD;
            ans[i]=w%MOD;
        }
    }
};

// 4次FFT
namespace _MTT_4{
    using namespace _FFT;

    void MTT(int f[],int g[],int n,int ans[]){
```

```
        static const int D=(1<<15);
        static const int MOD=998244353;
        static Complex a[M],b[M],c[M],d[M];

        int len=1;
        for(;len<=(n<<1);len<<=1);
        for(int i=0;i<len;i++)
            a[i]=b[i]=Complex(0,0);

        for(int i=0;i<=n;i++){
            a[i].x=f[i]/D; a[i].y=f[i]%D;
            b[i].x=g[i]/D; b[i].y=g[i]%D;
        }

        for(int i=1;i<len;i<<=1)
            for(int j=0;j<i;j++)
                rot[i+j]=Complex(cos(PI*j/i),sin(PI*j/i));

        FFT(a,len,1); FFT(b,len,1);

        for(int i=0;i<len;i++){
            Complex t,a0,a1,b0,b1;

            t = ~a[(len-i)%len];
            a0 = (a[i]-t)*Complex(0,-0.5);
            a1 = (a[i]+t)*Complex(0.5,0);
            t = ~b[(len-i)%len];
            b0 = (b[i]-t)*Complex(0,-0.5);
            b1 = (b[i]+t)*Complex(0.5,0);

            c[i] = a1*b1;
            d[i] = a1*b0+a0*b1+a0*b0*Complex(0,1);
        }

        FFT(c,len,-1); FFT(d,len,-1);

        for(int i=0;i<n;i++){
            ll w = 0;
            w += ll(round(c[i].x/len))%MOD*D*D;
            w += ll(round(d[i].x/len))%MOD*D;
            w += ll(round(d[i].y/len))%MOD;
            ans[i] = w%MOD;
        }
    }
};
```

多项式 / Polynomial

```
namespace _PolyInv{
```

```
    const int MOD=998244353;
    ll Inv(ll x);

    // MTT版本
    using namespace _MTT_4;
    void PolyInv(int a[],int b[],int n){
        if(n==1){
            b[0]=Inv(a[0]);
            return;
        }
        PolyInv(a,b,(n+1)/2);

        static int c[N];
        for(int i=0;i<n;i++)c[i]=0;

        MTT(a,b,n,c);
        for(int i=0;i<n;i++)c[i]=MOD-c[i];
        c[0]=(2+c[0])%MOD;
        MTT(c,b,n,b);
    }

    // NTT版本
    using namespace _NTT;
    void PolyInv_NTT(int a[],int b[],int n){
        if(n==1){
            b[0]=Inv(a[0]);
            return;
        }
        PolyInv(a,b,(n+1)/2);

        int len=1;
        for(;len<(n<<1);len<<=1);

        static int c[N];
        for(int i=0;i<n;i++)c[i]=a[i];
        for(int i=n;i<len;i++)c[i]=0;

        NTT(b,len,1); NTT(c,len,1);
        for(int i=0;i<len;i++)
            b[i]=(2LL-1LL*c[i]*b[i])%MOD*b[i]%MOD;
        NTT(b,len,-1);
        for(int i=n;i<len;i++)b[i]=0;
    }
};

namespace _PolyDiv{
    using namespace _PolyInv;
    const int MOD=998244353;

    // a = b*d + mod
    // FIXME: 改写成MTT形式
```

```
53
54      // NTT版本
55      using namespace _NTT;
56      void PolyDiv(int a[],int b[],int n,int m,int d[],int mod[]){
57          reverse(a,a+n+1); reverse(b,b+m+1);
58          static int inv_b[N];
59          PolyInv(b,inv_b,n-m+1);
60
61          int len=1;
62          for(;len<(n<<1);len<<=1);
63          for(int i=0;i<=n;i++)d[i]=a[i];
64          NTT(d,len,1); NTT(inv_b,len,1);
65          for(int i=0;i<len;i++)
66              d[i]=1LL*d[i]*inv_b[i]%MOD;
67          NTT(d,len,-1);
68          for(int i=n-m+1;i<len;i++)
69              d[i]=0;
70
71          reverse(a,a+n+1); reverse(b,b+m+1);
72          reverse(d,d+n-m+1);
73
74          static int _b[N],_d[N];
75          for(int i=0;i<=m;i++)_b[i]=b[i];
76          for(int i=0;i<=n-m;i++)_d[i]=d[i];
77          NTT(_b,len,1); NTT(_d,len,1);
78          for(int i=0;i<len;i++)
79              mod[i]=1LL*_b[i]*_d[i]%MOD;
80          NTT(mod,len,-1);
81
82          for(int i=0;i<m;i++)
83              mod[i]=(a[i]-mod[i]+MOD)%MOD;
84      }
85  };
```

树 / Tree Algorithm

```
1   // DEBUG:
2   namespace HeavyLightDecomposition{
3       SegTree t;
4       int dep[N],siz[N],pa[N],son[N],top[N],idx[N];
5       int nIdx;
6
7       void Build(){
8           nIdx=dep[0]=siz[0]=son[0]=0;
9           DFS1(); DFS2();
10      }
11      void DFS1(int u=1,int pa=0){
12          dep[u]=dep[HLDcp::pa[u]=pa]+1;
13          siz[u]=1;son[u]=0;
```

```
        for(int i=0;i<a[u].size();i++){
            int v=a[u][i];
            if(v==pa)continue;
            DFS1(v,u);
            if(siz[v]>siz[son[u]])son[u]=v;
            siz[u]+=siz[v];
        }
    }
    void DFS2(int u=1,int pa=0){
        idx[u]=++nIdx;top[u]=u;
        if(son[pa]==u)top[u]=top[pa];
        if(son[u])DFS2(son[u],u);
        for(int i=0;i<a[u].size();i++){
            int v=a[u][i];
            if(v==pa||v==son[u])continue;
            DFS2(v,u);
        }
    }
    void Add(int u){
        while(top[u]!=0){
            t.Update(idx[top[u]],idx[u],1);
            u=pa[top[u]];
        }
    }
    void Delete(int u){
        t.Update(idx[u],idx[u]+siz[u]-1,0);
    }
    // 对边操作，每个点代表(u,pa[u])这条边
    void Modify(int u,int v,int w){
        while(top[u]!=top[v]){
            if(dep[top[u]]<dep[top[v]])swap(u,v);
            t.Modify(idx[top[u]],idx[u],1,w,1,nIdx);
            u=pa[top[u]];
        }
        // 节点相同则退出
        if(u==v)return;
        if(dep[u]>dep[v])swap(u,v);
        t.Modify(idx[u]+1,idx[v],1,w,1,nIdx);
    }
};

// DEBUG:
// FIXME: 没有建树过程
namespace FHQTreap{
    struct Node{
        int v,w,siz,lazy; ll sum;
        Node *lch,*rch;

        Node(int _v=0){
            v=_v, w=rand(), siz=1;
            sum=v, lazy=0;
```

```
65              lch=rch=nullptr;
66          }
67          void Maintain(){
68              siz=1; sum=v;
69              if(lch!=nullptr)
70                  siz+=lch->siz,sum+=lch->sum;
71              if(rch!=nullptr)
72                  siz+=rch->siz,sum+=rch->sum;
73          }
74          void Pushdown(){
75              if((this==nullptr)||lazy==0)return;
76              if(lch!=nullptr)lch->lazy^=1;
77              if(rch!=nullptr)rch->lazy^=1;
78              swap(lch,rch); lazy=0;
79          }
80      };
81
82      typedef pair<Node*,Node*> pNode;
83      Node mp[N];
84
85      struct Treap{
86          Node *rt,*pit;
87
88          Treap(){
89              pit=mp; rt=nullptr;
90          }
91          Node* NewNode(int v){
92              *pit=Node(v);
93              return pit++;
94          }
95          void Insert(int v){
96              Node* o=NewNode(v);
97              rt=Merge(rt,o);
98          }
99          pNode Split(Node* o,int k){
100             pNode ret(nullptr,nullptr);
101             if(o==nullptr)return ret;
102
103             o->Pushdown();
104             int siz=(o->lch==nullptr)?0:o->lch->siz;
105
106             if(k<=siz){
107                 ret=Split(o->lch,k);
108                 o->lch=ret.second;
109                 o->Maintain();
110                 ret.second=o;
111             }else{
112                 ret=Split(o->rch,k-siz-1);
113                 o->rch=ret.first;
114                 o->Maintain();
115                 ret.first=o;
```

```
            }

            return ret;
        }
        Node* Merge(Node* a,Node* b){
            if(a==nullptr)return b;
            if(b==nullptr)return a;

            a->Pushdown(); b->Pushdown();
            if(a->w < b->w){
                a->rch=Merge(a->rch,b);
                a->Maintain();
                return a;
            }else{
                b->lch=Merge(a,b->lch);
                b->Maintain();
                return b;
            }
        }
        void Print(Node* o){
            if(o==nullptr)return;
            o->Pushdown();
            Print(o->lch);
            printf("%d ",o->v);
            Print(o->rch);
        }
        ll Inverse(int L,int R){
            pNode a=Split(rt,L-1);
            pNode b=Split(a.second,R-L+1);
            b.first->lazy^=1;       //b一定非空
            int ret=b.first->sum;
            rt=Merge(Merge(a.first,b.first),b.second);
            return ret;
        }
    };
};

// DEBUG:
namespace _DFS4Root{
    int _maxSiz;
    int maxSiz[N];
    void DFS(int u,int pa,int n){
        static int siz[N]; siz[u]=1;
        for(auto v:a[u]){
            if(v==pa)continue;
            DFS(v,u,n);
            siz[u]+=siz[v];
            maxSiz[u]=max(maxSiz[u],siz[v]);
        }
        maxSiz[u]=max(maxSiz[u],n-siz[u]);
        _maxSiz=min(_maxSiz,maxSiz[u]);
```

```
    }
};

// DEBUG:
namespace _TreeHash{
    const int P=99299299;
    // f[u] = xor f[v]*P+siz[v]
    ull TreeHash(int u,int pa){
        static int siz[N];
        siz[u]=1; ull ret=0;
        for(auto v:a[u]){
            if(v==pa)continue;
            auto _hash=TreeHash(v,u);
            ret^=_hash*P+siz[v];
            siz[u]+=siz[v];
        }
        return ret;
    }
};

// DEBUG:
namespace DSUOnTree{
    int ans[N],cnt[N],sum[N];

    void Modify(int u,int pa,int op,int son){
        if(op==1)sum[++cnt[c[u]]]++;
        else sum[cnt[c[u]]--]--;
        for(auto v:a[u])
            if(v!=pa && v!=son)
                Modify(v,u,op,son);
    }

    void DFS(int u,int pa,bool keep){
        int son=0;
        for(auto v:a[u])
            if(v!=pa && siz[v]>siz[son])
                son=v;
        for(auto v:a[u])
            if(v!=pa && v!=son)
                DFS(v,u,0);
        if(son)DFS(son,u,1);
        Modify(u,pa,1,son);

        for(auto p:qry[u])
            ans[p.first]=sum[p.second];

        if(!keep)Modify(u,pa,-1,0);
    }
};

// DEBUG:
```

```cpp
namespace DCOnTree{
    bool vst[N];
    int rt;
    int siz[N],maxSiz[N];

    void DFS4Rt(int u,int pa,int sum){
        siz[u]=1;
        for(auto v:a[u])
            if(v!=pa && !vst[v]){
                DFS4Rt(v,u,sum);
                siz[u]+=siz[v];
            }
        maxSiz[u]=max(siz[u],sum-siz[u]);
        if(maxSiz[u]>maxSiz[rt])rt=u;
    }

    void DFS(int u,int pa,int dep,ll w1,ll w2){
        for(auto v:a[u])
            if(v!=pa && !vst[v])
                DFS(v,u,dep+1,w1,w2);
    }

    int in0[N],in1[N],out0[N],out1[N];

    // 计算经过该点路径的贡献
    void Cal(int u){
        for(auto v:a[u]){
            if(vst[v])continue;
            DFS(v,u,1,w[u],0);      //remove w[u] in path w2
        }
    }

    void Solve(int u){
        vst[u]=1; Cal(u);
        for(auto v:a[u]){
            if(vst[v])continue;
            DFS4Rt(v,rt=0,siz[v]);      //init rt=0
            Solve(v);
        }
    }

    int main(){
        DFS4Rt(1,rt=0,n);
        Solve(rt);
    }
};
```

数论 / Number Theory

```
```

```cpp
// https://www.luogu.org/problem/P3383
// DEBUG:
namespace MillerRabin{
    bool MR(ll p){
        if(p==2)return 1;
        if(p<=1 || !(p&1))return 0;
        if(p==2152302898747LL)return 0;
        if(p==3215031751)return 0;

        mt19937_64 rng(time(0));
        for(int i=0;i<UPP;i++){
            ll a=rng()%(p-2)+2;
            for(ll k=p-1; !(k&1); k>>=1){
                ll t=QPow(a,k,p);
                if(t!=1 && t!=p-1)return 0;
                if(t==p-1)break;
            }
        }
        return 1;
    }
};

// DEBUG:
namespace PollardRho{
    ll QMul(i128 a,i128 b,ll mod){
        return a*b%mod;
    }

    inline ll Abs(ll x){
        return x>0?x:-x;
    }

    ll PR(ll x){
        if(MR(x))return x;

        mt19937_64 rng(time(0));
        ll t1=rng()%(x-1)+1;
        ll b=rng()%(x-1)+1;
        ll t2=(QMul(t1,t1,x)+b)%x;

        int cnt=0; ll p=1;
        while(t1!=t2){
            cnt++;
            p=QMul(p,Abs(t2-t1),x);
            if(p==0){
                ll g=__gcd(Abs(t2-t1),x);
                return max(PR(g),PR(x/g));
            }
            if(cnt==127){
                ll g=__gcd(p,x);
                if(g!=1 && g!=x)
```

```
                    return max(PR(g),PR(x/g));
                cnt=0; p=1;
            }
            t1=(QMul(t1,t1,x)+b)%x;
            t2=(QMul(t2,t2,x)+b)%x;
            t2=(QMul(t2,t2,x)+b)%x;
        }

        ll g=__gcd(p,x);
        if(g!=1 && g!=x)
            return max(PR(g),PR(x/g));

        return 0;
    }

    // 找到最大质因子
    // 先MR判定再PR
    ll Cal(ll x){
        if(MR(x))cout<<"Prime\n";
        else if(x==1)cout<<"1\n";
        else{
            ll ans=0;
            while(ans==0)
                ans=PR(x);
            cout<<ans<<endl;
        }
    }
};

// DEBUG:
namespace DuSieve{
    const int N=3e6+5;
    bool notPri[N];
    int pri[N],mu[N],phi[N];
    ll sumMu[N],sumPhi[N];

    void Init(){
        mu[1]=1; phi[1]=1;
        for(int i=2;i<N;i++){
            if(!notPri[i]){
                pri[++pri[0]]=i;
                phi[i]=i-1;
                mu[i]=-1;
            }
            for(int j=1;j<=pri[0] && i*pri[j]<N;j++){
                int x=i*pri[j]; notPri[x]=1;
                if(i%pri[j]){
                    mu[x]=-mu[i];
                    phi[x]=phi[i]*(pri[j]-1);
                }else{
                    mu[x]=0;
```

```
                    phi[x]=phi[i]*pri[j];
                    break;
                }
            }
        }

        for(int i=1;i<N;i++){
            sumMu[i]=sumMu[i-1]+mu[i];
            sumPhi[i]+=sumPhi[i-1]+phi[i];
        }
    }

    unordered_map<int,ll> _sumMu,_sumPhi;

    ll Mu(int n){
        if(n<N)return sumMu[n];
        if(_sumMu.count(n))return _sumMu[n];
        ll ret=1;
        // 实际上i和j可能爆int
        for(int i=2,j;i<=n;i=j+1){
            j=n/(n/i);
            ret-=Mu(n/i)*(j-i+1);
        }
        return _sumMu[n]=ret;
    }

    // 实际上ans可能爆ll
    ll Phi(int n){
        if(n<N)return sumPhi[n];
        if(_sumPhi.count(n))return _sumPhi[n];
        ll ret=1LL*(1+n)*n/2;
        // 实际上i和j可能爆int
        for(int i=2,j;i<=n;i=j+1){
            j=n/(n/i);
            ret-=Phi(n/i)*(j-i+1);
        }
        return _sumPhi[n]=ret;
    }
};
```

离散数学 / Discrete Maths

```
// DEBUG:
namespace BSGS{
    int BSGS(ll a,ll b,ll p){
        map<ll,int> s;
        int m=(int)ceil(sqrt(p));
        ll tmp=1;
        for(int i=0;i<m;i++,tmp=tmp*a%p)
```

```
            if(!s.count(tmp))s[tmp]=i;
        ll inv=Invert(tmp,p);tmp=b;
        for(int i=0;i<m;i++,tmp=tmp*inv%p)
            if(s.count(tmp))return i*m+s[tmp]+1;
        return -1;
    }
};

// DEBUG:
namespace ExGCD{
    void ExtendGCD(int a,int b,int &x,int &y,int &g){
        if(!b)x=1,y=0,g=a;
        else ExtendGCD(b,a%b,y,x,g),y-=x*(a/b);
    }
    // ax+by=c
    // 定义b'=b/g
    // 解集: x = x0+k*b', y = y0-k*a'
    // 最小非负解: x+ = (x0 % b'+ b') % (b'c')
    // 为啥mod b'c', b'c'>=b'不是吗, mod b'似乎就行
};

// DEBUG:
namespace ExCRT{
    ll ExtendCRT(){
        ll a0,p0,a1,p1; bool flag=1;
        cin>>p0>>a0;
        for(int i=2;i<=n;i++){
            ll x,y,g,c;
            cin>>p1>>a1;
            if(flag){
                ExtendGCD(p0,p1,x,y,g);
                c=a1-a0;
                if(c%g){flag=0;continue;}
                x=x*(c/g)%(p1/g);
                a0+=x*p0;p0=p0*p1/g;
                a0%=p0;
            }
        }
        if(flag)return (a0%p0+p0)%p0;
        else return -1;
    }
};
```

线性代数 / Linear Algebra

```
struct LBase{
    // 32位版本
    static const int S=32;
    ui b[S];
```

```
    LBase(){
        memset(b,0,sizeof(b));
    };
    void Insert(ui x){
        for(int i=S-1;i>=0;i--)
            if(x>>i){
                if(!b[i]){
                    b[i]=x; return;
                }else x^=b[i];
            }
    }
    bool Count(ui x){
        for(int i=S-1;i>=0;i--)
            if(x>>i){
                if(!b[i])return 0;
                else x^=b[i];
            }
        return 1;
    }
    // 线性基求交
    LBase operator &(LBase a){
        LBase tot=a, ret;
        for(int i=0;i<S;i++)
            if(b[i]){
                int now=b[i], det=0;
                for(int j=i;j>=0;j--)
                    if(now&(1<<j)){
                        if(tot.b[j]){
                            now^=tot.b[j];
                            det^=a.b[j];
                            if(now)continue;
                            ret.b[i]=det;
                        }else tot.b[j]=now, a.b[j]=det;
                        break;
                    }
            }
        return ret;
    }
    // 线性基求并
    LBase operator |(LBase a){
        LBase ret=a;
        for(int i=0;i<S;i++)
            ret.Insert(b[i]);
        return ret;
    }G
};
```

计算几何 / Computational Geometry

```
```

```cpp
// DEBUG:
namespace 2DGeometry{
    const double EPS=1e-8;

    struct Point{
        double x,y;
        Point operator-(Point b)const{
            return (Point){x-b.x,y-b.y};
        }
        // DCmp
        int operator*(Point b)const{
            double ret=x*b.y-y*b.x;
            if(ret>EPS)return 1;
            else if(ret<-EPS)return -1;
            else return 0;
        }
    };

    bool Cmp(Point a,Point b){
        return a.x==b.x ? a.y<b.y : a.x<b.x;
    }

    const int N=100000+5;

    // may return a point or a segment
    // the end equal to the start
    // when a-b-c is collinear, b will be inserted
    void ConvexHull(vector<Point> &p,vector<Point> &ret){
        static Point s[N]; int t=0;
        sort(p.begin(),p.end(),Cmp);

        int n=p.size();
        for(int i=0;i<n;i++){
            while(t>1 && (s[t]-s[t-1])*(p[i]-s[t])<0)t--;
            s[++t]=p[i];
        }
        int _t=t;
        for(int i=n-2;i>=0;i--){
            while(t>_t && (s[t]-s[t-1])*(p[i]-s[t])<0)t--;
            s[++t]=p[i];
        }
        for(int i=1;i<=t;i++)
            ret.push_back(s[i]);
    }
};
```

杂项 / Other

```cpp
struct Bignum{
```

```
    // 15*8 = 120位
    // 选8位是为了保证S*S*L不太大可以做除法
    // 但是其实也可以用个东西额外保存的，中间边乘法边取模
    const static int L=15+3, S=100000000;
    ll a[L];
    Bignum(){
        memset(a,0,sizeof(a));
    }
    void Set(int x){
        a[0]=x;
    }
    void operator+=(Bignum b){
        for(int i=0;i<L;i++)a[i]+=b.a[i];
        for(int i=0;i+1<L;i++){
            a[i+1]+=a[i]/S;
            a[i]%=S;
        }
    }
    Bignum operator*(Bignum b){
        Bignum ret;
        for(int i=0;i<L;i++)
            for(int j=0;j<=i;j++)
                ret.a[i]+=a[j]*b.a[i-j];
        for(int i=0;i+1<L;i++){
            ret.a[i+1]+=ret.a[i]/S;
            ret.a[i]%=S;
        }
        return ret;
    }
    void operator/=(int x){
        ll res=0;
        for(int i=L-1;i>=0;i--){
            res+=a[i]; a[i]=res/x;
            res=(res-a[i]*x)*S;
        }
    }
    void Print(){
        int p=L-1;
        while(p>0 && a[p]==0)p--;
        cout<<a[p];
        for(int i=p-1;i>=0;i--)
            cout<<setw(8)<<setfill('0')<<a[i];
    }
};

namespace _FastIO{
    template <typename T> inline T read(){
        T x=0, f=1; char ch=0;
        for(;!isdigit(ch);ch=getchar())
            if(ch=='-')f=-1;
        for(;isdigit(ch);ch=getchar())
```

```
            x=x*10+ch-'0';
        return x*s;
    }
    const int BUF=64+5;
    template <typename T> inline void write(T x){
        static int s[BUF]; int t=0;
        do{
            s[t++]=x%10, x/=10;
        }while(x);
        while(t)putchar(s[--t]+'0');
    }
};

// DEBUG:
// 能用__int128的时候就快得一批
namespace QMul{
    ll QMul(ll a,ll b){
        if(a>b)swap(a,b);
        ll ret=0;
        for(;b;b>>=1,(a<<=1)%=p)
            if(b&1)(ret+=a)%=p;
        return ret;
    }
};
```