Contents

1 Basic

1.1 .vimrc

```
syn on
se ai nu ru cul mouse=a
se cin et ts=2 sw=2 sts=2
so $VIMRUNTIME/mswin.vim
colo desert
se gfn=Monospace\ 14
```

1.2 Default code

```
#include<bits/stdc++.h>
#define io ios_base::sync_with_stdio(0);cin.tie(0);cout
    .tie(0);
#define endl '\n'
#define MOD 0x3f3f3f3f
#define llMOD 0x3f3f3f3f3f3f3f3f
typedef long long ll;
typedef unsigned long long ull;
using namespace std;


int main(){
    io;

}
```

1.3 Increase Stack Size

```
//stack resize (linux)
#include <sys/resource.h>
void increase_stack_size() {
  const rlim_t ks = 64*1024*1024;
  struct rlimit rl;
  int res=getrlimit(RLIMIT_STACK, &rl);
  if(res==0){
    if(rl.rlim_cur<ks){
      rl.rlim_cur=ks;
      res=setrlimit(RLIMIT_STACK, &rl);
} } }
```

2 Math

2.1 O(1)mul

```
LL mul(LL x,LL y,LL mod){
  LL ret=x*y-(LL)((long double)x/mod*y)*mod;
  // LL ret=x*y-(LL)((long double)x*y/mod+0.5)*mod;
  return ret<0?ret+mod:ret;
}
```

2.2 BigInt

```
struct Bigint{
  static const int LEN = 60;
  static const int BIGMOD = 10000;
  int s;
  int vl, v[LEN];
  //  vector<int> v;
  Bigint() : s(1) { vl = 0; }
  Bigint(long long a) {
    s = 1; vl = 0;
    if (a < 0) { s = -1; a = -a; }
    while (a) {
      push_back(a % BIGMOD);
      a /= BIGMOD;
  } }
  Bigint(string str) {
    s = 1; vl = 0;
    int stPos = 0, num = 0;
    if (!str.empty() && str[0] == '-') {
      stPos = 1;
      s = -1;
    }
    for (int i=SZ(str)-1, q=1; i>=stPos; i--) {
      num += (str[i] - '0') * q;
      if ((q *= 10) >= BIGMOD) {
        push_back(num);
        num = 0; q = 1;
    } }
    if (num) push_back(num);
    n();
  }
  int len() const {
    return vl;  // return SZ(v);
  }
  bool empty() const { return len() == 0; }
  void push_back(int x) {
    v[vl++] = x; // v.PB(x);
  }
  void pop_back() {
    vl--; // v.pop_back();
  }
  int back() const {
    return v[vl-1]; // return v.back();
  }
  void n() {
    while (!empty() && !back()) pop_back();
  }
  void resize(int nl) {
    vl = nl;
    fill(v, v+vl, 0);
    //    v.resize(nl);
    //    fill(ALL(v), 0);
  }
  void print() const {
    if (empty()) { putchar('0'); return; }
    if (s == -1) putchar('-');
    printf("%d", back());
    for (int i=len()-2; i>=0; i--) printf("%.4d",v[i]);
  }
  friend std::ostream& operator << (std::ostream& out,
      const Bigint &a) {
    if (a.empty()) { out << "0"; return out; }
    if (a.s == -1) out << "-";
    out << a.back();
    for (int i=a.len()-2; i>=0; i--) {
      char str[10];
      snprintf(str, 5, "%.4d", a.v[i]);
      out << str;
    }
    return out;
  }
  int cp3(const Bigint &b)const {
    if (s != b.s) return s - b.s;
    if (s == -1) return -(-*this).cp3(-b);
    if (len() != b.len()) return len()-b.len();//int
    for (int i=len()-1; i>=0; i--)
      if (v[i]!=b.v[i]) return v[i]-b.v[i];
    return 0;
  }
  bool operator<(const Bigint &b)const
    { return cp3(b)<0; }
  bool operator<=(const Bigint &b)const
    { return cp3(b)<=0; }
  bool operator==(const Bigint &b)const
    { return cp3(b)==0; }
  bool operator!=(const Bigint &b)const
    { return cp3(b)!=0; }
  bool operator>(const Bigint &b)const
    { return cp3(b)>0; }
  bool operator>=(const Bigint &b)const
    { return cp3(b)>=0; }
  Bigint operator - () const {
    Bigint r = (*this);
    r.s = -r.s;
    return r;
  }
  Bigint operator + (const Bigint &b) const {
    if (s == -1) return -(-(*this)+(-b));
    if (b.s == -1) return (*this)-(-b);
    Bigint r;
    int nl = max(len(), b.len());
    r.resize(nl + 1);
    for (int i=0; i<nl; i++) {
      if (i < len()) r.v[i] += v[i];
      if (i < b.len()) r.v[i] += b.v[i];
      if(r.v[i] >= BIGMOD) {
        r.v[i+1] += r.v[i] / BIGMOD;
        r.v[i] %= BIGMOD;
    } }
    r.n();
    return r;
  }
  Bigint operator - (const Bigint &b) const {
    if (s == -1) return -(-(*this)-(-b));
    if (b.s == -1) return (*this)+(-b);
    if ((*this) < b) return -(b-(*this));
    Bigint r;
    r.resize(len());
    for (int i=0; i<len(); i++) {
      r.v[i] += v[i];
      if (i < b.len()) r.v[i] -= b.v[i];
      if (r.v[i] < 0) {
        r.v[i] += BIGMOD;
        r.v[i+1]--;
    } }
    r.n();
    return r;
  }
  Bigint operator * (const Bigint &b) {
    Bigint r;
    r.resize(len() + b.len() + 1);
    r.s = s * b.s;
    for (int i=0; i<len(); i++) {
      for (int j=0; j<b.len(); j++) {
        r.v[i+j] += v[i] * b.v[j];
        if(r.v[i+j] >= BIGMOD) {
          r.v[i+j+1] += r.v[i+j] / BIGMOD;
          r.v[i+j] %= BIGMOD;
    } } }
    r.n();
    return r;
  }
  Bigint operator / (const Bigint &b) {
    Bigint r;
    r.resize(max(1, len()-b.len()+1));
    int oriS = s;
    Bigint b2 = b; // b2 = abs(b)
    s = b2.s = r.s = 1;
    for (int i=r.len()-1; i>=0; i--) {
      int d=0, u=BIGMOD-1;
      while(d<u) {
        int m = (d+u+1)>>1;
        r.v[i] = m;
        if((r*b2) > (*this)) u = m-1;
        else d = m;
      }
      r.v[i] = d;
    }
    s = oriS;
    r.s = s * b.s;
    r.n();
    return r;
  }
  Bigint operator % (const Bigint &b) {
    return (*this)-(*this)/b*b;
} };
```

2.3 Linear Recurrence

```
// Usage: linearRec({0, 1}, {1, 1}, k) //k'th fib
typedef vector<ll> Poly;
//S:前i項的值,tr:遞迴系數,k:求第k項
ll linearRec(Poly& S, Poly& tr, ll k) {
  int n = tr.size();
  auto combine = [&](Poly& a, Poly& b) {
    Poly res(n * 2 + 1);
    rep(i,0,n+1) rep(j,0,n+1)
      res[i+j]=(res[i+j] + a[i]*b[j])%mod;
    for(int i = 2*n; i > n; --i) rep(j,0,n)
      res[i-1-j]=(res[i-1-j] + res[i]*tr[j])%mod;
    res.resize(n + 1);
    return res;
  };
  Poly pol(n + 1), e(pol);
  pol[0] = e[1] = 1;
  for (++k; k; k /= 2) {
    if (k % 2) pol = combine(pol, e);
    e = combine(e, e);
  }
  ll res = 0;
  rep(i,0,n) res=(res + pol[i+1]*S[i])%mod;
  return res;
}
```

2.4 Fast Pow

```
ll mypow(ll m, ll n, ll mod){
    ll ans=1;
    for (; n > 0; n >>= 1){
        if (n&1)
            ans = ans * m % mod;
        m = m * m % mod;
    }
}
```

2.5 Miller Rabin

```
// n < 4,759,123,141        3 :  2, 7, 61
// n < 1,122,004,669,633    4 :  2, 13, 23, 1662803
// n < 3,474,749,660,383          6 :  pirmes <= 13
// n < 2^64                       7 :
// 2, 325, 9375, 28178, 450775, 9780504, 1795265022
// Make sure testing integer is in range [2, n□2] if
// you want to use magic.
LL magic[]={}
bool witness(LL a,LL n,LL u,int t){
  if(!a) return 0;
  LL x=mypow(a,u,n);
  for(int i=0;i<t;i++) {
    LL nx=mul(x,x,n);
    if(nx==1&&x!=1&&x!=n-1) return 1;
    x=nx;
  }
  return x!=1;
}
bool miller_rabin(LL n) {
  int s=(magic number size)
  // iterate s times of witness on n
  if(n<2) return 0;
  if(!(n&1)) return n == 2;
  ll u=n-1; int t=0;
  // n-1 = u*2^t
  while(!(u&1)) u>>=1, t++;
  while(s--){
    LL a=magic[s]%n;
    if(witness(a,n,u,t)) return 0;
  }
  return 1;
}
```

2.6 Simplex 線性規劃

```
const int MAXN = 111;
const int MAXM = 111;
const double eps = 1E-10;
double a[MAXN][MAXM], b[MAXN], c[MAXM], d[MAXN][MAXM];
double x[MAXM];
int ix[MAXN + MAXM]; // !!! array all indexed from 0
// max{cx} subject to {Ax<=b,x>=0}
// n: constraints, m: vars !!!
// x[] is the optimal solution vector
// usage :
// value = simplex(a, b, c, N, M);
double simplex(double a[MAXN][MAXM], double b[MAXN],
               double c[MAXM], int n, int m){
  ++m;
  int r = n, s = m - 1;
  memset(d, 0, sizeof(d));
  for (int i = 0; i < n + m; ++i) ix[i] = i;
  for (int i = 0; i < n; ++i) {
    for (int j = 0; j < m - 1; ++j) d[i][j] = -a[i][j];
    d[i][m - 1] = 1;
    d[i][m] = b[i];
    if (d[r][m] > d[i][m]) r = i;
  }
  for (int j = 0; j < m - 1; ++j) d[n][j] = c[j];
  d[n + 1][m - 1] = -1;
  for (double dd;; ) {
    if (r < n) {
      int t = ix[s]; ix[s] = ix[r + m]; ix[r + m] = t;
      d[r][s] = 1.0 / d[r][s];
      for (int j = 0; j <= m; ++j)
        if (j != s) d[r][j] *= -d[r][s];
      for (int i = 0; i <= n + 1; ++i) if (i != r) {
        for (int j = 0; j <= m; ++j) if (j != s)
          d[i][j] += d[r][j] * d[i][s];
        d[i][s] *= d[r][s];
      }
    }
    r = -1; s = -1;
    for (int j = 0; j < m; ++j)
      if (s < 0 || ix[s] > ix[j]) {
        if (d[n + 1][j] > eps ||
            (d[n + 1][j] > -eps && d[n][j] > eps))
          s = j;
      }
    if (s < 0) break;
    for (int i = 0; i < n; ++i) if (d[i][s] < -eps) {
      if (r < 0 ||
          (dd = d[r][m] / d[r][s] - d[i][m] / d[i][s])
              < -eps ||
          (dd < eps && ix[r + m] > ix[i + m]))
        r = i;
    }
    if (r < 0) return -1; // not bounded
  }
  if (d[n + 1][m] < -eps) return -1; // not executable
  double ans = 0;
  for(int i=0; i<m; i++) x[i] = 0;
  for (int i = m; i < n + m; ++i) { // the missing
      enumerated x[i] = 0
    if (ix[i] < m - 1){
      ans += d[i - m][m] * c[ix[i]];
      x[ix[i]] = d[i-m][m];
    }
  }
  return ans;
}
```

2.7 Faulhaber $(\sum_{i=1}^{n} i^p)$

```
/* faulhaber’ s formula -
 * cal power sum formula of all p=1~k in O(k^2) */
#define MAXK 2500
const int mod = 1000000007;
int b[MAXK]; // bernoulli number
int inv[MAXK+1]; // inverse
int cm[MAXK+1][MAXK+1]; // combinactories
int co[MAXK][MAXK+2]; // coeeficient of x^j when p=i
inline int getinv(int x) {
  int a=x,b=mod,a0=1,a1=0,b0=0,b1=1;
  while(b) {
    int q,t;
    q=a/b; t=b; b=a-b*q; a=t;
    t=b0; b0=a0-b0*q; a0=t;
    t=b1; b1=a1-b1*q; a1=t;
  }
  return a0<0?a0+mod:a0;
}
inline void pre() {
  /* combinational */
  for(int i=0;i<=MAXK;i++) {
    cm[i][0]=cm[i][i]=1;
    for(int j=1;j<i;j++)
      cm[i][j]=add(cm[i-1][j-1],cm[i-1][j]);
  }
  /* inverse */
  for(int i=1;i<=MAXK;i++) inv[i]=getinv(i);
  /* bernoulli */
  b[0]=1; b[1]=getinv(2); // with b[1] = 1/2
  for(int i=2;i<MAXK;i++) {
    if(i&1) { b[i]=0; continue; }
    b[i]=1;
    for(int j=0;j<i;j++)
      b[i]=sub(b[i],
               mul(cm[i][j],mul(b[j], inv[i-j+1])));
  }
  /* faulhaber */
  // sigma_x=1~n {x^p} =
  //   1/(p+1) * sigma_j=0~p {C(p+1,j)*Bj*n^(p-j+1)}
  for(int i=1;i<MAXK;i++) {
    co[i][0]=0;
    for(int j=0;j<=i;j++)
      co[i][i-j+1]=mul(inv[i+1], mul(cm[i+1][j], b[j]))
          ;
  }
}
/* sample usage: return f(n,p) = sigma_x=1~n (x^p) */
inline int solve(int n,int p) {
  int sol=0,m=n;
  for(int i=1;i<=p+1;i++) {
    sol=add(sol,mul(co[p][i],m));
    m = mul(m, n);
  }
  return sol;
}
```

2.8 Chinese Remainder

```
LL x[N],m[N];
LL CRT(LL x1, LL m1, LL x2, LL m2) {
  LL g = __gcd(m1, m2);
  if((x2 - x1) % g) return -1;// no sol
  m1 /= g; m2 /= g;
  pair<LL,LL> p = gcd(m1, m2);
  LL lcm = m1 * m2 * g;
  LL res = p.first * (x2 - x1) * m1 + x1;
  return (res % lcm + lcm) % lcm;
}
LL solve(int n){ // n>=2,be careful with no solution
  LL res=CRT(x[0],m[0],x[1],m[1]),p=m[0]/__gcd(m[0],m
      [1])*m[1];
  for(int i=2;i<n;i++){
    res=CRT(res,p,x[i],m[i]);
    p=p/__gcd(p,m[i])*m[i];
  }
  return res;
}
```

2.9 Pollard Rho

```
// does not work when n is prime  O(n^(1/4))
LL f(LL x, LL mod){ return add(mul(x,x,mod),1,mod); }
LL pollard_rho(LL n) {
  if(!(n&1)) return 2;
  while(true){
    LL y=2, x=rand()%(n-1)+1, res=1;
    for(int sz=2; res==1; sz*=2) {
      for(int i=0; i<sz && res<=1; i++) {
        x = f(x, n);
        res = __gcd(abs(x-y), n);
      }
      y = x;
    }
    if (res!=0 && res!=n) return res;
} }
```

2.10 Josephus Problem

```
int josephus(int n, int m){ //n人每m次
    int ans = 0;
    for (int i=1; i<=n; ++i)
        ans = (ans + m) % i;
    return ans;
}
```

2.11 Gaussian Elimination 高斯消

```
const int GAUSS_MOD = 100000007LL;
struct GAUSS{
    int n;
    vector<vector<int>> v;
    int ppow(int a , int k){
        if(k == 0) return 1;
        if(k % 2 == 0) return ppow(a * a % GAUSS_MOD ,
            k >> 1);
        if(k % 2 == 1) return ppow(a * a % GAUSS_MOD ,
            k >> 1) * a % GAUSS_MOD;
    }
    vector<int> solve(){
        vector<int> ans(n);
        REP(now , 0 , n){
            REP(i , now , n) if(v[now][now] == 0 && v[i
                ][now] != 0)
                swap(v[i] , v[now]); // det = -det;
            if(v[now][now] == 0) return ans;
            int inv = ppow(v[now][now] , GAUSS_MOD - 2)
                ;
            REP(i , 0 , n) if(i != now){
                int tmp = v[i][now] * inv % GAUSS_MOD;
                REP(j , now , n + 1) (v[i][j] +=
                    GAUSS_MOD - tmp * v[now][j] %
                    GAUSS_MOD) %= GAUSS_MOD;
            }
        }
        REP(i , 0 , n) ans[i] = v[i][n + 1] * ppow(v[i
            ][i] , GAUSS_MOD - 2) % GAUSS_MOD;
        return ans;
    }
    // gs.v.clear() , gs.v.resize(n , vector<int>(n + 1
         , 0));
} gs;
```

2.12 ax+by=gcd

```
PII gcd(int a, int b){
  if(b == 0) return {1, 0};
  PII q = gcd(b, a % b);
  return {q.second, q.first - q.second * (a / b)};
}
```

2.13 Discrete sqrt

```
void calcH(LL &t, LL &h, const LL p) {
  LL tmp=p-1; for(t=0;(tmp&1)==0;tmp/=2) t++; h=tmp;
}
// solve equation x^2 mod p = a
bool solve(LL a, LL p, LL &x, LL &y) {
  if(p == 2) { x = y = 1; return true; }
  int p2 = p / 2, tmp = mypow(a, p2, p);
  if (tmp == p - 1) return false;
  if ((p + 1) % 4 == 0) {
    x=mypow(a,(p+1)/4,p); y=p-x; return true;
  } else {
    LL t, h, b, pb; calcH(t, h, p);
    if (t >= 2) {
      do {b = rand() % (p - 2) + 2;
      } while (mypow(b, p / 2, p) != p - 1);
      pb = mypow(b, h, p);
    } int s = mypow(a, h / 2, p);
    for (int step = 2; step <= t; step++) {
      int ss = (((LL)(s * s) % p) * a) % p;
      for(int i=0;i<t-step;i++) ss=mul(ss,ss,p);
      if (ss + 1 == p) s = (s * pb) % p;
      pb = ((LL)pb * pb) % p;
    } x = ((LL)s * a) % p; y = p - x;
  } return true;
}
```

2.14 Romberg 定積分

```
// Estimates the definite integral of
// \int_a^b f(x) dx
template<class T>
double romberg( T& f, double a, double b, double eps=1e
    -8){
  vector<double>t; double h=b-a,last,curr; int k=1,i=1;
  t.push_back(h*(f(a)+f(b))/2);
  do{ last=t.back(); curr=0; double x=a+h/2;
    for(int j=0;j<k;j++) curr+=f(x), x+=h;
    curr=(t[0] + h*curr)/2; double k1=4.0/3.0,k2
        =1.0/3.0;
    for(int j=0;j<i;j++){ double temp=k1*curr-k2*t[j];
      t[j]=curr; curr=temp; k2/=4*k1-k2; k1=k2+1;
    } t.push_back(curr); k*=2; h/=2; i++;
  }while( fabs(last-curr) > eps);
  return t.back();
}
```

2.15 Roots of Polynomial 找多項式的根

```
const double eps = 1e-12;
const double inf = 1e+12;
double a[ 10 ], x[ 10 ]; // a[0..n](coef) must be
    filled
int n; // degree of polynomial must be filled
int sign( double x ){return (x < -eps)?(-1):(x>eps);}
double f(double a[], int n, double x){
  double tmp=1,sum=0;
  for(int i=0;i<=n;i++)
  { sum=sum+a[i]*tmp; tmp=tmp*x; }
  return sum;
}
double binary(double l,double r,double a[],int n){
  int sl=sign(f(a,n,l)),sr=sign(f(a,n,r));
  if(sl==0) return l; if(sr==0) return r;
  if(sl*sr>0) return inf;
  while(r-l>eps){
    double mid=(l+r)/2;
    int ss=sign(f(a,n,mid));
    if(ss==0) return mid;
    if(ss*sl>0) l=mid; else r=mid;
  }
  return l;
}
void solve(int n,double a[],double x[],int &nx){
  if(n==1){ x[1]=-a[0]/a[1]; nx=1; return; }
  double da[10], dx[10]; int ndx;
  for(int i=n;i>=1;i--) da[i-1]=a[i]*i;
  solve(n-1,da,dx,ndx);
  nx=0;
  if(ndx==0){
    double tmp=binary(-inf,inf,a,n);
    if (tmp<inf) x[++nx]=tmp;
    return;
  }
  double tmp;
  tmp=binary(-inf,dx[1],a,n);
  if(tmp<inf) x[++nx]=tmp;
  for(int i=1;i<=ndx-1;i++){
    tmp=binary(dx[i],dx[i+1],a,n);
    if(tmp<inf) x[++nx]=tmp;
  }
  tmp=binary(dx[ndx],inf,a,n);
  if(tmp<inf) x[++nx]=tmp;
} // roots are stored in x[1..nx]
```

2.16 Primes

```
/* 12721, 13331, 14341, 75577, 123457, 222557, 556679
* 999983, 1097774749, 1076767633, 100102021, 999997771
* 1001010013, 1000512343, 987654361, 999991231
* 999888733, 98789101, 987777733, 999991921, 1010101333
* 1010102101, 1000000000039, 10000000000000037
* 2305843009213693951, 4611686018427387847
* 9223372036854775783, 18446744073709551557 */
int mu[ N ] , p_tbl[ N ];
vector<int> primes;
void sieve() {
  mu[ 1 ] = p_tbl[ 1 ] = 1;
  for( int i = 2 ; i < N ; i ++ ){
    if( !p_tbl[ i ] ){
      p_tbl[ i ] = i;
      primes.push_back( i );
      mu[ i ] = -1;
    }
    for( int p : primes ){
      int x = i * p;
      if( x >= M ) break;
      p_tbl[ x ] = p;
      mu[ x ] = -mu[ i ];
      if( i % p == 0 ){
        mu[ x ] = 0;
        break;
} } } }
vector<int> factor( int x ){
  vector<int> fac{ 1 };
  while( x > 1 ){
    int fn = SZ(fac), p = p_tbl[ x ], pos = 0;
    while( x % p == 0 ){
      x /= p;
      for( int i = 0 ; i < fn ; i ++ )
        fac.PB( fac[ pos ++ ] * p );
  } }
  return fac;
}
```

2.17 Phi $\phi(n)$

```
ll phi(ll n){   // 計算小於n的數中與n互質的有幾個
    ll res = n, a=n;     // O(sqrtN)
    for(ll i=2;i*i<=a;i++){
        if(a%i==0){
            res = res/i*(i-1);
            while(a%i==0) a/=i;
        }
    }
    if(a>1) res = res/a*(a-1);
    return res;
}
```

2.18 Result

- Lucas’ Theorem :
  For $n, m \in \mathbb{Z}^*$ and prime P, $C(m,n)$ mod P $= \Pi(C(m_i, n_i))$ where m_i is the i-th digit of m in base P.

- Stirling approximation :
  $n! \approx \sqrt{2\pi n}(\frac{n}{e})^n e^{\frac{1}{12n}}$

- Stirling Numbers(permutation $|P| = n$ with k cycles):
  $S(n,k) =$ coefficient of x^k in $\Pi_{i=0}^{n-1}(x+i)$

- Stirling Numbers(Partition n elements into k non-empty set):
  $S(n,k) = \frac{1}{k!}\sum\limits_{j=0}^{k}(-1)^{k-j}\binom{k}{j}j^n$

- Pick’ s Theorem : $A = i + b/2 - 1$
  其面積 A 和內部格點數目 i、邊上格點數目 b 的關係

- Catalan number : $C_n = \binom{2n}{n}/(n+1)$
  $C_n^{n+m} - C_{n+1}^{n+m} = (m+n)!\frac{n-m+1}{n+1} \quad for \quad n \geq m$
  $C_n = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{(n+1)!n!}$
  $C_0 = 1 \quad and \quad C_{n+1} = 2(\frac{2n+1}{n+2})C_n$
  $C_0 = 1 \quad and \quad C_{n+1} = \sum_{i=0}^{n} C_i C_{n-i} \quad for \quad n \geq 0$

- Euler Characteristic:
  planar graph: $V - E + F - C = 1$
  convex polyhedron: $V - E + F = 2$
  V, E, F, C: number of vertices, edges, faces(regions), and components

- Kirchhoff’s theorem :
  $A_{ii} = deg(i), A_{ij} = (i,j) \in E\ ? -1 : 0$, Deleting any one row, one column, and cal the det(A)

- Polya’ theorem (c 為方法數，m 為總數):
  $(\sum_{i=1}^{m} c^{gcd(i,m)})/m$

- 錯排公式: (n 個人中，每個人皆不再原來位置的組合數):
  $dp[0] = 1; dp[1] = 0;$
  $dp[i] = (i-1) * (dp[i-1] + dp[i-2]);$

- Bell 數 (有 n 個人, 把他們拆組的方法總數) :
  $B_0 = 1$
  $B_n = \sum_{k=0}^{n} s(n,k) \quad (second - stirling)$
  $B_{n+1} = \sum_{k=0}^{n}\binom{n}{k}B_k$

- Wilson’s theorem :
  $(p-1)! \equiv -1(mod\ \ p)$

- Fermat’s little theorem :
  $a^p \equiv a(mod\ \ p)$

- Euler’s totient function:
  $A^{B^C} mod\ \ p = pow(A, pow(B, C, p-1))mod\ \ p$

- 歐拉函數降冪公式:
  A^B mod $C = A^{B \text{ mod } \phi(c) + \phi(c)}$ mod C

3 Geometry

3.1 definition

```
typedef long double ld;
const ld eps = 1e-8;
int dcmp(ld x) {
  if(abs(x) < eps) return 0;
  else return x < 0 ? -1 : 1;
}
struct Pt {
  ld x, y;
  Pt(ld _x=0, ld _y=0):x(_x), y(_y) {}

  Pt operator+(const Pt &a) const {
    return Pt(x+a.x, y+a.y);
  }
  Pt operator-(const Pt &a) const {
    return Pt(x-a.x, y-a.y);
  }
  Pt operator*(const ld &a) const {
    return Pt(x*a, y*a);
  }
  Pt operator/(const ld &a) const {
    return Pt(x/a, y/a);
  }
  ld operator*(const Pt &a) const {
    return x*a.x + y*a.y;
  }
  ld operator^(const Pt &a) const {
```

```cpp
    return x*a.y - y*a.x;
  }
  bool operator<(const Pt &a) const {
    return x < a.x || (x == a.x && y < a.y);
    //return dcmp(x-a.x) < 0 || (dcmp(x-a.x) == 0 &&
        dcmp(y-a.y) < 0);
  }
  bool operator==(const Pt &a) const {
    return dcmp(x-a.x) == 0 && dcmp(y-a.y) == 0;
  }
};
ld norm2(const Pt &a) {
  return a*a;
}
ld norm(const Pt &a) {
  return sqrt(norm2(a));
}
Pt perp(const Pt &a) {
  return Pt(-a.y, a.x);
}
Pt rotate(const Pt &a, ld ang) {
  return Pt(a.x*cos(ang)-a.y*sin(ang), a.x*sin(ang)+a.y
      *cos(ang));
}
struct Line {
  Pt s, e, v; // start, end, end-start
  ld ang;
  Line(Pt _s=Pt(0, 0), Pt _e=Pt(0, 0)):s(_s), e(_e) { v
       = e-s; ang = atan2(v.y, v.x); }

  bool operator<(const Line &L) const {
    return ang < L.ang;
  }
};
struct Circle {
  Pt o; ld r;
  Circle(Pt _o=Pt(0, 0), ld _r=0):o(_o), r(_r) {}
};
```

3.2 Intersection of 2 lines 兩線關係

```cpp
Pt LLIntersect(Line a, Line b) {
  Pt p1 = a.s, p2 = a.e, q1 = b.s, q2 = b.e;
  ld f1 = (p2-p1)^(q1-p1),f2 = (p2-p1)^(p1-q2),f;
  if(dcmp(f=f1+f2) == 0)
    return dcmp(f1)?Pt(NAN,NAN):Pt(INFINITY,INFINITY);
  return q1*(f2/f) + q2*(f1/f);
}
```

3.3 halfPlaneIntersection 半平面交

```cpp
// for point or line solution, change > to >=
bool onleft(Line L, Pt p) {
  return dcmp(L.v^(p-L.s)) > 0;
}
// assume that Lines intersect
vector<Pt> HPI(vector<Line>& L) {
  sort(L.begin(), L.end()); // sort by angle
  int n = L.size(), fir, las;
  Pt *p = new Pt[n];
  Line *q = new Line[n];
  q[fir=las=0] = L[0];
  for(int i = 1 ; i < n ; i++) {
    while(fir < las && !onleft(L[i], p[las-1])) las--;
    while(fir < las && !onleft(L[i], p[fir])) fir++;
    q[++las] = L[i];
    if(dcmp(q[las].v^q[las-1].v) == 0) {
      las--;
      if(onleft(q[las], L[i].s)) q[las] = L[i];
    }
    if(fir < las) p[las-1] = LLIntersect(q[las-1], q[
        las]);
  }
  while(fir < las && !onleft(q[fir], p[las-1])) las--;
  if(las-fir <= 1) return {};
  p[las] = LLIntersect(q[las], q[fir]);
  int m = 0;
  vector<Pt> ans(las-fir+1);
  for(int i = fir ; i <= las ; i++) ans[m++] = p[i];
  return ans;
}
```

3.4 Convex Hull 凸包

```cpp
double cross(Pt o, Pt a, Pt b){
  return (a-o) ^ (b-o);
}
vector<Pt> convex_hull(vector<Pt> pt){
  sort(pt.begin(),pt.end());
  int top=0;
  vector<Pt> stk(2*pt.size());
  for (int i=0; i<(int)pt.size(); i++){
    while (top >= 2 && cross(stk[top-2],stk[top-1],pt[i
        ]) <= 0)
      top--;
    stk[top++] = pt[i];
  }
  for (int i=pt.size()-2, t=top+1; i>=0; i--){
    while (top >= t && cross(stk[top-2],stk[top-1],pt[i
        ]) <= 0)
      top--;
    stk[top++] = pt[i];
  }
  stk.resize(top-1);
  return stk;
}
```

3.5 Convex Hull 3D

```cpp
struct Pt{
  Pt cross(const Pt &p) const
  { return Pt(y * p.z - z * p.y, z * p.x - x * p.z, x *
       p.y - y * p.x); }
} info[N];
int mark[N][N],n, cnt;;
double mix(const Pt &a, const Pt &b, const Pt &c)
{ return a * (b ^ c); }
double area(int a, int b, int c)
{ return norm((info[b] - info[a]) ^ (info[c] - info[a])
    ); }
double volume(int a, int b, int c, int d)
{ return mix(info[b] - info[a], info[c] - info[a], info
    [d] - info[a]); }
struct Face{
  int a, b, c; Face(){}
  Face(int a, int b, int c): a(a), b(b), c(c) {}
  int &operator [](int k)
  { if (k == 0) return a; if (k == 1) return b; return
      c; }
};
vector<Face> face;
void insert(int a, int b, int c)
{ face.push_back(Face(a, b, c)); }
void add(int v) {
  vector <Face> tmp; int a, b, c; cnt++;
  for (int i = 0; i < SIZE(face); i++) {
    a = face[i][0]; b = face[i][1]; c = face[i][2];
    if(Sign(volume(v, a, b, c)) < 0)
    mark[a][b] = mark[b][a] = mark[b][c] = mark[c][b] =
         mark[c][a] = mark[a][c] = cnt;
    else tmp.push_back(face[i]);
  } face = tmp;
  for (int i = 0; i < SIZE(tmp); i++) {
    a = face[i][0]; b = face[i][1]; c = face[i][2];
    if (mark[a][b] == cnt) insert(b, a, v);
    if (mark[b][c] == cnt) insert(c, b, v);
    if (mark[c][a] == cnt) insert(a, c, v);
}}
int Find(){
  for (int i = 2; i < n; i++) {
    Pt ndir = (info[0] - info[i]) ^ (info[1] - info[i])
        ;
    if (ndir == Pt()) continue; swap(info[i], info[2]);
    for (int j = i + 1; j < n; j++) if (Sign(volume(0,
        1, 2, j)) != 0) {
      swap(info[j], info[3]); insert(0, 1, 2); insert
          (0, 2, 1); return 1;
} } return 0; }
int main() {
  for (; scanf("%d", &n) == 1; ) {
    for (int i = 0; i < n; i++) info[i].Input();
    sort(info, info + n); n = unique(info, info + n) -
        info;
    face.clear(); random_shuffle(info, info + n);
    if (Find()) { memset(mark, 0, sizeof(mark)); cnt =
        0;
      for (int i = 3; i < n; i++) add(i); vector<Pt>
          Ndir;
      for (int i = 0; i < SIZE(face); ++i) {
        Pt p = (info[face[i][0]] - info[face[i][1]]) ^
               (info[face[i][2]] - info[face[i][1]]);
        p = p / norm( p ); Ndir.push_back(p);
      } sort(Ndir.begin(), Ndir.end());
      int ans = unique(Ndir.begin(), Ndir.end()) - Ndir
          .begin();
      printf("%d\n", ans);
    } else printf("1\n");
} }
double calcDist(const Pt &p, int a, int b, int c)
{ return fabs(mix(info[a] - p, info[b] - p, info[c] - p
    ) / area(a, b, c)); }
//compute the minimal distance of center of any faces
double findDist() { //compute center of mass
  double totalWeight = 0; Pt center(.0, .0, .0);
  Pt first = info[face[0][0]];
  for (int i = 0; i < SIZE(face); ++i) {
    Pt p = (info[face[i][0]]+info[face[i][1]]+info[face
        [i][2]]+first)*.25;
    double weight = mix(info[face[i][0]] - first, info[
        face[i][1]]
        - first, info[face[i][2]] - first);
    totalWeight += weight; center = center + p * weight
        ;
  } center = center / totalWeight;
  double res = 1e100; //compute distance
  for (int i = 0; i < SIZE(face); ++i)
    res = min(res, calcDist(center, face[i][0], face[i
        ][1], face[i][2]));
    return res; }
```

3.6 Intersection of 2 segments 線段交

```
int ori( const Pt& o , const Pt& a , const Pt& b ){
  LL ret = ( a - o ) ^ ( b - o );
  return (ret > 0) - (ret < 0);
}
// p1 == p2 || q1 == q2 need to be handled
bool banana( const Pt& p1 , const Pt& p2 ,
             const Pt& q1 , const Pt& q2 ){
  if( ( ( p2 - p1 ) ^ ( q2 - q1 ) ) == 0 ){ // parallel
    if( ori( p1 , p2 , q1 ) ) return false;
    return ( ( p1 - q1 ) * ( p2 - q1 ) ) <= 0 ||
           ( ( p1 - q2 ) * ( p2 - q2 ) ) <= 0 ||
           ( ( q1 - p1 ) * ( q2 - p1 ) ) <= 0 ||
           ( ( q1 - p2 ) * ( q2 - p2 ) ) <= 0;
  }
  return (ori( p1, p2, q1 ) * ori( p1, p2, q2 )<=0) &&
         (ori( q1, q2, p1 ) * ori( q1, q2, p2 )<=0);
}
```

3.7 Tangent line of two circles 兩圓共同切線

```
vector<Line> go( const Cir& c1 , const Cir& c2 , int
    sign1 ){
  // sign1 = 1 for outer tang, -1 for inter tang
  vector<Line> ret;
  double d_sq = norm2( c1.O - c2.O );
  if( d_sq < eps ) return ret;
  double d = sqrt( d_sq );
  Pt v = ( c2.O - c1.O ) / d;
  double c = ( c1.R - sign1 * c2.R ) / d;
  if( c * c > 1 ) return ret;
  double h = sqrt( max( 0.0 , 1.0 - c * c ) );
  for( int sign2 = 1 ; sign2 >= -1 ; sign2 -= 2 ){
    Pt n = { v.X * c - sign2 * h * v.Y ,
             v.Y * c + sign2 * h * v.X };
    Pt p1 = c1.O + n * c1.R;
    Pt p2 = c2.O + n * ( c2.R * sign1 );
    if( fabs( p1.X - p2.X ) < eps and
        fabs( p1.Y - p2.Y ) < eps )
      p2 = p1 + perp( c2.O - c1.O );
    ret.push_back( { p1 , p2 } );
  }
  return ret;
}
```

3.8 Heart of Triangle

```
Pt inCenter( Pt &A,  Pt &B,  Pt &C) { // ☒心
  double a = norm(B-C), b = norm(C-A), c = norm(A-B);
  return (A * a + B * b + C * c) / (a + b + c);
}
Pt circumCenter( Pt &a,  Pt &b,  Pt &c) { // 外心
  Pt bb = b - a, cc = c - a;
  double db=norm2(bb), dc=norm2(cc), d=2*(bb ^ cc);
  return a-Pt(bb.Y*dc-cc.Y*db, cc.X*db-bb.X*dc) / d;
}
Pt othroCenter( Pt &a,  Pt &b,  Pt &c) { // 垂心
  Pt ba = b - a, ca = c - a, bc = b - c;
  double Y = ba.Y * ca.Y * bc.Y,
    A = ca.X * ba.Y - ba.X * ca.Y,
    x0= (Y+ca.X*ba.Y*b.X-ba.X*ca.Y*c.X) / A,
    y0= -ba.X * (x0 - c.X) / ba.Y + ca.Y;
  return Pt(x0, y0);
}
```

4 Graph

4.1 MaximumClique 最大團

```
#define N 111
struct MaxClique{ // 0-base
  typedef bitset<N> Int;
  Int linkto[N] , v[N];
  int n;
  void init(int _n){
    n = _n;
    for(int i = 0 ; i < n ; i ++){
      linkto[i].reset(); v[i].reset();
  } }
  void addEdge(int a , int b)
  { v[a][b] = v[b][a] = 1; }
  int popcount(const Int& val)
  { return val.count(); }
  int lowbit(const Int& val)
  { return val._Find_first(); }
  int ans , stk[N];
  int id[N] , di[N] , deg[N];
  Int cans;
  void maxclique(int elem_num, Int candi){
    if(elem_num > ans){
      ans = elem_num; cans.reset();
      for(int i = 0 ; i < elem_num ; i ++)
        cans[id[stk[i]]] = 1;
    }
    int potential = elem_num + popcount(candi);
    if(potential <= ans) return;
    int pivot = lowbit(candi);
    Int smaller_candi = candi & (~linkto[pivot]);
    while(smaller_candi.count() && potential > ans){
      int next = lowbit(smaller_candi);
      candi[next] = !candi[next];
      smaller_candi[next] = !smaller_candi[next];
      potential --;
      if(next == pivot || (smaller_candi & linkto[next
          ]).count()){
        stk[elem_num] = next;
        maxclique(elem_num + 1, candi & linkto[next]);
  } } }
  int solve(){
    for(int i = 0 ; i < n ; i ++){
      id[i] = i; deg[i] = v[i].count();
    }
    sort(id , id + n , [&](int id1, int id2){
          return deg[id1] > deg[id2]; });
    for(int i = 0 ; i < n ; i ++) di[id[i]] = i;
    for(int i = 0 ; i < n ; i ++)
      for(int j = 0 ; j < n ; j ++)
        if(v[i][j]) linkto[di[i]][di[j]] = 1;
    Int cand; cand.reset();
    for(int i = 0 ; i < n ; i ++) cand[i] = 1;
    ans = 1;
    cans.reset(); cans[0] = 1;
    maxclique(0, cand);
    return ans;
} }solver;
```

4.2 MaximalClique 極大團

```
#define N 80
struct MaxClique{ // 0-base
  typedef bitset<N> Int;
  Int lnk[N] , v[N];
  int n;
  void init(int _n){
    n = _n;
    for(int i = 0 ; i < n ; i ++){
      lnk[i].reset(); v[i].reset();
  } }
  void addEdge(int a , int b)
  { v[a][b] = v[b][a] = 1; }
  int ans , stk[N], id[N] , di[N] , deg[N];
  Int cans;
  void dfs(int elem_num, Int candi, Int ex){
    if(candi.none()&&ex.none()){
      cans.reset();
      for(int i = 0 ; i < elem_num ; i ++)
        cans[id[stk[i]]] = 1;
      ans = elem_num; // cans is a maximal clique
      return;
    }
    int pivot = (candi|ex)._Find_first();
    Int smaller_candi = candi & (~lnk[pivot]);
    while(smaller_candi.count()){
      int nxt = smaller_candi._Find_first();
      candi[nxt] = smaller_candi[nxt] = 0;
      ex[nxt] = 1;
      stk[elem_num] = nxt;
      dfs(elem_num+1,candi&lnk[nxt],ex&lnk[nxt]);
  } }
  int solve(){
    for(int i = 0 ; i < n ; i ++){
      id[i] = i; deg[i] = v[i].count();
    }
    sort(id , id + n , [&](int id1, int id2){
          return deg[id1] > deg[id2]; });
    for(int i = 0 ; i < n ; i ++) di[id[i]] = i;
    for(int i = 0 ; i < n ; i ++)
      for(int j = 0 ; j < n ; j ++)
        if(v[i][j]) lnk[di[i]][di[j]] = 1;
    ans = 1; cans.reset(); cans[0] = 1;
    dfs(0, Int(string(n,'1')), 0);
    return ans;
} }solver;
```

4.3 Strongly Connected Component 強連通分量

```
struct Scc{
  int n, nScc, vst[MXN], bln[MXN];
  vector<int> E[MXN], rE[MXN], vec;
  void init(int _n){
    n = _n;
    for (int i=0; i<MXN; i++)
      E[i].clear(), rE[i].clear();
  }
  void addEdge(int u, int v){
    E[u].PB(v); rE[v].PB(u);
  }
  void DFS(int u){
    vst[u]=1;
    for (auto v : E[u]) if (!vst[v]) DFS(v);
    vec.PB(u);
  }
  void rDFS(int u){
    vst[u] = 1; bln[u] = nScc;
    for (auto v : rE[u]) if (!vst[v]) rDFS(v);
  }
  void solve(){
    nScc = 0;
    vec.clear();
    FZ(vst);
    for (int i=0; i<n; i++)
      if (!vst[i]) DFS(i);
    reverse(vec.begin(),vec.end());
    FZ(vst);
    for (auto v : vec)
      if (!vst[v]){
        rDFS(v); nScc++;
      }
  }
};
```

4.4 Min Mean Cycle 最小環平均

```
/* minimum mean cycle O(VE) */
struct MMC{
#define E 101010
#define V 1021
#define inf 1e9
#define eps 1e-6
  struct Edge { int v,u; double c; };
  int n, m, prv[V][V], prve[V][V], vst[V];
  Edge e[E];
  vector<int> edgeID, cycle, rho;
  double d[V][V];
  void init( int _n )
  { n = _n; m = 0; }
  // WARNING: TYPE matters
  void addEdge( int vi , int ui , double ci )
  { e[ m ++ ] = { vi , ui , ci }; }
  void bellman_ford() {
    for(int i=0; i<n; i++) d[0][i]=0;
    for(int i=0; i<n; i++) {
      fill(d[i+1], d[i+1]+n, inf);
      for(int j=0; j<m; j++) {
        int v = e[j].v, u = e[j].u;
        if(d[i][v]<inf && d[i+1][u]>d[i][v]+e[j].c) {
          d[i+1][u] = d[i][v]+e[j].c;
          prv[i+1][u] = v;
          prve[i+1][u] = j;
  } } } }
  double solve(){
    // returns inf if no cycle, mmc otherwise
    double mmc=inf;
    int st = -1;
    bellman_ford();
    for(int i=0; i<n; i++) {
      double avg=-inf;
      for(int k=0; k<n; k++) {
        if(d[n][i]<inf-eps) avg=max(avg,(d[n][i]-d[k][i
            ])/(n-k));
        else avg=max(avg,inf);
      }
      if (avg < mmc) tie(mmc, st) = tie(avg, i);
    }
    fill(vst,0); edgeID.clear(); cycle.clear(); rho.
        clear();
    for (int i=n; !vst[st]; st=prv[i--][st]) {
      vst[st]++;
      edgeID.PB(prve[i][st]);
      rho.PB(st);
    }
    while (vst[st] != 2) {
      if(rho.empty()) return inf;
      int v = rho.back(); rho.pop_back();
      cycle.PB(v);
      vst[v]++;
    }
    reverse(ALL(edgeID));
    edgeID.resize(SZ(cycle));
    return mmc;
} }mmc;
```

4.5 Directed Graph Min Cost Cycle 最小環

```
// works in O(N M)
#define INF 100000000000000LL
#define N 5010
#define M 200010
struct edge{
  int to; LL w;
  edge(int a=0, LL b=0): to(a), w(b){}
};
struct node{
  LL d; int u, next;
  node(LL a=0, int b=0, int c=0): d(a), u(b), next(c){}
}b[M];
struct DirectedGraphMinCycle{
  vector<edge> g[N], grev[N];
  LL dp[N][N], p[N], d[N], mu;
bool inq[N];
int n, bn, bsz, hd[N];
void b_insert(LL d, int u){
  int i = d/mu;
  if(i >= bn) return;
  b[++bsz] = node(d, u, hd[i]);
  hd[i] = bsz;
}
void init( int _n ){
  n = _n;
  for( int i = 1 ; i <= n ; i ++ )
    g[ i ].clear();
}
void addEdge( int ai , int bi , LL ci )
{ g[ai].push_back(edge(bi,ci)); }
LL solve(){
  fill(dp[0], dp[0]+n+1, 0);
  for(int i=1; i<=n; i++){
    fill(dp[i]+1, dp[i]+n+1, INF);
    for(int j=1; j<=n; j++) if(dp[i-1][j] < INF){
      for(int k=0; k<(int)g[j].size(); k++)
        dp[i][g[j][k].to] =min(dp[i][g[j][k].to],
                               dp[i-1][j]+g[j][k].w);
  } }
  mu=INF; LL bunbo=1;
  for(int i=1; i<=n; i++) if(dp[n][i] < INF){
    LL a=-INF, b=1;
    for(int j=0; j<=n-1; j++) if(dp[j][i] < INF){
      if(a*(n-j) < b*(dp[n][i]-dp[j][i])){
        a = dp[n][i]-dp[j][i];
        b = n-j;
    } }
    if(mu*b > bunbo*a)
      mu = a, bunbo = b;
  }
  if(mu < 0) return -1; // negative cycle
  if(mu == INF) return INF; // no cycle
  if(mu == 0) return 0;
  for(int i=1; i<=n; i++)
    for(int j=0; j<(int)g[i].size(); j++)
    g[i][j].w *= bunbo;
  memset(p, 0, sizeof(p));
  queue<int> q;
  for(int i=1; i<=n; i++){
    q.push(i);
    inq[i] = true;
  }
  while(!q.empty()){
    int i=q.front(); q.pop(); inq[i]=false;
    for(int j=0; j<(int)g[i].size(); j++){
      if(p[g[i][j].to] > p[i]+g[i][j].w-mu){
        p[g[i][j].to] = p[i]+g[i][j].w-mu;
        if(!inq[g[i][j].to]){
          q.push(g[i][j].to);
          inq[g[i][j].to] = true;
  } } } }
  for(int i=1; i<=n; i++) grev[i].clear();
  for(int i=1; i<=n; i++)
    for(int j=0; j<(int)g[i].size(); j++){
      g[i][j].w += p[i]-p[g[i][j].to];
      grev[g[i][j].to].push_back(edge(i, g[i][j].w));
    }
  LL mldc = n*mu;
  for(int i=1; i<=n; i++){
    bn=mldc/mu, bsz=0;
    memset(hd, 0, sizeof(hd));
    fill(d+i+1, d+n+1, INF);
    b_insert(d[i]=0, i);
    for(int j=0; j<=bn-1; j++) for(int k=hd[j]; k; k=
        b[k].next){
      int u = b[k].u;
      LL du = b[k].d;
      if(du > d[u]) continue;
      for(int l=0; l<(int)g[u].size(); l++) if(g[u][l
          ].to > i){
        if(d[g[u][l].to] > du + g[u][l].w){
          d[g[u][l].to] = du + g[u][l].w;
          b_insert(d[g[u][l].to], g[u][l].to);
    } } }
    for(int j=0; j<(int)grev[i].size(); j++) if(grev[
        i][j].to > i)
      mldc=min(mldc,d[grev[i][j].to] + grev[i][j].w);
    }
    return mldc / bunbo;
} }graph;
```

4.6 K-th Shortest Path 第 K 短路徑

```
// time: O(|E| \lg |E| + |V| \lg |V| + K)
// memory: O(|E| \lg |E| + |V|)
struct KSP{ // 1-base
  struct nd{
    int u, v; ll d;
    nd(int ui = 0, int vi = 0, ll di = INF)
    { u = ui; v = vi; d = di; }
  };
  struct heap{
    nd* edge; int dep; heap* chd[4];
  };
  static int cmp(heap* a,heap* b)
  { return a->edge->d > b->edge->d; }
  struct node{
    int v; ll d; heap* H; nd* E;
    node(){}
    node(ll _d, int _v, nd* _E)
    { d =_d; v = _v; E = _E; }
    node(heap* _H, ll _d)
    { H = _H; d = _d; }
    friend bool operator<(node a, node b)
    { return a.d > b.d; }
  };
  int n, k, s, t;
  ll dst[ N ];
  nd *nxt[ N ];
  vector<nd*> g[ N ], rg[ N ];
  heap *nullNd, *head[ N ];
  void init( int _n , int _k , int _s , int _t ){
    n = _n; k = _k; s = _s; t = _t;
    for( int i = 1 ; i <= n ; i ++ ){
      g[ i ].clear(); rg[ i ].clear();
      nxt[ i ] = NULL; head[ i ] = NULL;
      dst[ i ] = -1;
  } }
  void addEdge( int ui , int vi , ll di ){
    nd* e = new nd(ui, vi, di);
    g[ ui ].push_back( e );
    rg[ vi ].push_back( e );
  }
  queue<int> dfsQ;
  void dijkstra(){
    while(dfsQ.size()) dfsQ.pop();
    priority_queue<node> Q;
    Q.push(node(0, t, NULL));
    while (!Q.empty()){
      node p = Q.top(); Q.pop();
      if(dst[p.v] != -1) continue;
      dst[ p.v ] = p.d;
      nxt[ p.v ] = p.E;
      dfsQ.push( p.v );
      for(auto e: rg[ p.v ])
        Q.push(node(p.d + e->d, e->u, e));
  } }
  heap* merge(heap* curNd, heap* newNd){
    if(curNd == nullNd) return newNd;
    heap* root = new heap;
    memcpy(root, curNd, sizeof(heap));
    if(newNd->edge->d < curNd->edge->d){
      root->edge = newNd->edge;
      root->chd[2] = newNd->chd[2];
      root->chd[3] = newNd->chd[3];
      newNd->edge = curNd->edge;
      newNd->chd[2] = curNd->chd[2];
      newNd->chd[3] = curNd->chd[3];
    }
    if(root->chd[0]->dep < root->chd[1]->dep)
      root->chd[0] = merge(root->chd[0],newNd);
    else
      root->chd[1] = merge(root->chd[1],newNd);
    root->dep = max(root->chd[0]->dep, root->chd[1]->
        dep) + 1;
    return root;
  }
  vector<heap*> V;
  void build(){
    nullNd = new heap;
    nullNd->dep = 0;
    nullNd->edge = new nd;
    fill(nullNd->chd, nullNd->chd+4, nullNd);
    while(not dfsQ.empty()){
      int u = dfsQ.front(); dfsQ.pop();
      if(!nxt[ u ]) head[ u ] = nullNd;
      else head[ u ] = head[nxt[ u ]->v];
      V.clear();
      for( auto&& e : g[ u ] ){
        int v = e->v;
        if( dst[ v ] == -1 ) continue;
        e->d += dst[ v ] - dst[ u ];
        if( nxt[ u ] != e ){
          heap* p = new heap;
          fill(p->chd, p->chd+4, nullNd);
          p->dep = 1;
          p->edge = e;
          V.push_back(p);
      } }
      if(V.empty()) continue;
      make_heap(V.begin(), V.end(), cmp);
#define L(X) ((X<<1)+1)
#define R(X) ((X<<1)+2)
      for( size_t i = 0 ; i < V.size() ; i ++ ){
        if(L(i) < V.size()) V[i]->chd[2] = V[L(i)];
        else V[i]->chd[2]=nullNd;
        if(R(i) < V.size()) V[i]->chd[3] = V[R(i)];
        else V[i]->chd[3]=nullNd;
      }
      head[u] = merge(head[u], V.front());
  } }
  vector<ll> ans;
  void first_K(){
    ans.clear();
    priority_queue<node> Q;
    if( dst[ s ] == -1 ) return;
    ans.push_back( dst[ s ] );
    if( head[s] != nullNd )
      Q.push(node(head[s], dst[s]+head[s]->edge->d));
    for( int _ = 1 ; _ < k and not Q.empty() ; _ ++ ){
      node p = Q.top(), q; Q.pop();
      ans.push_back( p.d );
      if(head[ p.H->edge->v ] != nullNd){
        q.H = head[ p.H->edge->v ];
        q.d = p.d + q.H->edge->d;
        Q.push(q);
      }
      for( int i = 0 ; i < 4 ; i ++ )
        if( p.H->chd[ i ] != nullNd ){
          q.H = p.H->chd[ i ];
          q.d = p.d - p.H->edge->d + p.H->chd[ i ]->
              edge->d;
          Q.push( q );
  } }   }
  void solve(){ // ans[i] stores the i-th shortest path
    dijkstra();
    build();
    first_K(); // ans.size() might less than k
} }solver;
```

4.7 SPFA

```
bool spfa(){
    deque<int> dq;
    dis[0]=0;
    dq.push_back(0);
    inq[0]=1;
    while(!dq.empty()){
        int u=dq.front();
        dq.pop_front();
        inq[u]=0;
        for(auto i:edge[u]){
            if(dis[i.first]>i.second+dis[u]){
                dis[i.first]=i.second+dis[u];
                len[i.first]=len[u]+1;
                if(len[i.first]>n)  return 1;
                if(inq[i.first])  continue;
                if(!dq.empty()&&dis[dq.front()]>dis[i.
                    first])
                    dq.push_front(i.first);
                else
                    dq.push_back(i.first);
                inq[i.first]=1;
    } } }
    return 0;
}
```

4.8 差分約束

約束條件 $V_j - V_i \leq W$ 建邊 $V_i - > V_j$ 權重為 W -> bellman-ford or spfa

4.9 eulerPath

```
#define FOR(i,a,b) for(int i=a;i<=b;i++)
int dfs_st[10000500],dfn=0;
int ans[10000500],cnt=0,num=0;
vector<int>G[1000050];
int cur[1000050];
int ind[1000050],out[1000050];
void dfs(int x){
    FOR(i,1,n)sort(G[i].begin(),G[i].end());
    dfs_st[++dfn]=x;
    memset(cur,-1,sizeof(cur));
    while(dfn>0){
        int u=dfs_st[dfn];
        int complete=1;
        for(int i=cur[u]+1;i<G[u].size();i++){
            int v=G[u][i];
            num++;
            dfs_st[++dfn]=v;
            cur[u]=i;
            complete=0;
            break;
        }
        if(complete)ans[++cnt]=u,dfn--;
    }
}
bool check(int &start){
    int l=0,r=0,mid=0;
    FOR(i,1,n){
        if(ind[i]==out[i]+1)l++;
        if(out[i]==ind[i]+1)r++,start=i;
        if(ind[i]==out[i])mid++;
    }
    if(l==1&&r==1&&mid==n-2)return true;
    l=1;
    FOR(i,1,n)if(ind[i]!=out[i])l=0;
    if(l){
        FOR(i,1,n)if(out[i]>0){
            start=i;
            break;
        }
        return true;
    }
    return false;
}
int main(){
    cin>>n>>m;
    FOR(i,1,m){
        int x,y;scanf("%d%d",&x,&y);
        G[x].push_back(y);
        ind[y]++,out[x]++;
    }
    int start=-1,ok=true;
    if(check(start)){
        dfs(start);
        if(num!=m){
            puts("What a shame!");
            return 0;
        }
        for(int i=cnt;i>=1;i--)
            printf("%d ",ans[i]);
        puts("");
    }
    else puts("What a shame!");
}
```

5 String

5.1 PalTree

```
// len[s]是對應的回文長度
// num[s]是有幾個回文後綴
// cnt[s]是這個回文子字串在整個字串中的出現次數
// fail[s]是他長度次長的回文後綴，aba的fail是a
const int MXN = 1000010;
struct PalT{
  int nxt[MXN][26],fail[MXN],len[MXN];
  int tot,lst,n,state[MXN],cnt[MXN],num[MXN];
  int diff[MXN],sfail[MXN],fac[MXN],dp[MXN];
  char s[MXN]={-1};
  int newNode(int l,int f){
    len[tot]=l,fail[tot]=f,cnt[tot]=num[tot]=0;
    memset(nxt[tot],0,sizeof(nxt[tot]));
    diff[tot]=(l>0?l-len[f]:0);
    sfail[tot]=(l>0&&diff[tot]==diff[f]?sfail[f]:f);
    return tot++;
  }
  int getfail(int x){
    while(s[n-len[x]-1]!=s[n]) x=fail[x];
    return x;
  }
  int getmin(int v){
    dp[v]=fac[n-len[sfail[v]]-diff[v]];
    if(diff[v]==diff[fail[v]])
        dp[v]=min(dp[v],dp[fail[v]]);
    return dp[v]+1;
  }
  int push(){
    int c=s[n]-'a',np=getfail(lst);
    if(!(lst=nxt[np][c])){
      lst=newNode(len[np]+2,nxt[getfail(fail[np])][c]);
      nxt[np][c]=lst; num[lst]=num[fail[lst]]+1;
    }
    fac[n]=n;
    for(int v=lst;len[v]>0;v=sfail[v])
        fac[n]=min(fac[n],getmin(v));
    return ++cnt[lst],lst;
  }
  void init(const char *_s){
    tot=lst=n=0;
    newNode(0,1),newNode(-1,1);
    for(;_s[n];) s[n+1]=_s[n],++n,state[n-1]=push();
    for(int i=tot-1;i>1;i--) cnt[fail[i]]+=cnt[i];
  }
}palt;
```

5.2 KMP

```
/*
len-failure[k]:
在k結尾的情況下，這個子字串可以由開頭
長度為(len-failure[k])的部分重複出現來表達

failure[k]:
failure[k]為次長相同前綴後綴
如果我們不只想求最多，而且以0-base做為考量
，那可能的長度由大到小會是
failuer[k]、failure[failuer[k]-1]
、failure[failure[failuer[k]-1]-1]..
直到有值為0為止
*/
int failure[MXN];
void KMP(string& t, string& p)
{
    if (p.size() > t.size()) return;
    for (int i=1, j=failure[0]=-1; i<p.size(); ++i)
    {
        while (j >= 0 && p[j+1] != p[i])
            j = failure[j];
        if (p[j+1] == p[i]) j++;
        failure[i] = j;
    }
    for (int i=0, j=-1; i<t.size(); ++i)
    {
        while (j >= 0 && p[j+1] != t[i])
            j = failure[j];
        if (p[j+1] == t[i]) j++;
        if (j == p.size()-1)
        {
            cout << i - p.size() + 1<<" ";
            j = failure[j];
}   }   }
```

5.3 SAIS

```
const int N = 300010;
struct SA{
#define REP(i,n) for ( int i=0; i<int(n); i++ )
#define REP1(i,a,b) for ( int i=(a); i<=int(b); i++ )
  bool _t[N*2];
  int _s[N*2], _sa[N*2], _c[N*2], x[N], _p[N], _q[N*2],
       hei[N], r[N];
  int operator [] (int i){ return _sa[i]; }
  void build(int *s, int n, int m){
    memcpy(_s, s, sizeof(int) * n);
    sais(_s, _sa, _p, _q, _t, _c, n, m);
    mkhei(n);
  }
  void mkhei(int n){
    REP(i,n) r[_sa[i]] = i;
    hei[0] = 0;
    REP(i,n) if(r[i]) {
      int ans = i>0 ? max(hei[r[i-1]] - 1, 0) : 0;
      while(_s[i+ans] == _s[_sa[r[i]-1]+ans]) ans++;
      hei[r[i]] = ans;
    }
  }
  void sais(int *s, int *sa, int *p, int *q, bool *t,
      int *c, int n, int z){
    bool uniq = t[n-1] = true, neq;
    int nn = 0, nmxz = -1, *nsa = sa + n, *ns = s + n,
        lst = -1;
#define MS0(x,n) memset((x),0,n*sizeof(*(x)))
#define MAGIC(XD) MS0(sa, n); \
    memcpy(x, c, sizeof(int) * z); \
    XD; \
    memcpy(x + 1, c, sizeof(int) * (z - 1)); \
    REP(i,n) if(sa[i] && !t[sa[i]-1]) sa[x[s[sa[i
        ]-1]]++] = sa[i]-1; \
    memcpy(x, c, sizeof(int) * z); \
    for(int i = n - 1; i >= 0; i--) if(sa[i] && t[sa[i
        ]-1]) sa[--x[s[sa[i]-1]]] = sa[i]-1;
    MS0(c, z);
    REP(i,n) uniq &= ++c[s[i]] < 2;
    REP(i,z-1) c[i+1] += c[i];
    if (uniq) { REP(i,n) sa[--c[s[i]]] = i; return; }
    for(int i = n - 2; i >= 0; i--) t[i] = (s[i]==s[i
        +1] ? t[i+1] : s[i]<s[i+1]);
    MAGIC(REP1(i,1,n-1) if(t[i] && !t[i-1]) sa[--x[s[i
        ]]]=p[q[i]=nn++]=i);
    REP(i, n) if (sa[i] && t[sa[i]] && !t[sa[i]-1]) {
      neq=lst<0||memcmp(s+sa[i],s+lst,(p[q[sa[i]]+1]-sa
          [i])*sizeof(int));
      ns[q[lst=sa[i]]]=nmxz+=neq;
    }
    sais(ns, nsa, p + nn, q + n, t + n, c + z, nn, nmxz
         + 1);
    MAGIC(for(int i = nn - 1; i >= 0; i--) sa[--x[s[p[
        nsa[i]]]]] = p[nsa[i]]);
  }
}sa;
int H[ N ], SA[ N ];
void suffix_array(int* ip, int len) {
  // should padding a zero in the back
  // ip is int array, len is array length
  // ip[0..n-1] != 0, and ip[len] = 0
  ip[len++] = 0;
  sa.build(ip, len, 128);
  for (int i=0; i<len; i++) {
    H[i] = sa.hei[i + 1];
    SA[i] = sa._sa[i + 1];
  }
  // resulting height, sa array \in [0,len)
}
```

5.4 SuffixAutomata

```
// any path start from root forms a substring of S
// occurrence of P : iff SAM can run on input word P
// number of different substring : ds[1]-1
// total length of all different substring : dsl[1]
// max/min length of state i : mx[i]/mx[mom[i]]+1
// assume a run on input word P end at state i:
// number of occurrences of P : cnt[i]
// first occurrence position of P : fp[i]-|P|+1
// all position of P : fp of "dfs from i through rmom"
const int MXM = 1000010;
```

```
struct SAM{
  int tot, root, lst, mom[MXM], mx[MXM]; //ind[MXM]
  int nxt[MXM][33]; //cnt[MXM],ds[MXM],dsl[MXM],fp[MXM]
  // bool v[MXM]
  int newNode(){
    int res = ++tot;
    fill(nxt[res], nxt[res]+33, 0);
    mom[res] = mx[res] = 0; //cnt=ds=dsl=fp=v=0
    return res;
  }
  void init(){
    tot = 0;
    root = newNode();
    lst = root;
  }
  void push(int c){
    int p = lst;
    int np = newNode(); //cnt[np]=1
    mx[np] = mx[p]+1; //fp[np]=mx[np]-1
    for(; p && nxt[p][c] == 0; p = mom[p])
      nxt[p][c] = np;
    if(p == 0) mom[np] = root;
    else{
      int q = nxt[p][c];
      if(mx[p]+1 == mx[q]) mom[np] = q;
      else{
        int nq = newNode(); //fp[nq]=fp[q]
        mx[nq] = mx[p]+1;
        for(int i = 0; i < 33; i++)
          nxt[nq][i] = nxt[q][i];
        mom[nq] = mom[q];
        mom[q] = nq;
        mom[np] = nq;
        for(; p && nxt[p][c] == q; p = mom[p])
          nxt[p][c] = nq;
    } }
    lst = np;
  }
  void calc(){
    calc(root);
    iota(ind,ind+tot,1);
    sort(ind,ind+tot,[&](int i,int j){return mx[i]<mx[j
      ];});
    for(int i=tot-1;i>=0;i--)
    cnt[mom[ind[i]]]+=cnt[ind[i]];
  }
  void calc(int x){
    v[x]=ds[x]=1;dsl[x]=0; //rmom[mom[x]].push_back(x);
    for(int i=1;i<=26;i++){
      if(nxt[x][i]){
        if(!v[nxt[x][i]]) calc(nxt[x][i]);
        ds[x]+=ds[nxt[x][i]];
        dsl[x]+=ds[nxt[x][i]]+dsl[nxt[x][i]];
  } } }
  void push(const string& str){
    for(int i = 0; i < str.size() ; i++)
      push(str[i]-'a'+1);
  }
} sam;
```

5.5 Aho-Corasick AC!!!!!

```
struct ACautomata{
  struct Node{
    int cnt,i;
    Node *go[26], *fail, *dic;
    Node (){
      cnt = 0; fail = 0; dic=0;
      memset(go,0,sizeof(go));
    }
  }pool[1048576],*root;
  int nMem,n_pattern;
  Node* new_Node(){
    pool[nMem] = Node();
    return &pool[nMem++];
  }
  void init() {nMem=0;root=new_Node();n_pattern=0;}
  void add(const string &str) { insert(root,str,0); }
  void insert(Node *cur, const string &str, int pos){
    for(int i=pos;i<str.size();i++){
      if(!cur->go[str[i]-'a'])
        cur->go[str[i]-'a'] = new_Node();
      cur=cur->go[str[i]-'a'];
    }
    cur->cnt++; cur->i=n_pattern++;
  }
  void make_fail(){
    queue<Node*> que;
    que.push(root);
    while (!que.empty()){
      Node* fr=que.front(); que.pop();
      for (int i=0; i<26; i++){
        if (fr->go[i]){
          Node *ptr = fr->fail;
          while (ptr && !ptr->go[i]) ptr = ptr->fail;
          fr->go[i]->fail=ptr=(ptr?ptr->go[i]:root);
          fr->go[i]->dic=(ptr->cnt?ptr:ptr->dic);
          que.push(fr->go[i]);
  } } } }
  void query(string s){
      Node *cur=root;
      for(int i=0;i<(int)s.size();i++){
          while(cur&&!cur->go[s[i]-'a']) cur=cur->fail;
          cur=(cur?cur->go[s[i]-'a']:root);
          if(cur->i>=0) ans[cur->i]++;
          for(Node *tmp=cur->dic;tmp;tmp=tmp->dic)
              ans[tmp->i]++;
  } }// ans[i] : number of occurrence of pattern i
}AC;
```

5.6 Z Value

```
char s[MAXN];
int len,z[MAXN];
void Z_value() { //z[i] = lcp(s[1...],s[i...])
  int i,j,left,right;
  left=right=0; z[0]=len;
  for(i=1;i<len;i++) {
    j=max(min(z[i-left],right-i),0);
    for(;i+j<len&&s[i+j]==s[j];j++);
    z[i]=j;
    if(i+z[i]>right) {
      right=i+z[i];
      left=i;
}   }   }
```

5.7 BWT

```
struct BurrowsWheeler{
#define SIGMA 26
#define BASE 'a'
  vector<int> v[ SIGMA ];
  void BWT(char* ori, char* res){
    // make ori -> ori + ori
    // then build suffix array
  }
  void iBWT(char* ori, char* res){
    for( int i = 0 ; i < SIGMA ; i ++ )
      v[ i ].clear();
    int len = strlen( ori );
    for( int i = 0 ; i < len ; i ++ )
      v[ ori[i] - BASE ].push_back( i );
    vector<int> a;
    for( int i = 0 , ptr = 0 ; i < SIGMA ; i ++ )
      for( auto j : v[ i ] ){
        a.push_back( j );
        ori[ ptr ++ ] = BASE + i;
      }
    for( int i = 0 , ptr = 0 ; i < len ; i ++ ){
      res[ i ] = ori[ a[ ptr ] ];
      ptr = a[ ptr ];
    }
    res[ len ] = 0;
  }
} bwt;
```

5.8 ZValue Palindrome

```
void z_value_pal(char *s,int len,int *z){
  len=(len<<1)+1;
  for(int i=len-1;i>=0;i--)
    s[i]=i&1?s[i>>1]:'@';
  z[0]=1;
  for(int i=1,l=0,r=0;i<len;i++){
    z[i]=i<r?min(z[l+l-i],r-i):1;
    while(i-z[i]>=0&&i+z[i]<len&&s[i-z[i]]==s[i+z[i]])
        ++z[i];
    if(i+z[i]>r) l=i,r=i+z[i];
} }
```

5.9 Smallest Rotation

```
//rotate(begin(s),begin(s)+minRotation(s),end(s))
int minRotation(string s) {
  int a = 0, N = s.size(); s += s;
  rep(b,0,N) rep(k,0,N) {
    if(a+k == b || s[a+k] < s[b+k])
      {b += max(0, k-1); break;}
    if(s[a+k] > s[b+k]) {a = b; break;}
  } return a;
}
```

5.10 Cyclic LCS

```
#define L 0
#define LU 1
#define U 2
const int mov[3][2]={0,-1, -1,-1, -1,0};
int al,bl;
char a[MAXL*2],b[MAXL*2]; // 0-indexed
int dp[MAXL*2][MAXL];
char pred[MAXL*2][MAXL];
inline int lcs_length(int r) {
  int i=r+al,j=bl,l=0;
  while(i>r) {
    char dir=pred[i][j];
    if(dir==LU) l++;
    i+=mov[dir][0];
    j+=mov[dir][1];
  }
  return l;
}
inline void reroot(int r) { // r = new base row
  int i=r,j=1;
  while(j<=bl&&pred[i][j]!=LU) j++;
  if(j>bl) return;
  pred[i][j]=L;
  while(i<2*al&&j<=bl) {
    if(pred[i+1][j]==U) {
      i++;
      pred[i][j]=L;
    } else if(j<bl&&pred[i+1][j+1]==LU) {
      i++;
      j++;
      pred[i][j]=L;
    } else {
      j++;
} } }
int cyclic_lcs() {
  // a, b, al, bl should be properly filled
  // note: a WILL be altered in process
  //          -- concatenated after itself
  char tmp[MAXL];
  if(al>bl) {
    swap(al,bl);
    strcpy(tmp,a);
    strcpy(a,b);
    strcpy(b,tmp);
  }
  strcpy(tmp,a);
  strcat(a,tmp);
  // basic lcs
  for(int i=0;i<=2*al;i++) {
    dp[i][0]=0;
    pred[i][0]=U;
  }
  for(int j=0;j<=bl;j++) {
    dp[0][j]=0;
    pred[0][j]=L;
  }
  for(int i=1;i<=2*al;i++) {
    for(int j=1;j<=bl;j++) {
      if(a[i-1]==b[j-1]) dp[i][j]=dp[i-1][j-1]+1;
      else dp[i][j]=max(dp[i-1][j],dp[i][j-1]);
      if(dp[i][j-1]==dp[i][j]) pred[i][j]=L;
      else if(a[i-1]==b[j-1]) pred[i][j]=LU;
      else pred[i][j]=U;
  } }
  // do cyclic lcs
  int clcs=0;
  for(int i=0;i<al;i++) {
    clcs=max(clcs,lcs_length(i));
    reroot(i+1);
  }
  // recover a
  a[al]='\0';
  return clcs;
}
```

6 Data Structure

6.1 Link List

```
struct linklist{
    struct node{
        int value;
        node *front, *back;
        node(){
            front = back = nullptr;
        }
    }*begin, *end;
    int size = 0;
    linklist(){
        begin = end = new node();
        size = 0;
    }
    void push_back(int k){
        node *tmp;
        if(size == 0){
            begin->value = k;
            size++;
            return;
        }
        tmp = begin;
        node *a = new node();
        a->value = k;
        while(tmp->back != nullptr){
            tmp = tmp->back;
        }
        tmp->back = a;
        a->front = tmp;
        end = a;
        size++;
    }
    void insert(int loc, int k){
        node *tmp = begin, *a = new node(), *tmp2 = tmp
            ;
        a->value = k;
        int now = 0;
        while(now != loc){
            now++;
            tmp2 = tmp;
            tmp = tmp->back;
        }
        tmp2->back = a;
        tmp->front = a;
        a->front = tmp->front;
        a->back = tmp;
        size++;
    }
    void push_front(int k){
        if(size == 0){
            begin->value = k;
            size++;
            return;
        }
        node *a = new node();
        a->value = k;
        begin->front = a;
        a->back = begin;
        begin = a;
        size++;
    }
    void remove(int loc){
        node *tmp = begin, *tmp2 = begin;
        int now = 0;
        while(now != loc){
            now++;
            tmp2 = tmp;
            tmp = tmp->back;
        }
        tmp2->back = tmp->back;
        tmp->back->front = tmp2;
        delete tmp;
        size--;
    }
    void pop_back(){
        if(size == 1){
            size = 0;
            return;
        }
        end = end->front;
        delete end->back;
        end->back = nullptr;
        size--;
    }
    void pop_front(){
        if(size == 1){
            size = 0;
            return;
        }
        begin = begin->back;
        delete begin->front;
        begin->front = nullptr;
        size--;
    }
    void print(){
        node *tmp = begin;
        while(tmp != nullptr and size != 0){
            //print something
            tmp = tmp->back;
        }
    }
    int front(){ return begin->value; }
    int back(){ return end->value; }
    bool empty(){ return size == 0; }
};
```

6.2 Treap

```
struct Treap{
  int sz , val , pri , tag;
  Treap *l , *r;
  Treap( int _val ){
    val = _val; sz = 1;
    pri = rand(); l = r = NULL; tag = 0;
  }
};
void push( Treap * a ){
  if( a->tag ){
    Treap *swp = a->l; a->l = a->r; a->r = swp;
    int swp2;
    if( a->l ) a->l->tag ^= 1;
    if( a->r ) a->r->tag ^= 1;
    a->tag = 0;
} }
inline int Size( Treap * a ){ return a ? a->sz : 0; }
void pull( Treap * a ){
  a->sz = Size( a->l ) + Size( a->r ) + 1;
}
Treap* merge( Treap *a , Treap *b ){
  if( !a || !b ) return a ? a : b;
  if( a->pri > b->pri ){
    push( a );
    a->r = merge( a->r , b );
    pull( a );
    return a;
  }else{
    push( b );
    b->l = merge( a , b->l );
    pull( b );
    return b;
} }
void split_kth( Treap *t , int k, Treap*&a, Treap*&b ){
  if( !t ){ a = b = NULL; return; }
  push( t );
  if( Size( t->l ) + 1 <= k ){
    a = t;
    split_kth( t->r , k - Size( t->l ) - 1 , a->r , b )
        ;
    pull( a );
  }else{
    b = t;
    split_kth( t->l , k , a , b->l );
    pull( b );
} }
void split_key(Treap *t, int k, Treap*&a, Treap*&b){
  if(!t){ a = b = NULL; return; }
  push(t);
  if(k<=t->val){
    b = t;
    split_key(t->l,k,a,b->l);
    pull(b);
  }
  else{
    a = t;
    split_key(t->r,k,a->r,b);
    pull(a);
} }
```

6.3 Link-Cut Tree

```
struct Splay {
  static Splay nil, mem[MEM], *pmem;
  Splay *ch[2], *f;
  int val, rev, size;
  Splay (int _val=-1) : val(_val), rev(0), size(1)
  { f = ch[0] = ch[1] = &nil; }
  bool isr()
  { return f->ch[0] != this && f->ch[1] != this; }
  int dir()
  { return f->ch[0] == this ? 0 : 1; }
  void setCh(Splay *c, int d){
    ch[d] = c;
    if (c != &nil) c->f = this;
    pull();
  }
  void push(){
    if( !rev ) return;
    swap(ch[0], ch[1]);
    if (ch[0] != &nil) ch[0]->rev ^= 1;
    if (ch[1] != &nil) ch[1]->rev ^= 1;
    rev=0;
  }
  void pull(){
    size = ch[0]->size + ch[1]->size + 1;
    if (ch[0] != &nil) ch[0]->f = this;
    if (ch[1] != &nil) ch[1]->f = this;
  }
} Splay::nil, Splay::mem[MEM], *Splay::pmem = Splay::
    mem;
Splay *nil = &Splay::nil;
void rotate(Splay *x){
  Splay *p = x->f;
  int d = x->dir();
  if (!p->isr()) p->f->setCh(x, p->dir());
  else x->f = p->f;
  p->setCh(x->ch[!d], d);
  x->setCh(p, !d);
  p->pull(); x->pull();
}
vector<Splay*> splayVec;
void splay(Splay *x){
  splayVec.clear();
  for (Splay *q=x;; q=q->f){
    splayVec.push_back(q);
    if (q->isr()) break;
  }
  reverse(begin(splayVec), end(splayVec));
  for (auto it : splayVec) it->push();
  while (!x->isr()) {
    if (x->f->isr()) rotate(x);
    else if (x->dir()==x->f->dir())
      rotate(x->f),rotate(x);
    else rotate(x),rotate(x);
  }
}
int id(Splay *x) { return x - Splay::mem + 1; }
Splay* access(Splay *x){
  Splay *q = nil;
  for (;x!=nil;x=x->f){
    splay(x);
    x->setCh(q, 1);
    q = x;
  }
  return q;
}
void chroot(Splay *x){
  access(x);
  splay(x);
  x->rev ^= 1;
  x->push(); x->pull();
}
void link(Splay *x, Splay *y){
  access(x);
  splay(x);
  chroot(y);
  x->setCh(y, 1);
}
void cut_p(Splay *y) {
  access(y);
  splay(y);
  y->push();
  y->ch[0] = y->ch[0]->f = nil;
}
void cut(Splay *x, Splay *y){
  chroot(x);
  cut_p(y);
}
Splay* get_root(Splay *x) {
  access(x);
  splay(x);
  for(; x->ch[0] != nil; x = x->ch[0])
    x->push();
  splay(x);
  return x;
}
bool conn(Splay *x, Splay *y) {
  x = get_root(x);
  y = get_root(y);
  return x == y;
}
Splay* lca(Splay *x, Splay *y) {
  access(x);
  access(y);
  splay(x);
  if (x->f == nil) return x;
  else return x->f;
}
```

6.4 Disjoint Set

```
struct DisjointSet {
  int fa[MXN], h[MXN], top;
  struct Node {
    int x, y, fa, h;
    Node(int _x = 0, int _y = 0, int _fa = 0, int _h =
        0)
        : x(_x), y(_y), fa(_fa), h(_h) {}
  } stk[MXN];
  void init(int n) {
    top = 0;
    for (int i = 1; i <= n; i++) fa[i] = i, h[i] = 0;
  }
  int find(int x) { return x == fa[x] ? x : find(fa[x])
      ; }
  void merge(int u, int v) {
    int x = find(u), y = find(v);
    if (h[x] > h[y]) swap(x, y);
    stk[top++] = Node(x, y, fa[x], h[y]);
    if (h[x] == h[y]) h[y]++;
    fa[x] = y;
  }
  void undo(int k=1) {  //undo k times
    for (int i = 0; i < k; i++) {
      Node &it = stk[--top];
      fa[it.x] = it.fa;
      h[it.y] = it.h;
} } }djs;
```

6.5 Segment Tree

```
struct seg_tree{
  ll a[MXN],val[MXN*4],tag[MXN*4],NO_TAG=0;
  void push(int i,int l,int r){
    if(tag[i]!=NO_TAG){
      val[i]+=tag[i]; // update by tag
      if(l!=r){
        tag[cl(i)]+=tag[i]; // push
        tag[cr(i)]+=tag[i]; // push
      }
      tag[i]=NO_TAG;
  } }
  void pull(int i,int l,int r){
    int mid=(l+r)>>1;
    push(cl(i),l,mid);push(cr(i),mid+1,r);
    val[i]=max(val[cl(i)],val[cr(i)]); // pull
  }
  void build(int i,int l,int r){
    if(l==r){
      val[i]=a[l]; // set value
      return;
    }
    int mid=(l+r)>>1;
    build(cl(i),l,mid);build(cr(i),mid+1,r);
    pull(i,l,r);
  }
  void update(int i,int l,int r,int ql,int qr,int v){
    push(i,l,r);
    if(ql<=l&&r<=qr){
      tag[i]+=v; // update tag
      return;
    }
    int mid=(l+r)>>1;
    if(ql<=mid) update(cl(i),l,mid,ql,qr,v);
    if(qr>mid) update(cr(i),mid+1,r,ql,qr,v);
    pull(i,l,r);
  }
  ll query(int i,int l,int r,int ql,int qr){
    push(i,l,r);
    if(ql<=l&&r<=qr)
      return val[i]; // update answer
      ll mid=(l+r)>>1,ret=0;
    if(ql<=mid) ret=max(ret,query(cl(i),l,mid,ql,qr));
    if(qr>mid) ret=max(ret,query(cr(i),mid+1,r,ql,qr));
    return ret;
} }tree;
```

6.6 Bit Index Tree

```
//N -> BIT陣列大小bit.size()
int query(int x){
    int ret = 0;
    while(x){
        ret += b[x];
        x -= x & (-x);
    }
    return ret;
}

void update(int x, int d){
    while(x <= N){
        b[x] += d;
        x += x & (-x);
    }
}
```

6.7 持久化 Segment Tree

```
struct node{
    int data;
    node *lch,*rch;
    node(int data):data(data),lch(nullptr),rch(nullptr)
        {}
    void pull(){
        data=0;
        if(lch!=nullptr) data+=lch->data;
        if(rch!=nullptr) data+=rch->data;
    }
};
void modify(int l,int r,int pos,node *pre,node *now,int
     data){
    if(l==r)
        now->data=data;
    else{
        now->lch=pre->lch;
        now->rch=pre->rch;
        int mid=(l+r)>>1;
        if(pos<=mid){
            now->lch=new node(0);
            modify(l,mid,pos,pre->lch,now->rch,data);
        }
        else{
            now->lch=new node(0);
            modify(mid+1,r,pos,pre->rch,now->rch,data);
        }
        now->pull();
    }
}
int find(int l,int r,node *p,int k){
    if(l==r) return l;
    int mid=(l+r)>>1;
    int l_size = p->lch->data;
    if(k<=l_size)
        return find(l,mid,p->lch,k);
    else
        return find(mid+1,r,p->rch,k-l_size);
}
void build(int l,int r,node *p){
    if(l==r) return;
    int mid=(l+r)>>1;
    p->lch=new node(0);
    build(l,mid,p->lch);
    p->rch=new node(0);
    build(mid+1,r,p->rch);
}
const int maxn=1000005;
int arr[maxn];
node *T[maxn];
int main(){
    int N,Q;
    cin>>N>>Q;
    vector<int> dct;
    for(int i=0;i<=N;i++)
        cin>>arr[i],dct.push_back(arr[i]);
    sort(dct.begin(),dct.end());
    dct.resize(unique(dct.begin(),dct.end(),arr[i])-dct
        .begin());
    T[0]=build(0,(int)dct.size()-1);
    for(int i=1;i<=N;i++){
        arr[i]=lower_bound(dct.begin(),dct.end(),arr[i
            ])-dct.begin();
        T[i]=new node(0);
        modify(0,(int)dct.size()-1,T[i-1],T[i],arr[i]);
    }
    while(Q--){
        int l,r,k;
        cin>>l>>r>>k;
        cout<<dct[find(0,(int)dct.size()-1,T[l-1],T[r],
            k)]<<endl;
    }
}
```

6.8 Black Magic

```
#include <bits/extc++.h>
using namespace __gnu_pbds;
typedef tree<int,null_type,less<int>,rb_tree_tag,
    tree_order_statistics_node_update> set_t;
#include <ext/pb_ds/assoc_container.hpp>
typedef cc_hash_table<int,int> umap_t;
typedef priority_queue<int> heap;
#include<ext/rope>
using namespace __gnu_cxx;
int main(){
  // Insert some entries into s.
  set_t s; s.insert(12); s.insert(505);
  // The order of the keys should be: 12, 505.
  assert(*s.find_by_order(0) == 12);
  assert(*s.find_by_order(3) == 505);
  // The order of the keys should be: 12, 505.
  assert(s.order_of_key(12) == 0);
  assert(s.order_of_key(505) == 1);
  // Erase an entry.
  s.erase(12);
  // The order of the keys should be: 505.
  assert(*s.find_by_order(0) == 505);
  // The order of the keys should be: 505.
  assert(s.order_of_key(505) == 0);

  heap h1 , h2; h1.join( h2 );

  rope<char> r[ 2 ];
  r[ 1 ] = r[ 0 ]; // persistenet
  string t = "abc";
  r[ 1 ].insert( 0 , t.c_str() );
  r[ 1 ].erase( 1 , 1 );
  cout << r[ 1 ].substr( 0 , 2 );
}
```

7 Others

7.1 Find max tangent(x,y is increasing)

```
const int MAXN = 100010;
Pt sum[MAXN], pnt[MAXN], ans, calc;
inline bool cross(Pt a, Pt b, Pt c){
  return (c.y-a.y)*(c.x-b.x) > (c.x-a.x)*(c.y-b.y);
}//pt[0]=(0,0);pt[i]=(i,pt[i-1].y+dy[i-1]),i=1~n;dx>=l
double find_max_tan(int n,int l,LL dy[]){
  int np, st, ed, now;
  sum[0].x = sum[0].y = np = st = ed = 0;
  for (int i = 1, v; i <= n; i++)
    sum[i].x=i,sum[i].y=sum[i-1].y+dy[i-1];
  ans.x = now = 1,ans.y = -1;
  for (int i = 0; i <= n - l; i++){
    while(np>1&&cross(pnt[np-2],pnt[np-1],sum[i]))
      np--;
    if (np < now && np != 0) now = np;
    pnt[np++] = sum[i];
    while(now<np&&!cross(pnt[now-1],pnt[now],sum[i+l]))
      now++;
    calc = sum[i + l] - pnt[now - 1];
    if (ans.y * calc.x < ans.x * calc.y)
      ans = calc,st = pnt[now - 1].x,ed = i + l;
  }
  return (double)(sum[ed].y-sum[st].y)/(sum[ed].x-sum[
      st].x);
}
```

7.2 Exact Cover Set

```
// given n*m 0-1 matrix
// find a set of rows s.t.
// for each column, there's exactly one 1
#define N 1024 //row
#define M 1024 //column
#define NM ((N+2)*(M+2))
char A[N][M]; //n*m 0-1 matrix
int used[N]; //answer: the row used
int id[N][M];
int L[NM],R[NM],D[NM],U[NM],C[NM],S[NM],ROW[NM];
void remove(int c){
  L[R[c]]=L[c]; R[L[c]]=R[c];
  for( int i=D[c]; i!=c; i=D[i] )
    for( int j=R[i]; j!=i; j=R[j] ){
      U[D[j]]=U[j]; D[U[j]]=D[j]; S[C[j]]--;
}   }
void resume(int c){
  for( int i=D[c]; i!=c; i=D[i] )
    for( int j=L[i]; j!=i; j=L[j] ){
      U[D[j]]=D[U[j]]=j; S[C[j]]++;
    }
  L[R[c]]=R[L[c]]=c;
}
int dfs(){
  if(R[0]==0) return 1;
  int md=100000000,c;
  for( int i=R[0]; i!=0; i=R[i] )
    if(S[i]<md){ md=S[i]; c=i; }
  if(md==0) return 0;
  remove(c);
  for( int i=D[c]; i!=c; i=D[i] ){
    used[ROW[i]]=1;
    for( int j=R[i]; j!=i; j=R[j] ) remove(C[j]);
    if(dfs()) return 1;
    for( int j=L[i]; j!=i; j=L[j] ) resume(C[j]);
    used[ROW[i]]=0;
  }
  resume(c);
  return 0;
}
int exact_cover(int n,int m){
  for( int i=0; i<=m; i++ ){
    R[i]=i+1; L[i]=i-1; U[i]=D[i]=i;
    S[i]=0; C[i]=i;
  }
  R[m]=0; L[0]=m;
  int t=m+1;
  for( int i=0; i<n; i++ ){
    int k=-1;
    for( int j=0; j<m; j++ ){
      if(!A[i][j]) continue;
      if(k==-1) L[t]=R[t]=t;
      else{ L[t]=k; R[t]=R[k]; }
      k=t; D[t]=j+1; U[t]=U[j+1];
      L[R[t]]=R[L[t]]=U[D[t]]=D[U[t]]=t;
      C[t]=j+1; S[C[t]]++; ROW[t]=i; id[i][j]=t++;
  } }
  for( int i=0; i<n; i++ ) used[i]=0;
  return dfs();
}
```

7.3 逆序數對

```
//BIT at here
int main(){
    vector<int> arr, idx, res;
    bit.resize(arr.size() + 10);
    res.resize(idx.size());
    idx.resize(unique(idx.begin(), idx.end()) - idx.
        begin());
    sort(idx.begin(), idx.end());
    for(int i = 0; i < res.size(); i++)
        res[i] = lower_bound(idx.begin(), idx.end(),
            all[i]) - idx.begin() + 1;
    int ans = 0;
    for(int i = all.size() - 2; i >= 0; i -= 2){
        ans += query(res[i] - 1);
        update(res[i], 1);
    }
    cout << ans << endl;
}
```