泡泡猿 ACM 模板

Rand0w & REXWIND & Dallby

2021 年 11 月 8 日

目录

1 头文件

1.1 头文件 (Rand0w)

```
#include <bits/stdc++.h>
//#include <bits/extc++.h>
//using namespace __gnu_pbds;
//using namespace __gnu_cxx;
using namespace std;
#pragma optimize(2)
//#pragma GCC optimize("Ofast,no-stack-protector")
//#pragma GCC target("sse,sse2,sse3,ssse3,sse4,popcnt,abm,mmx,avx,avx2,tune=native")
#define rbset(T) tree<T,null_type,less<T>,rb_tree_tag,tree_order_statistics_node_update>
const int inf = 0x7FFFFFFF;
typedef long long ll;
typedef double db;
typedef long double ld;
template<class T>inline void MAX(T &x,T y){if(y>x)x=y;}
template<class T>inline void MIN(T &x,T y){if(y<x)x=y;}
namespace FastIO
{
char buf[1 << 21], buf2[1 << 21], a[20], *p1 = buf, *p2 = buf, hh = '\n';
int p, p3 = -1;
void read() {}
void print() {}
inline int getc()
{
return p1 == p2 && (p2 = (p1 = buf) + fread(buf, 1, 1 << 21, stdin), p1 == p2) ? EOF : *p1++;
}
inline void flush()
{
fwrite(buf2, 1, p3 + 1, stdout), p3 = -1;
}
template <typename T, typename... T2>
inline void read(T &x, T2 &... oth)
{
int f = 0;x = 0;char ch = getc();
while (!isdigit(ch)){if (ch == '-')f = 1;ch = getc();}
while (isdigit(ch)){x = x * 10 + ch - 48;ch = getc();}
x = f ? -x : x;read(oth...);
}
template <typename T, typename... T2>
inline void print(T x, T2... oth)
{
if (p3 > 1 << 20)flush();
if (x < 0)buf2[++p3] = 45, x = -x;
do{a[++p] = x % 10 + 48;}while (x /= 10);
do{buf2[++p3] = a[p];}while (--p);
buf2[++p3] = hh;
print(oth...);
}
} // namespace FastIO
#define read FastIO::read
#define print FastIO::print
```

```cpp
#define flush FastIO::flush
#define spt fixed<<setprecision
#define endll '\n'
#define mul(a,b,mod) (__int128)(a)*(b)%(mod)
#define pii(a,b) pair<a,b>
#define pow powmod
#define X first
#define Y second
#define lowbit(x) (x&-x)
#define MP make_pair
#define pb push_back
#define pt putchar
#define yx_queue priority_queue
#define lson(pos) (pos<<1)
#define rson(pos) (pos<<1|1)
#define y1 code_by_Rand0w
#define yn A_muban_for_ACM
#define j1 it_is just_an_eastegg
#define lr hope_you_will_be_happy_to_see_this
#define int long long
#define rep(i, a, n) for (register int i = a; i <= n; ++i)
#define per(i, a, n) for (register int i = n; i >= a; --i)
const ll llinf = 4223372036854775851;
const ll mod = (0 ? 1000000007 : 998244353);
ll pow(ll a,ll b,ll md=mod) {ll res=1;a%=md; assert(b>=0); for(;b;b>>=1){if(b&1)res=mul(res,a,md);a=mul(a,a,
    md);}return res;}
const ll mod2 = 999998639;
const int m1 = 998244353;
const int m2 = 1000001011;
const int pr=233;
const double eps = 1e-7;
const int maxm= 1;
const int maxn = 510000;
void work()
{
}
signed main()
{
   #ifndef ONLINE_JUDGE
   //freopen("in.txt","r",stdin);
    //freopen("out.txt","w",stdout);
#endif
    //std::ios::sync_with_stdio(false);
    //cin.tie(NULL);
    int t = 1;
    //cin>>t;
    for(int i=1;i<=t;i++){
        //cout<<"Case #"<<i<<":"<<endll;
        work();
    }
    return 0;
}
```

1.2 头文件 (REXWind)

```
#include<iostream>
#include<cstring>
#include<cstdio>
#include<algorithm>
#include<vector>
#include<map>
#include<queue>
#include<cmath>
using namespace std;

template<class T>inline void read(T &x){
    x=0;char o,f=1;
    while(o=getchar(),o<48)if(o==45)f=-f;
    do x=(x<<3)+(x<<1)+(o^48);while(o=getchar(),o>47);x*=f;}
int cansel_sync=(ios::sync_with_stdio(0),cin.tie(0),0);
#define ll long long
#define ull unsigned long long
#define rep(i,a,b) for(int i=(a);i<=(b);i++)
#define repb(i,a,b) for(int i=(a);i>=b;i--)
#define mkp make_pair
#define ft first
#define sd second
#define log(x) (31-__builtin_clz(x))
#define INF 0x3f3f3f3f
typedef pair<int,int> pii;
typedef pair<ll,ll> pll;
ll gcd(ll a,ll b){ while(b^=a^=b^=a%=b); return a; }
//#define INF 0x7fffffff

void solve(){

}

int main(){
    int z;
    cin>>z;
    while(z--) solve();
}
```

1.3 头文件 (Dallby)

```
#include<bits/stdc++.h>
// #pragma GCC optimize(3)
#include<bits/stdc++.h>
using namespace std;
#define rep(i,x,y) for(int i=(x);i<=(y);++i)
#define dep(i,x,y) for(int i=(x);i>=(y);--i)
#define mst(a,x) memset(a,x,sizeof(a))
#define endl "\n"
#define fr first
#define sc second
#define debug cout<<"DEBUG\n";
```

```cpp
#define OMG(a,n) rep(i,1,n) cout<<a[i]<<" "; cout<<endl;
#define OMG2(a,n,m) rep(i,1,n) {rep(i,1,m) cout<<a[i][j]<<" "; cout<<endl;}
template <typename Type> void RIP(Type x) {cout<<x<<endl;}template <typename Type, typename... Targs>void
    RIP(Type x, Targs... args) {cout<<x<<" ";RIP(args...);}
mt19937 rnd(chrono::high_resolution_clock::now().time_since_epoch().count());
typedef long long ll; typedef unsigned long long ull; typedef pair<int,ll>pil; typedef pair<int,int>pii;
    typedef pair<ll,ll>pll;
const int N=1e6+10; const double eps=1e-9;
const int inf=0x3f3f3f3f; const ll INF=0x3f3f3f3f3f3f3f3f;
const int mo=(1?998244353:1000000007); ll mul(ll a,ll b,ll m=mo){return a*b%m;} ll fpow(ll a,ll b,ll m=mo){
    ll ans=1; for(;b;a=mul(a,a,m),b>>=1)if(b&1)ans=mul(ans,a,m); return ans;}
inline ll read(){ll x=0,tag=1; char c=getchar();for(;!isdigit(c);c=getchar())if(c=='-')tag=-1;for(; isdigit(
    c);c=getchar())x=x*10+c-48;return x*tag;}
typedef double lf; const lf pi=acos(-1.0); lf readf(){lf x; if(scanf("%lf",&x)!=1)exit(0); return x;}
    template<typename T> T sqr(T x){return x*x;}
ll a[N],b[N];
void Solve(){

}
int main(){
   //freopen("D:\\in.txt","r",stdin);
   //freopen("D:\\out.txt","w",stdout);
   //ios::sync_with_stdio(false); cin.tie(0); cout.tie(0);
   int T=1; //T=read();
   rep(kase,1,T){
      Solve();
   }
   return 0;
}
```

2 数论

2.1 欧拉筛

$O(n)$ 筛素数

```cpp
int primes[maxn+5],tail;
bool is_prime[maxn+5];
void euler(){
  is_prime[1] = 1;
  for (int i = 2; i < maxn; i++)
  {
    if (!is_prime[i])
    primes[++tail]=i;
    for (int j = 0; j < primes.size() && i * primes[j] < maxn; j++)
    {
      is_prime[i * primes[j]] = 1;
      if ((i % primes[j]) == 0)
        break;
    }
  }
}
```

2.2 Exgcd

求出 $ax + by = gcd(a,b)$ 的一组可行解 $O(logn)$

```
void Exgcd(ll a,ll b,ll &d,ll &x,ll &y){
   if(!b){d=a;x=1;y=0;}
   else{Exgcd(b,a%b,d,y,x);y-=x*(a/b);}
}
```

2.3 Excrt 扩展中国剩余定理

$$\text{求解同余方程组}\begin{cases} x\ \%\ b_1 \equiv\ a_1 \\ x\ \%\ b_2 \equiv\ a_2 \\ \qquad\vdots \\ x\ \%\ b_n \equiv\ a_n \end{cases}$$

```
int excrt(int a[],int b[],int n){
   int lc=1;
   for(int i=1;i<=n;i++)
      lc=lcm(lc,a[i]);
   for(int i=1;i<n;i++){
      int p,q,g;
      g=exgcd(a[i],a[i+1],p,q);
      int k=(b[i+1]-b[i])/g;
      q=-q;p*=k;q*=k;
      b[i+1]=a[i]*p%lc+b[i];
      b[i+1]%=lc;
      a[i+1]=lcm(a[i],a[i+1]);
   }
   return (b[n]%lc+lc)%lc;
}
```

2.4 线性求逆元

```
void init(int p){
   inv[1] = 1;
   for(int i=2;i<=n;i++){
      inv[i] = (ll)(p-p/i)*inv[p%i]%p;
   }
}
```

2.5 多项式

2.5.1 FFT 快速傅里叶变换

```
const int SIZE=(1<<21)+5;
const double PI=acos(-1);
struct CP{
   double x,y;
   CP(double x=0,double y=0):x(x),y(y){}
   CP operator +(const CP &A)const{return CP(x+A.x,y+A.y);}
   CP operator -(const CP &A)const{return CP(x-A.x,y-A.y);}
   CP operator *(const CP &A)const{return CP(x*A.x-y*A.y,x*A.y+y*A.x);}
```

```
};
int limit,rev[SIZE];
void DFT(CP *F,int op){
   for(int i=0;i<limit;i++)if(i<rev[i])swap(F[i],F[rev[i]]);
   for(int mid=1;mid<limit;mid<<=1){
      CP wn(cos(PI/mid),op*sin(PI/mid));
      for(int len=mid<<1,k=0;k<limit;k+=len){
         CP w(1,0);
         for(int i=k;i<k+mid;i++){
            CP tmp=w*F[i+mid];
            F[i+mid]=F[i]-tmp;
            F[i]=F[i]+tmp;
            w=w*wn;
         }
      }
   }
   if(op==-1)for(int i=0;i<limit;i++)F[i].x/=limit;
}
void FFT(int n,int m,CP *F,CP *G){
   for(limit=1;limit<=n+m;limit<<=1);
   for(int i=0;i<limit;i++)rev[i]=(rev[i>>1]>>1)|((i&1)?limit>>1:0);
   DFT(F,1),DFT(G,1);
   for(int i=0;i<limit;i++)F[i]=F[i]*G[i];
   DFT(F,-1);
}
```

2.5.2 NTT 快速数论变换

```
const int SIZE=(1<<21)+5;
int limit,rev[SIZE];
void DFT(ll *f, int op) {
   const ll G = 3;
   for(int i=0; i<limit; ++i) if(i<rev[i]) swap(f[i],f[rev[i]]);
   for(int len=2; len<=limit; len<<=1) {
      ll w1=pow(pow(G,(mod-1)/len),~op?1:mod-2);
      for(int l=0, hf=len>>1; l<limit; l+=len) {
         ll w=1;
         for(int i=l; i<l+hf; ++i) {
            ll tp=w*f[i+hf]%mod;
            f[i+hf]=(f[i]-tp+mod)%mod;
            f[i]=(f[i]+tp)%mod;
            w=w*w1%mod;
         }
      }
   }
   if(op==-1) for(int i=0, inv=pow(limit,mod-2); i<limit; ++i) f[i]=f[i]*inv%mod;
}
void NTT(int n,int m,int *F,int *G){
   for(limit=1;limit<=n+m;limit<<=1);
   for(int i=0;i<limit;i++)rev[i]=(rev[i>>1]>>1)|((i&1)?limit>>1:0);
   DFT(F,1),DFT(G,1);
   for(int i=0;i<limit;i++)F[i]=F[i]*G[i];
   DFT(F,-1);
}
```

2.5.3 MTT 任意模数 FFT

FFT 版常数巨大，慎用。

```
struct MTT{
   static const int N=1<<20;
   struct cp{
      long double a,b;
      cp(){a=0,b=0;}
      cp(const long double &a,const long double &b):a(a),b(b){}
      cp operator+(const cp &t)const{return cp(a+t.a,b+t.b);}
      cp operator-(const cp &t)const{return cp(a-t.a,b-t.b);}
      cp operator*(const cp &t)const{return cp(a*t.a-b*t.b,a*t.b+b*t.a);}
      cp conj()const{return cp(a,-b);}
   };
   cp wn(int n,int f){
      static const long double pi=acos(-1.0);
      return cp(cos(pi/n),f*sin(pi/n));
   }
   int g[N];
   void dft(cp a[],int n,int f){
      for(int i=0;i<n;i++)if(i>g[i])swap(a[i],a[g[i]]);
      for(int i=1;i<n;i<<=1){
         cp w=wn(i,f);
         for(int j=0;j<n;j+=i<<1){
            cp e(1,0);
            for(int k=0;k<i;e=e*w,k++){
               cp x=a[j+k],y=a[j+k+i]*e;
               a[j+k]=x+y,a[j+k+i]=x-y;
            }
         }
      }
      if(f==-1){
         cp Inv(1.0/n,0);
         for(int i=0;i<n;i++)a[i]=a[i]*Inv;
      }
   }
   cp a[N],b[N],Aa[N],Ab[N],Ba[N],Bb[N];
   vector<ll> conv_mod(const vector<ll> &u,const vector<ll> &v,ll mod){ // 任意模数fft
      const int n=(int)u.size()-1,m=(int)v.size()-1,M=sqrt(mod)+1;
      const int k=32-__builtin_clz(n+m+1),s=1<<k;
      g[0]=0; for(int i=1;i<s;i++)g[i]=(g[i/2]/2)|((i&1)<<(k-1));
      for(int i=0;i<s;i++){
         a[i]=i<=n?cp(u[i]%mod%M,u[i]%mod/M):cp();
         b[i]=i<=m?cp(v[i]%mod%M,v[i]%mod/M):cp();
      }
      dft(a,s,1); dft(b,s,1);
      for(int i=0;i<s;i++){
         int j=(s-i)%s;
         cp t1=(a[i]+a[j].conj())*cp(0.5,0);
         cp t2=(a[i]-a[j].conj())*cp(0,-0.5);
         cp t3=(b[i]+b[j].conj())*cp(0.5,0);
         cp t4=(b[i]-b[j].conj())*cp(0,-0.5);
```

```
            Aa[i]=t1*t3,Ab[i]=t1*t4,Ba[i]=t2*t3,Bb[i]=t2*t4;
        }
        for(int i=0;i<s;i++){
            a[i]=Aa[i]+Ab[i]*cp(0,1);
            b[i]=Ba[i]+Bb[i]*cp(0,1);
        }
        dft(a,s,-1); dft(b,s,-1);
        vector<ll> ans;
        for(int i=0;i<n+m+1;i++){
            ll t1=llround(a[i].a)%mod;
            ll t2=llround(a[i].b)%mod;
            ll t3=llround(b[i].a)%mod;
            ll t4=llround(b[i].b)%mod;
            ans.push_back((t1+(t2+t3)*M%mod+t4*M*M)%mod);
        }
        return ans;
    }
}mtt;
```

2.5.4 FWT 快速沃尔什变换

计算

$$C_i = \sum_{j \bigoplus k=i}^{n} A_j \times B_k$$

$\bigoplus$ 可以是与、或、异或

```
void FWT(ll *f, int op) {
    for(int len=2; len<=up; len<<=1) {
        for(int l=0, hf=len>>1; l<up; l+=len) {
            for(int i=l; i<l+hf; ++i) {
                ll x=f[i], y=f[i+hf];
                if(op>0) {
                    if(op==1) f[i]=(x+y)%mod, f[i+hf]=(x-y+mod)%mod; //xor
                    else if(op==2) f[i]=(x+y)%mod; //and
                    else f[i+hf]=(x+y)%mod; //or
                }
                else {
                    if(op==-1) f[i]=(x+y)*inv2%mod, f[i+hf]=(x-y+mod)*inv2%mod; //xor
                    else if(op==-2) f[i]=(x-y+mod)%mod; //and
                    else f[i+hf]=(y-x+mod)%mod; //or
                }
            }
        }
    }
}
```

2.6 组合数

预处理阶乘，并通过逆元实现相除

```
ll jc[MAXN];
ll qpow(ll d,ll c){//快速幂
    ll res = 1;
```

```
    while(c){
        if(c&1) res=res*d%med;
        d=d*d%med;c>>=1;
    }return res;
}
inline ll niyuan(ll x){return qpow(x,med-2);}
void initjc(){//初始化阶乘
    jc[0] = 1;
    rep(i,1,MAXN-1) jc[i] = jc[i-1]*i%med;
}
inline int C(int n,int m){//n是下面的
    if(n<m) return 0;
    return jc[n]*niyuan(jc[n-m])%med*niyuan(jc[m])%med;
}
int main(){
    initjc();
    int n,m;
    while(cin>>n>>m) cout<<C(n,m)<<endl;
}
```

2.7 矩阵快速幂

```
struct Matrix{
    ll a[MAXN][MAXN];
    Matrix(ll x=0){
        for(int i=0;i<n;i++){
            for(int j=0;j<n;j++){
                a[i][j]=x*(i==j);
            }
        }
    }
    Matrix operator *(const Matrix &b)const{//通过重载运算符实现矩阵乘法
        Matrix res(0);
        for(int i=0;i<n;i++){
            for(int j=0;j<n;j++){
                for(int k=0;k<n;k++){
                    ll &ma = res.a[i][j];
                    ma = (ma+a[i][k]*b.a[k][j])%mod;
                }
            }
        }
        return res;
    }
};
Matrix qpow(Matrix d,ll m){//底数和幂次数
    Matrix res(1);//构造E单位矩阵
    while(m){
        if(m&1)
            res=res*d;
        d=d*d;
        m>>=1;
    }
    return res;
}
```

2.8 高斯消元

$O(n^3)$ 复杂度，需要用 double 存储。

```
double date[110][110];
bool guass(int n){
    for(int i=1;i<=n;i++){
        int mix=-1;
        for(int j=i;j<=n;j++)
            if(date[j][i]!=0){
                mix=j;break;
            }
        if(mix==-1)
            return false;
        if(mix!=i)
            for(int j=1;j<=n+1;j++)
                swap(date[mix][j],date[i][j]);
        double t=date[i][i];
        for(int j=i;j<=n+1;j++){
            date[i][j]=date[i][j]/t;
        }
        for(int j=1;j<=n;j++){
            if(date[j][i]==0||j==i)
                continue;
            double g=date[j][i]/date[i][i];
            for(int k=1;k<=n+1;k++)
                date[j][k]-=date[i][k]*g;
        }
    }
    return true;
}
```

2.9 欧拉降幂

$$a^b \equiv \begin{cases} a^{b\%\phi(p)}, & \gcd(a,p)=1 \\ a^b, & \gcd(a,p)\neq 1, b<\phi(p) \\ a^{b\%\phi(p)+\phi(p)}, & \gcd(a,p)\neq 1, b\geq\phi(p) \end{cases} \pmod p$$

3 图论

3.1 二分图最大匹配

用最大流求解，一边连源点流量 1，一遍连汇点流量 1，连上匹配边流量 1，最大流即为最大匹配数

时间复杂度趋近 n

3.2 dijkstra

```
vector<pii> G[N];
int dis[N];
bool vis[N];
```

```cpp
void dijkstra(int n,int pos){
    rep(i,1,n) dis[i]=inf,vis[i]=0;
    priority_queue<pii,vector<pii>,greater<pii>> pq;
    pq.push({0,pos});
    dis[pos]=0;
    while(!pq.empty()){
        pii top=pq.top(); pq.pop();
        int u=top.sc,d=top.fr;
        if(vis[u]) continue;
        for(auto e:G[u]){
            int v=e.fr;
            if(dis[v]>d+e.sc) {
                dis[v]=d+e.sc;
                pq.push({dis[v],v});
            }
        }
    }
}
```

3.3 2-Sat

```cpp
//
struct TwoSat{
    //适用于存在多个二选一抉择(互斥)，且不同编号抉择间存在互斥关系的问题
    //考虑到不会撤回操作 复杂度O(N)
    //编号从0开始
    int n;
    vector<int> G[2*N];
    int S[2*N],c=0;
    bool mark[2*N],ok;
    void init(int n){
        this->n=n,ok=1;
        rep(i,0,n*2) mark[i]=0;
        rep(i,0,n*2) G[i].clear();
    }
    bool dfs(int u){
        if(mark[u^1]) return false;
        if(mark[u]) return true;
        S[++c]=u;
        mark[u]=1;
        for(auto v:G[u]){
            if(!dfs(v)) return false;
        }
        return true;
    }
    //x,y代表抉择编号,val代表选0/1
    void add(int x,int xval,int y,int yval){
        x=x*2+xval,y=y*2+yval;
        G[x^1].push_back(y);
        G[y^1].push_back(x);
    }
    void solve(){
        rep(i,0,n-1){
            c=0;
```

```
            if(!dfs(i*2)){
                while(c) mark[S[c--]]=0;
                if(!dfs(i*2+1)) ok=0;
            }
        }
    }
}solver;
```

3.4 最大团

```
//假了
```

3.5 稳定婚姻

```
//构造一种匹配方式，使得不存在一组未被选中且双方好感更高的匹配
int B[N][N],G[N][N];//x对x的好感
vector<int> vec[N];
int pos[N],px[N],py[N],cur;
queue<int> que;
bool cmp(const int x,const int y){
    return B[cur][x]>B[cur][y];
}
void stable_marriage(int n){
     memset(pos,0,sizeof(pos));
     memset(py,0,sizeof(py));
     rep(i,1,n) vec[i].clear();
     rep(i,1,n) rep(j,1,n) vec[i].push_back(j);
     rep(i,1,n) {
        cur=i;
        sort(vec[i].begin(),vec[i].end(),cmp);
     }
     rep(i,1,n) que.push(i);
     while(!que.empty()){
        queue<int> tq;
        while(!que.empty()){
            int b=que.front(); que.pop();
            int p=pos[b]++,g=vec[b][p];
            if(!py[g]||G[g][b]>G[g][py[g]]){
                if(py[g]) tq.push(py[g]);
                py[g]=b,px[b]=g;
            }
            else tq.push(b);
        }
        que=tq;
     }
}
```

3.6 点双连通分量

```
vector<int> G[N],bcc[N];
int pre[N]={0},low[N];
int cur,bcnt,bccno[N]; //=0;
```

```
bool iscut[N];
stack<pii> S;
void dfs(int u,int fa){
    low[u]=pre[u]=++cur;
    int child=0;
    for(auto v:G[u]){
        if(v==fa) continue;
        if(pre[v]&&pre[v]<pre[u]) {
            //S.push({u,v});
            low[u]=min(low[u],pre[v]);
        }
        else if(!pre[v]){
            S.push({u,v});
            child++;
            dfs(v,u);
            low[u]=min(low[u],low[v]);
            if(low[v]>=pre[u]){
                iscut[u]=1;
                bcnt++; bcc[bcnt].clear();
                while(1){
                    pii x=S.top(); S.pop();
                    if(bccno[x.fr]!=bcnt){
                        bcc[bcnt].push_back(x.fr);
                        bccno[x.fr]=bcnt;
                    }
                    if(bccno[x.sc]!=bcnt){
                        bcc[bcnt].push_back(x.sc);
                            bccno[x.sc]=bcnt;
                    }
                    if(x.fr==u&&x.sc==v) break;
                }
            }
        }
    }
    if(fa==-1&&child==1) iscut[u]=0;
```

3.7 强连通分量

```
vector<int> G[N];
int sccno[N],dfn[N],low[N],scnt,tot;
stack<int> S;
void tarjan(int u){
    dfn[u]=low[u]=++tot;
    S.push(u);
    for(auto v:G[u]){
        if(sccno[v]) continue;
        if(!dfn[v]){
            tarjan(v);
            low[u]=min(low[u],low[v]);
        }
        else low[u]=min(low[u],dfn[v]); //low[v] is also ok.
    }
    if(low[u]==dfn[u]){
        scnt++;
```

```cpp
        int top=-1;
        while(top!=u){
            top=S.top(); S.pop();
            sccno[top]=scnt;
        }
    }
}
```

3.8 最小树形图

```cpp
struct Edge{int u,v,w,b;};
vector<Edge> eset,E;
int fa[N],minw[N],id[N],top[N];
ll solve(int n,int rt){
    ll res=0;
    while(1){
        int cnt=0;
        rep(i,1,n) minw[i]=inf,id[i]=top[i]=0;
        for(auto &e:eset){
            if(e.u!=e.v&&e.w<minw[e.v]){
                minw[e.v]=e.w;
                fa[e.v]=e.u;
            }
        }
        minw[rt]=0;
        rep(i,1,n){
            if(minw[i]==inf) return -1;
            res+=minw[i];
            for(int u=i;!id[u]&&u!=rt;u=fa[u]){
                if(top[u]==i){
                    id[u]=++cnt;
                    for(int v=fa[u];v!=u;v=fa[v]){
                        id[v]=cnt;
                    }
                    break;
                }
                else top[u]=i;
            }
        }
        if(!cnt) return res;
        rep(i,1,n) if(!id[i]) id[i]=++cnt;
        for(auto &e:eset){
            e.w-=minw[e.v];
            e.u=id[e.u],e.v=id[e.v];
        }
        n=cnt;
        rt=id[rt];
```

3.9 KM

```cpp
int n;
ll G[N][N];
int px[N],py[N],vx[N],vy[N],pre[N];
```

```
ll lx[N],ly[N],slack[N],d;
queue<int> que;
void upd(int v){
   int t;
   while(v){
      t=px[pre[v]];
      py[v]=pre[v];
      px[pre[v]]=v;
      v=t;
   }
}
void bfs(int x){
   rep(i,1,n) vy[i]=vx[i]=0,slack[i]=inf;
   while(!que.empty()) que.pop();
   que.push(x);
   while(1){
      while(!que.empty()){
         int u=que.front(); que.pop();
         vx[u]=1;
         rep(v,1,n) if(!vy[v]){
            if(lx[u]+ly[v]-G[u][v]<slack[v]){
               slack[v]=lx[u]+ly[v]-G[u][v];
               pre[v]=u;
               if(!slack[v]){
                  vy[v]=1;
                  if(!py[v]) {upd(v); return;}
                  else que.push(py[v]);
               }
            }
         }
      }
      d=inf;
      rep(i,1,n) if(slack[i]) d=min(d,slack[i]);
      rep(i,1,n) {
         if(vx[i]) lx[i]-=d;
         if(vy[i]) ly[i]+=d;
         else slack[i]-=d;
      }
      rep(i,1,n){
         if(!vy[i]&&!slack[i]) {
            vy[i]=1;
            if(!py[i]) {upd(i); return;}
            else que.push(py[i]);
         }
      }
   }
}
ll KM(){
   ll res=0;
   fill(lx+1,lx+n+1,-inf); memset(ly,0,sizeof(ly));
   memset(px,0,sizeof(px)); memset(py,0,sizeof(py));
   memset(pre,0,sizeof(pre));
   rep(i,1,n) rep(j,1,n) lx[i]=max(lx[i],G[i][j]);
   rep(i,1,n) bfs(i);
   rep(i,1,n) res+=G[i][px[i]];
```

```
    return res;
}
```

3.10 差分约束

```
//介绍WIP
vector<pair<int,ll>> G[N];
bool inque[N];
int incnt[N];
ll dis[N];
bool bellman_ford(ll x){
    queue<int> que;
    rep(i,1,n) {
        dis[i]=0,inque[i]=1,incnt[i]=0;
        que.push(i);
    }
    while(!que.empty()){
        int u=que.front(); que.pop();
        inque[u]=0;
        for(auto e:G[u]){
            int v=e.fr;
            //cout<<dis[u]<<" "<<e.sc<<" "<<dis[v]<<endl;
            if(dis[u]+e.sc<dis[v]){
                dis[v]=dis[u]+e.sc;
                if(!inque[v]){
                    inque[v]=1;
                    que.push(v);
                    if(++incnt[v]>n) return true;
                }
            }
        }
    }
    return false;
}
```

3.11 一般图最大权匹配带花树 (hjt)

```
int n;
vector<int> G[N];
struct DSU { // join: d[x] = d[y], query: d[x] == d[y]
    int a[N];
    void init(int n) { iota(a, a + n + 1, 0); }
    int fa(int x) { return a[x] == x ? x : a[x] = fa(a[x]); }
    int &operator[](int x) { return a[fa(x)]; }
} d;
deque<int> q;
int mch[N],vis[N],dfn[N],fa[N],dcnt=0;
int lca(int x,int y){
    dcnt++;
    while(1){
        if(x==0)swap(x,y); x=d[x];
        if(dfn[x]==dcnt)return x;
        else dfn[x]=dcnt,x=fa[mch[x]];
```

```
    }
}
void shrink(int x,int y,int p){
    while(d[x]!=p){
        fa[x]=y; y=mch[x];
        if(vis[y]==2)vis[y]=1,q.push_back(y);
        if(d[x]==x)d[x]=p;
        if(d[y]==y)d[y]=p;
        x=fa[y];
    }
}
bool match(int s){
    d.init(n); fill(fa,fa+n+1,0);
    fill(vis,vis+n+1,0); vis[s]=1;
    q.assign(1,s);
    while(!q.empty()){
        int x=q.front(); q.pop_front();
        for(auto p:G[x]){
            if(d[x]==d[p] || vis[p]==2)continue;
            if(!vis[p]){
                vis[p]=2; fa[p]=x;
                if(!mch[p]){
                    for(int now=p,last,tmp;now;now=last){
                        last=mch[tmp=fa[now]];
                        mch[now]=tmp,mch[tmp]=now;
                    }
                    return 1;
                }
                vis[mch[p]]=1; q.push_back(mch[p]);
            }
            else if(vis[p]==1){
                int l=lca(x,p);
                shrink(x,p,l);
                shrink(p,x,l);
            }
        }
    }
    return 0;
}
```

4 计算几何

4.1 三点求圆心

```
struct point{
    double x;
    double y;
};

point cal(point a,point b,point c){
    double x1 = a.x;double y1 = a.y;
    double x2 = b.x;double y2 = b.y;
    double x3 = c.x; double y3 = c.y;
```

```
    double a1 = 2*(x2-x1); double a2 = 2*(x3-x2);
    double b1 = 2*(y2-y1); double b2 = 2*(y3-y2);
    double c1 = x2*x2 + y2*y2 - x1*x1 - y1*y1;
    double c2 = x3*x3 + y3*y3 - x2*x2 - y2*y2;
    double rx = (c1*b2-c2*b1)/(a1*b2-a2*b1);
    double ry = (c2*a1-c1*a2)/(a1*b2-a2*b1);
    return point{rx,ry};
}
```

4.2 欧拉降幂

$$a^b \equiv \begin{cases} a^{b\%\phi(p)}, & \gcd(a,p)=1 \\ a^b, & \gcd(a,p)\neq 1, b<\phi(p) \\ a^{b\%\phi(p)+\phi(p)}, & \gcd(a,p)\neq 1, b\geq \phi(p) \end{cases} \pmod p$$

4.3 拉格朗日插值

```
namespace polysum {
#define rep(i,a,n) for (int i=a;i<n;i++)
#define per(i,a,n) for (int i=n-1;i>=a;i--)
const int D = 1010000; ///可能需要用到的最高次
LL a[D], f[D], g[D], p[D], p1[D], p2[D], b[D], h[D][2], C[D];
LL powmod(LL a, LL b) {
    LL res = 1;
    a %= mod;
    assert(b >= 0);

    for (; b; b >>= 1) {
        if (b & 1)
            res = res * a % mod;

        a = a * a % mod;
    }

    return res;
}

///函数用途：给出数列的（d+1）项，其中d为最高次方项
///求出数列的第n项，数组下标从0开始
LL calcn(int d, LL *a, LL n) { /// a[0].. a[d] a[n]
    if (n <= d)
        return a[n];

    p1[0] = p2[0] = 1;
    rep(i, 0, d + 1) {
        LL t = (n - i + mod) % mod;
        p1[i + 1] = p1[i] * t % mod;
    }
    rep(i, 0, d + 1) {
        LL t = (n - d + i + mod) % mod;
        p2[i + 1] = p2[i] * t % mod;
    }
    LL ans = 0;
```

```
    rep(i, 0, d + 1) {
        LL t = g[i] * g[d - i] % mod * p1[i] % mod * p2[d - i] % mod * a[i] % mod;

        if ((d - i) & 1)
            ans = (ans - t + mod) % mod;
        else
            ans = (ans + t) % mod;
    }
    return ans;
}
void init(int M) {///用到的最高次
    f[0] = f[1] = g[0] = g[1] = 1;
    rep(i, 2, M + 5) f[i] = f[i - 1] * i % mod;
    g[M + 4] = powmod(f[M + 4], mod - 2);
    per(i, 1, M + 4) g[i] = g[i + 1] * (i + 1) % mod; ///费马小定理筛逆元
}

///函数用途：给出数列的（m+1）项，其中m为最高次方
///求出数列的前（n-1）项的和（从第0项开始）
LL polysum(LL m, LL *a, LL n) { /// a[0].. a[m] \sum_{i=0}^{n-1} a[i]
    for (int i = 0; i <= m; i++)
        b[i] = a[i];

    ///前n项和，其最高次幂加1
    b[m + 1] = calcn(m, b, m + 1);
    rep(i, 1, m + 2) b[i] = (b[i - 1] + b[i]) % mod;
    return calcn(m + 1, b, n - 1);
}
LL qpolysum(LL R, LL n, LL *a, LL m) { /// a[0].. a[m] \sum_{i=0}^{n-1} a[i]*R^i
    if (R == 1)
        return polysum(n, a, m);

    a[m + 1] = calcn(m, a, m + 1);
    LL r = powmod(R, mod - 2), p3 = 0, p4 = 0, c, ans;
    h[0][0] = 0;
    h[0][1] = 1;
    rep(i, 1, m + 2) {
        h[i][0] = (h[i - 1][0] + a[i - 1]) * r % mod;
        h[i][1] = h[i - 1][1] * r % mod;
    }
    rep(i, 0, m + 2) {
        LL t = g[i] * g[m + 1 - i] % mod;

        if (i & 1)
            p3 = ((p3 - h[i][0] * t) % mod + mod) % mod, p4 = ((p4 - h[i][1] * t) % mod + mod) % mod;
        else
            p3 = (p3 + h[i][0] * t) % mod, p4 = (p4 + h[i][1] * t) % mod;
    }
    c = powmod(p4, mod - 2) * (mod - p3) % mod;
    rep(i, 0, m + 2) h[i][0] = (h[i][0] + h[i][1] * c) % mod;
    rep(i, 0, m + 2) C[i] = h[i][0];
    ans = (calcn(m, C, n) * powmod(R, n) - c) % mod;

    if (ans < 0)
        ans += mod;
```

```
    return ans;
}
}
```

5 数据结构

5.1 扫描线

扫描线是离散化后，使用类似权值线段树来维护每个截面上的线段长度。
通过把二维平面上的四边形拆分成入边和出边两段，在遇到边的时候对对应的区间进行区间加/减即可。
每个节点上需要维护被完全覆盖的次数和实际长度。

```
#define ls (x<<1)
#define rs (x<<1|1)//这种方法感觉还挺好的

int cansel_sync=(ios::sync_with_stdio(0),cin.tie(0),0);
const int MAXN = 2e5+5;//这里要开n的两倍
//线结构体
struct Line{
    ll l,r,h;
    int qz;//记录位置和权值
    bool operator < (Line &rhs){
        return h < rhs.h;
    }
}line[MAXN];
int n;
ll x1,y1,x2,y2;
ll X[MAXN];
//线段树
struct Segt{
    int l,r;//是X的下标,即离散化后的
    int sum;//sum是被完全覆盖的次数
    ll len;//len是区间内被盖住的长度
    //因为每次查询都是查询根节点,所以这边不需要懒惰标记
}t[MAXN<<3];//一个边有两个点,所以这里要开8倍
void build(int x,int l,int r){
    t[x].l = l;t[x].r = r;
    t[x].len = t[x].sum = 0;
    if(l==r) return;//到了叶子节点
    int mid = (l+r)>>1;
    build(ls,l,mid);
    build(rs,mid+1,r);
}
void push_up(int x){
    int l = t[x].l,r = t[x].r;
    if(t[x].sum) t[x].len = X[r+1]-X[l];//x的区间是X[l]到X[r+1]-1
    else t[x].len = t[ls].len + t[rs].len;//合并儿子的信息
}
void update(int x,int L,int R,int v){//这里的LR存的是实际值
    //这里如果是线段L,R,线段树上是L到R-1的部分维护
    int l = t[x].l,r = t[x].r;
    if(X[r+1]<=L||R<=X[l]) return;//加等于,不然会搞到无辜的线
    if(L<=X[l]&&X[r+1]<=R){
```

```
        t[x].sum += v;//修改覆盖次数
        push_up(x);
        return;
    }
    update(ls,L,R,v);
    update(rs,L,R,v);
    push_up(x);
}
int main(){
    cin>>n;
    rep(i,1,n){
        cin>>x1>>y1>>x2>>y2;
        X[2*i-1] = x1,X[2*i] = x2;//一会儿离散化要用的,这里存实际值
        line[2*i-1] = Line{x1,x2,y1,1};//开始的线
        line[2*i] = Line{x1,x2,y2,-1};//结束的线
    }
    n<<=1;//line的数量是四边形数量的2倍
    sort(line+1,line+1+n);
    sort(X+1,X+1+n);
    int tot = unique(X+1,X+n+1)-(X+1);//去除重复相邻元素,并且tot记录总数
    build(1,1,tot-1);//为什么是tot-1?
    //因为线段树只需要维护X[1]到X[tot]-1这一段的,实际长度是向右贴的
    ll res = 0;
    rep(i,1,n-1){//每次高度是line[i+1].h-line[i].h,所以是到n-1就行
        update(1,line[i].l,line[i].r,line[i].qz);//扫描线加入线段树
        res += t[1].len*(line[i+1].h-line[i].h);
    }
    cout<<res<<endl;
}
```

5.2 ST 表求 RMQ

$O(nlog_n)$ 预处理，$O(1)$ 查询

```
#define log(x) (31-__builtin_clz(x))
const int MAXN = 1e5+10;
const int LOGN = log(MAXN)/log(2)+5;
int M[MAXN][LOGN];
int a[MAXN];
int z,m,n;
void init(){//初始化，复杂度O(nlogn)
    for(int i=1;i<=n;i++) M[i][0]=i;//长度为1的区间最值是自己
    for(int j=1;j<=LOGN;j++){
        for(int i=1;i<=n-(1<<j)+1;i++){
            if(a[M[i][j-1]]<a[M[i+(1<<(j-1))][j-1]]) M[i][j] = M[i][j-1];//这里以最小值为例
            else M[i][j] = M[i+(1<<j-1)][j-1];
        }
    }
}
int query(int l,int r){
    int k = log(r-l+1)/log(2);//向下取整
    if(a[M[l][k]]<a[M[r-(1<<k)+1][k]]) return M[l][k];
    else return M[r-(1<<k)+1][k];
}
```

5.3 并查集系列

5.3.1 普通并查集

带路径压缩,$O(1)$ 复杂度

```
int fa[maxn];
int find(int x){if(fa[x]^x)return fa[x]=find(fa[x]);return x;}
void merge(int a,int b){fa[find(a)]=find(b);}
```

5.3.2 按秩合并并查集

```
int fa[maxn];
int dep[maxn];
int find(int x){int now=x; while(fa[now]^now)now=fa[now];return now;}
void merge(int a,int b){
    int l=find(a),r=find(b);
    if(l==r) return;
    if(dep[l]>dep[r])swap(l,r);
    fa[l]=r;
    dep[r]+=dep[l]==dep[r];
}
```

5.3.3 可持久化并查集

```
struct chair_man_tree{
    struct node{
        int lson,rson;
    }tree[maxn<<5];
    int tail=0;
    int tail2=0;
    int fa[maxn<<2];
    int depth[maxn<<2];
    inline int getnew(int pos){
        tree[++tail]=tree[pos];
        return tail;
    }
    int build(int l,int r){

        if(l==r){
            fa[++tail2]=l;
            depth[tail2]=1;
            return tail2;
        }
        int now=tail++;
        int mid=(l+r)>>1;
        tree[now].lson=build(l,mid);
        tree[now].rson=build(mid+1,r);
        return now;
    }
    int query(int pos,int l,int r,int qr){
        if(l==r)
            return pos;
        int mid=(l+r)>>1;
```

```
        if(qr<=mid)
            return query(tree[pos].lson,l,mid,qr);
        else return query(tree[pos].rson,mid+1,r,qr);
    }
    int update(int pos,int l,int r,int qr,int val){
        if(l==r){
            depth[++tail2]=depth[pos];
            fa[tail2]=val;
            return tail2;
        }
        int now=getnew(pos);
        int mid=(l+r)>>1;
        if(mid>=qr)
            tree[now].lson=update(tree[now].lson,l,mid,qr,val);
        else tree[now].rson=update(tree[now].rson,mid+1,r,qr,val);
        return now;
    }
    int add(int pos,int l,int r,int qr){
        if(l==r){
            depth[++tail2]=depth[pos]+1;
            fa[tail2]=fa[pos];
            return tail2;
        }
        int now=getnew(pos);
        int mid=(l+r)>>1;
        if(mid>=qr)
            tree[now].lson=add(tree[now].lson,l,mid,qr);
        else tree[now].rson=add(tree[now].rson,mid+1,r,qr);
        return now;
    }
    int getfa(int root,int qr){
        int t=fa[query(root,1,n,qr)];
        if(qr==t)
        return qr;
        else return getfa(root,t);
    }
}t;
```

5.3.4 可撤销并查集

```
struct DSU{
    int *rnk,*f,top;
    pii *stk;
    DSU(int n=N){
        rnk=new int[n],f=new int[n];
        stk=new pii[n];
    }
    void init(int n){
        rep(i,1,n) rnk[i]=0,f[i]=i;
        top=0;
    }
    int getf(int u){return f[u]==u?u:getf(f[u]);}
    void unite(int u,int v){
        u=getf(u),v=getf(v);
```

```
        if(u==v) return;
        if(rnk[u]>rnk[v]){
            stk[++top]={v,rnk[v]};
            f[v]=u;
        }
        else{
            stk[++top]={u,rnk[u]};
            f[u]=v;
            if(rnk[u]==rnk[v]) rnk[v]++;
        }
    }
    void undo(int pos){
        while(top>pos){
            int u=stk[top].fr,r=stk[top].sc;
            f[u]=u,rnk[u]=r;
            --top;
        }
    }
}dsu;
```

5.3.5 ETT 维护动态图连通性

待补

5.4 平衡树系列

5.4.1 fhq_treap

无旋 treap，可持久化，常数大

```
mt19937 rnd(514114);
struct fhq_treap{
    struct node{
        int l, r;
        int val, key;
        int size;
    } fhq[maxn];
    int cnt, root;
    inline int newnode(int val){
        fhq[++cnt].val = val;
        fhq[cnt].key = rnd();
        fhq[cnt].size = 1;
        fhq[cnt].l = fhq[cnt].r = 0;
        return cnt;
    }
    inline void pushup(int now){
    fhq[now].size = fhq[fhq[now].l].size + fhq[fhq[now].r].size + 1;
    }
    void split(int now, int val, int &x, int &y){
        if (!now){
            x = y = 0;
            return;
        }
        else if (fhq[now].val <= val){
        x = now;
```

```
        split(fhq[now].r, val, fhq[now].r, y);
        }
        else{
        y = now;
        split(fhq[now].l, val, x, fhq[now].l);
        }
    pushup(now);
    }
    int merge(int x, int y){
        if (!x || !y)
            return x + y;
        if (fhq[x].key > fhq[y].key){
            fhq[x].r = merge(fhq[x].r, y);
            pushup(x);
            return x;
        }else{
            fhq[y].l = merge(x, fhq[y].l);
            pushup(y);
            return y;
        }
    }
    inline void insert(int val){
        int x, y;
        split(root, val, x, y);
        root = merge(merge(x, newnode(val)), y);
    }
    inline void del(int val){
        int x, y, z;
        split(root, val - 1, x, y);
        split(y, val, y, z);
        y = merge(fhq[y].l, fhq[y].r);
        root = merge(merge(x, y), z);
    }
    inline int getrk(int num){
        int x, y;
        split(root, num - 1, x, y);
        int ans = fhq[x].size + 1;
        root = merge(x, y);
        return ans;
    }
    inline int getnum(int rank){
        int now=root;
        while(now)
        {
            if(fhq[fhq[now].l].size+1==rank)
                break;
            else if(fhq[fhq[now].l].size>=rank)
                now=fhq[now].l;
            else{
                rank-=fhq[fhq[now].l].size+1;
                now=fhq[now].r;
            }
        }
        return fhq[now].val;
    }
```

```
    inline int pre(int val){
        int x, y, ans;
        split(root, val - 1, x, y);
        int t = x;
        while (fhq[t].r)
            t = fhq[t].r;
        ans = fhq[t].val;
        root = merge(x, y);
        return ans;
    }
    inline int aft(int val){
        int x, y, ans;
        split(root, val, x, y);
        int t = y;
        while (fhq[t].l)
            t = fhq[t].l;
        ans = fhq[t].val;
        root = merge(x, y);
        return ans;
    }
} tree;
```

5.4.2 替罪羊树

```
struct node
{
        int l, r, val;
        int size, fact;
        bool exsit;
};
class tzy_tree
{
    public:
    double alpha=0.75;
    node tzy[1100000];
    int root=0,cnt=0;
    vector<int> tt;
    inline void newnode(int &now,int v){
        now=++cnt;
        tzy[now].val=v;
        tzy[now].l=tzy[now].r=0;
        tzy[now].size=tzy[now].fact=tzy[now].exsit=1;
    }
    inline bool imbalanced(int now){
        if(max(tzy[tzy[now].l].size,tzy[tzy[now].r].size)>tzy[now].size*alpha||tzy[now].size-tzy[now].fact>tzy
            [now].size*0.3)
            return true;
        return false;
    }
    void zhongxu(int now){
        if(!now)
            return;
        zhongxu(tzy[now].l);
        if(tzy[now].exsit)
```

```
            tt.push_back(now);
        zhongxu(tzy[now].r);
    }
    void lift(int l,int r,int &now){
        if(l==r){
            now=tt[l];
            tzy[now].l=tzy[now].r=0;
            tzy[now].fact=tzy[now].size=1;
            return;
        }
        int m=(l+r)>>1;
        while(l<m&&tzy[tt[m]].val==tzy[tt[m-1]].val)
            m--;
        now=tt[m];
        if(l<m) lift(l,m-1,tzy[now].l);
        else tzy[now].l=0;
        lift(m+1,r,tzy[now].r);
        tzy[now].size=tzy[tzy[now].l].size+tzy[tzy[now].r].size+1;
        tzy[now].fact=tzy[tzy[now].l].fact+tzy[tzy[now].r].fact+1;
    }
    void rebuild(int &now){
        tt.clear();
        zhongxu(now);
        if(tt.empty()){
            now=0;
            return;
        }
        lift(0,tt.size()-1,now);
    }
    void update(int now,int end){
        if(!now)return;
        if(tzy[end].val<tzy[now].val)
            update(tzy[now].l,end);
        else update(tzy[now].r,end);
        tzy[now].size=tzy[tzy[now].l].size+tzy[tzy[now].r].size+1;
    }
    void check(int &now,int end){
        if(now==end)
            return;
        if(imbalanced(now)){
            rebuild(now);
            update(root,now);
            return;
        }
        if(tzy[end].val<tzy[now].val)
            check(tzy[now].l,end);
        else
            check(tzy[now].r,end);
    }
    inline void init(){
        root=1;
        cnt=0;
    }
    void insert(int &now,int val){
```

```
        if(!now){
            newnode(now,val);
            check(root,now);
            return;
        }
        tzy[now].fact++,tzy[now].size++;
        if(val<tzy[now].val)
            insert(tzy[now].l,val);
        else
            insert(tzy[now].r,val);
    }
    void erase(int now,int val){
        if(tzy[now].exsit&&tzy[now].val==val){
            tzy[now].exsit=false;
            tzy[now].fact--;
            check(root,now);
            return;
        }
        tzy[now].fact--;
        if(val<tzy[now].val)
            erase(tzy[now].l,val);
        else erase(tzy[now].r,val);
    }
    int getrank(int val){
        int now=root,rk=1;
        while(now){
            if(val<=tzy[now].val)
                now=tzy[now].l;
            else rk+=tzy[now].exsit+tzy[tzy[now].l].fact,now=tzy[now].r;
        }
        return rk;
    }
    int getnum(int rk){
        int now=root;
        while(now){
            if(tzy[now].exsit&&tzy[tzy[now].l].fact+tzy[now].exsit==rk)
                break;
            else if(tzy[tzy[now].l].fact>=rk){
                now=tzy[now].l;
            }else{
                rk-=tzy[tzy[now].l].fact+tzy[now].exsit;
                now=tzy[now].r;
            }
        }
        return tzy[now].val;
    }
}tree;
```

5.5 KD tree

```
inline void updmin(int &x,int y){x=min(x,y);}
inline void updmax(int &x,int y){x=max(x,y);}
int Dim;
struct KDT{
```

```
    int root;
    int dim,top=0,tot=0,*rub;
    struct point{
        int x[2],w;
        bool operator<(const point&b){
            return x[Dim]<b.x[Dim];
        }
    }*p;
    struct node{
        int mi[2],mx[2],sum=0,ls,rs,sz=0;
        point p;
    }*tr;
    int newnode(){
        if(top) return rub[top--];
        else return ++tot;
    }
    void up(int u){
        int l=tr[u].ls,r=tr[u].rs;
        rep(i,0,1){
            tr[u].mi[i]=tr[u].mx[i]=tr[u].p.x[i];
            if(l){
                updmin(tr[u].mi[i],tr[l].mi[i]);
                updmax(tr[u].mx[i],tr[l].mx[i]);
            }
            if(r){
                updmin(tr[u].mi[i],tr[r].mi[i]);
                updmax(tr[u].mx[i],tr[r].mx[i]);
            }
        }
        tr[u].sum=tr[l].sum+tr[r].sum+tr[u].p.w;
        tr[u].sz=tr[l].sz+tr[r].sz+1;
    }
    int build(int l,int r,int dim){
        if(l>r) return 0;
        int mid=l+r>>1,u=newnode();
        Dim=dim; nth_element(p+l,p+mid,p+r+1);
        tr[u].p=p[mid];
        tr[u].ls=build(l,mid-1,dim^1),tr[u].rs=build(mid+1,r,dim^1);
        up(u);
        return u;
    }
    void pia(int u,int num){//传统?
        int l=tr[u].ls,r=tr[u].rs;
        if(l) pia(l,num);
        p[tr[l].sz+1+num]=tr[u].p,rub[++top]=u;
        if(r) pia(r,num+1+tr[l].sz);
    }
    void balance(int &u,int dim){
        if(tr[u].sz*0.75<tr[tr[u].ls].sz||
           tr[u].sz*0.75<tr[tr[u].rs].sz){
            pia(u,0); u=build(1,tr[u].sz,dim);
        }
    }
    void insert(int &u,point p,int dim){
        if(!u) {
```

```
            u=newnode(),tr[u].p=p;
            tr[u].ls=tr[u].rs=0;
            up(u); return; //待修改
        }
        if(p.x[dim]<=tr[u].p.x[dim]) insert(tr[u].ls,p,dim^1);
        else insert(tr[u].rs,p,dim^1);
        up(u); balance(u,dim);
    }
    int in(int x1,int y1,int x2,int y2,int X1,int Y1,int X2,int Y2){
        return x1>=X1&&x2<=X2&&y1>=Y1&&y2<=Y2;
    }//左是否在右内
    int out(int x1,int y1,int x2,int y2,int X1,int Y1,int X2,int Y2){
        return x2<X1||x1>X2||y2<Y1||y1>Y2;
    }//左是否在右外
    int query(int u,int x1,int y1,int x2,int y2){
        if(!u) return 0;
        auto mx=tr[u].mx,mi=tr[u].mi,x=tr[u].p.x;
        int res=0;
        if(in(mi[0],mi[1],mx[0],mx[1],x1,y1,x2,y2)) return tr[u].sum;
        if(out(mi[0],mi[1],mx[0],mx[1],x1,y1,x2,y2)) return 0;
        if(in(x[0],x[1],x[0],x[1],x1,y1,x2,y2)) res+=tr[u].p.w;
        res+=query(tr[u].ls,x1,y1,x2,y2)+query(tr[u].rs,x1,y1,x2,y2);
        return res;
    }
    KDT(int maxn=1e6+10){
        tr=new node[maxn],p=new point[maxn];
        rub = new int[maxn],root=0;
    }
    void insert(int x,int y,int k){insert(root,(point){x,y,k},0);}
    int query(int x1,int y1,int x2,int y2){return query(root,x1,y1,x2,y2);}
};
```

6 字符串

6.1 KMP

```
const int MAXN = 2e6+5;
int pi[MAXN];//MAXN记得开大一点,因为这里要存到m+n+1长度的
vector<int> res;//储存答案

void getpi(const string &s){ //求s的前缀函数
    pi[0]=0;
    int j=0;
    rep(i,1,s.length()-1){
        while(j>0&&s[i]!=s[j]) j=pi[j-1];//找到合适且最长的j
        if(s[i]==s[j])j++;//能成功匹配的情况
        pi[i]=j;
    }
}

void kmp(string s,string t){ //在主串t中找模式串s
    getpi(s+'#'+t);
    int n=(int)s.length(),m=(int)t.length();
```

```
    rep(i,n+1,m+n+1-1)
        if(pi[i]==n) res.push_back(i-2*s.size()); //i-2n计算得左端点
}
```

6.2 AC 自动机

```
const int MAXN = 1e5+5;
int jdbh[MAXN];//记录第i个模式串对应的节点编号
int cntcx[MAXN];//记录第i个模式串出现的次数
inline int idx(char c){return c-'a';}
struct Node{
    int son[26],flag,fail;//cnt记录次数,flag记录编号
    void clr(){
        memset(son,0,sizeof(son));
        flag=0;
    }
}trie[MAXN*10];
int n,cntt;//cntt记录总点数
string s,ms[166];
int maxx;
queue<int>q;
inline void insert(string &s,int num){
    int siz = s.size(),v,u=1;
    rep(i,0,siz-1){
        v = idx(s[i]);
        if(!trie[u].son[v]){trie[u].son[v] = ++cntt;trie[cntt].clr();}
        u = trie[u].son[v];
    }
    trie[u].flag = num;//标记为单词,flag记录编号
    //保证每个模式串只出现一次
    cntcx[num] = 0;
    jdbh[num] = u;//记录当前单词对应的节点编号
}
inline void getfail(){
    rep(i,0,25) trie[0].son[i] = 1;
    trie[0].flag = 0;
    q.push(1);
    trie[1].fail = 0;
    int u,v,ufail;
    while(!q.empty()){
        u = q.front();q.pop();
        rep(i,0,25){
            v = trie[u].son[i];
            ufail = trie[u].fail;
            if(!v){trie[u].son[i]=trie[ufail].son[i];continue;}//画好一条跳fail的路
            trie[v].fail = trie[ufail].son[i];
            q.push(v);
        }
    }
}
inline void query(string &s){
    int siz = s.size(),u = 1,v,k;
    rep(i,0,siz-1){
        v = idx(s[i]);
```

```
        k = trie[u].son[v];
        while(k){
            if(trie[k].flag){
                cntcx[trie[k].flag]++;//计数
                maxx = max(maxx,cntcx[trie[k].flag]);
            }
            k = trie[k].fail;//跳fail
        }
        u = trie[u].son[v];//这一句其实也有跳fail的功能，很精妙
    }
}
inline void solve(){
    cntt = 1;
    trie[0].clr();
    trie[1].clr();
    rep(i,1,n){
        cin>>ms[i];
        insert(ms[i],i);
    }
    getfail();
    cin>>s;
    maxx = 0;
    query(s);
    cout<<maxx<<endl;
    rep(i,1,n){
        if(cntcx[i]==maxx) cout<<ms[i]<<endl;
    }
}
```

6.3 FFT 解决字符串匹配问题

可以用来解决含有通配符的字符串匹配问题定义匹配函数

$$(x,y) = (A_x - B_x)^2$$

如果两个字符相同，则满足 $C(x,y) = 0$
定义模式串和文本串 x 位置对齐时候的完全匹配函数为

$$P(x) = \sum C(i, x+i)$$

模式串在位置 x 上匹配时,$p(x) = 0$
通过将模式串 reverse 后卷积，可以快速处理每个位置 x 上的完全匹配函数 $P(x)$ 同理，如果包含通配符，则设通配符的值为 0，可以构造损失函数

$$C(x,y) = (A_x - B_x)^2 \cdot A_x \cdot B_x = A_x^3 B_x + A_x B_x^3 - 2A_x^2 B_x^2$$

通过三次 FFT 即可求得每个位置上的 P(x)
以下是用 FFT 解决普通字符串匹配问题的代码
即实现 KMP 的功能，复杂度较高，为 $O(nlog_n)$

```
void solve(){
    limit = 1,l=0;
    cin>>n>>m;
    cin>>s1>>s2;
    rep(i,0,n-1) B[i].x = s1[i]-'a'+1;
    rep(i,0,m-1) A[i].x = s2[i]-'a'+1;
    double T = 0;
```

```
    //T = sigma A[i]^A[i] i=0~m-1
    rep(i,0,m-1) T += A[i].x*A[i].x;
    //f[x] = sigma B[i]^B[i] i=0~x
    f[0] = B[0].x*B[0].x;
    rep(i,1,n-1) f[i] = f[i-1]+B[i].x*B[i].x;
    //g[x] = S[i]*B[j] i+j==x
    reverse(A,A+m);//S = A.reverse
    //FFT预处理
    while(limit<=n+m-2) limit<<=1,l++;
    rep(i,0,limit-1)
        r[i]= ( r[i>>1]>>1 )| ( (i&1)<<(l-1) );

    FFT(A,1);FFT(B,1);
    rep(i,0,limit) A[i]=A[i]*B[i];
    FFT(A,-1);
    rep(i,0,n-1) g[i] = (int)(A[i].x/limit+0.5);//四舍五入

    //T + f(x) - f(x-m) - 2g(x);
    double tmp;
    rep(x,m-1,n-1){
        tmp = T+f[x]-2*g[x];
        if(x!=m-1) tmp -= f[x-m];
        //cout<<tmp<<' ';
        if(fabs(tmp)<eps) cout<<x-(m-1)+1<<endl;//输出匹配上的位置
    }
    cout<<endl;
}
```

6.4 字符串哈希

快速取子串哈希值

```
const int b = 131;//推荐的base，可以选其他质数
void init(int n){//初始化
    pw[0] = 1;
    for (int i = 1; i <= n; i ++ ) {
        h[i] = h[i-1]*b + str[i];//做每个前缀的哈希值
        pw[i] = pw[i-1]*b;//预处理b^k的值
    }
}
// 计算子串 str[l ~ r] 的哈希值
ull get(int l, int r) {
    return h[r] - h[l-1]*pw[r-l+1];
}
```

6.5 后缀数组 SA+LCP

LCP(i,j) 后缀 i 和后缀 j 的最长公共前缀

```
int n,m;
string s;
int rk[MAXN],sa[MAXN],c[MAXN],rk2[MAXN];
//sa[i]存排名i的原始编号 rk[i]存编号i的排名 第二关键字rk2
inline void get_SA(){
    rep(i,1,n) ++c[rk[i]=s[i]];//基数排序
```

```
    rep(i,2,m) c[i] += c[i-1];
    //c做前缀和，可以知道每个关键字的排名最低在哪里
    repb(i,n,1) sa[c[rk[i]]--] = i;//记录每个排名的原编号

    for(int w=1;w<=n;w<<=1){//倍增
        int num = 0;
        rep(i,n-w+1,n) rk2[++num] = i;//没有第二关键字的排在前面
        rep(i,1,n) if(sa[i]>w) rk2[++num] = sa[i]-w;
        //编号sa[i]大于w的才能作为编号sa[i]-w的第二关键字
        rep(i,1,m) c[i] = 0;
        rep(i,1,n) ++c[rk[i]];
        rep(i,2,m) c[i]+=c[i-1];
        repb(i,n,1) sa[c[rk[rk2[i]]]--]=rk2[i],rk2[i]=0;
        //同一个桶中按照第二关键字排序
        swap(rk,rk2);
        //这时候的rk2时这次排序用到的上一轮的rk,要计算出新的rk给下一轮排序

        rk[sa[1]]=1,num=1;
        rep(i,2,n)
            rk[sa[i]] = (rk2[sa[i]]==rk2[sa[i-1]]&&rk2[sa[i]+w]==rk2[sa[i-1]+w])?num:++num;
        //下一次排名的第一关键字,相同的两个元素排名也相同
        if(num==n) break;//rk都唯一时，排序结束
        m=num;
    }
}
int height[MAXN];
inline void get_height(){
    int k = 0,j;
    rep(i,1,n) rk[sa[i]] = i;
    rep(i,1,n){
        if(rk[i]==1) continue;//第一名往前没有前缀
        if(k) k--;//h[i]>=h[i-1]-1 即height[rk[i]]>=height[rk[i-1]]-1
        j = sa[rk[i]-1];//找排在rk[i]前面的
        while(j+k<=n&&i+k<=n&&s[i+k]==s[j+k]) ++k;//逐字符比较
        //因为每次k只会-1，故++k最多只会加2n次
        height[rk[i]] = k;
    }
}
inline void solve(){
    cin>>s;
    s = ' '+s;
    n = s.size()-1,m = 122;//m为字符个数'z'=122
    get_SA();
    rep(i,1,n) cout<<sa[i]<<' ';
    cout<<endl;
}
```

6.6 后缀自动机 SAM

```
struct state{
    int len,link;
    map<char,int> nxt;//也可以用数组,空间换时间
};
state sta[MAXN<<1];//状态数需要设定为两倍
```

```
int sz,last;//sz为自动机大小
inline void init_SAM(){
    sta[0].len = 0;sta[0].link = -1;//虚拟状态t0
    sz = 1;
    last = 0;
}
int cnt[MAXN<<1];
void SAM_extend(char c){
    int cur = sz++;
    cnt[cur] = 1;
    sta[cur].len = sta[last].len+1;
    int p = last;
    //沿着last的link添加到c的转移，直到找到已经有c转移的状态p
    while(p!=-1&&!sta[p].nxt.count(c)){
        sta[p].nxt[c] = cur;
        p = sta[p].link;
    }
    if(p==-1) sta[cur].link = 0;//情况1,没有符合的p
    else{
        int q = sta[p].nxt[c];
        if(sta[q].len==sta[p].len+1)//情况2,稳定的转移(lenq=lenp+1,前面没有增加)
            sta[cur].link = q;
        else{//情况3,把q的lenp+1的部分拿出来(clone),p到clone的转移是稳定的
            int clone = sz++;
            cnt[clone] = 0;
            sta[clone].len = sta[p].len+1;
            sta[clone].nxt = sta[q].nxt;
            sta[clone].link = sta[q].link;
            while(p!=-1 && sta[p].nxt[c]==q){//把向q的转移指向clone
                sta[p].nxt[c]=clone;
                p=sta[p].link;
            }
            sta[q].link = sta[cur].link = clone;//clone是q的后缀,故linkq=clone
        }
    }
    last = cur;//sta[last]包含目前处理的整个前缀!
}
string s;
vector<int> e[MAXN<<1];
void dfs(int now){
    for(auto to:e[now]){
        dfs(to);
        cnt[now] += cnt[to];
    }
}
inline void solve(){
    cin>>s;
    init_SAM();
    int siz = s.size();
    rep(i,0,siz-1) SAM_extend(s[i]);
    rep(i,1,sz-1) e[sta[i].link].push_back(i);//link边反过来构造树
    dfs(0);
    ll maxx = 0;
    rep(i,1,sz-1)
        if(cnt[i]!=1) maxx = max(maxx,1ll*cnt[i]*sta[i].len);
```

```
    cout<<maxx<<endl;
}
int main(){
    solve();
}
```

6.7 广义 SAM

```
//广义sam只需要在插入新串时把last设为1
//其实就是存一下自己的sam
struct SAM{
    static const int sigma=26,c0='a';
    struct Node{
        int fa=0,to[sigma],len=0;
        int &operator [] (int x) {return to[x];}
        Node(){
            rep(i,0,sigma-1) to[i]=0;
        }
    }a[N*2];
    int last=1,tot=1;
    int newnode(){
        a[++tot]=Node(); return tot;
    }

    ll sz[N*2],t[N*2],val[N*2];

    void insert(int c){
        bool fl=0; int z;
        c-=c0;
        //if(a[last][c]&&a[last].len+1==a[a[last][c]].len) return a[last][c];
        int x=last,next=newnode();
        sz[next]=1;
        last=next;
        a[next].len=a[x].len+1;
        for(;x&&!a[x][c];x=a[x].fa) a[x][c]=next;
        if(!x) a[next].fa=1;
        else{
            int y=a[x][c];
            if(a[y].len==a[x].len+1){
                a[next].fa=y;
            }
            else{
                if(a[x].len+1==a[next].len) fl=1;
                z=newnode();
                a[z]=a[y],a[z].len=a[x].len+1;
                //sz[z]=1;
                for(;x&&a[x][c]==y;x=a[x].fa) a[x][c]=z;
                a[y].fa=a[next].fa=z;
            }
        }
    }
    bool vis[N*2];
    void dfs(int u){
        vis[u]=1;
```

```
      sz[u]=1;
      rep(i,0,25) if(a[u][i]) {
         if(!vis[a[u][i]]) dfs(a[u][i]);
         sz[u]+=sz[a[u][i]];
      }
   }
   void solve(){
      int n=read();
      rep(i,1,n){
         last=1;
         string s; cin>>s;
         for(auto c:s) insert(c);
      }
      dfs(1);
      cout<<sz[1]-1<<endl;
   }

}sam;
```

7 其他

7.1 三分

```
while(l<r){//类似求导的方式求极值
   int x=(l+r)/2,y=x+1; //l+(r-l)/2
   if(f(x)<f(y))l=x+1; else r=y-1; //最大值
   //if(f(x)<f(y)) r-y-1; else l=x+1; //最小值
}
```

7.2 数位 dp

```
ll dp[20][2][2],bit[20]; //这个[2]表示lz状态，如果lz被使用了的话就需要记录
ll dfs(int pos,ll s,bool lim=1,bool lz=1){
   if(pos==-1)return !s; //返回该状态是否符合要求(0或1)
   ll &x=dp[pos][s][0];
   if(!lim && x!=-1)return x;
   ll ans=0;
   int maxi=lim?bit[pos]:9;
   if(s){
      if(maxi>=2) rep(i,0,1)
         ans+=dfs(pos-1,i,lim&&2==maxi,0);
   }
   else{
      rep(i,0,maxi){
         if(i==2||i==4) continue;
         //...//状态转移
         //if(lz && i==0)...//可能要用lz，其他地方都不用
         ans+=dfs(pos-1,0,
            lim && i==maxi,
            lz && i==0);
         if(i!=6) ans+=dfs(pos-1,1,lim&&i==maxi,lz&&i==0);
      }
```

```
    }
    if(!lim)x=ans; //不限制的时候才做存储 yyds
    return ans;
}
ll calc(ll n){
    int len=0;
    while(n)bit[len++]=n%10,n/=10;
    return dfs(len-1,0)+dfs(len-1,1);
}
```

7.3 线性基

```
struct basis{//线性基超大常数版
    ll p[64];
    #define B(x,i) ((x>>i)&1)
    //yyds
    void init(){ memset(p,0,sizeof(p)); }
    void add(ll x){
        dep(i,62,0){
            if(!B(x,i)) continue;
            if(!p[i]) {p[i]=x; break;}
            x^=p[i];
        }
    }
    ll qmx(ll x){
        dep(i,62,0) if(!B(x,i)&&p[i]) x^=p[i];
        return x;
    }
    ll qmn(ll x){
        dep(i,62,0) if(B(x,i)&&p[i]) x^=p[i];
        return x;
    }
}b;
```

7.4 莫队

```
int cnt[MAXN];//记录数字在区间[l,r]内出现的次数
int pos[MAXN],a[MAXN];
ll ans[MAXN];
int n,m,k,res;
struct Q{
    int l,r,k;//k记录原来的编号
    friend bool operator < (Q x,Q y){//同一个分块内r小的排前面;不同分块则按分块靠前的
        return pos[x.l]==pos[y.l]?x.r<y.r:pos[x.l]<pos[y.l];
        //return (pos[a.l]^pos[b.l])?pos[a.l]<pos[b.l]:((pos[a.l]&1)?a.r<b.r:a.r>b.r);
        //这条第一个和==是一样的,后面的是对于左端点在同一奇数块的区间，右端点按升序排列，反之降序
    }
}q[MAXN];

void Add(int pos){
    res -= cnt[a[pos]]*cnt[a[pos]];
    cnt[a[pos]]++;
    res += cnt[a[pos]]*cnt[a[pos]];
```

```
}
void Sub(int pos){
    res -= cnt[a[pos]]*cnt[a[pos]];
    cnt[a[pos]]--;
    res += cnt[a[pos]]*cnt[a[pos]];
}
int main(){
    cin>>n>>m>>k;//k为数字范围
    memset(cnt,0,sizeof(cnt));
    int siz = sqrt(n);//每个分块的大小
    rep(i,1,n){
        cin>>a[i];
        pos[i] = i/siz;//分块
    }
    rep(i,1,m){
        cin>>q[i].l>>q[i].r;
        q[i].k = i;//记录原来的编号,用于打乱顺序后的还原
    }
    sort(q+1,q+1+m);
    res = 0;//初始化res
    int l = 1,r = 0;//当前知道的区间
    //因为是闭区间,如果是[1,1]的话则一开始就包含一个元素了
    rep(i,1,m){//莫队的核心,注意加减的顺序
        while(q[i].l<l) Add(--l);
        while(q[i].l>l) Sub(l++);
        while(q[i].r<r) Sub(r--);
        while(q[i].r>r) Add(++r);
        ans[q[i].k] = res;
    }
    rep(i,1,m) cout<<ans[i]<<endl;
}
```

7.5 带修莫队

```
int a[MAXN],b[MAXN];//a读入一开始的序列,b记录修改后的
int pos[MAXN];//分块
int cq,cr;//统计查询修改次数
int R[MAXN][3];//0记位置,1记原本的值,2记修改后的值
ll res;
int ans[MAXN];//记录结果
int n,m;
void Add(int x){if(cnt[x]==0)res++;cnt[x]++;}//带修莫队的add和sub有区别
void Sub(int x){if(cnt[x]==1)res--;cnt[x]--;}
struct Q{
    int l,r,k,t;
    friend bool operator < (Q a,Q b){
        return (pos[a.l]^pos[b.l])?pos[a.l]<pos[b.l]:((pos[a.r]^pos[b.r])?a.r<b.r:a.t<b.t);
        //增加第三关键字,询问的先后顺序,用t或者k应该都行
    }
}q[MAXN];
int main(){
    cin>>n>>m;
    cq = cr = 0;
    int siz = pow(n,2.0/3.0);//这么分块最好,别问
```

```
    rep(i,1,n){
        cin>>a[i];
        b[i]=a[i];
        pos[i] = i/siz;
    }
    char hc;
    rep(i,1,m){//读入修改和询问
        cin>>hc;
        if(hc=='Q'){
            cin>>q[cq].l>>q[cq].r;
            q[cq].k=cq;q[cq].t=cr;//注意这时候R[cr]还是没有的,这次询问是在R[cr-1]之后的
            cq++;
        }
        else{
            cin>>R[cr][0]>>R[cr][2];
            R[cr][1] = b[R[cr][0]];
            b[R[cr][0]] = R[cr][2];//在b数组中记录更改
            cr++;
        }
    }
    sort(q,q+cq);
    int l=1,r=0,sjc=0;//时间戳
    res = 0;
    rep(i,0,cq-1){
        while(sjc<q[i].t){
            if(l<=R[sjc][0]&&R[sjc][0]<=r)//判断修改是否在该区间内
                Sub(R[sjc][1]),Add(R[sjc][2]);
            a[R[sjc][0]] = R[sjc][2];//在a上也进行更改
            sjc++;
        }
        while(sjc>q[i].t){
            sjc--;
            if(l<=R[sjc][0]&&R[sjc][0]<=r)//判断修改是否在该区间内
                Sub(R[sjc][2]),Add(R[sjc][1]);
            a[R[sjc][0]] = R[sjc][1];//在a上也进行更改
        }
        while(l>q[i].l) Add(a[--l]);
        while(l<q[i].l) Sub(a[l++]);
        while(r<q[i].r) Add(a[++r]);
        while(r>q[i].r) Sub(a[r--]);
        ans[q[i].k] = res;
    }
    rep(i,0,cq-1) cout<<ans[i]<<endl;
}
```

8 STL 等小技巧

8.1 集合 set

还可以通过 lower_bound 和 upper_bound 返回迭代器来找前驱, 后继

```
//并交集
vector<int> ANS;
set_union(s1.begin(),s1.end(),s2.begin(),s2.end(),inserter(ANS,ANS.begin()));//set_intersection()
```

```cpp

//通过迭代器遍历集合
set<char>::iterator iter = temp1.begin();
while (iter!=temp1.end()){
    cout<<*iter;
    iter++;
}
```

8.2 快读快写 (短)

```cpp
template<class T>inline void read(T &x){x=0;char o,f=1;while(o=getchar(),o<48)if(o==45)f=-f;do x=(x<<3)+(x
    <<1)+(o^48);while(o=getchar(),o>47);x*=f;}
template<class T>
void wt(T x){//快写
   if(x < 0) putchar('-'), x = -x;
   if(x >= 10) wt(x / 10);
   putchar('0' + x % 10);
}
```

8.3 GCD(压行)

```cpp
ll gcd(ll a,ll b){ while(b^=a^=b^=a%=b); return a; }
```

8.4 计时

```cpp
inline double run_time(){
    return 1.0*clock()/CLOCKS_PER_SEC;
}
```

8.5 替换 unorderedset 的 hash 函数

```cpp
struct VectorHash {
    size_t operator()(const std::vector<int>& v) const {
        std::hash<int> hasher;
        size_t seed = 0;
        for (int i : v) {
            seed ^= hasher(i) + 0x9e3779b9 + (seed<<6) + (seed>>2);
        }
        return seed;
    }
};
unordered_set<vector<int>,VectorHash> st;
```