Algorithm Library

magic::team.getname()

South China Normal University

September 7, 2024

Contents

头文件

DEBUG 头

```cpp
#include <bits/stdc++.h>
using namespace std;
using i64=long long;
using i128=__int128;

namespace DBG
{
    template <class T>
    void _dbg(const char *f,T t) { cerr<<f<<'='<<t<<'\n'; }

    template <class A,class... B>
    void _dbg(const char *f,A a,B... b)
    {
        while (*f!=',') cerr<<*f++;
        cerr<<'='<<a<<",";
        _dbg(f+1,b...);
    }

    template <class T>
    ostream& operator << (ostream& os,const vector<T> &v)
    {
        os<<"[ ";
        for (const auto &x:v) os<<x<<", ";
        os<<"]";
        return os;
    }

    #define dbg(...) _dbg(#__VA_ARGS__, __VA_ARGS__)
}

using namespace DBG;
```

__int128 输出流

```cpp
ostream &operator << (ostream &os,i128 n)
{
    string s;
    bool neg=n<0;
    if (neg) n=-n;
    while (n)
    {
        s+='0'+n%10;
        n/=10;
    }
    if (neg) s+='-';
    reverse(s.begin(),s.end());
    if (s.empty()) s+='0';
    return os<<s;
}
```

常用数学函数

```cpp
i64 ceilDiv(i64 n,i64 m)
{
    if (n>=0) return (n+m-1)/m;
    else return n/m;
}

i64 floorDiv(i64 n,i64 m)
{
    if (n>=0) return n/m;
    else return (n-m+1)/m;
}

i128 gcd(i128 a,i128 b)
{
```

```
    return b?gcd(b,a%b):a;
}
```

数学

欧拉筛

```
vector<int> minp,primes;

void sieve(int n)
{
    minp.assign(n+1,0);
    primes.clear();
    for (int i=2;i<=n;i++)
    {
        if (!minp[i])
        {
            minp[i]=i;
            primes.push_back(i);
        }
        for (auto p:primes)
        {
            if (i*p>n) break;
            minp[i*p]=p;
            if (p==minp[i]) break;
        }
    }
}
```

取模类（MInt）

```
template <class T>
constexpr T power(T a,i64 b)
{
    T res=1;
    for (;b;b>>=1,a*=a)
        if (b&1) res*=a;
    return res;
}

template <int P>
struct MInt
{
    int x;
    constexpr MInt():x{} {}
    constexpr MInt(i64 x):x{norm(x%getMod())} {}

    static int Mod;
    constexpr static int getMod()
    {
        if (P>0) return P;
        else return Mod;
    }

    constexpr static void setMod(int Mod_) { Mod=Mod_; }

    constexpr int norm(int x) const
    {
        if (x<0) x+=getMod();
        if (x>=getMod()) x-=getMod();
        return x;
    }

    constexpr int val() const { return x; }

    explicit constexpr operator int () const { return x; }

    constexpr MInt operator - () const
    {
```

```
        MInt res;
        res.x=norm(getMod()-x);
        return res;
    }

    constexpr MInt inv() const
    {
        assert(x!=0);
        return power(*this,getMod()-2);
    }

    constexpr MInt &operator *= (MInt rhs) &
    {
        x=1ll*x*rhs.x%getMod();
        return *this;
    }

    constexpr MInt &operator += (MInt rhs) &
    {
        x=norm(x+rhs.x);
        return *this;
    }

    constexpr MInt &operator -= (MInt rhs) &
    {
        x=norm(x-rhs.x);
        return *this;
    }

    constexpr MInt &operator /= (MInt rhs) &
    {
        return *this*=rhs.inv();
    }

    friend constexpr MInt operator * (MInt lhs,MInt rhs)
    {
        MInt res=lhs;
        res*=rhs;
        return res;
    }

    friend constexpr MInt operator + (MInt lhs,MInt rhs)
    {
        MInt res=lhs;
        res+=rhs;
        return res;
    }

    friend constexpr MInt operator - (MInt lhs,MInt rhs)
    {
        MInt res=lhs;
        res-=rhs;
        return res;
    }

    friend constexpr MInt operator / (MInt lhs,MInt rhs)
    {
        MInt res=lhs;
        res/=rhs;
        return res;
    }

    friend constexpr istream &operator >> (istream &is,MInt &a)
    {
        i64 v;
        is>>v;
        a=MInt(v);
        return is;
    }

    friend constexpr ostream &operator << (ostream &os,const MInt &a) { return os<<a.val(); }
```

```
    friend constexpr bool operator == (MInt lhs,MInt rhs) { return lhs.val()==rhs.val(); }

    friend constexpr bool operator != (MInt lhs,MInt rhs) { return lhs.val()!=rhs.val(); }
};

template<>
int MInt<0>::Mod=1;

template<int V,int P>
constexpr MInt<P> CInv=MInt<P>(V).inv();
```

多项式

```
namespace polygone{
    //慎用!!!
    //需要根据实际情况来调整 函数 poly operator * (poly f,poly g)
    //和 数据类型 (变为 int) 因为取模频繁 而且我这一块处理的不好导致常数很大
    //涵盖了 多项式 求导, 积分, 加减乘 , 快速幂, ln , exp , 开根
    //没有涵盖 除法 多点求值 复合运算 等复杂部分 (学艺不精呜呜呜)
    //带有模数 没有支持无模
    #define N 3000006
    #define ll long long
    long long read(){
        ll f=1,s=0;char ch=getchar();
        while(ch<'0'||ch>'9'){if(ch=='-')f=-1;ch=getchar();}
        while(ch>='0'&&ch<='9')s=(s<<1)+(s<<3)+ch-'0',ch=getchar();
        return s*f;
    }

    long long reading(ll mod){
        ll f=1,s=0;char ch=getchar();
        while(ch<'0'||ch>'9'){if(ch=='-')f=-1;ch=getchar();}
        while(ch>='0'&&ch<='9')s=(s<<1)+(s<<3)+ch-'0',s%=mod,ch=getchar();
        return s*f;
    }
    const double pi=acos(-1.0);
    const long long  mod = 998244353, g = 3, g1 = 332748118;
    ll add(ll x,ll y){x+=y;return x>=mod?x-mod:x;}
    ll rdu(ll x,ll y){x-=y;return x<0?x+mod:x;}
    ll mul(ll x,ll y){return 1ll*x*y%mod;}

    #define poly vector<ll>
    #define plen(x) ((int)x.size())

    ll finv[N],r[N],lim,lg;

    ll qpow(ll x,ll y){
        if(y==1) return x;
        ll res=qpow(x,y>>1);
        if(y&1) return (((res*res)%mod)*x)%mod;
        return (res*res)%mod;
    }ll ginv(ll x){return qpow(x,mod-2);}

    void init(ll n){
        for(lim=1,lg=0;lim<=n;lim<<=1,lg++);
        for(int i=0;i<lim;i++) r[i]=((r[i>>1]>>1)|((i&1)<<(lg-1)));
    }

    void cinv(ll n){
        finv[1]=1;
        for(int i=2;i<=n;i++) finv[i]=mul(mod-mod/i,finv[mod%i]);
    }

    poly operator - (ll x,poly f){ll len=plen(f);for(int i=0;i<len;i++) f[i]=mod-f[i];f[0]=add(f[0],x);return f;}
    poly operator - (poly f,ll x){f[0]=rdu(f[0],x);return f;}
    poly operator - (poly f,poly g){ll n=max(plen(f),plen(g));f.resize(n);g.resize(n);for(int i=0;i<n;i++)
    ↪ f[i]=rdu(f[i],g[i]);return f;}
    poly operator * (poly f,ll x){ll len=plen(f);for(int i=0;i<len;i++) f[i]=mul(f[i],x);return f;}
    poly operator + (poly f,poly g){ll n=max(plen(f),plen(g));f.resize(n);g.resize(n);for(int i=0;i<n;i++)
    ↪ f[i]=add(f[i],g[i]);return f;}
```

```
56
57      void ntt(poly &f,int op){
58          for(int i=0;i<lim;i++) if(i<r[i]) swap(f[i],f[r[i]]);
59          for(int mid=1;mid<lim;mid<<=1){
60              ll wn=qpow(op==1?g:g1,(mod-1)/(mid<<1));
61              for(int j=0;j<lim;j+=(mid<<1)){
62                  ll w=1;
63                  for(int k=0;k<mid;k++,w=(w*wn)%mod){
64                      ll x=f[j+k],y=w*f[j+k+mid]%mod;
65                      f[j+k]=(x+y)%mod;
66                      f[j+k+mid]=(x-y+mod)%mod;
67                  }
68              }
69          }
70          if(op==1) return;
71          ll inv=qpow(lim,mod-2);
72          for(int i=0;i<lim;i++) f[i]=1ll*f[i]*inv%mod;
73      }
74
75      poly operator * (poly f,poly g){
76          ll n=plen(f)+plen(g)-1;
77          init(n);
78          f.resize(lim);g.resize(lim);
79          ntt(f,1);ntt(g,1);
80          for(int i=0;i<lim;i++) f[i]=mul(f[i],g[i]);
81          ntt(f,-1);
82          f.resize(min(n,1ll*100000));
83          return f;
84      }
85
86      poly inv(poly f){
87          poly g=poly(1,ginv(f[0]));
88          ll len=plen(f);
89          for(int i=2;i<(len<<1);i<<=1){
90              poly A=f;
91              A.resize(i);
92              g=g*(2-(g*A));
93          }
94          g.resize(len);
95          return g;
96      }
97
98      poly qiudao(poly f){
99          ll len=plen(f);
100         for(int i=0;i<len-1;i++) f[i]=mul(i+1,f[i+1]);
101         f.resize(len-1);
102         return f;
103     }
104
105     poly jifen(poly f){
106         ll len=plen(f);
107         f.resize(len+1);
108         for(int i=len-1;i>=1;i--) f[i]=mul(f[i-1],finv[i]);
109         f[0]=0;
110         return f;
111     }
112
113     poly ln(poly f){
114         poly g=jifen(qiudao(f)*inv(f));
115         g.resize(plen(f));
116         return g;
117     }
118
119     poly exp(poly f){
120         poly g=poly(1,1);
121         ll len=plen(f);
122         for(int i=2;i<(len<<2);i<<=1){
123             poly A=f;
124             A.resize(i);
125             g=g*((1-ln(g))+A);
126         }
```

```
        g.resize(len);
        return g;
    }

    poly Pow(poly f,ll k){
        f=ln(f);f=f*k;f=exp(f);
        return f;
    }

    poly sqrt(poly f){
        poly g=poly(1,1);
        ll len=plen(f);
        ll inv2=ginv(2);
        for(int i=2;i<(len<<2);i<<=1){
            poly A=f;
            A.resize(i);
            g=(g+(A*inv(g)))*inv2;
        }
        g.resize(len);
        return g;
    }
}
using namespace polygone;
```

杜教筛

```
void prime(){
    u[1]=1;v[1]=1;phi[1]=1;
    for(ll i=2;i<=N;i++){
        if(!v[i]) {u[i]=-1;p[++cnt]=i;phi[i]=i-1;}
        for(ll j=1;j<=cnt&&p[j]*i<=N;j++){
            v[p[j]*i]=1;
            if(!(i%p[j])) {phi[i*p[j]]=phi[i]*p[j];break;}
            else u[i*p[j]]=-u[i],phi[i*p[j]]=phi[i]*phi[p[j]];
        }
    }
    for(ll i=1;i<=N;i++) u[i]+=u[i-1],phi[i]+=phi[i-1];
}

ll sumu(ll n) {
    if(n<=N) return u[n];
    if(M[n]) return M[n];
    ll res=1;
    for(ll l=2,r=0;l<=n;l=r+1){
        r=n/(n/l);
        res-=1ll*sumu(n/l)*(r-l+1);
    }
    M[n]=res;
    return M[n];
}
```

Min_25 筛

```
void prime(ll n){
    for(ll i=2;i<=n;i++){
        if(!vis[i]) p[++cnt]=i;
        for(ll j=1;j<=cnt&&p[j]<=n/i;j++){
            vis[i*p[j]]=1;
            if(i%p[j]==0) break;
        }
    }
}

ll qz1(ll x){return x%=mod,x*(x+1)%mod*inv2%mod;}
ll qz2(ll x){return x%=mod,x*(x+1)%mod*(2*x+1)%mod*inv6%mod;}
ll get(ll x){return x<N?id1[x]:id2[n/x];}
ll sq(ll x) {return x%=mod,x*x%mod;}
ll F(ll x)  {return x%=mod,(sq(x)+mod-x)%mod;}

void getg(){
```

```
    for(ll l=1,r;l<=n;l=r+1){
        r=n/(n/l);
        v[++m]=n/l;
        if(v[m]<N) id1[v[m]]=m;
        else id2[n/v[m]]=m;
        g1[m]=(qz1(v[m])-1+mod)%mod;
        g2[m]=(qz2(v[m])-1+mod)%mod;
    }
    for(ll j=1;j<=cnt;j++){
        for(ll i=1;i<=m&&p[j]<=v[i]/p[j];i++){
            g1[i]=(g1[i]-p[j]*(g1[get(v[i]/p[j])]-g1[get(p[j-1])])%mod+mod)%mod;
            g1[i]=(g1[i]+mod)%mod;
            g2[i]=(g2[i]-sq(p[j])*(g2[get(v[i]/p[j])]-g2[get(p[j-1])])%mod+mod)%mod;
            g2[i]=(g2[i]+mod)%mod;
        }
    }
}

ll S(ll x,ll y){
    if(p[y]>=x) return 0;
    ll res=(g2[get(x)]-g1[get(x)]-g2[get(p[y])]+g1[get(p[y])]+mod)%mod;//g(n);
    for(ll i=y+1;i<=cnt&&p[i]<=x/p[i];i++){
        ll P=p[i];
        for(ll j=1;P<=x/p[i];j++,P*=p[i]){//p^e
            res=(res+F(P)*S(x/P,i)%mod+F(P*p[i]))%mod;
        }
    }
    return res;
}

ll qpow(ll x,ll y){
    if(y==1) return x;
    ll res=qpow(x,y>>1);
    if(y&1) return (((res*res)%mod)*x)%mod;
    return (res*res)%mod;
}ll inv(ll x){return qpow(x,mod-2);}

int main(){
    inv2=inv(2),inv6=inv(6);
    n=read();
    ll T=sqrt(n)+1;
    prime(T);
    getg();
    ll ans=(S(n,0)+1+mod)%mod;
    cout<<ans;
    return 0;
}
```

线性基

```
struct LB
{
    static constexpr int L=60;
    array<i64,L+1> a{};

    LB(){}

    LB(const vector<i64> &v) { init(v); }

    bool insert(i64 t)
    {
        for (int i=L;i>=0;i--)
            if (t&(1ll<<i))
            {
                if (!a[i])
                {
                    a[i]=t;
                    return 1;
                }
                else t^=a[i];
            }
```

```
        return 0;
    }

    void init(const vector<i64> &v) { for (auto x:v) insert(x); }

    bool check(i64 t)
    {
        for (int i=L;i>=0;i--)
            if (t&(1ll<<i))
                if (!a[i]) return 0;
                else t^=a[i];
        return 1;
    }

    i64 QueryMax()
    {
        i64 res=0;
        for (int i=L;i>=0;i--)
            res=max(res,res^a[i]);
        return res;
    }

    i64 QueryMin()
    {
        for (int i=0;i<=L;i++)
            if (a[i]) return a[i];
        return 0;
    }

    i64 QueryKth(int k)
    {
        i64 res=0;
        int cnt=0;
        array<i64,L+1> tmp{};
        for (int i=0;i<=L;i++)
        {
            for (int j=i-1;j>=0;j--)
                if (a[i]&(1ll<<j)) a[i]^=a[j];
            if (a[i]) tmp[cnt++]=a[i];
        }
        if (k>=(1ll<<cnt)) return -1;
        for (int i=0;i<cnt;i++)
            if (k&(1ll<<i)) res^=tmp[i];
        return res;
    }
};
```

数据结构

并查集（启发式合并 + 带撤销）

```
struct DSU
{
    int n=0;
    vector<int> fa,siz;
    stack<int> s;

    DSU(int n) { init(n); }

    void init(int n)
    {
        fa.resize(n);
        iota(fa.begin(),fa.end(),0);
        siz.assign(n,1);
        while (!s.empty()) s.pop();
    }

    int get(int x) { return fa[x]==x?x:get(fa[x]); }

    void merge(int x,int y)
```

```
    {
        x=get(x),y=get(y);
        if (x==y) return;
        if (siz[x]<siz[y]) swap(x,y);
        s.push(y),fa[y]=x,siz[x]+=siz[y];
    }

    void undo()
    {
        if (s.empty()) return;
        int y=s.top();
        s.pop();
        siz[fa[y]]-=siz[y];
        fa[y]=y;
    }

    void back(int t=0) { while (s.size()>t) undo(); }
};
```

状压 RMQ

```
template <class T,class Cmp=less<T>>
struct RMQ
{
    const Cmp cmp=Cmp();
    static constexpr unsigned B=64;
    using u64=unsigned long long;
    int n;
    vector<vector<T>> a;
    vector<T> pre,suf,ini;
    vector<u64> stk;

    RMQ() {}
    RMQ(const vector<T> &v) { init(v); }

    void init(const vector<T> &v)
    {
        n=v.size();
        pre=suf=ini=v;
        stk.resize(n);
        if (!n) return;
        const int M=(n-1)/B+1;
        const int lg=__lg(M);
        a.assign(lg+1,vector<T>(M));
        for (int i=0;i<M;i++)
        {
            a[0][i]=v[i*B];
            for (int j=1;j<B&&i*B+j<n;j++)
                a[0][i]=min(a[0][i],v[i*B+j],cmp);
        }
        for (int i=1;i<n;i++)
            if (i%B) pre[i]=min(pre[i],pre[i-1],cmp);
        for (int i=n-2;i>=0;i--)
            if (i%B!=B-1) suf[i]=min(suf[i],suf[i+1],cmp);
        for (int j=0;j<lg;j++)
            for (int i=0;i+(2<<j)<=M;i++)
                a[j+1][i]=min(a[j][i],a[j][i+(1<<j)],cmp);
        for (int i=0;i<M;i++)
        {
            const int l=i*B;
            const int r=min(1U*n,l+B);
            u64 s=0;
            for (int j=l;j<r;j++)
            {
                while (s&&cmp(v[j],v[__lg(s)+l])) s^=1ULL<<__lg(s);
                s|=1ULL<<(j-l);
                stk[j]=s;
            }
        }
    }

```

```
    //查询区间 [l,r) 的 RMQ
    T operator()(int l,int r)
    {
        if (l/B!=(r-1)/B)
        {
            T ans=min(suf[l],pre[r-1],cmp);
            l=l/B+1,r=r/B;
            if (l<r)
            {
                int k=__lg(r-l);
                ans=min({ans,a[k][l],a[k][r-(1<<k)]},cmp);
            }
            return ans;
        }
        else
        {
            int x=B*(l/B);
            return ini[__builtin_ctzll(stk[r-1]>>(l-x))+l];
        }
    }
};
```

树状数组

```
template <class T>
struct BIT
{
    int n;
    vector<T> a;

    BIT(int n_=0) { init(n_); }

    void init(int n_)
    {
        n=n_;
        a.assign(n,T{});
    }

    void add(int x,const T &v)
    {
        for (int i=x+1;i<=n;i+=i&-i)
            a[i-1]=a[i-1]+v;
    }

    //查询区间 [0,x)
    T sum(int x)
    {
        T ans{};
        for (int i=x;i>0;i-=i&-i)
            ans=ans+a[i-1];
        return ans;
    }

    //查询区间 [l,r)
    T rangeSum(int l,int r) { return sum(r)-sum(l); }

    int select(const T &k)
    {
        int x=0;
        T cur{};
        for (int i=1<<__lg(n);i;i>>=1)
        {
            if (x+i<=n&&cur+a[x+i-1]<=k)
            {
                x+=i;
                cur=cur+a[x-1];
            }
        }
        return x;
    }
};
```

线段树

```
template <class Info,class Tag>
struct SGT
{
    int n;
    vector<Info> info;
    vector<Tag> tag;

    SGT():n(0) {}
    SGT(int n_,Info v_=Info()) { init(n_,v_); }

    template <class T>
    SGT(vector<T> init_) { init(init_); }

    void init(int n_,Info v_=Info()) { init(vector(n_,v_)); }

    template <class T>
    void init(vector<T> init_)
    {
        n=init_.size();
        info.assign(4<<__lg(n),Info());
        tag.assign(4<<__lg(n),Tag());
        function<void(int,int,int)> build=[&](int p,int l,int r)
        {
            if (r-l==1)
            {
                info[p]=init_[l];
                return;
            }
            int m=(l+r)>>1;
            build(p<<1,l,m);
            build(p<<1|1,m,r);
            pushup(p);
        };
        build(1,0,n);
    }

    void pushup(int p) { info[p]=info[p<<1]+info[p<<1|1]; }

    void apply(int p,const Tag &v)
    {
        info[p].apply(v);
        tag[p].apply(v);
    }

    void pushdown(int p)
    {
        apply(p<<1,tag[p]);
        apply(p<<1|1,tag[p]);
        tag[p]=Tag();
    }

    void modify(int p,int l,int r,int x,const Info &v)
    {
        if (r-l==1)
        {
            info[p]=v;
            return;
        }
        int m=(l+r)>>1;
        pushdown(p);
        if (x<m) modify(p<<1,l,m,x,v);
        else modify(p<<1|1,m,r,x,v);
        pushup(p);
    }

    //O(log n) 单点修改
    void modify(int p,const Info &v) { modify(1,0,n,p,v); }

    Info rangeQuery(int p,int l,int r,int x,int y)
```

```
    {
        if (l>=y||r<=x) return Info();
        if (l>=x&&r<=y) return info[p];
        int m=(l+r)>>1;
        pushdown(p);
        return rangeQuery(p<<1,l,m,x,y)+rangeQuery(p<<1|1,m,r,x,y);
    }

    //O(log n) 区间查询 [l,r)
    Info rangeQuery(int l,int r) { rangeQuery(1,0,n,l,r); }

    void rangeApply(int p,int l,int r,int x,int y,const Tag &v)
    {
        if (l>=y||r<=x) return;
        if (l>=x&&r<=y)
        {
            apply(p,v);
            return;
        }
        int m=(l+r)>>1;
        pushdown(p);
        rangeApply(p<<1,l,m,x,y,v);
        rangeApply(p<<1|1,m,r,x,y,v);
        pushup(p);
    }

    //O(log n) 区间操作 [l,r)
    void rangeApply(int l,int r,const Tag &v) { rangeApply(1,0,n,l,r,v); }

    //O(log n) 区间 [l,r) 内查找第一个合法位置
    template <class F>
    int findFirst(int p,int l,int r,int x,int y,F pred)
    {
        if (l>=y||r<=x||!pred(info[p])) return -1;
        if (r-l==1) return l;
        int m=(l+r)>>1;
        pushdown(p);
        int res=findFirst(p<<1,l,m,x,y,pred);
        if (res==-1) res=findFirst(p<<1|1,m,r,x,y,pred);
        return res;
    }

    template <class F>
    int findFirst(int l,int r,F pred) { return findFirst(1,0,n,l,r,pred); }

    template <class F>
    int findLast(int p,int l,int r,int x,int y,F pred)
    {
        if (l>=y||r<=x||!pred(info[p])) return -1;
        if (r-l==1) return l;
        int m=(l+r)>>1;
        pushdown(p);
        int res=findFirst(p<<1|1,m,r,x,y,pred);
        if (res==-1) res=findFirst(p<<1,l,m,x,y,pred);
        return res;
    }

    template <class F>
    int findLast(int l,int r,F pred) { return findLast(1,0,n,l,r,pred); }
};

//这里默认乘法优先 (x*a+b)*c+d=x*(a*c)+(b*c+d)
struct Tag
{
    i64 a=1,b=0;
    void apply(Tag t)
    {
        a*=t.a;
        b=b*t.a+t.b;
    }
};
```

```
struct Info
{
    i64 x=0,l=0,r=0;
    void apply(Tag t)
    {
        int len=r-l+1;
        x=x*t.a+len*t.b;
    }
};

Info operator + (Info a,Info b)
{
    return {a.x+b.x,min(a.l,b.l),max(a.r,b.r)};
}
```

字符串

字符串哈希（随机模数）

```
bool isPrime(int n)
{
    if (n<=1) return 0;
    for (int i=2;i*i<=n;i++)
        if (n%i==0) return 0;
    return 1;
}

int findPrime(int n)
{
    while (!isPrime(n)) n++;
    return n;
}

mt19937 rng(time(0));
const int P=findPrime(rng()%900000000+100000000);
struct StrHash
{
    int n;
    vector<int> h,p;

    StrHash(const string &s){ init(s); }

    void init(const string &s)
    {
        n=s.size();
        h.resize(n+1);
        p.resize(n+1);
        p[0]=1;
        for (int i=0;i<n;i++) h[i+1]=(10ll*h[i]+s[i]-'a')%P;
        for (int i=0;i<n;i++) p[i+1]=10ll*p[i]%P;
    }

    //查询 [l,r) 的区间哈希
    int get(int l,int r) { return (h[r]+1ll*(P-h[l])*p[r-l])%P; }
};
```

KMP

```
vector<int> KMP(const string &s)
{
    int now=0;
    vector<int> pre(s.size(),0);
    for (int i=1;i<s.size();i++)
    {
        while (now&&s[i]!=s[now]) now=pre[now-1];
        if (s[i]==s[now]) now++;
        pre[i]=now;
    }
```

```
    return pre;
}
```

Z 函数

```
vector<int> zFunction(string s)
{
    int n=s.size();
    vector<int> z(n);
    z[0]=n;
    for (int i=1,j=1;i<n;i++)
    {
        z[i]=max(0,min(j+z[j]-i,z[i-j]));
        while (i+z[i]<n&&s[z[i]]==s[i+z[i]]) z[i]++;
        if (i+z[i]>j+z[j]) j=i;
    }
    return z;
}
```

AC 自动机

```
struct ACAM
{
    static constexpr int ALPHABET=26;
    struct Node
    {
        int len;
        int link;
        array<int,ALPHABET> next;
        Node():len{0},link{0},next{}{}
    };

    vector<Node> t;

    ACAM() { init(); }

    void init()
    {
        t.assign(2,Node());
        t[0].next.fill(1);
        t[0].len=-1;
    }

    int newNode()
    {
        t.emplace_back();
        return t.size()-1;
    }

    int add(const string &a)
    {
        int p=1;
        for (auto c:a)
        {
            int x=c-'a';
            if (t[p].next[x]==0)
            {
                t[p].next[x]=newNode();
                t[t[p].next[x]].len=t[p].len+1;
            }
            p=t[p].next[x];
        }
        return p;
    }

    void work()
    {
        queue<int> q;
        q.push(1);
        while (!q.empty())
```

```
        {
            int x=q.front();
            q.pop();
            for (int i=0;i<ALPHABET;i++)
            {
                if (t[x].next[i]==0) t[x].next[i]=t[t[x].link].next[i];
                else
                {
                    t[t[x].next[i]].link=t[t[x].link].next[i];
                    q.push(t[x].next[i]);
                }
            }
        }
    }

    int next(int p,int x) { return t[p].next[x]; }

    int link(int p) { return t[p].link; }

    int size() { return t.size(); }
};
```

后缀数组

```
struct SA
{
    int n;
    vector<int> sa,rk,lc;
    SA(const string &s)
    {
        n=s.length();
        sa.resize(n);
        rk.resize(n);
        lc.resize(n-1);
        iota(sa.begin(),sa.end(),0);
        sort(sa.begin(),sa.end(),[&](int a,int b){ return s[a]<s[b]; });
        rk[sa[0]]=0;
        for (int i=1;i<n;i++) rk[sa[i]]=rk[sa[i-1]]+(s[sa[i]]!=s[sa[i-1]]);
        int k=1;
        vector<int> tmp,cnt(n);
        tmp.reserve(n);
        while (rk[sa[n-1]]<n-1)
        {
            tmp.clear();
            for (int i=0;i<k;i++) tmp.push_back(n-k+i);
            for (auto i:sa)
                if (i>=k) tmp.push_back(i-k);
            fill(cnt.begin(),cnt.end(),0);
            for (int i=0;i<n;i++) cnt[rk[i]]++;
            for (int i=1;i<n;i++) cnt[i]+=cnt[i-1];
            for (int i=n-1;i>=0;i--) sa[--cnt[rk[tmp[i]]]]=tmp[i];
            swap(rk,tmp);
            rk[sa[0]]=0;
            for (int i=1;i<n;i++)
                rk[sa[i]]=rk[sa[i-1]]+(tmp[sa[i-1]]<tmp[sa[i]]||sa[i-1]+k==n||tmp[sa[i-1]+k]<tmp[sa[i]+k]);
            k<<=1;
        }
        for (int i=0,j=0;i<n;i++)
        {
            if (rk[i]==0) j=0;
            else
            {
                for (j-=j>0;i+j<n&&sa[rk[i]-1]+j<n&&s[i+j]==s[sa[rk[i]-1]+j];) j++;
                lc[rk[i]-1]=j;
            }//lc[i]:lcp(sa[i],sa[i+1]),lcp(sa[i],sa[j])=min{lc[i...j-1]}
        }
    }
};
```

（广义）后缀自动机

```cpp
struct SAM
{
    static constexpr int ALPHABET=26;
    struct Node
    {
        int len;
        int link;
        array<int,ALPHABET> next;
        Node():len{},link{},next{} {}
    };

    vector<Node> t;

    SAM() { init(); }

    void init()
    {
        t.assign(2,Node());
        t[0].next.fill(1);
        t[0].len=-1;
    }

    int newNode()
    {
        t.emplace_back();
        return t.size()-1;
    }

    int extend(int lst,int c)
    {
        if (t[lst].next[c]&&t[t[lst].next[c]].len==t[lst].len+1)
            return t[lst].next[c];
        int p=lst,np=newNode(),flag=0;
        t[np].len=t[p].len+1;
        while (!t[p].next[c])
        {
            t[p].next[c]=np;
            p=t[p].link;
        }
        if (!p)
        {
            t[np].link=1;
            return np;
        }
        int q=t[p].next[c];
        if (t[q].len==t[p].len+1)
        {
            t[np].link=q;
            return np;
        }
        if (p==lst) flag=1,np=0,t.pop_back();
        int nq=newNode();
        t[nq].link=t[q].link;
        t[nq].next=t[q].next;
        t[nq].len=t[p].len+1;
        t[q].link=t[np].link=nq;
        while (p&&t[p].next[c]==q)
        {
            t[p].next[c]=nq;
            p=t[p].link;
        }
        return flag?nq:np;
    }

    int add(const string &a)
    {
        int p=1;
        for (auto c:a) p=extend(p,c-'a');
        return p;
```

```
    }

    int next(int p,int x) { return t[p].next[x]; }

    int link(int p) { return t[p].link; }

    int len(int p) { return t[p].len; }

    int size() { return t.size(); }
};
```

Manacher

```
vector<int> manacher(vector<int> s)
{
    vector<int> t{0};
    for (auto c:s)
    {
        t.push_back(c);
        t.push_back(0);
    }
    int n=t.size();
    vector<int> r(n);
    for (int i=0,j=0;i<n;i++)
    {
        if (j*2-i>=0&&j+r[j]>i) r[i]=min(r[j*2-i],j+r[j]-i);
        while (i-r[i]>=0&&i+r[i]<n&&t[i-r[i]]==t[i+r[i]]) r[i]++;
        if (i+r[i]>j+r[j]) j=i;
    }
    return r;
}
```

回文自动机

```
struct PAM
{
    static constexpr int ALPHABET_SIZE=28;
    struct Node
    {
        int len,link,cnt;
        array<int,ALPHABET_SIZE> next;
        Node():len{},link{},cnt{},next{}{}
    };
    vector<Node> t;
    int suff;
    string s;

    PAM() { init(); }

    void init()
    {
        t.assign(2,Node());
        t[0].len=-1;
        suff=1;
        s.clear();
    }

    int newNode()
    {
        t.emplace_back();
        return t.size()-1;
    }

    bool add(char c,char offset='a')
    {
        int pos=s.size();
        s+=c;
        int let=c-offset;
        int cur=suff,curlen=0;
        while (1)
```

```
        {
            curlen=t[cur].len;
            if (pos-curlen-1>=0&&s[pos-curlen-1]==s[pos]) break;
            cur=t[cur].link;
        }
        if (t[cur].next[let])
        {
            suff=t[cur].next[let];
            return 0;
        }
        int num=newNode();
        suff=num;
        t[num].len=t[cur].len+2;
        t[cur].next[let]=num;
        if (t[num].len==1)
        {
            t[num].link=t[num].cnt=1;
            return 1;
        }
        while (1)
        {
            cur=t[cur].link;
            curlen=t[cur].len;
            if (pos-curlen-1>=0&&s[pos-curlen-1]==s[pos])
            {
                t[num].link=t[cur].next[let];
                break;
            }
        }
        t[num].cnt=t[t[num].link].cnt+1;
        return 1;
    }
};
```

图论

强连通分量

```
struct SCC
{
    int n,cur,cnt;
    vector<vector<int>> adj;
    vector<int> stk,dfn,low,bel;

    SCC() {}
    SCC(int n) { init(n); }

    void init(int n)
    {
        this->n=n;
        adj.assign(n,{});
        stk.clear();
        dfn.assign(n,-1);
        low.resize(n);
        bel.assign(n,-1);
        cur=cnt=0;
    }

    void add(int u,int v) { adj[u].push_back(v); }

    void dfs(int x)
    {
        dfn[x]=low[x]=cur++;
        stk.push_back(x);
        for (auto y:adj[x])
        {
            if (dfn[y]==-1)
            {
                dfs(y);
                low[x]=min(low[x],low[y]);
```

```
            }
            else if (bel[y]==-1) low[x]=min(low[x],dfn[y]);
        }
        if (dfn[x]==low[x])
        {
            int y;
            do
            {
                y=stk.back();
                bel[y]=cnt;
                stk.pop_back();
            } while (y!=x);
            cnt++;
        }
    }

    vector<int> work()
    {
        for (int i=0;i<n;i++)
            if (dfn[i]==-1) dfs(i);
        return bel;
    }

    struct Graph
    {
        int n;
        vector<pair<int,int>> edges;
        vector<int> siz,cnte;
    };

    Graph compress()
    {
        Graph G;
        G.n=cnt;
        G.siz.resize(cnt);
        G.cnte.resize(cnt);
        for (int i=0;i<n;i++)
        {
            G.siz[bel[i]]++;
            for (auto j:adj[i])
                if (bel[i]!=bel[j])
                    G.edges.emplace_back(bel[j],bel[i]);
        }
        return G;
    };
};
```

边双连通分量

```
struct EBCC
{
    int n;
    vector<vector<int>> adj;
    vector<int> stk,dfn,low,bel;
    int cur,cnt;

    EBCC() {}
    EBCC(int n) { init(n); }

    void init(int n)
    {
        this->n=n;
        adj.assign(n,{});
        dfn.assign(n,-1);
        low.resize(n);
        bel.assign(n,-1);
        stk.clear();
        cur=cnt=0;
    }

    void add(int u,int v)
```

```
    {
        adj[u].push_back(v);
        adj[v].push_back(u);
    }

    void dfs(int x,int p)
    {
        dfn[x]=low[x]=cur++;
        stk.push_back(x);
        for (auto y:adj[x])
        {
            if (y==p) continue;
            if (dfn[y]==-1)
            {
                dfs(y,x);
                low[x]=min(low[x],low[y]);
            }
            else if (bel[y]==-1&&dfn[y]<dfn[x]) low[x]=min(low[x],dfn[y]);
        }
        if (dfn[x]==low[x])
        {
            int y;
            do
            {
                y=stk.back();
                bel[y]=cnt;
                stk.pop_back();
            } while (y!=x);
            cnt++;
        }
    }

    vector<int> work()
    {
        dfs(0,-1);
        return bel;
    }

    struct Graph
    {
        int n;
        vector<pair<int,int>> edges;
        vector<int> siz,cnte;
    };

    Graph compress()
    {
        Graph G;
        G.n=cnt;
        G.siz.resize(cnt);
        G.cnte.resize(cnt);
        for (int i=0;i<n;i++)
        {
            G.siz[bel[i]]++;
            for (auto j:adj[i])
            {
                if (bel[i]<bel[j]) G.edges.emplace_back(bel[i],bel[j]);
                else if (i<j) G.cnte[bel[i]]++;
            }
        }
        return G;
    };
};
```

轻重链剖分

```
struct HLD
{
    int n;
    vector<int> siz,top,dep,pa,in,out,seq;
    vector<vector<int>> adj;
```

```
    int cur;

    HLD(){}
    HLD(int n) { init(n); }

    void init(int n)
    {
        this->n=n;
        siz.resize(n);
        top.resize(n);
        dep.resize(n);
        pa.resize(n);
        in.resize(n);
        out.resize(n);
        seq.resize(n);
        cur=0;
        adj.assign(n,{});
    }

    void addEdge(int u,int v)
    {
        adj[u].push_back(v);
        adj[v].push_back(u);
    }

    void work(int rt=0)
    {
        top[rt]=rt;
        dep[rt]=0;
        pa[rt]=-1;
        dfs1(rt);
        dfs2(rt);
    }

    void dfs1(int u)
    {
        if (pa[u]!=-1) adj[u].erase(find(adj[u].begin(),adj[u].end(),pa[u]));
        siz[u]=1;
        for (auto &v:adj[u])
        {
            pa[v]=u;
            dep[v]=dep[u]+1;
            dfs1(v);
            siz[u]+=siz[v];
            if (siz[v]>siz[adj[u][0]])
                swap(v,adj[u][0]);
        }
    }

    void dfs2(int u)
    {
        in[u]=cur++;
        seq[in[u]]=u;
        for (auto v:adj[u])
        {
            top[v]=(v==adj[u][0])?top[u]:v;
            dfs2(v);
        }
        out[u]=cur;
    }

    int lca(int u,int v)
    {
        while (top[u]!=top[v])
        {
            if (dep[top[u]]>dep[top[v]]) u=pa[top[u]];
            else v=pa[top[v]];
        }
        return dep[u]<dep[v]?u:v;
    }

```

```
    int dist(int u,int v) { return dep[u]+dep[v]-(dep[lca(u,v)]<<1); }

    int jump(int u,int k)
    {
        if (dep[u]<k) return -1;
        int d=dep[u]-k;
        while (dep[top[u]]>d) u=pa[top[u]];
        return seq[in[u]-dep[u]+d];
    }

    bool isAncester(int u,int v) { return in[u]<=in[v]&&in[v]<out[u]; }

    int rootedParent(int u,int v)//u->root,v->point
    {
        if (u==v) return u;
        if (!isAncester(v,u)) return pa[v];
        auto it=upper_bound(adj[v].begin(),adj[v].end(),u,[&](int x,int y){ return in[x]<in[y]; })-1;
        return *it;
    }

    int rootedSize(int u,int v)//same as rootedParent
    {
        if (u==v) return n;
        if (!isAncester(v,u)) return siz[v];
        return n-siz[rootedParent(u,v)];
    }

    int rootedLca(int a,int b,int c) { return lca(a,b)^lca(b,c)^lca(c,a); }
};
```

最大流

```
bool bfs(ll s,ll t){
    queue<ll>q;
    for(int i=1;i<=n;i++) dis[i]=-1;dis[s]=0;
    for(int i=1;i<=n;i++) cur[i]=frm[i];
    q.push(s);
    while(!q.empty()){
        ll x=q.front();q.pop();
        for(int i=frm[x];i;i=e[i].net){
            ll v=e[i].to;
            if(dis[v]==-1&&e[i].val) dis[v]=dis[x]+1,q.push(v);
        }
    }
    return dis[t]!=-1;
}

ll dfs(ll s,ll flow){
    if(s==t||!flow) return flow;
    if(dis[s]>=dis[t]) return 0;
    ll now=0,res=0;
    for(int i=cur[s];i;i=e[i].net){
        ll x=e[i].to;
        if(dis[x]==dis[s]+1&&e[i].val){
            res=dfs(x,min(flow-now,e[i].val));
            if(!res) continue;
            e[i].val-=res;e[i^1].val+=res;now+=res;flow-=res;
            cur[s]=i;
            if(!flow) break;
        }
    }
    if(!now) return (dis[s]=0);
    return now;
}

void dinic(ll s,ll t){
    while(bfs(s,t)){
        ll last=dfs(s,1e18);
        while(last) ans+=last,last=dfs(s,1e18);
    }
}
```

最大流条件下最小费用（费用流）

```
bool bfs(ll s,ll t){
    queue<ll>q;
    memset(dis,127,sizeof dis);memset(vis,0,sizeof vis);memset(pre,-1,sizeof pre);
    ll inf=dis[0];dis[s]=0;vis[s]=1;
    dis[s]=pre[s]=0;flow[s]=inf;q.push(s);
    while(!q.empty()){
        ll x=q.front();q.pop();vis[x]=0;
        for(int i=frm[x];i;i=e[i].net){
            ll v=e[i].to;
            if(dis[v]>dis[x]+e[i].cost&&e[i].val){
                dis[v]=dis[x]+e[i].cost;
                xb[v]=i;pre[v]=x;flow[v]=min(flow[x],e[i].val);
                if(!vis[v]) q.push(v),vis[v]=1;
            }
        }
    }
    if(dis[t]==inf) return 0;
    return 1;
}

void EK(ll s,ll t){
    while(bfs(s,t)){
        ll k=t;
        while(s!=k){
            e[xb[k]].val-=flow[t];e[xb[k]^1].val+=flow[t];
            k=pre[k];
        }
        maxflow+=flow[t];mincost+=flow[t]*dis[t];
    }
}
//add(x,y,v,c);add(x,y,0,-c);
```