Contents

1 Basic

1.1 default code

```
#pragma GCC optimize("O3,unroll-loops")
#pragma GCC target("avx2,bmi,bmi2,lzcnt,popcnt")
#include <bits/stdc++.h>
using namespace std;
ios_base::sync_with_stdio(0); cin.tie(0); cout.tie(0);
```

1.2 .vimrc

```
set nu rnu ts=4 sw=4 bs=2 ai hls cin mouse=a
color default
sy on
inoremap {<CR> {<CR>}<C-o>O
inoremap jk <Esc>
nnoremap J 5j
nnoremap K 5k
nnoremap run :w<bar>!g++ -std=c++14 -DLOCAL -Wfatal-
    errors -o test "%" && echo "done." && time ./test<
    CR>
```

1.3 Increase Stack Size (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);
} } }
```

1.4 Misc

```
編譯參數：-std=c++14 -Wall -Wshadow (-fsanitize=
    undefined)

mt19937 gen(chrono::steady_clock::now().
    time_since_epoch().count());
int randint(int lb, int ub)
{ return uniform_int_distribution<int>(lb, ub)(gen); }

#define SECs ((double)clock() / CLOCKS_PER_SEC)
double startTime;
bool TIME() { // 比最大可執行時間小一點
    return SECs - startTime > 0.8;
}
int main() {
    startTime = SECs;
}
```

```
struct KeyHasher {
  size_t operator()(const Key& k) const {
    return k.first + k.second * 100000;
} };
typedef unordered_map<Key,int,KeyHasher> map_t;
// builtin function 可以代的值為int32
__builtin_popcountll    // 二進位有幾個1
__builtin_clzll         // 左起第一個1之前0的個數
__builtin_parityll      // 1的個數的奇偶性
__builtin_mul_overflow(a,b,&h) // a*b是否溢位
```

1.5 check

```
for ((i=0;;i++))
do
    echo "$i"
    python3 gen.py > input
    ./ac < input > ac.out
    ./wa < input > wa.out
    diff ac.out wa.out || break
done
```

1.6 python-related

```
parser:
int(eval(num.replace("/","//")))

from fractions import Fraction
from decimal import Decimal, getcontext, ROUND_HALF_UP,
     ROUND_CEILING, ROUND_FLOOR
getcontext().prec = 250 # set precision
getcontext().rounding = ROUND_HALF_UP

itwo = Decimal(0.5)
two = Decimal(2)

format(x, '0.10f') # set precision

N = 200
def angle(cosT):
  """given cos(theta) in decimal return theta"""
  for i in range(N):
    cosT = ((cosT + 1) / two) ** itwo
  sinT = (1 - cosT * cosT) ** itwo
  return sinT * (2 ** N)
pi = angle(Decimal(-1))

"""round to 2 decimal places"""
sum = Decimal(input())
sum.quantize(Decimal('.00'), ROUND_HALF_UP)

"""Fraction"""
x = Fraction(1, 3) # 1/3
x.as_integer_ratio() # (1, 3)

"""input list of integers"""
arr = list(map(int, input().split()))
```

2 flow

2.1 ISAP $O(V^3)$

```
struct Maxflow {
  static const int MAXV = 20010;
  static const int INF  = 1000000;
  struct Edge {
    int v, c, r;
    Edge(int _v, int _c, int _r):
      v(_v), c(_c), r(_r) {}
  };
  int s, t;
  vector<Edge> G[MAXV*2];
  int iter[MAXV*2], d[MAXV*2], gap[MAXV*2], tot;
  void init(int x) {
    tot = x+2;
    s = x+1, t = x+2;
    for(int i = 0; i <= tot; i++) {
      G[i].clear();
      iter[i] = d[i] = gap[i] = 0;
  } }
  void addEdge(int u, int v, int c) {
    G[u].push_back(Edge(v, c, SZ(G[v]) ));
    G[v].push_back(Edge(u, 0, SZ(G[u]) - 1));
  }
  int dfs(int p, int flow) {
    if(p == t) return flow;
    for(int &i = iter[p]; i < SZ(G[p]); i++) {
      Edge &e = G[p][i];
      if(e.c > 0 && d[p] == d[e.v]+1) {
        int f = dfs(e.v, min(flow, e.c));
        if(f) {
          e.c -= f;
          G[e.v][e.r].c += f;
          return f;
    } } }
    if( (--gap[d[p]]) == 0) d[s] = tot;
    else {
      d[p]++;
      iter[p] = 0;
      ++gap[d[p]];
    }
    return 0;
  }
  int solve() {
    int res = 0;
    gap[0] = tot;
    for(res = 0; d[s] < tot; res += dfs(s, INF));
    return res;
  }
  void reset() {
    for(int i=0;i<=tot;i++) {
      iter[i]=d[i]=gap[i]=0;
} } }flow;
```

2.2 MinCostFlow

```
struct zkwflow{
  static const int maxN=10000;
  struct Edge{ int v,f,re; ll w;};
  int n,s,t,ptr[maxN]; bool vis[maxN]; ll dis[maxN];
  vector<Edge> E[maxN];
  void init(int _n,int _s,int _t){
    n=_n,s=_s,t=_t;
    for(int i=0;i<n;i++) E[i].clear();
  }
  void addEdge(int u,int v,int f,ll w){
    E[u].push_back({v,f,(int)E[v].size(),w});
    E[v].push_back({u,0,(int)E[u].size()-1,-w});
  }
  bool SPFA(){
    fill_n(dis,n,LLONG_MAX); fill_n(vis,n,false);
    queue<int> q; q.push(s); dis[s]=0;
    while (!q.empty()){
      int u=q.front(); q.pop(); vis[u]=false;
      for(auto &it:E[u]){
        if(it.f>0&&dis[it.v]>dis[u]+it.w){
          dis[it.v]=dis[u]+it.w;
          if(!vis[it.v]){
            vis[it.v]=true; q.push(it.v);
    } } } }
    return dis[t]!=LLONG_MAX;
  }
  int DFS(int u,int nf){
    if(u==t) return nf;
    int res=0; vis[u]=true;
    for(int &i=ptr[u];i<(int)E[u].size();i++){
      auto &it=E[u][i];
      if(it.f>0&&dis[it.v]==dis[u]+it.w&&!vis[it.v]){
        int tf=DFS(it.v,min(nf,it.f));
        res+=tf,nf-=tf,it.f-=tf;
        E[it.v][it.re].f+=tf;
        if(nf==0){ vis[u]=false; break; }
      }
```

```
    }
    return res;
  }
  pair<int,ll> flow(){
    int flow=0; ll cost=0;
    while (SPFA()){
      fill_n(ptr,n,0);
      int f=DFS(s,INT_MAX);
      flow+=f; cost+=dis[t]*f;
    }
    return{ flow,cost };
  } // reset: do nothing
} flow;
```

2.3 Dinic $O(V^2E)$

```
#define SZ(x) (int)x.size()
#define PB push_back
struct Dinic{
  struct Edge{ int v,f,re; };
  int n,s,t,level[MXN];
  vector<Edge> E[MXN];
  void init(int _n, int _s, int _t){
    n = _n; s = _s; t = _t;
    for (int i=0; i<n; i++) E[i].clear();
  }
  void add_edge(int u, int v, int f){
    E[u].PB({v,f,SZ(E[v])});
    E[v].PB({u,0,SZ(E[u])-1});
  }
  bool BFS(){
    for (int i=0; i<n; i++) level[i] = -1;
    queue<int> que;
    que.push(s);
    level[s] = 0;
    while (!que.empty()){
      int u = que.front(); que.pop();
      for (auto it : E[u]){
        if (it.f > 0 && level[it.v] == -1){
          level[it.v] = level[u]+1;
          que.push(it.v);
    } } }
    return level[t] != -1;
  }
  int DFS(int u, int nf){
    if (u == t) return nf;
    int res = 0;
    for (auto &it : E[u]){
      if (it.f > 0 && level[it.v] == level[u]+1){
        int tf = DFS(it.v, min(nf,it.f));
        res += tf; nf -= tf; it.f -= tf;
        E[it.v][it.re].f += tf;
        if (nf == 0) return res;
    } }
    if (!res) level[u] = -1;
    return res;
  }
  int flow(int res=0){
    while ( BFS() )
      res += DFS(s,2147483647);
    return res;
} }flow;
```

2.4 匈牙利演算法

```
#define NIL -1
#define INF 100000000
int n,matched;
int cost[MAXN][MAXN];
bool sets[MAXN]; // whether x is in set S
bool sett[MAXN]; // whether y is in set T
int xlabel[MAXN],ylabel[MAXN];
int xy[MAXN],yx[MAXN]; // matched with whom
int slack[MAXN]; // given y: min{xlabel[x]+ylabel[y]-
    cost[x][y]} | x not in S
int prev[MAXN]; // for augmenting matching
inline void relabel() {
  int i,delta=INF;
  for(i=0;i<n;i++) if(!sett[i]) delta=min(slack[i],
      delta);
  for(i=0;i<n;i++) if(sets[i]) xlabel[i]-=delta;
  for(i=0;i<n;i++) {
    if(sett[i]) ylabel[i]+=delta;
    else slack[i]-=delta;
  }
}
inline void add_sets(int x) {
  int i;
  sets[x]=1;
  for(i=0;i<n;i++) {
    if(xlabel[x]+ylabel[i]-cost[x][i]<slack[i]) {
      slack[i]=xlabel[x]+ylabel[i]-cost[x][i];
      prev[i]=x;
    }
  }
}
inline void augment(int final) {
  int x=prev[final],y=final,tmp;
  matched++;
  while(1) {
    tmp=xy[x]; xy[x]=y; yx[y]=x; y=tmp;
    if(y==NIL) return;
    x=prev[y];
  }
}
inline void phase() {
  int i,y,root;
  for(i=0;i<n;i++) { sets[i]=sett[i]=0; slack[i]=INF; }
  for(root=0;root<n&&xy[root]!=NIL;root++);
  add_sets(root);
  while(1) {
    relabel();
    for(y=0;y<n;y++) if(!sett[y]&&slack[y]==0) break;
    if(yx[y]==NIL) { augment(y); return; }
    else { add_sets(yx[y]); sett[y]=1; }
  }
}
inline int hungarian() {
  int i,j,c=0;
  for(i=0;i<n;i++) {
    xy[i]=yx[i]=NIL;
    xlabel[i]=ylabel[i]=0;
    for(j=0;j<n;j++) xlabel[i]=max(cost[i][j],xlabel[i
        ]);
  }
  for(i=0;i<n;i++) phase();
  for(i=0;i<n;i++) c+=cost[i][xy[i]];
  return c;
}
```

2.5 Kuhn Munkres 最大完美二分匹配 $O(n^3)$

```
struct KM{ // max weight, for min negate the weights
  int n, mx[MXN], my[MXN], pa[MXN];
  ll g[MXN][MXN], lx[MXN], ly[MXN], sy[MXN];
  bool vx[MXN], vy[MXN];
  void init(int _n) { // 1-based
    n = _n;
    for(int i=1; i<=n; i++) fill(g[i], g[i]+n+1, 0);
  }
  void addEdge(int x, int y, ll w) {g[x][y] = w;}
  void augment(int y) {
    for(int x, z; y; y = z)
      x=pa[y], z=mx[x], my[y]=x, mx[x]=y;
  }
  void bfs(int st) {
    for(int i=1; i<=n; ++i) sy[i]=INF, vx[i]=vy[i]=0;
    queue<int> q; q.push(st);
    for(;;) {
      while(q.size()) {
        int x=q.front(); q.pop(); vx[x]=1;
        for(int y=1; y<=n; ++y) if(!vy[y]){
          ll t = lx[x]+ly[y]-g[x][y];
          if(t==0){
            pa[y]=x;
            if(!my[y]){augment(y);return;}
            vy[y]=1, q.push(my[y]);
          }else if(sy[y]>t) pa[y]=x,sy[y]=t;
```

```
      } }
      ll cut = INF;
      for(int y=1; y<=n; ++y)
        if(!vy[y]&&cut>sy[y]) cut=sy[y];
      for(int j=1; j<=n; ++j){
        if(vx[j]) lx[j] -= cut;
        if(vy[j]) ly[j] += cut;
        else sy[j] -= cut;
      }
      for(int y=1; y<=n; ++y) if(!vy[y]&&sy[y]==0){
        if(!my[y]){augment(y);return;}
        vy[y]=1, q.push(my[y]);
  } } }
  ll solve(){
    fill(mx, mx+n+1, 0); fill(my, my+n+1, 0);
    fill(ly, ly+n+1, 0); fill(lx, lx+n+1, -INF);
    for(int x=1; x<=n; ++x) for(int y=1; y<=n; ++y)
      lx[x] = max(lx[x], g[x][y]);
    for(int x=1; x<=n; ++x) bfs(x);
    ll ans = 0;
    for(int y=1; y<=n; ++y) ans += g[my[y]][y];
    return ans;
} }graph;
```

2.6 Flow Method

```
Maximize c^T x subject to Ax ≤ b, x ≥ 0;
with the corresponding symmetric dual problem,
Minimize b^T y subject to A^T y ≥ c, y ≥ 0.

Maximize c^T x subject to Ax ≤ b;
with the corresponding asymmetric dual problem,
Minimize b^T y subject to A^T y = c, y ≥ 0.

Minimum vertex cover on bipartite graph =
Maximum matching on bipartite graph

Minimum edge cover on bipartite graph =
vertex number - Minimum vertex cover(Maximum matching)

Independent set on bipartite graph =
vertex number - Minimum vertex cover(Maximum matching)

找出最小點覆蓋，做完dinic之後，從源點dfs只走還有流量的
    邊，左邊沒被走到的點跟右邊被走到的點就是答案，其他
    點為最大獨立集
```

Maximum density subgraph ($\sum W_e + \sum W_v$) / |V|

```
Binary search on answer:
For a fixed D, construct a Max flow model as follow:
Let S be Sum of all weight( or inf)
1. from source to each node with cap = S
2. For each (u,v,w) in E, (u->v,cap=w), (v->u,cap=w)
3. For each node v, from v to sink with cap = S + 2 * D
    - deg[v] - 2 * (W of v)
```

where deg[v] = $\sum weightofedgeassociatedwithv$

```
If maxflow < S * |V|, D is an answer.

Requiring subgraph: all vertex can be reached from
    source with
edge whose cap > 0.
```

3 Math

3.1 FFT

```
// const int MXN = 262144 (MXN must be 2^k)
// before any usage, run pre_fft() first
typedef long double ld;
typedef complex<ld> cplx; //real() ,imag()
const ld PI = acosl(-1);
const cplx I(0, 1);

struct FFT{
  cplx omega[MXN+1];
  FFT(){ //pre_fft
    for(int i=0; i<=MXN; i++)
      omega[i] = exp(i * 2 * PI / MXN * I);
  }
  // n must be 2^k
  void fft(int n, cplx a[], bool inv=false){
    int basic = MXN / n;
    int theta = basic;
    for (int m = n; m >= 2; m >>= 1) {
      int mh = m >> 1;
      for (int i = 0; i < mh; i++) {
      cplx w = omega[inv ? MXN-(i*theta%MXN) : i*theta%
          MXN];
      for (int j = i; j < n; j += m) {
        int k = j + mh;
        cplx x = a[j] - a[k];
        a[j] += a[k];
        a[k] = w * x;
      } }
      theta = (theta * 2) % MXN;
    }
    int i = 0;
    for (int j = 1; j < n - 1; j++) {
      for (int k = n >> 1; k > (i ^= k); k >>= 1);
      if (j < i) swap(a[i], a[j]);
    }
    if(inv) for (i = 0; i < n; i++) a[i] /= n;
  }
  cplx arr[MXN+1];
  inline void mul(int _n,ll a[],int _m,ll b[],ll ans[])
      {
    int n=1,sum=_n+_m-1;
    while(n<sum)
      n<<=1;
    for(int i=0;i<n;i++) {
      double x=(i<_n?a[i]:0),y=(i<_m?b[i]:0);
      arr[i]=complex<double>(x+y,x-y);
    }
    fft(n,arr);
    for(int i=0;i<n;i++)
      arr[i]=arr[i]*arr[i];
    fft(n,arr,true);
    for(int i=0;i<sum;i++)
      ans[i]=(long long)(arr[i].real()/4+0.5);
  }
}fft;
```

3.2 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;
}
```

3.3 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;
}
```

3.4 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;
}
```

3.5 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;
}
```

3.6 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;
} }
```

3.7 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;
}
```

3.8 Matrix

```
//矩陣乘法
for(int i = 0; i < n; i++){
    for(int j = 0; j < n; j++){
        for(int k = 0; k < n; k++){
            ret[i][j] += a[i][k] * b[k][j];
        }
    }
}
//矩陣快速冪
int base[2][2] = {   int ans[2][2] = {
  {1, 1},              {1, 0},
  {1, 0}               {0, 1}
};                   };

int mypow(int y){
  while(y){
    if( y&1 ) { ans = mul(ans, base); } //實作矩陣乘法
    base = mul(base, base);//實作矩陣乘法
    y >>= 1;
  }
  return ans[0][0];
}
```

3.9 Gaussian Elimination

```
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;
```

3.10 Inverse Matrix

```
int GAUSS_MOD;
struct GAUSS{
  int n;
  vector<vector<int> > v;
  vector<vector<int> > rev;
  int mul(int x,int y,int mod){
    int ret=x*y-(int)((long double)x/mod*y)*mod;
    return ret<0?ret+mod:ret;
  }
  int ppow(int a, int b){//res=(a^b)%m
    int res=1, k=a;
    while(b){
      if((b&1)) res=mul(res,k,GAUSS_MOD)%GAUSS_MOD;
      k=mul(k,k,GAUSS_MOD)%GAUSS_MOD;
      b>>=1;
    }
    return res%GAUSS_MOD;
  }
  bool solve(){
    for(int now = 0; now < n; now++){
      int ch;
      for(ch = now; ch < n && !v[ch][now]; ch++);
      if(ch >= n) return 0;
      for(int i = now; i < n; i++) if(v[now][now] == 0
          && v[i][now] != 0){
          swap(v[i] , v[now]); // det = -det;
          swap(rev[i], rev[now]);
      }
      if(v[now][now] == 0) return 0;
      int inv = ppow(v[now][now] , GAUSS_MOD - 2);
      for(int i = 0; i < n; i++) if(i != now){
        int tmp = v[i][now] * inv % GAUSS_MOD;
        for(int j = 0; j < n; j++) {
          (v[i][j] += GAUSS_MOD - tmp * v[now][j] %
              GAUSS_MOD) %= GAUSS_MOD;
          (rev[i][j] += GAUSS_MOD - tmp * rev[now][j] %
               GAUSS_MOD) %= GAUSS_MOD;
        }
      }
    }
    return 1;
}} gs;

signed main(){
  int n, p; //n*n matrix, MOD=p
  cin>>n>>p; //if(!n && !p) return 0;
  GAUSS_MOD = p; gs.n = n;
  gs.v.clear() , gs.v.resize(n + 1, vector<int>(n + 2 ,
       0));
  gs.rev.clear() , gs.rev.resize(n + 1, vector<int>(n +
       2 , 0));
  for(int i = 0; i < n; i++){
    for(int j = 0; j < n; j++){
      cin>>gs.v[i][j];
      if(i == j) gs.rev[i][j] = 1;
    }
  }
  if(!gs.solve()) cout << "singular\n";
  else{
    for(int i = 0; i < n; i++){
      int inv = gs.ppow(gs.v[i][i] , p - 2);
      for(int j = 0; j < n; j++)
          cout << (gs.rev[i][j] * inv % p) <<" ";
      cout<<"\n";
    }
  }
  cout << "\n";
}
```

3.11 模反元素

```
long long inv(long long a,long long m){
    long long x,y;
    long long d=exgcd(a,m,x,y);
    if(d==1) return (x+m)%m;
    else return -1; //-1為無解
}
```

3.12 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)};
}

int exgcd(int a,int b,long long &x,long long &y) {
    if(b == 0){x=1,y=0;return a;}
    int now=exgcd(b,a%b,y,x);
    y-=a/b*x;
    return now;
}
```

3.13 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;
}
```

3.14 Prefix Inverse

```
void solve( int m ){
  inv[ 1 ] = 1;
  for( int i = 2 ; i < m ; i ++ )
    inv[ i ] = ((LL)(m - m / i) * inv[m % i]) % m;
}
```

3.15 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]
```

3.16 Combination thearom

```
const ll mod = 1e9 + 7;
ll fac[(int)2e6 + 1], inv[(int)2e6 + 1];
ll getinv(ll a){ return qpow(a, mod-2); }
void init(int n){
  fac[0] = 1;
  for(int i = 1; i <= n; i++){
    fac[i] = fac[i-1] * i % mod;
  }
  inv[n] = getinv(fac[n]);
  for(int i = n - 1; i >= 0; i--){
    inv[i] = inv[i + 1] * (i + 1) % mod;
  }
}
ll C(int n, int m){
  if(m > n) return 0;
  return fac[n] * inv[m] % mod * inv[n-m] % mod;
}
```

3.17 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;
}
```

3.18 Phi

```
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;
  for(ll i=2;i*i<=a;i++){ // O(sqrtN)
    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;
}
```

3.19 Int Sqrt

```
LL intSqrt(LL S) { //return origin val when S <= 0
    if (S <= 0) return S;
    LL x = S;
    for (LL nx;;x = nx){
        nx = (x+S/x)>>1LL;
        if(nx >= x) break;
    }
    return x;
}
```

3.20 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):
  $S(n,k) = \frac{1}{k!}\sum_{j=0}^{k}(-1)^{k-j}\binom{k}{j}j^n$

- Pick′ s Theorem : $A = i + b/2 - 1$
  在二維座標平面中畫上網格，對於任何簡單多邊形
  A: 面積、i: 內部的格點數、b: 邊上的格點數

- 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 is number of color，m is the number of cycle size):
  $(\sum_{i=1}^{m} c^{gcd(i,m)})/m$

- Burnside lemma:
  $|X/G| = \frac{1}{|G|}\sum_{g \in G}|X^g|$

- 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:
  $A^B \ \text{mod} \ C = A^{B \ \text{mod} \ \phi(c) + \phi(c)} \ \text{mod} \ C$

- 用歐拉函數求模反元素:
  如果 a 和 n 互質, 則 a 對 n 的模反元素
  $a^{-1} \equiv a^{\phi(n)-1}(mod \ n)$

- 6 的倍數:
  $(a-1)^3 + (a+1)^3 + (-a)^3 + (-a)^3 = 6a$

- 上高斯 (向上取整) :
  $\lceil \frac{a}{b} \rceil = \frac{a+b-1}{b}$

- 點到直線距離公式:
  $d = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2+b^2}}$

4 Geometry

4.1 definition

```
#define all(a) a.begin(),a.end()
ostream& operator<<(ostream& os, const Pt& pt) {
  return os << "(" << pt.x << ", " << pt.y << ")";}
typedef long double ld;
const ld eps = 1e-8;
const ld pi = acos(-1);
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 {
    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 Circle {
  Pt o; ld r;
  Circle(Pt _o=Pt(0, 0), ld _r=0):o(_o), r(_r) {}
};
```

4.2 Intersection of 2 lines

```
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);
}
```

4.3 halfPlaneIntersection

```
// O(nlogn)
// 傳入 vector<Line>
// (半平面為點 st 往 ed 的逆時針方向)
// 回傳值為形成的凸多邊形的頂點 vector
// assume that Lines intersect
vector<Pt> HPI(vector<Line> P) {
    sort(P.begin(), P.end(), [&](Line l, Line m) {
        if (argcmp(l.v, m.v)) return true;
        if (argcmp(m.v, l.v)) return false;
        return PtSide(l.s, m) > 0;
    });
    int n = P.size(), l = 0, r = -1;
    for (int i = 0; i < n; i++) {
        if (i and !argcmp(P[i - 1].v, P[i].v)) continue
            ;
        while (l < r and PtSide(LLIntersect(P[r-1], P[r
            ]), P[i]) <= 0) r--;
        while (l < r and PtSide(LLIntersect(P[l], P[l
            +1]), P[i]) <= 0) l++;
        P[++r] = P[i];
    }
    while (l < r and PtSide(LLIntersect(P[r-1], P[r]),
        P[l]) <= 0) r--;
    while (l < r and PtSide(LLIntersect(P[l], P[l+1]),
        P[r]) <= 0) l++;
    if (r - l <= 1 or !argcmp(P[l].v, P[r].v))
        return {}; // empty
    if (PtSide(LLIntersect(P[l], P[r]), P[l+1]) <= 0) {
        assert(0);
        return {}; // infinity
    }
    vector<Line> lns = vector(P.begin() + l, P.begin()
        + r + 1);
  lns.push_back(lns[0]);
  vector<Pt> hpi;
```

```
  for(int i = 1; i < lns.size(); i++) hpi.push_back(
      LLIntersect(lns[i-1], lns[i]));
  return hpi;
}
```

4.4 Convex Hull

```
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) // 如果想要有點共線的點，把 <= 改成 <
      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;
}
```

4.5 Convex Hull trick

```
struct Convex {
  int n;
  vector<Pt> A, V, L, U;
  Convex(const vector<Pt> &_A) : A(_A), n(_A.size()) {
      // n >= 3
    auto it = max_element(all(A));
    L.assign(A.begin(), it + 1);
    U.assign(it, A.end()), U.push_back(A[0]);
    for (int i = 0; i < n; i++) {
      V.push_back(A[(i + 1) % n] - A[i]);
    }
  }
  int PtSide(Pt p, Line L) {
    return dcmp((L.b - L.a)^(p - L.a));
  }
  int inside(Pt p, const vector<Pt> &h, auto f) {
    auto it = lower_bound(all(h), p, f);
    if (it == h.end()) return 0;
    if (it == h.begin()) return p == *it;
    return 1 - dcmp((p - *prev(it))^(*it  - *prev(it)))
        ;
  }
  // 1. whether a given point is inside the CH
  // ret 0: out, 1: on, 2: in
  int inside(Pt p) {
    return min(inside(p, L, less{}), inside(p, U,
        greater{}));
  }
  static bool cmp(Pt a, Pt b) { return dcmp(a ^ b) > 0;
      }
  // 2. Find tangent points of a given vector
  // ret the idx of far/closer tangent point
  int tangent(Pt v, bool close = true) {
    assert(v != Pt{});
    auto l = V.begin(), r = V.begin() + L.size() - 1;
    if (v < Pt{}) l = r, r = V.end();
    if (close) return (lower_bound(l, r, v, cmp) - V.
        begin()) % n;
    return (upper_bound(l, r, v, cmp) - V.begin()) % n;
  }
  // 3. Find 2 tang pts on CH of a given outside point
  // return index of tangent points
  // return {-1, -1} if inside CH
  array<int, 2> tangent2(Pt p) {
    array<int, 2> t{-1, -1};
    if (inside(p) == 2) return t;
    if (auto it = lower_bound(all(L), p); it != L.end()
        and p == *it) {
      int s = it - L.begin();
      return {(s + 1) % n, (s - 1 + n) % n};
    }
    if (auto it = lower_bound(all(U), p, greater{}); it
        != U.end() and p == *it) {
      int s = it - U.begin() + L.size() - 1;
      return {(s + 1) % n, (s - 1 + n) % n};
    }
    for (int i = 0; i != t[0]; i = tangent((A[t[0] = i]
        - p), 0));
    for (int i = 0; i != t[1]; i = tangent((p - A[t[1]
        = i]), 1));
    return t;
  }
  int find(int l, int r, Line L) {
    if (r < l) r += n;
    int s = PtSide(A[l % n], L);
    return *ranges::partition_point(views::iota(l, r),
      [&](int m) {
        return PtSide(A[m % n], L) == s;
      }) - 1;
  };
  // 4. Find intersection point of a given line
  // intersection is on edge (i, next(i))
  vector<int> intersect(Line L) {
    int l = tangent(L.a - L.b), r = tangent(L.b - L.a);
    if(PtSide(A[l], L) == 0) return {l};
    if(PtSide(A[r], L) == 0) return {r};
    if (PtSide(A[l], L) * PtSide(A[r], L) > 0) return
        {};
    return {find(l, r, L) % n, find(r, l, L) % n};
  }
};
```

4.6 Intersection of 2 segments

```
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);
}
```

4.7 Intersection of Polygon and Circle

```
ld PCIntersect(vector<Pt> v, Circle cir) {
  for(int i = 0 ; i < (int)v.size() ; ++i) v[i] = v[i]
      - cir.o;
  ld ans = 0, r = cir.r;
  int n = v.size();
  for(int i = 0 ; i < n ; ++i) {
    Pt pa = v[i], pb = v[(i+1)%n];
    if(norm(pa) < norm(pb)) swap(pa, pb);
    if(dcmp(norm(pb)) == 0) continue;
    ld s, h, theta;
    ld a = norm(pb), b = norm(pa), c = norm(pb-pa);
    ld cosB = (pb*(pb-pa))/a/c, B = acos(cosB);
    if(cosB > 1) B = 0;
    else if(cosB < -1) B = PI;
    ld cosC = (pa*pb)/a/b, C = acos(cosC);
    if(cosC > 1) C = 0;
    else if(cosC < -1) C = PI;
    if(a > r) {
      s = (C/2)*r*r;
      h = a*b*sin(C)/c;
```

```
      if(h < r && B < PI/2) s -= (acos(h/r)*r*r - h*
          sqrt(r*r-h*h));
    }
    else if(b > r) {
      theta = PI - B - asin(sin(B)/r*a);
      s = 0.5*a*r*sin(theta) + (C-theta)/2*r*r;
    }
    else s = 0.5*sin(C)*a*b;
    ans += abs(s)*dcmp(v[i]^v[(i+1)%n]);
  }
  return abs(ans);
}
```

4.8 Circle cover

```
#define N 1021
#define D long double
struct CircleCover{
  int C; Circle c[ N ]; //填入C(圓數量),c(圓陣列)
  bool g[ N ][ N ], overlap[ N ][ N ];
  // Area[i] : area covered by at least i circles
  D Area[ N ];
  void init( int _C ){ C = _C; }
  bool CCinter( Circle& a , Circle& b , Pt& p1 , Pt& p2
      ){
    Pt o1 = a.o , o2 = b.o;
    D r1 = a.r , r2 = b.r;
    if( norm( o1 - o2 ) > r1 + r2 ) return {};
    if( norm( o1 - o2 ) < max(r1, r2) - min(r1, r2) )
        return {};
    D d2 = ( o1 - o2 ) * ( o1 - o2 );
    D d = sqrt(d2);
    if( d > r1 + r2 ) return false;
    Pt u=(o1+o2)*0.5 + (o1-o2)*((r2*r2-r1*r1)/(2*d2));
    D A=sqrt((r1+r2+d)*(r1-r2+d)*(r1+r2-d)*(-r1+r2+d));
    Pt v=Pt( o1.y-o2.y , -o1.x + o2.x ) * A / (2*d2);
    p1 = u + v; p2 = u - v;
    return true;
  }
  struct Teve {
    Pt p; D ang; int add;
    Teve() {}
    Teve(Pt _a, D _b, int _c):p(_a), ang(_b), add(_c){}
    bool operator<(const Teve &a)const
    {return ang < a.ang;}
  }eve[ N * 2 ];
  // strict: x = 0, otherwise x = -1
  bool disjuct( Circle& a, Circle &b, int x )
  {return dcmp( norm( a.o - b.o ) - a.r - b.r ) > x;}
  bool contain( Circle& a, Circle &b, int x )
  {return dcmp( a.r - b.r - norm( a.o - b.o ) ) > x;}
  bool contain(int i, int j){
    /* c[j] is non-strictly in c[i]. */
    return (dcmp(c[i].r - c[j].r) > 0 ||
            (dcmp(c[i].r - c[j].r) == 0 && i < j) ) &&
                contain(c[i], c[j], -1);
  }
  void solve(){
    for( int i = 0 ; i <= C + 1 ; i ++ )
      Area[ i ] = 0;
    for( int i = 0 ; i < C ; i ++ )
      for( int j = 0 ; j < C ; j ++ )
        overlap[i][j] = contain(i, j);
    for( int i = 0 ; i < C ; i ++ )
      for( int j = 0 ; j < C ; j ++ )
        g[i][j] = !(overlap[i][j] || overlap[j][i] ||
            disjuct(c[i], c[j], -1));
    for( int i = 0 ; i < C ; i ++ ){
      int E = 0, cnt = 1;
      for( int j = 0 ; j < C ; j ++ )
        if( j != i && overlap[j][i] )
          cnt ++;
      for( int j = 0 ; j < C ; j ++ )
        if( i != j && g[i][j] ){
          Pt aa, bb;
          CCinter(c[i], c[j], aa, bb);
          D A=atan2(aa.y - c[i].o.y, aa.x - c[i].o.x);
          D B=atan2(bb.y - c[i].o.y, bb.x - c[i].o.x);
          eve[E ++] = Teve(bb, B, 1);
          eve[E ++] = Teve(aa, A, -1);
          if(B > A) cnt ++;
        }
      if( E == 0 ) Area[ cnt ] += pi * c[i].r * c[i].r;
      else{
        sort( eve , eve + E );
        eve[E] = eve[0];
        for( int j = 0 ; j < E ; j ++ ){
          cnt += eve[j].add;
          Area[cnt] += (eve[j].p ^ eve[j + 1].p) * 0.5;
          D theta = eve[j + 1].ang - eve[j].ang;
          if (theta < 0) theta += 2.0 * pi;
          Area[cnt] +=
            (theta - sin(theta)) * c[i].r*c[i].r * 0.5;
}}}}};
```

4.9 Tangent line of two circles

```
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;
}
```

4.10 Minimum distance of two convex

```
double TwoConvexHullMinDis(Pt P[],Pt Q[],int n,int m){
  int mn=0,mx=0; double tmp,ans=1e9;
  for(int i=0;i<n;++i) if(P[i].y<P[mn].y) mn=i;
  for(int i=0;i<m;++i) if(Q[i].y>Q[mx].y) mx=i;
  P[n]=P[0]; Q[m]=Q[0];
  for (int i=0;i<n;++i) {
    while(tmp=((Q[mx+1]-P[mn+1])^(P[mn]-P[mn+1]))>((Q[
        mx]-P[mn+1])^(P[mn]-P[mn+1]))) mx=(mx+1)%m;
    if(tmp<0) // pt to segment distance
      ans=min(ans,dis(Line(P[mn],P[mn+1]),Q[mx]));
    else // segment to segment distance
      ans=min(ans,dis(Line(P[mn],P[mn+1]),Line(Q[mx],Q[
          mx+1])));
    mn=(mn+1)%n;
  }
  return ans;
}
```

4.11 Poly Union

```
struct PY{
  int n; Pt pt[5]; double area;
  Pt& operator[](const int x){ return pt[x]; }
  void init(){ //n,pt[0~n-1] must be filled
    area=pt[n-1]^pt[0];
    for(int i=0;i<n-1;i++) area+=pt[i]^pt[i+1];
    if((area/=2)<0)reverse(pt,pt+n),area=-area;
} };
PY py[500]; pair<double,int> c[5000];
inline double segP(Pt &p,Pt &p1,Pt &p2){
  if(dcmp(p1.x-p2.x)==0) return (p.y-p1.y)/(p2.y-p1.y);
  return (p.x-p1.x)/(p2.x-p1.x);
}
```

```
double polyUnion(int n){ //py[0~n-1] must be filled
  int i,j,ii,jj,ta,tb,r,d; double z,w,s,sum=0,tc,td;
  for(i=0;i<n;i++) py[i][py[i].n]=py[i][0];
  for(i=0;i<n;i++){
    for(ii=0;ii<py[i].n;ii++){
      r=0;
      c[r++]=make_pair(0.0,0); c[r++]=make_pair(1.0,0);
      for(j=0;j<n;j++){
        if(i==j) continue;
        for(jj=0;jj<py[j].n;jj++){
          ta=dcmp(tri(py[i][ii],py[i][ii+1],py[j][jj]))
            ;
          tb=dcmp(tri(py[i][ii],py[i][ii+1],py[j][jj
              +1]));
          if(ta==0 && tb==0){
            if((py[j][jj+1]-py[j][jj])*(py[i][ii+1]-py[
                i][ii])>0&&j<i){
              c[r++]=make_pair(segP(py[j][jj],py[i][ii
                  ],py[i][ii+1]),1);
              c[r++]=make_pair(segP(py[j][jj+1],py[i][
                  ii],py[i][ii+1]),-1);
            }
          }else if(ta>=0 && tb<0){
            tc=tri(py[j][jj],py[j][jj+1],py[i][ii]);
            td=tri(py[j][jj],py[j][jj+1],py[i][ii+1]);
            c[r++]=make_pair(tc/(tc-td),1);
          }else if(ta<0 && tb>=0){
            tc=tri(py[j][jj],py[j][jj+1],py[i][ii]);
            td=tri(py[j][jj],py[j][jj+1],py[i][ii+1]);
            c[r++]=make_pair(tc/(tc-td),-1);
      } } }
      sort(c,c+r);
      z=min(max(c[0].first,0.0),1.0); d=c[0].second; s
          =0;
      for(j=1;j<r;j++){
        w=min(max(c[j].first,0.0),1.0);
        if(!d) s+=w-z;
        d+=c[j].second; z=w;
      }
      sum+=(py[i][ii]^py[i][ii+1])*s;
  } }
  return sum/2;
}
```

4.12 Minkowski sum

```
// P, Q, R(return) are counterclockwise order convex
    polygon
#define all(a) a.begin(),a.end()
vector<Pt> Minkowski(vector<Pt> P, vector<Pt> Q) {
    auto cmp = [&](Pt a, Pt b) {
        return Pt{a.y, a.x} < Pt{b.y, b.x};
    };
    auto reorder = [&](auto &R) {
        rotate(R.begin(), min_element(all(R), cmp), R.
            end());
        R.push_back(R[0]), R.push_back(R[1]);
    };
    const int n = P.size(), m = Q.size();
    reorder(P), reorder(Q);
    vector<Pt> R;
    for (int i = 0, j = 0, s; i < n or j < m; ) {
        R.push_back(P[i] + Q[j]);
        s = dcmp((P[i + 1] - P[i]) ^ (Q[j + 1] - Q[j]))
            ;
        if (s >= 0) i++;
        if (s <= 0) j++;
    }
  rotate(R.begin(), min_element(all(R)), R.end());
    return R;
}
```

4.13 Area of Rectangles

```
struct AreaofRectangles{
#define cl(x) (x<<1)
#define cr(x) (x<<1|1)
    ll n, id, sid;
    pair<ll,ll> tree[MXN<<3];   // count, area
    vector<ll> ind;
    tuple<ll,ll,ll,ll> scan[MXN<<1];
    void pull(int i, int l, int r){
        if(tree[i].first)  tree[i].second = ind[r+1] -
            ind[l];
        else if(l != r){
            int mid = (l+r)>>1;
            tree[i].second = tree[cl(i)].second + tree[
                cr(i)].second;
        }
        else    tree[i].second = 0;
    }
    void upd(int i, int l, int r, int ql, int qr, int v
        ){
        if(ql <= l && r <= qr){
            tree[i].first += v;
            pull(i, l, r); return;
        }
        int mid = (l+r) >> 1;
        if(ql <= mid) upd(cl(i), l, mid, ql, qr, v);
        if(qr > mid) upd(cr(i), mid+1, r, ql, qr, v);
        pull(i, l, r);
    }
    void init(int _n){
        n = _n; id = sid = 0;
        ind.clear(); ind.resize(n<<1);
        fill(tree, tree+(n<<2), make_pair(0, 0));
    }
    void addRectangle(int lx, int ly, int rx, int ry){
        ind[id++] = lx; ind[id++] = rx;
        scan[sid++] = make_tuple(ly, 1, lx, rx);
        scan[sid++] = make_tuple(ry, -1, lx, rx);
    }
    ll solve(){
        sort(ind.begin(), ind.end());
        ind.resize(unique(ind.begin(), ind.end()) - ind
            .begin());
        sort(scan, scan + sid);
        ll area = 0, pre = get<0>(scan[0]);
        for(int i = 0; i < sid; i++){
            auto [x, v, l, r] = scan[i];
            area += tree[1].second * (x-pre);
            upd(1, 0, ind.size()-1, lower_bound(ind.
                begin(), ind.end(), l)-ind.begin(),
                lower_bound(ind.begin(),ind.end(),r)-
                ind.begin()-1, v);
            pre = x;
        }
        return area;
    }rect;
```

4.14 Min dist on Cuboid

```
typedef LL T;
T r;
void turn(T i, T j, T x, T y, T z,
          T x0, T y0, T L, T W, T H) {
  if (z==0) { T R = x*x+y*y; if (R<r) r=R; return; }
  if(i>=0 && i< 2) turn(i+1, j, x0+L+z, y, x0+L-x,
                      x0+L, y0, H, W, L);
  if(j>=0 && j< 2) turn(i, j+1, x, y0+W+z, y0+W-y,
                      x0, y0+W, L, H, W);
  if(i<=0 && i>-2) turn(i-1, j, x0-z, y, x-x0,
                      x0-H, y0, H, W, L);
  if(j<=0 && j>-2) turn(i, j-1, x, y0-z, y-y0,
                      x0, y0-H, L, H, W);
}
T solve(T L, T W, T H,
        T x1, T y1, T z1, T x2, T y2, T z2){
  if( z1!=0 && z1!=H ){
    if( y1==0 || y1==W )
      swap(y1,z1), swap(y2,z2), swap(W,H);
  else swap(x1,z1), swap(x2,z2), swap(L,H);
  }
  if (z1==H) z1=0, z2=H-z2;
  r=INF; turn(0,0,x2-x1,y2-y1,z2,-x1,-y1,L,W,H);
  return r;
}
```

4.15 Distance of Line and Point

```
ld Distance_of_Line_and_Point(Line l, Pt p) {
    ld cross_product = abs((p - l.s) ^ l.v);
    ld line_length = sqrtl(l.v * l.v);
    return cross_product / line_length;
}
```

4.16 Angle of two vector

```
ld Angle_of_two_vector(Pt A, Pt B, Pt O) {
    ld a = (A - O) * (B - O);
    ld b = (A - O) ^ (B - O);
    ld theta = atan2(b, a);
    return theta;
}
```

4.17 極角排序

```
//極角排序
//atan2(y, x) version
// p is reference point
// 180 度開始, 逆時針排序, 剛好在 180 度會排最後
bool cmp(Pt &lhs, Pt rhs) {
    return atan2((lhs - p).y, (lhs - p).x) < atan2((rhs
         - p).y, (rhs - p).x);
}

//cross product version
// p is reference point
// 270 度開始, 逆時針排序, 剛好在 270 度會排最後
bool cmp(const Pt& lhs, const Pt& rhs) {
    if ((lhs < p) ^ (rhs < p)) return (lhs < p) < (rhs
        < p);
    return ((lhs - p) ^ (rhs - p)) > 0;
}
```

4.18 Heart of Triangle

```
Pt inCenter( Pt &A,  Pt &B,  Pt &C) { // 內心
  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);
}
```

5 Graph

5.1 Lowest Common Ancestor $O(lgn)$

```
struct LCA {
  int n, ti, lgN;
  int anc[MXN + 5][__lg(MXN) + 1] = {0};
  int MaxLength[MXN][__lg(MXN) + 1] = {0};
  int time_in[MXN] = {0};
  int time_out[MXN] = {0};
  LCA(int _n, int f):n(_n), ti(0), lgN(__lg(n)) {
    dfs(f, f, 0);
    build();
  }
  void dfs(int now, int f, int len_to_father) {  // dfs
       for anc, time, Lenth
    ti++;
    anc[now][0] = f;
    time_in[now] = ti;
    MaxLength[now][0] = len_to_father;
    for (auto i : graph[now]) {
        if (i.first == f) continue;
        dfs(i.first, now, i.second);
    }
    time_out[now] = ti;
  }
  void build() {  // build anc[][], MaxLength[][]
    for (int i = 1; i <= lgN; ++i) {
      for (int u = 1; u <= n; ++u) {
        anc[u][i] = anc[anc[u][i - 1]][i - 1];
        MaxLength[u][i] = max(MaxLength[u][i - 1],
             MaxLength[anc[u][i - 1]][i - 1]);
        // dis[u][i] += dis[anc[u][i - 1]][i - 1]
        // + dis[u][i - 1];
      }
    }
  }
  bool isAncestor(int x, int y) {
    return time_in[x] <= time_in[y] && time_out[x] >=
        time_out[y];
  }
  int getLCA(int u, int v) {
    if (isAncestor(u, v)) return u;
    if (isAncestor(v, u)) return v;
    for (int i = lgN; i >= 0; --i) {
      if (!isAncestor(anc[u][i], v)) {
        u = anc[u][i];
      }
    }
    return anc[u][0];
  }
  int getMAX(int u, int v) { //獲得路徑上最大邊權
    int lca = getLCA(u, v);
    int maxx = -1;
    for (int i = lgN; i >= 0; --i) {
      // u to lca
      if (!isAncestor(anc[u][i], lca)) {
        maxx = max(maxx, MaxLength[u][i]);
        u = anc[u][i];
      }

      // v to lca
      if (!isAncestor(anc[v][i], lca)) {
        maxx = max(maxx, MaxLength[v][i]);
        v = anc[v][i];
      }
    }
    if (u != lca) maxx = max(maxx, MaxLength[u][0]);
    if (v != lca) maxx = max(maxx, MaxLength[v][0]);
    return maxx;
  }
};
```

5.2 Hamiltonian path $O(n^2 2^n)$

```
//dp[i][j] = 目前在j節點走過{i}節點的最短路徑
for(int i=1 ; i < (1 << n) ; i++ ) {
  for(int j = 1 ; j < n ; j++ ) {
    if(!((1 << j) & i)&&(i&1)) {
      for( int k = 0 ; k < n ; k++ ) {
        if(j == k) continue;
        if( (1<<k)&i ) dp[j][i|(1<<j)]=
          min(dp[j][i|(1<<j)],dp[k][i]+dis[k][j]);
      }
    }
  }
}
```

5.3 MaximumClique 最大團

```
#define N 111
struct MaxClique{ // 0-base
```

```cpp
  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;
```

5.4 MaximalClique 極大團

```cpp
#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;
```

5.5 BCC based on vertex 點雙聯通分量

```cpp
#define PB push_back
#define REP(i, n) for(int i = 0; i < n; i++)
struct BccVertex {
  int n,nScc,step,dfn[MXN],low[MXN];
  vector<int> E[MXN],sccv[MXN];
  int top,stk[MXN];
  void init(int _n) { // 初始化n點
    n = _n; nScc = step = 0;
    for (int i=0; i<n; i++) E[i].clear();
  }
  void addEdge(int u, int v)  // 無向邊
  { E[u].PB(v); E[v].PB(u); }
  void DFS(int u, int f) {
    dfn[u] = low[u] = step++;
    stk[top++] = u;
    for (auto v:E[u]) {
      if (v == f) continue;
      if (dfn[v] == -1) {
        DFS(v,u);
        low[u] = min(low[u], low[v]);
        if (low[v] >= dfn[u]) {
          int z;
          sccv[nScc].clear();
          do {
            z = stk[--top];
            sccv[nScc].PB(z);
          } while (z != v);
          sccv[nScc++].PB(u);
        }
      }else
        low[u] = min(low[u],dfn[v]);
  } }
  vector<vector<int>> solve() { // 回傳(size=2 橋, size
      >2 點雙連通分量)
    vector<vector<int>> res;
    for (int i=0; i<n; i++)
      dfn[i] = low[i] = -1;
    for (int i=0; i<n; i++)
      if (dfn[i] == -1) {
        top = 0;
        DFS(i,i);
      }
    REP(i,nScc) res.PB(sccv[i]);
    return res;
  }
}graph;
```

5.6 Strongly Connected Component 強連通分量

```cpp
#define PB push_back
#define FZ(x) memset(x, 0, sizeof(x)) //fill zero
```

```
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++;
      }
  }
};
```

5.7 ManhattanMST

```
//return {{u,v},w}: u <-> v (w), 需要再手動去重
//need Point definition
vector<pair<pair<int,int>, int>> ManhattanMST(vector<Pt
    > P) {
  vector<int> id(P.size());
  iota(id.begin(),id.end(), 0);
  vector<pair<pair<int,int>, int>> edg;
  for (int k = 0; k < 4; k++) {
    sort(id.begin(),id.end(), [&](int i, int j) {
      return (P[i] - P[j]).x < (P[j] - P[i]).y;
    });
    map<int, int> sweep;
    for (int i : id) {
      auto it = sweep.lower_bound(-P[i].y);
      while (it != sweep.end()) {
        int j = it->second;
        Pt d = P[i] - P[j];
        if (d.y > d.x) break;
        edg.push_back({{i, j},d.x + d.y});
        it = sweep.erase(it);
      }
      sweep[-P[i].y] = i;
    }
    for (Pt &p : P) {
      if (k % 2) p.x = -p.x;
      else swap(p.x, p.y);
    }
  }
  return edg;
}
```

5.8 Min Mean Cycle

```
/* 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;
```

5.9 Directed Graph Min Cost Cycle

```
// works in O(N M)
#define INF 1000000000000000LL
#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;
```

5.10 DominatorTree

```
struct DominatorTree{ // O(N)
#define REP(i,s,e) for(int i=(s);i<=(e);i++)
#define REPD(i,s,e) for(int i=(s);i>=(e);i--)
  int n , m , s;
  vector< int > g[ MAXN ] , pred[ MAXN ];
  vector< int > cov[ MAXN ];
  int dfn[ MAXN ] , nfd[ MAXN ] , ts;
  int par[ MAXN ]; //idom[u] s到u的最後一個必經點
  int sdom[ MAXN ] , idom[ MAXN ];
  int mom[ MAXN ] , mn[ MAXN ];
  inline bool cmp( int u , int v )
  { return dfn[ u ] < dfn[ v ]; }
  int eval( int u ){
    if( mom[ u ] == u ) return u;
    int res = eval( mom[ u ] );
    if(cmp( sdom[ mn[ mom[ u ] ] ] , sdom[ mn[ u ] ] ))
      mn[ u ] = mn[ mom[ u ] ];
    return mom[ u ] = res;
  }
  void init( int _n , int _m , int _s ){
    ts = 0; n = _n; m = _m; s = _s;
    REP( i, 1, n ) g[ i ].clear(), pred[ i ].clear();
  }
  void addEdge( int u , int v ){
    g[ u ].push_back( v );
    pred[ v ].push_back( u );
  }
  void dfs( int u ){
    ts++;
    dfn[ u ] = ts;
    nfd[ ts ] = u;
    for( int v : g[ u ] ) if( dfn[ v ] == 0 ){
      par[ v ] = u;
      dfs( v );
  } }
  void build(){
    REP( i , 1 , n ){
      dfn[ i ] = nfd[ i ] = 0;
      cov[ i ].clear();
      mom[ i ] = mn[ i ] = sdom[ i ] = i;
    }
    dfs( s );
    REPD( i , n , 2 ){
      int u = nfd[ i ];
      if( u == 0 ) continue ;
      for( int v : pred[ u ] ) if( dfn[ v ] ){
        eval( v );
        if( cmp( sdom[ mn[ v ] ] , sdom[ u ] ) )
          sdom[ u ] = sdom[ mn[ v ] ];
      }
      cov[ sdom[ u ] ].push_back( u );
      mom[ u ] = par[ u ];
      for( int w : cov[ par[ u ] ] ){
        eval( w );
        if( cmp( sdom[ mn[ w ] ] , par[ u ] ) )
          idom[ w ] = mn[ w ];
        else idom[ w ] = par[ u ];
      }
      cov[ par[ u ] ].clear();
    }
    REP( i , 2 , n ){
      int u = nfd[ i ];
      if( u == 0 ) continue ;
      if( idom[ u ] != sdom[ u ] )
        idom[ u ] = idom[ idom[ u ] ];
} } }domT;
```

5.11 K-th Shortest Path

```
// 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;
```

5.12 Floryd Warshall

```
for( int k=0 ; k < n ; k++ )
  for( int i=0 ; i < n ; i++ )
    for( int j=0 ; j < n ; j++ )
      if( dis[i][j] > dis[i][k]+dis[k][j] && dis[i][k
          ] < INF && dis[k][j] < INF )
        dis[i][j]=dis[i][k]+dis[k][j];
  for( int i=0 ; i < n ; i++ )
    for( int j=0 ; j < n ; j++ )
      for( int k=0 ; k < n && dis[i][j] != negINF ; k++
          )
        if( dis[k][k] < 0 && dis[i][k] != INF && dis[k
            ][j] != INF )
          dis[i][j]=negINF;
```

5.13 虛樹

```
vector<int> virTree(vector<int> ver, LCA &lca) {
  auto cmp = [&](int u, int v){return time_in[u] <
      time_in[v];};
    sort(ver.begin(),ver.end(),cmp); //用dfn排序
    vector<int>res(ver.begin(),ver.end());
    for(int i = 1; i < ver.size(); i++){
        res.push_back(lca.getLCA(ver[i-1],ver[i]));//把
            LCA丟進虛樹內
    }
    sort(res.begin(),res.end(),cmp); //再用dfn排序
    res.erase(unique(res.begin(),res.end()), res.end())
        ; //去掉重複的點
    return res;
}
```

5.14 Tree Hash

```
map<vector<int>, int> id;
int dfs(int x, int f){
  vector<int> sub;
  for (int v : edge[x]){
    if (v != f)
      sub.push_back(dfs(v, x));
  }
  sort(sub.begin(), sub.end());
  if (!id.count(sub))
    id[sub] = id.size();
  return id[sub];
}
```

5.15 HeavyLightDecomposition

```
// 詢問,修改複雜度 O(log^2 n)
// 1-base

int sz[MXN], dep[MXN], son[MXN], fa[MXN];

// 第一次 dfs
// 找重兒子 需要紀錄當前節點的子樹大小(sz)、深度(dep)、
   重兒子(son)、父節點(fa)
// 沒有子節點 son[x] = 0
void dfs_sz(int x, int f, int d) { //當前節點 x ，父節
   點 f，深度 d
    sz[x] = 1;  dep[x] = d;  fa[x] = f;
    for(int i : edge[x]) {
        if(i == f)    continue;
        dfs_sz(i, x, d+1);
        sz[x] += sz[i];
        if(sz[son[x]] < sz[i])    son[x] = i;
    }
}

// 第二次 dfs
int top[MXN]; // 每個節點所在的鏈的頂端節點
int dfn[MXN]; // 節點編號,編號為在線段樹上的位置
int rnk[MXN]; // 編號為哪個節點
int bottom[MXN]; // 維護每個節點的子樹中最大 dfn 編號
int cnt = 0;
int dfs_hld(int x, int f){
    top[x] = (son[fa[x]] == x ? top[fa[x]] : x);
    rnk[cnt] = x;
    bottom[x] = dfn[x] = cnt++;
    if(son[x])    bottom[x] = max(bottom[x], dfs_hld(
        son[x], x)); // 更新子樹最大編號
    for(int i : edge[x]){
        if(i == f || i == son[x])    continue;
        bottom[x] = max(bottom[x], dfs_hld(i, x)); //
            更新子樹最大編號
    }
    return bottom[x];
}

// 求出 lca
// 不斷跳鏈，直到 u,v 跳到同一條鏈上為止
// 每次跳鏈選所在的鏈頂端深度較深的一端往上跳
int getLca(int u, int v) {
    while(top[u] != top[v]){
      if(dep[top[u]] > dep[top[v]])
          u = fa[top[u]];
      else
          v = fa[top[v]];
    }
    return dep[u] > dep[v] ? v : u;
}

// 路徑權重總和
int query(int u, int v) {
    int ret = 0;
    while(top[u] != top[v]){
        if (dep[top[u]] > dep[top[v]]){
            ret += segtree.query(dfn[top[u]], dfn[u]);
            u = fa[top[u]];
        }
        else{
            ret += segtree.query(dfn[top[v]], dfn[v]);
            v = fa[top[v]];
        }
    }
    // 最後到同一條鏈上
    ret += segtree.query(min(dfn[u], dfn[v]), max(dfn[u
        ], dfn[v]));
    return ret;
}
```

5.16 Graph Thearom

- 差分約束條件:
  約束條件 $V_j - V_i \leq W$ addEdge(V_i, V_j, W) and run bellman-ford or spfa
- 龜兔賽跑演算法:
  開始賽跑，兔子一次走兩格、烏龜一次走一格直到他們相遇停止
  此時讓兔子返回起始點，兩者以相同走一格的速度繼續前進，他們就會在環入口會合
- 2-SAT 條件:
  滿足 $(x_1 ory_1)and(x_2ory_2)and$ ... 對於一個限制 $(xory)$，則加兩條邊 $x \to \neg y$, $y \to \neg x$

6 String

6.1 PalTree $O(n)$

```
// state[i]代表第i個字元為結尾的最長回文編號
// 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;
```

6.2 Longest Increasing Subsequence

```
vector<int> getLIS(vector<int> a){
  vector<int> lis;
  for(int i : a){
    if(lis.empty() || lis.back() < i)    lis.push_back(
        i);
    else      *lower_bound(lis.begin(), lis.end(), i) =
         i;
  }
  return lis;
}
```

6.3 Longest Common Subsequence $O(nlgn)$

```
int LCS(string& s1, string& s2) {
  vector<int> p[128];  // 假設字元範圍為 0 ~ 127
  for (int i = 0; i < s2.size(); ++i) p[s2[i]].
      push_back(i);
  vector<int> v;
  v.push_back(-1);

  for (int i = 0; i < s1.size(); ++i)
    for (int j = p[s1[i]].size() - 1; j >= 0; --j) {
      int n = p[s1[i]][j];

      if (n > v.back())
        v.push_back(n);
      else
        *lower_bound(v.begin(), v.end(), n) = n;
    }
  return v.size() - 1;
};
```

6.4 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]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];
vector<int> KMP(string& t, string& p) {
    vector<int> ret;
    if(p.size() > t.size()) return ret;
    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) {
            ret.push_back(i - p.size() + 1);
            j = failure[j];
        }
    }
    return ret;
}
```

6.5 SAIS $O(n)$

```
/*** SA：將字串的所有後綴排序後的數組  ***/
/* SA[i]儲存排序後第i小的後綴從哪裡開始  */
/**** H[i]為第i小的字串跟第i-1小的LCP  ***/
/**** 註：LCP(Longest Common Prefix)  ****/
/**** ex:S = "babd", SA[0] = 1("abd") ****/
/** SA[1] = 0("babd"), SA[2] = 2("bd")  **/
/*** H[0] = 0, H[1] = 0, H[2] = 1("b") ***/
/* 傳入參數:ip 陣列放字串，len為字串長度 */
/* 需保證ip[len]為0, 且字串裡的元素不為0 */
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)
}
```

6.6 Z Value $O(n)$

```
//z[i] = lcp(s[1...n-1],s[i...n-1])
int z[MAXN];
void Z_value(const string& s) {
  int i, j, left, right, len = s.size();
  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;
}   }   }
```

6.7 Manacher Algorithm $O(n)$

```
// 求以每個字元為中心的最長回文半徑
// 頭尾以及每個字元間都加入一個
// 沒出現過的字元，這邊以'@'為例
// s為傳入的字串，len為字串長度
// z為儲存答案的陣列(有包含'@'要小心)
// ex: s = "abaac" -> "@a@b@a@a@c@"
// z =                [12141232121]
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];
} }
```

6.8 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;
}
```

6.9 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;
}
```

6.10 Hash

```
//字串雜湊前的idx是0-base，雜湊後為1-base
//即區間為 [0,n-1] -> [1,n]
//若要取得區間[L,R]的值則
//H[R] - H[L-1] * p^(R-L+1)
//cmp為比較從i開始長度為len的字串和
//(h[i+len-1] - h[i-1] * qpow(p, len) % modl + modl)
//從j開始長度為len的字串是否相同
#define x first
#define y second
pair<int,int> Hash[MXN];
void build(const string& s){
  pair<int,int> val = make_pair(0,0);
  Hash[0]=val;
  for(int i=1; i<=s.size(); i++){
  val.x = (val.x * P1 + s[i-1]) % MOD;
  val.y = (val.y * P2 + s[i-1]) % MOD;
  Hash[i] = val;
  }
}
bool cmp( int i, int j, int len ) {
    return ((Hash[i+len-1].x-Hash[i-1].x*qpow(P1,len)%
        MOD+MOD)%MOD == (Hash[j+len-1].x-Hash[j-1].x*
        qpow(P1,len)%MOD+MOD)%MOD)
    && ((Hash[i+len-1].y-Hash[i-1].y*qpow(P2,len)%MOD+
        MOD)%MOD == (Hash[j+len-1].y-Hash[j-1].y*qpow(
        P2,len)%MOD+MOD)%MOD);
}
```

7 Data Structure

7.1 Segment tree

```
// !!!注意build()時初始化用的陣列也是1-base
// !!!query(0, 0) 會報錯
#define cl(x) (x*2)
#define cr(x) (x*2+1)

struct segmentTree {
    int n;
    vector<int> seg, tag, cov;
    segmentTree(int _n): n(_n) {
        seg = tag = cov = vector<int>(n * 4, 0);
    }
    void push(int i, int L, int R) {
        if(cov[i]) {
            seg[i] = cov[i] * (R - L + 1);
            if(L < R) {
                cov[cl(i)] = cov[cr(i)] = cov[i];
                tag[cl(i)] = tag[cr(i)] = 0;
            }
            cov[i] = 0;
        }
        if(tag[i]) {
            seg[i] += tag[i] * (R - L + 1);
            if(L < R) {
                tag[cl(i)] += tag[i];
                tag[cr(i)] += tag[i];
            }
            tag[i] = 0;
        }
    }
    void pull(int i, int L, int R) {
        if(L >= R) return;
        int mid = L + R >> 1;
        push(cl(i), L, mid);
        push(cr(i), mid + 1, R);
        seg[i] = seg[cl(i)] + seg[cr(i)];
    }
    void build(vector<int>& arr, int i = 1, int L = 1,
      int R = -1) {
        if(R == -1) R = n;
        if(L == R) return void(seg[i] = arr[L]);
        int mid = L + R >> 1;
        build(arr, cl(i), L, mid);
        build(arr, cr(i), mid + 1, R);
        pull(i, L, R);
    }
    int query(int rL, int rR, int i = 1, int L = 1, int
      R = -1) {
        if(R == -1) R = n;
        push(i, L, R);
        if(rL <= L && R <= rR) return seg[i];
        int mid = L + R >> 1, ret = 0;
        if(rL <= mid) ret += query(rL, rR, cl(i), L,
          mid);
        if(mid < rR ) ret += query(rL, rR, cr(i), mid +
          1, R);
        return ret;
    }
    void update(int rL, int rR, int val, int i = 1, int
      L = 1, int R = -1) {
        if(R == -1) R = n;
        push(i, L, R);
        if(rL <= L && R <= rR) return void(tag[i] = val
          );
        int mid = L + R >> 1;
        if(rL <= mid) update(rL, rR, val, cl(i), L, mid
          );
        if(mid < rR ) update(rL, rR, val, cr(i), mid +
          1, R);
        pull(i, L, R);
    }
    void cover(int rL, int rR, int val, int i = 1, int
      L = 1, int R = -1) {
        if(R == -1) R = n;
        push(i, L, R);
        if(rL <= L && R <= rR) return void(cov[i] = val
          );
        int mid = L + R >> 1;
        if(rL <= mid) cover(rL, rR, val, cl(i), L, mid)
          ;
        if(mid < rR ) cover(rL, rR, val, cr(i), mid +
          1, R);
        pull(i, L, R);
    }
};
/*  Test Case:
    4
    1 2 3 4
    5
    2 1 3
    1 1 3 1
    2 1 3
    1 1 4 1
    2 1 4
*/
```

7.2 持久化 SMT

```
struct node{
  node *l, *r;
  int val;
};

vector<node *> ver;
int arr[MXN] = {0};

//0-base
struct SegmentTree{
  int n;
  node *root;
  void build(int _n){
    n = _n;
    root = build(0, n-1);
  }
  node* build(int L, int R){
    node *x = new node();
    if(L == R){ x->val = arr[L]; return x;}
    int mid = (L+R)/2;
    x->l = build(L, mid);
    x->r  = build(mid + 1, R);
    x->val = x->l->val + x->r->val;
    return x;
  }
  int query(node *ro, int L, int R){return query(ro, 0,
      n-1, L, R);}
  int query(int L, int R){return query(root, 0, n-1, L,
      R);}
  int query(node *x, int L, int R, int recL, int recR){
    if(recL <= L && R <= recR) return x->val;
    int mid = (L+R)/2, res = 0;
    if(recL <= mid) res += query(x->l, L, mid, recL,
        recR);
    if(mid  < recR) res += query(x->r, mid+1, R, recL,
        recR);
    return res;
  }
  void update(int pos, int v){update(root, 0, n-1, pos,
      v);}
  void update(node *x, int L, int R, int pos, int v){
    if(L == R){ x->val = v; arr[L] = v; return;}
    int mid = (L+R)/2;
    if(pos <= mid) update(x->l, L, mid, pos, v);
    else           update(x->r, mid+1, R, pos, v);
    x->val = x->l->val + x->r->val;
  }
  node *update_ver(node *pre, int l, int r, int pos,
      int v){
    node *x = new node();    //當前位置建立新節點
    if(l == r){
      x->val = v;
      return x;
    }
    int mid = (l+r)>>1;
    if(pos <= mid){ //更新左邊
      x->l = update_ver(pre->l, l, mid, pos, v); //左邊
          節點連向新節點
      x->r = pre->r; //右邊連到原本的右邊
```

```
    }
    else{ //更新右邊
      x->l = pre->l; //左邊連到原本的左邊
      x->r = update_ver(pre->r, mid+1, r, pos, v);  //
          右邊節點連向新節點
    }
    x->val = x->l->val + x->r->val;
    return x;
}} seg;

void add_ver(int x,int v){    //修改位置 x 的值為 v
    ver.push_back(seg.update_ver(ver.back(), 0, seg.n
        -1, x, v));
}
```

7.3 持久化並查集

```
struct DSU {
    int n;
    vector<int> fa, sz;
    vector<tuple<int, int, int, int>> ver;
    DSU(int _n): n(_n), fa(n), sz(n, 1) {
        iota(fa.begin(), fa.end(), 0);
    }
    int find(int x) {
        return fa[x] == x ? x : find(fa[x]);
    }
    void merge(int x, int y) {
        x = find(x), y = find(y);
        if(sz[x] < sz[y]) swap(x, y);
        ver.push_back({x, sz[x], y, fa[y]});
        if(x == y) return;
        sz[x] += sz[y];
        fa[y] = x;
    }
    void undo() {
        if(ver.empty()) return;
        auto [x, szx, y, fy] = ver.back();
        ver.pop_back();
        sz[x] = szx;
        fa[y] = fy;
    }
};
```

7.4 Trie

```
struct trie{
  trie *nxt[26];
  int cnt;     //紀錄有多少個字串以此節點結尾
  int sz;      //有多少字串的前綴包括此節點
  trie():cnt(0),sz(0){
      memset(nxt,0,sizeof(nxt));
  }
};

trie *root = new trie(); //創建新的字典樹

void insert(string& s){
  trie *now = root;  // 每次從根結點出發
  for(auto i:s){
    now->sz++;
    if(now->nxt[i-'a'] == NULL){
      now->nxt[i-'a'] = new trie();
    }
    now = now->nxt[i-'a'];  //走到下一個字母
  }
  now->cnt++; now->sz++;
}

int query_prefix(string& s){  //查詢有多少前綴為 s
  trie *now = root;    // 每次從根結點出發
  for(auto i:s){
    if(now->nxt[i-'a'] == NULL){
      return 0;
    }
    now = now->nxt[i-'a'];
  }
  return now->sz;
}

int query_count(string& s){  //查詢字串 s 出現次數
  trie *now = root;    // 每次從根結點出發
  for(auto i:s){
    if(now->nxt[i-'a'] == NULL){
      return 0;
    }
    now = now->nxt[i-'a'];
  }
  return now->cnt;
}
```

7.5 Treap (interval reverse)

```
//拆出[a,b]區間就如同下面所展示先使用splitByTh()拆出
//左右,再把左區間拆成l, m最後merge()回去
//反轉區間時又記得使用^=可以直接反轉01

//treap拆區間時從後面拆是因為這樣[a,b]的關係
//不用重新考慮．要是先拆前面b的位置會變成b-a+1
//0-base
//splitByTh(root,a-1,l,m);
//splitByTh(m,b-a+1,m,r);

mt19937 gen(chrono::steady_clock::now().
    time_since_epoch().count());
struct Treap {
  int key, pri, sz, tag, sum;
  Treap *L, *R;
  Treap( int val ) {
    sum=key=val, pri=gen(), sz=1, tag=0;
    L=R=NULL;
};};
int Size( Treap *a ) { return !a?0:a->sz;}
void pull( Treap *a ) {
  a->sz=Size(a->L)+Size(a->R)+1;
  a->sum=a->key;
  if( a->L ) a->sum+=a->L->sum;
  if( a->R ) a->sum+=a->R->sum;
}
void push( Treap *a ) {
  if( a && a->tag ) {
    swap(a->L,a->R);
    if( a->L ) a->L->tag^=1;
    if( a->R ) a->R->tag^=1;
    a->tag=0;
}}
Treap *merge(Treap *a, Treap *b) {
  if( !a || !b ) return a?a:b;
  push(a), push(b);
  if( a->pri > b->pri ) {
    a->R=merge(a->R,b);
    pull(a); return a;
  }
  b->L=merge(a,b->L);
  pull(b); return b;
}
void print(Treap *a) {
  if( !a ) return;
  push(a);
  print(a->L);
  cout.put(a->key);
  print(a->R);
}
Treap *buildTreap( int n, string& str ) {
  Treap *root=NULL;
  for( int i=0 ; i < n ; i++ )
    root=merge(root,new Treap(str[i]));
  return root;
}
void splitbyk( Treap *x, int k, Treap *&a, Treap *&b )
    {
  if(!x) a=b=NULL;
  else if( x->key <= k ) {
    a=x;
    splitbyk(x->R,k,a->R,b);
    pull(a);
  }
```

```
  else {
    b=x;
    splitbyk(x->L,k,a,b->L);
    pull(b);
  }
}
void splitByTh( Treap *x, int k, Treap *&a, Treap *&b )
     {
  if( !x ) { a=b=NULL; return; }
  push(x);
  if( Size(x->L)+1 <= k ) {
    a=x;
    splitByTh(x->R,k-Size(x->L)-1,a->R,b);
    pull(a);
  }
  else {
    b=x;
    splitByTh(x->L,k,a,b->L);
    pull(b);
  }
}
signed main() {
  string str;
  int n, m;
  cin>>n>>m>>str;
  Treap *root;
  root=buildTreap(n,str);
  for( int i=0 ; i < m ; i++ ) {
    int a, b;
    cin>>a>>b;
    Treap *l, *m, *r;
    splitByTh(root,b,l,r);
    splitByTh(l,a-1,l,m);
    m->tag^=1;
    root=merge(l,merge(m,r));
  }
  print(root);
}
```

7.6 Treap (interval erase)

```
//區間移除使用bitset維護區間值
mt19937 gen(chrono::steady_clock::now().
    time_since_epoch().count());
struct Treap {
char key;
int pri, sz;
bitset<128> tag;
  Treap *L, *R;
  Treap( char val ) {
    key=val, pri=gen(), sz=1;
    L=R=NULL;
    tag.set(key);
}; };
int Size( Treap *a ) { return !a?0:a->sz;}
void pull( Treap *a ) {
  if( !a ) return;
  a->sz=Size(a->L)+Size(a->R)+1;
  a->tag=a->tag.reset();
  a->tag=a->tag.set(a->key);
  if( a->L ) a->tag|=a->L->tag;
  if( a->R ) a->tag|=a->R->tag;
}
Treap *merge( Treap *a, Treap *b ) {
  if( !a || !b ) return a?a:b;
  if( a->pri > b->pri ) {
    a->R=merge(a->R,b);
    pull(a);
    return a;
  }
  b->L=merge(a,b->L);
  pull(b);
  return b;
}
Treap *buildTreap( int n, string& str ) {
  Treap *root=NULL;
  for( int i=0 ; i < n ; i++ )
    root=merge(root,new Treap(str[i]));
  return root;
}
void print( Treap *a ) {
  if( !a ) return;
  print(a->L);
  cout.put(a->key);
  print(a->R);
}
void splitByTh( Treap *x, int k, Treap *&a, Treap *&b )
     {
  if( !x ) { a=b=NULL; return; }
  if( Size(x->L)+1 <= k ) {
    a=x;
    splitByTh(x->R,k-Size(x->L)-1,a->R,b);
    pull(a);
  }
  else {
    b=x;
    splitByTh(x->L,k,a,b->L);
    pull(b);
  }
}
void erase( Treap *&x, char ch ) {
  if( !x || !x->tag.test(ch) ) return;
  erase(x->L,ch);
  erase(x->R,ch);
  if( x->key == ch ) {
    Treap *l=x->L, *r=x->R;
    x=NULL;
    x=merge(l,r);
  }
  pull(x);
}
signed main() {
  string str;
  int n, m;
  cin>>n>>m>>str;
  Treap *root;
  root=buildTreap(n,str);
  for( int i=0 ; i < m ; i++ ) {
    char c;
    int a, b;
    cin>>a>>b>>c;
    Treap *l, *m, *r;
    if( !root || !root->tag.test(c) ) continue;
    splitByTh(root,b,l,r);
    splitByTh(l,a-1,l,m);
    if( m || !m->tag.test(c) ) erase(m,c);
    root=merge(l,merge(m,r));
  }
  print(root);
}
```

7.7 BIT

```
#define lowbit(x) (x&-x)
struct BIT {
    int n;
    vector<int> bit;
    BIT(int _n):n(_n), bit(n + 1) {}
    void update(int x, int val) {
        for(; x <= n; x += lowbit(x)) bit[x] += val;
    }
    void update(int L, int R, int val) {
        update(L, val), update(R + 1, -val);
    }
    int query(int x) {
        int res = 0;
        for(; x; x -= lowbit(x)) res += bit[x];
        return res;
    }
    int query(int L, int R) {
        return query(R) - query(L - 1);
    }
};
```

7.8 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;
tree<int,null_type,less_equal<int>,rb_tree_tag,
    tree_order_statistics_node_update> mt_t;
#include <ext/pb_ds/assoc_container.hpp>
typedef cc_hash_table<int,int> umap_t;
// gp_hash_table<int, int>
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);
  // if we want to delete less_equal tag tree
  mt_t.erase(mt_t.find_by_order(mt_t.order_of_key(val))
      );

  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 );
}
```

8 Others

8.1 SOS dp

```
for(int i = 0; i<(1<<N); ++i)
  F[i] = A[i];
for(int i = 0;i < N; ++i) for(int mask = 0; mask < (1<<
    N); ++mask){
  if(mask & (1<<i))
    F[mask] += F[mask^(1<<i)];
}
```

8.2 De Brujin sequence

```
// return cyclic array of length k^n such that every
// array of length n using 0~k-1 appears as a subarray.
vector<int> DeBruijn(int k,int n){
  if(k==1) return {0};
  vector<int> aux(k*n),res;
  function<void(int,int)> f=[&](int t,int p)->void{
    if(t>n){ if(n%p==0)
        for(int i=1;i<=p;++i) res.push_back(aux[i]);
    }else{
      aux[t]=aux[t-p]; f(t+1,p);
      for(aux[t]=aux[t-p]+1;aux[t]<k;++aux[t]) f(t+1,t)
          ;
    }
  };
  f(1,1); return res;
}
```

8.3 CDQ 分治

```
//cdq分治使用的結構u, v, w為排序物的三個維度
//ans記錄了有幾項三維都小於等於自己
//cnt記錄了相同物有幾個，在使用cdq之前必先去重，
//並且將相同元素紀錄至cnt中，可使用map來做到這步
//cdq使用的BIT就是普通求和的BIT，大小就開維度的
//值域範圍，若值域大於2e6則要先進行離散化
struct triple {int u, v, w, ans, cnt;};
BIT *bt;
void cdq(int L, int R, vector<triple>& arr) {
  if(R - L <= 1) return;
  int mid = L + R >> 1;
  vector<triple> temp;
  cdq(L, mid, arr), cdq(mid, R, arr);
  for(int i = L, j = mid; i < mid || j < R;) {
    for(; i < mid && (j >= R || arr[i].v <= arr[j].v);
        i++) {
      bt->update(arr[i].w, arr[i].cnt);
      temp.push_back(arr[i]);
    }
    if(j < R) {
      arr[j].ans += bt->query(arr[j].w);
      temp.push_back(arr[j]);
      j++;
    }
  }
  for(int i = L; i < mid; i++)
    bt->update(arr[i].w, -arr[i].cnt);
  copy(temp.begin(), temp.end(), arr.begin() + L);
}
signed main()
{
  // n 個數 k 值域範圍
  int n, k;
  cin >> n >> k;
  map<tuple<int, int, int>, int> mp;
  vector<int> res(n, 0);
  vector<triple> arr;
  bt = new BIT(k + 1);
  for(int i = 0; i < n; i++) {
      int x, y, z;
      cin >> x >> y >> z;
      mp[{x, y, z}]++;
  }
  for(auto t : mp)
    arr.push_back({get<0>(t.first), get<1>(t.first),
        get<2>(t.first), 0, t.second});
  cdq(0, arr.size(), arr);
  for(auto &[x,y,z,a,b] : arr) res[a + b - 1] += b;
  for(int i : res) cout << i << '\n';
}
```

8.4 3D LIS

```
#define lowbit(x) (x&-x)
const int MAXN=1e5+5;
struct BIT {
  int n;
  vector<int> bit;
  BIT( int _n ):n(_n), bit(_n+1,0) {}
  int query( int x ) {
    int res=0;
    for(; x > 0 ; x-=lowbit(x) ) res=max(res,bit[x]);
    return res;
  }
  void update( int x, int val ) {
    for(; x <= n ; x+=lowbit(x) ) {
      if( val < 0 ) bit[x]=0;
      else bit[x]=max(bit[x],val);
    }
  }
}bt(MAXN);
struct triple {
  int u, v, w, ans, cnt;
  bool operator<( triple b ) { return u<b.u; }
};
bool cmp( triple a, triple b ) {return a.v<b.v;}
void cdq( int L, int R, vector<triple>& arr ) {
  if( R-L <= 1 ) return;
  int mid=L+R>>1;
  cdq(L,mid,arr);
  sort(arr.begin()+L,arr.begin()+mid,cmp);
```

```
  sort(arr.begin()+mid,arr.begin()+R,cmp);
  for( int i=L, j=mid ; i < mid || j < R ; ) {
    for(; i < mid && ( j >= R || arr[i].v < arr[j].v )
        ; i++ ) bt.update(arr[i].w,arr[i].ans);
    if( j < R ) {
      arr[j].ans=max(bt.query(arr[j].w-1)+1,arr[j].ans)
          ;
      j++;
    }
  }
  for( int i=L ; i < mid ; i++ ) bt.update(arr[i].w,-1)
      ;
  sort(arr.begin()+L,arr.begin()+R);
  cdq(mid,R,arr);
}
signed main()
{
  ios_base::sync_with_stdio(0); cin.tie(0); cout.tie(0)
      ;
  int n, res=0;
  cin>>n;
  vector<int> ls;
  vector<triple> arr;
  for( int i=0 ; i < n ; i++ ) {
    int a, b;
    cin>>a>>b;
    arr.push_back({i,a,b,1,1});//{第一維,第二維,第三維,
        答案,數量}
    ls.push_back(b);
  }
  sort(ls.begin(),ls.end());
  ls.resize(unique(ls.begin(),ls.end())-ls.begin());
  for( auto &t : arr ) t.w=lower_bound(ls.begin(),ls.
      end(),t.w)-ls.begin()+1;
  n=arr.size();
  cdq(0,n,arr);
  for( int i=0 ; i < n ; i++ ) res=max(res,arr[i].ans);
  cout<<res<<'\n';
}
```

8.5 Ternary Search

```
while(L <= R) {
    int ml = L + (R - L) / 3, mr = R - (R - L) / 3;
    if(L == R) return L;
    else if( checker(ml) < checker(mr) ) L = ml + 1;
    else R = mr - 1;
}
```

8.6 Max Subrectangle

```
const int N = 1e5+5;
int n, a[N],l[N], r[N];
long long ans;
int main() {
  while (cin>>n) {
    ans = 0;
    for (int i = 1; i <= n; i++) cin>>a[i], l[i] = r[i]
        = i;
    for (int i = 1; i <= n; i++)
      while (l[i] > 1 && a[i] <= a[l[i] - 1]) l[i] = l[
          l[i] - 1];
    for (int i = n; i >= 1; i--)
      while (r[i] < n && a[i] <= a[r[i] + 1]) r[i] = r[
          r[i] + 1];
    for (int i = 1; i <= n; i++)
      ans = max(ans, (long long)(r[i] - l[i] + 1) * a[i
          ]);
    cout<<ans<<"\n";
  }
}
```

8.7 Maximal Rectangle

```
const int MXN = 300;
int maximalRectangle(vector<vector<char>>& matrix) {
    int a[MXN]{}, l[MXN]{}, r[MXN]{};
    int n = matrix.size(), m = matrix[0].size(), ans =
        0;
    for(int i = 1; i <= n; i++) {
        for(int j = 1; j <= m; j++) l[j] = r[j] = j;
        char c;
        for(int j = 1; j <= m; j++) { //對每一個直行做
            統計，若是上一個a[j]也是1則會變成2
            c = matrix[i - 1][j - 1];
            if (c == '1') a[j]++;
            else if (c == '0') a[j] = 0;
        }
        for(int j = 1; j <= m; j++) while(l[j] != 1 &&
            a[l[j] - 1] >= a[j]) l[j] = l[l[j] - 1];
        for(int j = m; j >= 1; j--) while(r[j] != m &&
            a[r[j] + 1] >= a[j]) r[j] = r[r[j] + 1];
        for(int j = 1; j <= m; j++) ans = max(ans, (r[j
            ] - l[j] + 1) * a[j]);
    }
    return ans;
}
```

8.8 p-Median

```
for(int i = 1; i <= n; i++) {
  for(int j = i; j <= n; j++) {
    dis[i][j] = 0;
    for(int k = i; k <= j; k++) dis[i][j] += abs(arr[k]
        - arr[i + j >> 1]);
    if(i == 1) dp[i][j] = dis[i][j];
  }
}
for(int i = 2; i <= p; i++) {
  for(int j = i; j <= n; j++) {
    dp[i][j] = INF;
    for(int k = i; k <= j; k++) {
      if(dp[i][j] > dp[i - 1][k - 1] + dis[k][j]) {
        dp[i][j] = dp[i - 1][k - 1] + dis[k][j];
        fa[i][j] = k - 1;
      }
    }
  }
}
```

8.9 Tree Knapsack

```
int dfs(int u) {
  int p = 1;
  dp[u][1] = s[u];
  for (int v : edge[u]) {
    int siz = dfs(v);
    for (int i = min(p, m + 1); i; i--)
      for (int j = 1; j <= siz && i + j <= m + 1; j++)
        dp[u][i + j] = max(dp[u][i + j], dp[u][i] + dp[
            v][j]);
    p += siz;
  }
  return p;
}
```

8.10 AC-Automaton

```
// 1-based
// n is the number of patterns
struct Automaton {
    static const int MXN = 1e6;
    int n, cnt, vis[MXN], rev[MXN], indeg[MXN], ans[MXN
        ];
    queue<int> q;
    struct trie_node {
        vector<int> son;
        int fail, flag, ans;
        trie_node(): son(27), fail(0), flag(0) {}
```

```cpp
    } trie[MXN];
    void init(int _n) {
        n = _n, cnt = 1;
        for (int i = 1; i <= n; i++) vis[i] = 0;
    }
    // insert a string s with number num
    // num is the index of the pattern
    void insert(string s, int num) {
        int u = 1, len = s.size();
        for (int i = 0; i < len; i++) {
            int v = s[i] - 'a';
            if (!trie[u].son[v]) trie[u].son[v] = ++cnt
                ;
            u = trie[u].son[v];
        }
        if (!trie[u].flag) trie[u].flag = num;
        rev[num] = trie[u].flag;
    }
    void getfail() {
        for (int i = 0; i < 26; i++) trie[0].son[i] =
            1;
        q.push(1);
        trie[1].fail = 0;
        while (q.size()) {
            int u = q.front(); q.pop();
            int Fail = trie[u].fail;
            for (int i = 0; i < 26; i++) {
                int v = trie[u].son[i];
                if (!v) {
                    trie[u].son[i] = trie[Fail].son[i];
                    continue;
                }
                trie[v].fail = trie[Fail].son[i];
                indeg[trie[Fail].son[i]]++;
                q.push(v);
            }
        }
    }
    void topu() {
        for (int i = 1; i <= cnt; i++)
            if (!indeg[i]) q.push(i);
        while (q.size()) {
            int fr = q.front(); q.pop();
            vis[trie[fr].flag] = trie[fr].ans;
            int u = trie[fr].fail;
            trie[u].ans += trie[fr].ans;
            if (!--indeg[u]) q.push(u);
        }
    }
    void query(string &s) {
        int u = 1, len = s.size();
        for (int i = 0; i < len; i++) u = trie[u].son[s
            [i] - 'a'], trie[u].ans++;
    }
    void solve(string &s) {
        getfail();
        query(s);
        topu();
        for (int i = 1; i <= n; i++) ans[i] = vis[rev[i
            ]];
    }
} AC;
```