Algorithm Library

magic::team.getname()

South China Normal University

September 1, 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)
{
```

```
15        return b?gcd(b,a%b):a;
16    }
```

数学

欧拉筛

```
1    vector<int> minp,primes;
2
3    void sieve(int n)
4    {
5        minp.assign(n+1,0);
6        primes.clear();
7        for (int i=2;i<=n;i++)
8        {
9            if (!minp[i])
10           {
11               minp[i]=i;
12               primes.push_back(i);
13           }
14           for (auto p:primes)
15           {
16               if (i*p>n) break;
17               minp[i*p]=p;
18               if (p==minp[i]) break;
19           }
20       }
21   }
```

取模类（MInt）

```
1    template <class T>
2    constexpr T power(T a,i64 b)
3    {
4        T res=1;
5        for (;b;b>>=1,a*=a)
6            if (b&1) res*=a;
7        return res;
8    }
9
10   template <int P>
11   struct MInt
12   {
13       int x;
14       constexpr MInt():x{} {}
15       constexpr MInt(i64 x):x{norm(x%getMod())} {}
16
17       static int Mod;
18       constexpr static int getMod()
19       {
20           if (P>0) return P;
21           else return Mod;
22       }
23
24       constexpr static void setMod(int Mod_) { Mod=Mod_; }
25
26       constexpr int norm(int x) const
27       {
28           if (x<0) x+=getMod();
29           if (x>=getMod()) x-=getMod();
30           return x;
31       }
32
33       constexpr int val() const { return x; }
34
35       explicit constexpr operator int () const { return x; }
36
37       constexpr MInt operator - () const
38       {
```

```cpp
39          MInt res;
40          res.x=norm(getMod()-x);
41          return res;
42      }
43
44      constexpr MInt inv() const
45      {
46          assert(x!=0);
47          return power(*this,getMod()-2);
48      }
49
50      constexpr MInt &operator *= (MInt rhs) &
51      {
52          x=1ll*x*rhs.x%getMod();
53          return *this;
54      }
55
56      constexpr MInt &operator += (MInt rhs) &
57      {
58          x=norm(x+rhs.x);
59          return *this;
60      }
61
62      constexpr MInt &operator -= (MInt rhs) &
63      {
64          x=norm(x-rhs.x);
65          return *this;
66      }
67
68      constexpr MInt &operator /= (MInt rhs) &
69      {
70          return *this*=rhs.inv();
71      }
72
73      friend constexpr MInt operator * (MInt lhs,MInt rhs)
74      {
75          MInt res=lhs;
76          res*=rhs;
77          return res;
78      }
79
80      friend constexpr MInt operator + (MInt lhs,MInt rhs)
81      {
82          MInt res=lhs;
83          res+=rhs;
84          return res;
85      }
86
87      friend constexpr MInt operator - (MInt lhs,MInt rhs)
88      {
89          MInt res=lhs;
90          res-=rhs;
91          return res;
92      }
93
94      friend constexpr MInt operator / (MInt lhs,MInt rhs)
95      {
96          MInt res=lhs;
97          res/=rhs;
98          return res;
99      }
100
101     friend constexpr istream &operator >> (istream &is,MInt &a)
102     {
103         i64 v;
104         is>>v;
105         a=MInt(v);
106         return is;
107     }
108
109     friend constexpr ostream &operator << (ostream &os,const MInt &a) { return os<<a.val(); }
```

```cpp
      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

```cpp
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;}
```

```cpp
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        }
```

```
127            g.resize(len);
128            return g;
129        }
130
131        poly Pow(poly f,ll k){
132            f=ln(f);f=f*k;f=exp(f);
133            return f;
134        }
135
136        poly sqrt(poly f){
137            poly g=poly(1,1);
138            ll len=plen(f);
139            ll inv2=ginv(2);
140            for(int i=2;i<(len<<2);i<<=1){
141                poly A=f;
142                A.resize(i);
143                g=(g+(A*inv(g)))*inv2;
144            }
145            g.resize(len);
146            return g;
147        }
148    }
149    using namespace polygone;
```

杜教筛

```
1   void prime(){
2       u[1]=1;v[1]=1;phi[1]=1;
3       for(ll i=2;i<=N;i++){
4           if(!v[i]) {u[i]=-1;p[++cnt]=i;phi[i]=i-1;}
5           for(ll j=1;j<=cnt&&p[j]*i<=N;j++){
6               v[p[j]*i]=1;
7               if(!(i%p[j])) {phi[i*p[j]]=phi[i]*p[j];break;}
8               else u[i*p[j]]=-u[i],phi[i*p[j]]=phi[i]*phi[p[j]];
9           }
10      }
11      for(ll i=1;i<=N;i++) u[i]+=u[i-1],phi[i]+=phi[i-1];
12  }
13
14  ll sumu(ll n) {
15      if(n<=N) return u[n];
16      if(M[n]) return M[n];
17      ll res=1;
18      for(ll l=2,r=0;l<=n;l=r+1){
19          r=n/(n/l);
20          res-=1ll*sumu(n/l)*(r-l+1);
21      }
22      M[n]=res;
23      return M[n];
24  }
```

Min_25 筛

```
1   void prime(ll n){
2       for(ll i=2;i<=n;i++){
3           if(!vis[i]) p[++cnt]=i;
4           for(ll j=1;j<=cnt&&p[j]<=n/i;j++){
5               vis[i*p[j]]=1;
6               if(i%p[j]==0) break;
7           }
8       }
9   }
10
11  ll qz1(ll x){return x%=mod,x*(x+1)%mod*inv2%mod;}
12  ll qz2(ll x){return x%=mod,x*(x+1)%mod*(2*x+1)%mod*inv6%mod;}
13  ll get(ll x){return x<N?id1[x]:id2[n/x];}
14  ll sq(ll x) {return x%=mod,x*x%mod;}
15  ll F(ll x)  {return x%=mod,(sq(x)+mod-x)%mod;}
16
17  void getg(){
```

```
18      for(ll l=1,r;l<=n;l=r+1){
19          r=n/(n/l);
20          v[++m]=n/l;
21          if(v[m]<N) id1[v[m]]=m;
22          else id2[n/v[m]]=m;
23          g1[m]=(qz1(v[m])-1+mod)%mod;
24          g2[m]=(qz2(v[m])-1+mod)%mod;
25      }
26      for(ll j=1;j<=cnt;j++){
27          for(ll i=1;i<=m&&p[j]<=v[i]/p[j];i++){
28              g1[i]=(g1[i]-p[j]*(g1[get(v[i]/p[j])]-g1[get(p[j-1])])%mod+mod)%mod;
29              g1[i]=(g1[i]+mod)%mod;
30              g2[i]=(g2[i]-sq(p[j])*(g2[get(v[i]/p[j])]-g2[get(p[j-1])])%mod+mod)%mod;
31              g2[i]=(g2[i]+mod)%mod;
32          }
33      }
34  }
35
36  ll S(ll x,ll y){
37      if(p[y]>=x) return 0;
38      ll res=(g2[get(x)]-g1[get(x)]-g2[get(p[y])]+g1[get(p[y])]+mod)%mod;//g(n);
39      for(ll i=y+1;i<=cnt&&p[i]<=x/p[i];i++){
40          ll P=p[i];
41          for(ll j=1;P<=x/p[i];j++,P*=p[i]){//p^e
42              res=(res+F(P)*S(x/P,i)%mod+F(P*p[i]))%mod;
43          }
44      }
45      return res;
46  }
47
48  ll qpow(ll x,ll y){
49      if(y==1) return x;
50      ll res=qpow(x,y>>1);
51      if(y&1) return (((res*res)%mod)*x)%mod;
52      return (res*res)%mod;
53  }ll inv(ll x){return qpow(x,mod-2);}
54
55  int main(){
56      inv2=inv(2),inv6=inv(6);
57      n=read();
58      ll T=sqrt(n)+1;
59      prime(T);
60      getg();
61      ll ans=(S(n,0)+1+mod)%mod;
62      cout<<ans;
63      return 0;
64  }
```

数据结构

并查集（启发式合并 + 带撤销）

```
1   struct DSU
2   {
3       int n=0;
4       vector<int> fa,siz;
5       stack<int> s;
6
7       DSU(int n) { init(n); }
8
9       void init(int n)
10      {
11          fa.resize(n);
12          iota(fa.begin(),fa.end(),0);
13          siz.assign(n,1);
14          while (!s.empty()) s.pop();
15      }
16
17      int get(int x) { return fa[x]==x?x:get(fa[x]); }
18
```

```
19      void merge(int x,int y)
20      {
21          x=get(x),y=get(y);
22          if (x==y) return;
23          if (siz[x]<siz[y]) swap(x,y);
24          s.push(y),fa[y]=x,siz[x]+=siz[y];
25      }
26
27      void undo()
28      {
29          if (s.empty()) return;
30          int y=s.top();
31          s.pop();
32          siz[fa[y]]-=siz[y];
33          fa[y]=y;
34      }
35
36      void back(int t=0) { while (s.size()>t) undo(); }
37  };
```

状压 RMQ

```
1   template <class T,class Cmp=less<T>>
2   struct RMQ
3   {
4       const Cmp cmp=Cmp();
5       static constexpr unsigned B=64;
6       using u64=unsigned long long;
7       int n;
8       vector<vector<T>> a;
9       vector<T> pre,suf,ini;
10      vector<u64> stk;
11
12      RMQ() {}
13      RMQ(const vector<T> &v) { init(v); }
14
15      void init(const vector<T> &v)
16      {
17          n=v.size();
18          pre=suf=ini=v;
19          stk.resize(n);
20          if (!n) return;
21          const int M=(n-1)/B+1;
22          const int lg=__lg(M);
23          a.assign(lg+1,vector<T>(M));
24          for (int i=0;i<M;i++)
25          {
26              a[0][i]=v[i*B];
27              for (int j=1;j<B&&i*B+j<n;j++)
28                  a[0][i]=min(a[0][i],v[i*B+j],cmp);
29          }
30          for (int i=1;i<n;i++)
31              if (i%B) pre[i]=min(pre[i],pre[i-1],cmp);
32          for (int i=n-2;i>=0;i--)
33              if (i%B!=B-1) suf[i]=min(suf[i],suf[i+1],cmp);
34          for (int j=0;j<lg;j++)
35              for (int i=0;i+(2<<j)<=M;i++)
36                  a[j+1][i]=min(a[j][i],a[j][i+(1<<j)],cmp);
37          for (int i=0;i<M;i++)
38          {
39              const int l=i*B;
40              const int r=min(1U*n,l+B);
41              u64 s=0;
42              for (int j=l;j<r;j++)
43              {
44                  while (s&&cmp(v[j],v[__lg(s)+l])) s^=1ULL<<__lg(s);
45                  s|=1ULL<<(j-l);
46                  stk[j]=s;
47              }
48          }
49      }
```

```
50
51      //查询区间 [l,r) 的 RMQ
52      T operator()(int l,int r)
53      {
54          if (l/B!=(r-1)/B)
55          {
56              T ans=min(suf[l],pre[r-1],cmp);
57              l=l/B+1,r=r/B;
58              if (l<r)
59              {
60                  int k=__lg(r-l);
61                  ans=min({ans,a[k][l],a[k][r-(1<<k)]},cmp);
62              }
63              return ans;
64          }
65          else
66          {
67              int x=B*(l/B);
68              return ini[__builtin_ctzll(stk[r-1]>>(l-x))+l];
69          }
70      }
71  };
```

树状数组

```
1   template <class T>
2   struct BIT
3   {
4       int n;
5       vector<T> a;
6
7       BIT(int n_=0) { init(n_); }
8
9       void init(int n_)
10      {
11          n=n_;
12          a.assign(n,T{});
13      }
14
15      void add(int x,const T &v)
16      {
17          for (int i=x+1;i<=n;i+=i&-i)
18              a[i-1]=a[i-1]+v;
19      }
20
21      //查询区间 [0,x)
22      T sum(int x)
23      {
24          T ans{};
25          for (int i=x;i>0;i-=i&-i)
26              ans=ans+a[i-1];
27          return ans;
28      }
29
30      //查询区间 [l,r)
31      T rangeSum(int l,int r) { return sum(r)-sum(l); }
32
33      int select(const T &k)
34      {
35          int x=0;
36          T cur{};
37          for (int i=1<<__lg(n);i;i>>=1)
38          {
39              if (x+i<=n&&cur+a[x+i-1]<=k)
40              {
41                  x+=i;
42                  cur=cur+a[x-1];
43              }
44          }
45          return x;
46      }
```

```
47    };
```

线段树

```
1    template <class Info,class Tag>
2    struct SGT
3    {
4        int n;
5        vector<Info> info;
6        vector<Tag> tag;
7
8        SGT():n(0) {}
9        SGT(int n_,Info v_=Info()) { init(n_,v_); }
10
11       template <class T>
12       SGT(vector<T> init_) { init(init_); }
13
14       void init(int n_,Info v_=Info()) { init(vector(n_,v_)); }
15
16       template <class T>
17       void init(vector<T> init_)
18       {
19           n=init_.size();
20           info.assign(4<<__lg(n),Info());
21           tag.assign(4<<__lg(n),Tag());
22           function<void(int,int,int)> build=[&](int p,int l,int r)
23           {
24               if (r-l==1)
25               {
26                   info[p]=init_[l];
27                   return;
28               }
29               int m=(l+r)>>1;
30               build(p<<1,l,m);
31               build(p<<1|1,m,r);
32               pushup(p);
33           };
34           build(1,0,n);
35       }
36
37       void pushup(int p) { info[p]=info[p<<1]+info[p<<1|1]; }
38
39       void apply(int p,const Tag &v)
40       {
41           info[p].apply(v);
42           tag[p].apply(v);
43       }
44
45       void pushdown(int p)
46       {
47           apply(p<<1,tag[p]);
48           apply(p<<1|1,tag[p]);
49           tag[p]=Tag();
50       }
51
52       void modify(int p,int l,int r,int x,const Info &v)
53       {
54           if (r-l==1)
55           {
56               info[p]=v;
57               return;
58           }
59           int m=(l+r)>>1;
60           pushdown(p);
61           if (x<m) modify(p<<1,l,m,x,v);
62           else modify(p<<1|1,m,r,x,v);
63           pushup(p);
64       }
65
66       //O(log n) 单点修改
67       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;
```

```
139         }
140     };
141
142     struct Info
143     {
144         i64 x=0,l=0,r=0;
145         void apply(Tag t)
146         {
147             int len=r-l+1;
148             x=x*t.a+len*t.b;
149         }
150     };
151
152     Info operator + (Info a,Info b)
153     {
154         return {a.x+b.x,min(a.l,b.l),max(a.r,b.r)};
155     }
```

字符串

字符串哈希（随机模数）

```
1   bool isPrime(int n)
2   {
3       if (n<=1) return 0;
4       for (int i=2;i*i<=n;i++)
5           if (n%i==0) return 0;
6       return 1;
7   }
8
9   int findPrime(int n)
10  {
11      while (!isPrime(n)) n++;
12      return n;
13  }
14
15  mt19937 rng(time(0));
16  const int P=findPrime(rng()%900000000+100000000);
17  struct StrHash
18  {
19      int n;
20      vector<int> h,p;
21
22      StrHash(const string &s){ init(s); }
23
24      void init(const string &s)
25      {
26          n=s.size();
27          h.resize(n+1);
28          p.resize(n+1);
29          p[0]=1;
30          for (int i=0;i<n;i++) h[i+1]=(10ll*h[i]+s[i]-'a')%P;
31          for (int i=0;i<n;i++) p[i+1]=10ll*p[i]%P;
32      }
33
34      //查询 [l,r) 的区间哈希
35      int get(int l,int r) { return (h[r]+1ll*(P-h[l])*p[r-l])%P; }
36  };
```

KMP

```
1   vector<int> KMP(const string &s)
2   {
3       int now=0;
4       vector<int> pre(s.size(),0);
5       for (int i=1;i<s.size();i++)
6       {
7           while (now&&s[i]!=s[now]) now=pre[now-1];
8           if (s[i]==s[now]) now++;
```

```
9          pre[i]=now;
10     }
11     return pre;
12 }
```

Z 函数

```
1  vector<int> zFunction(string s)
2  {
3      int n=s.size();
4      vector<int> z(n);
5      z[0]=n;
6      for (int i=1,j=1;i<n;i++)
7      {
8          z[i]=max(0,min(j+z[j]-i,z[i-j]));
9          while (i+z[i]<n&&s[z[i]]==s[i+z[i]]) z[i]++;
10         if (i+z[i]>j+z[j]) j=i;
11     }
12     return z;
13 }
```

AC 自动机

```
1  struct ACAM
2  {
3      static constexpr int ALPHABET=26;
4      struct Node
5      {
6          int len;
7          int link;
8          array<int,ALPHABET> next;
9          Node():len{0},link{0},next{}{}
10     };
11
12     vector<Node> t;
13
14     ACAM() { init(); }
15
16     void init()
17     {
18         t.assign(2,Node());
19         t[0].next.fill(1);
20         t[0].len=-1;
21     }
22
23     int newNode()
24     {
25         t.emplace_back();
26         return t.size()-1;
27     }
28
29     int add(const string &a)
30     {
31         int p=1;
32         for (auto c:a)
33         {
34             int x=c-'a';
35             if (t[p].next[x]==0)
36             {
37                 t[p].next[x]=newNode();
38                 t[t[p].next[x]].len=t[p].len+1;
39             }
40             p=t[p].next[x];
41         }
42         return p;
43     }
44
45     void work()
46     {
47         queue<int> q;
```

```
48          q.push(1);
49          while (!q.empty())
50          {
51              int x=q.front();
52              q.pop();
53              for (int i=0;i<ALPHABET;i++)
54              {
55                  if (t[x].next[i]==0) t[x].next[i]=t[t[x].link].next[i];
56                  else
57                  {
58                      t[t[x].next[i]].link=t[t[x].link].next[i];
59                      q.push(t[x].next[i]);
60                  }
61              }
62          }
63      }
64
65      int next(int p,int x) { return t[p].next[x]; }
66
67      int link(int p) { return t[p].link; }
68
69      int size() { return t.size(); }
70  };
```

后缀数组

```
1   struct SA
2   {
3       int n;
4       vector<int> sa,rk,lc;
5       SA(const string &s)
6       {
7           n=s.length();
8           sa.resize(n);
9           rk.resize(n);
10          lc.resize(n-1);
11          iota(sa.begin(),sa.end(),0);
12          sort(sa.begin(),sa.end(),[&](int a,int b){ return s[a]<s[b]; });
13          rk[sa[0]]=0;
14          for (int i=1;i<n;i++) rk[sa[i]]=rk[sa[i-1]]+(s[sa[i]]!=s[sa[i-1]]);
15          int k=1;
16          vector<int> tmp,cnt(n);
17          tmp.reserve(n);
18          while (rk[sa[n-1]]<n-1)
19          {
20              tmp.clear();
21              for (int i=0;i<k;i++) tmp.push_back(n-k+i);
22              for (auto i:sa)
23                  if (i>=k) tmp.push_back(i-k);
24              fill(cnt.begin(),cnt.end(),0);
25              for (int i=0;i<n;i++) cnt[rk[i]]++;
26              for (int i=1;i<n;i++) cnt[i]+=cnt[i-1];
27              for (int i=n-1;i>=0;i--) sa[--cnt[rk[tmp[i]]]]=tmp[i];
28              swap(rk,tmp);
29              rk[sa[0]]=0;
30              for (int i=1;i<n;i++)
31                  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]);
32              k<<=1;
33          }
34          for (int i=0,j=0;i<n;i++)
35          {
36              if (rk[i]==0) j=0;
37              else
38              {
39                  for (j-=j>0;i+j<n&&sa[rk[i]-1]+j<n&&s[i+j]==s[sa[rk[i]-1]+j];) j++;
40                  lc[rk[i]-1]=j;
41              }//lc[i]:lcp(sa[i],sa[i+1]),lcp(sa[i],sa[j])=min{lc[i...j-1]}
42          }
43      }
44  };
```

15

（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;
```

```
70        }
71
72        int next(int p,int x) { return t[p].next[x]; }
73
74        int link(int p) { return t[p].link; }
75
76        int len(int p) { return t[p].len; }
77
78        int size() { return t.size(); }
79  };
```

Manacher

```
1   vector<int> manacher(vector<int> s)
2   {
3       vector<int> t{0};
4       for (auto c:s)
5       {
6           t.push_back(c);
7           t.push_back(0);
8       }
9       int n=t.size();
10      vector<int> r(n);
11      for (int i=0,j=0;i<n;i++)
12      {
13          if (j*2-i>=0&&j+r[j]>i) r[i]=min(r[j*2-i],j+r[j]-i);
14          while (i-r[i]>=0&&i+r[i]<n&&t[i-r[i]]==t[i+r[i]]) r[i]++;
15          if (i+r[i]>j+r[j]) j=i;
16      }
17      return r;
18  }
```

回文自动机

```
1   struct PAM
2   {
3       static constexpr int ALPHABET_SIZE=28;
4       struct Node
5       {
6           int len,link,cnt;
7           array<int,ALPHABET_SIZE> next;
8           Node():len{},link{},cnt{},next{}{}
9       };
10      vector<Node> t;
11      int suff;
12      string s;
13
14      PAM() { init(); }
15
16      void init()
17      {
18          t.assign(2,Node());
19          t[0].len=-1;
20          suff=1;
21          s.clear();
22      }
23
24      int newNode()
25      {
26          t.emplace_back();
27          return t.size()-1;
28      }
29
30      bool add(char c,char offset='a')
31      {
32          int pos=s.size();
33          s+=c;
34          int let=c-offset;
35          int cur=suff,curlen=0;
36          while (1)
```

```
37          {
38              curlen=t[cur].len;
39              if (pos-curlen-1>=0&&s[pos-curlen-1]==s[pos]) break;
40              cur=t[cur].link;
41          }
42          if (t[cur].next[let])
43          {
44              suff=t[cur].next[let];
45              return 0;
46          }
47          int num=newNode();
48          suff=num;
49          t[num].len=t[cur].len+2;
50          t[cur].next[let]=num;
51          if (t[num].len==1)
52          {
53              t[num].link=t[num].cnt=1;
54              return 1;
55          }
56          while (1)
57          {
58              cur=t[cur].link;
59              curlen=t[cur].len;
60              if (pos-curlen-1>=0&&s[pos-curlen-1]==s[pos])
61              {
62                  t[num].link=t[cur].next[let];
63                  break;
64              }
65          }
66          t[num].cnt=t[t[num].link].cnt+1;
67          return 1;
68      }
69  };
```

图论

强连通分量

```
1   struct SCC
2   {
3       int n,cur,cnt;
4       vector<vector<int>> adj;
5       vector<int> stk,dfn,low,bel;
6
7       SCC() {}
8       SCC(int n) { init(n); }
9
10      void init(int n)
11      {
12          this->n=n;
13          adj.assign(n,{});
14          stk.clear();
15          dfn.assign(n,-1);
16          low.resize(n);
17          bel.assign(n,-1);
18          cur=cnt=0;
19      }
20
21      void add(int u,int v) { adj[u].push_back(v); }
22
23      void dfs(int x)
24      {
25          dfn[x]=low[x]=cur++;
26          stk.push_back(x);
27          for (auto y:adj[x])
28          {
29              if (dfn[y]==-1)
30              {
31                  dfs(y);
32                  low[x]=min(low[x],low[y]);
```

```cpp
                }
                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

```cpp
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)
```

```
23          {
24              adj[u].push_back(v);
25              adj[v].push_back(u);
26          }
27
28          void dfs(int x,int p)
29          {
30              dfn[x]=low[x]=cur++;
31              stk.push_back(x);
32              for (auto y:adj[x])
33              {
34                  if (y==p) continue;
35                  if (dfn[y]==-1)
36                  {
37                      dfs(y,x);
38                      low[x]=min(low[x],low[y]);
39                  }
40                  else if (bel[y]==-1&&dfn[y]<dfn[x]) low[x]=min(low[x],dfn[y]);
41              }
42              if (dfn[x]==low[x])
43              {
44                  int y;
45                  do
46                  {
47                      y=stk.back();
48                      bel[y]=cnt;
49                      stk.pop_back();
50                  } while (y!=x);
51                  cnt++;
52              }
53          }
54
55          vector<int> work()
56          {
57              dfs(0,-1);
58              return bel;
59          }
60
61          struct Graph
62          {
63              int n;
64              vector<pair<int,int>> edges;
65              vector<int> siz,cnte;
66          };
67
68          Graph compress()
69          {
70              Graph G;
71              G.n=cnt;
72              G.siz.resize(cnt);
73              G.cnte.resize(cnt);
74              for (int i=0;i<n;i++)
75              {
76                  G.siz[bel[i]]++;
77                  for (auto j:adj[i])
78                  {
79                      if (bel[i]<bel[j]) G.edges.emplace_back(bel[i],bel[j]);
80                      else if (i<j) G.cnte[bel[i]]++;
81                  }
82              }
83              return G;
84          };
85      };
```

轻重链剖分

```
1   struct HLD
2   {
3       int n;
4       vector<int> siz,top,dep,pa,in,out,seq;
5       vector<vector<int>> adj;
```

```
6        int cur;
7
8        HLD(){}
9        HLD(int n) { init(n); }
10
11       void init(int n)
12       {
13           this->n=n;
14           siz.resize(n);
15           top.resize(n);
16           dep.resize(n);
17           pa.resize(n);
18           in.resize(n);
19           out.resize(n);
20           seq.resize(n);
21           cur=0;
22           adj.assign(n,{});
23       }
24
25       void addEdge(int u,int v)
26       {
27           adj[u].push_back(v);
28           adj[v].push_back(u);
29       }
30
31       void work(int rt=0)
32       {
33           top[rt]=rt;
34           dep[rt]=0;
35           pa[rt]=-1;
36           dfs1(rt);
37           dfs2(rt);
38       }
39
40       void dfs1(int u)
41       {
42           if (pa[u]!=-1) adj[u].erase(find(adj[u].begin(),adj[u].end(),pa[u]));
43           siz[u]=1;
44           for (auto &v:adj[u])
45           {
46               pa[v]=u;
47               dep[v]=dep[u]+1;
48               dfs1(v);
49               siz[u]+=siz[v];
50               if (siz[v]>siz[adj[u][0]])
51                   swap(v,adj[u][0]);
52           }
53       }
54
55       void dfs2(int u)
56       {
57           in[u]=cur++;
58           seq[in[u]]=u;
59           for (auto v:adj[u])
60           {
61               top[v]=(v==adj[u][0])?top[u]:v;
62               dfs2(v);
63           }
64           out[u]=cur;
65       }
66
67       int lca(int u,int v)
68       {
69           while (top[u]!=top[v])
70           {
71               if (dep[top[u]]>dep[top[v]]) u=pa[top[u]];
72               else v=pa[top[v]];
73           }
74           return dep[u]<dep[v]?u:v;
75       }
76
```

```
77        int dist(int u,int v) { return dep[u]+dep[v]-(dep[lca(u,v)]<<1); }
78
79        int jump(int u,int k)
80        {
81            if (dep[u]<k) return -1;
82            int d=dep[u]-k;
83            while (dep[top[u]]>d) u=pa[top[u]];
84            return seq[in[u]-dep[u]+d];
85        }
86
87        bool isAncester(int u,int v) { return in[u]<=in[v]&&in[v]<out[u]; }
88
89        int rootedParent(int u,int v)//u->root,v->point
90        {
91            if (u==v) return u;
92            if (!isAncester(v,u)) return pa[v];
93            auto it=upper_bound(adj[v].begin(),adj[v].end(),u,[&](int x,int y){ return in[x]<in[y]; })-1;
94            return *it;
95        }
96
97        int rootedSize(int u,int v)//same as rootedParent
98        {
99            if (u==v) return n;
100           if (!isAncester(v,u)) return siz[v];
101           return n-siz[rootedParent(u,v)];
102       }
103
104       int rootedLca(int a,int b,int c) { return lca(a,b)^lca(b,c)^lca(c,a); }
105  };
```

最大流

```
1   bool bfs(ll s,ll t){
2       queue<ll>q;
3       for(int i=1;i<=n;i++) dis[i]=-1;dis[s]=0;
4       for(int i=1;i<=n;i++) cur[i]=frm[i];
5       q.push(s);
6       while(!q.empty()){
7           ll x=q.front();q.pop();
8           for(int i=frm[x];i;i=e[i].net){
9               ll v=e[i].to;
10              if(dis[v]==-1&&e[i].val) dis[v]=dis[x]+1,q.push(v);
11          }
12      }
13      return dis[t]!=-1;
14  }
15
16  ll dfs(ll s,ll flow){
17      if(s==t||!flow) return flow;
18      if(dis[s]>=dis[t]) return 0;
19      ll now=0,res=0;
20      for(int i=cur[s];i;i=e[i].net){
21          ll x=e[i].to;
22          if(dis[x]==dis[s]+1&&e[i].val){
23              res=dfs(x,min(flow-now,e[i].val));
24              if(!res) continue;
25              e[i].val-=res;e[i^1].val+=res;now+=res;flow-=res;
26              cur[s]=i;
27              if(!flow) break;
28          }
29      }
30      if(!now) return (dis[s]=0);
31      return now;
32  }
33
34  void dinic(ll s,ll t){
35      while(bfs(s,t)){
36          ll last=dfs(s,1e18);
37          while(last) ans+=last,last=dfs(s,1e18);
38      }
39  }
```

最大流条件下最小费用（费用流）

```cpp
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);
```