Contents

1 Pre-work

1.1 Linux

```
ls        //show the file
mkdir test  //make a directory - test
cd test    //go to test directory
cd ..     //back to last directory
mv test test2 //if test2 exist,move test to test2,else
    rename test to test2
rm test   //delete test

vim a.cpp  //write code
{
  i     //insert mode(type)
  esc   //normal mode(command)
  :wq   //save and leave
  normal + gg + G + d //delete all
  normal +the line   your mouse stop+d  //delete the
      line
  normal + gg + V + G + " = " //indent(tab)
  normal + u  //return to last step
  the leter your mouse stop + s //delete the letter and
       start insert mode
  !!DO NOT press ctrl + s!! //if so ,use ctrl+q
}
g++ a.cpp -o a.out        //compile
./a.out   //run
```

1.2 Python

```
{
  cat > a.in   //copy input to a.in,use ctrl+d to
      finish
  . / a.out < a.in        //enter input in a.in and run
  . / a.out<a.in>a.o  //enter output to a.o
  cat a.o     //show up the content in a.o
}
{                  //should use python
  from random import*
    randint(a, b)     //produce a random int in a-b
    random(a, b)     //maybe produce a float
}
ctrl + c    //force to stop

python3 test.py     //run
diff a.out b.out  //look difference between two file
{
  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
}     //match

a,b = b,a #swap

from sys import stdin //讀整行
n = stdin.readline()

a,b = map(int,input().split().) //映射輸入

s.strip() //刪頭尾空格

i='A' //字元轉數字
i=ord(i) #65
i=ord('B') - ord('A')

x=bin(x) //2(以0b開頭)
x=oct(x) //8(以0o開頭)
x=hex(x) //16(以0x開頭)
x=int(x,2) //2轉10
x=int(x,8) //8轉10
x=int(x,16) //16轉10

for s in stdin:  //連續輸入

print("%g" %(R)) //去除小數的0 輸出法
```

1.3 Header

```
#include<bits/stdc++.h>
#define PI acos(-1.0)
#define e 2.7182818
#define LL long long
#define lowbit(x) (x&-x)
using namespace std;
ios::sync_with_stdio(0);
cin.tie(0);  cout.tie(0);
cout << fixed << setprecision(10) << ;
```

1.4 Int

```
int -2,147,483,648 ~ 2,147,483,647

unsigned 0 ~ 4,294,967,295

long long -9,223,372,036,854,775,808 ~
    9,223,372,036,854,775,807

unsigned long long 0 ~ 18,446,744,073,709,551,615
```

2 String

2.1 String to int

```
int num = atoi(str.c_str());
long long num = atoll(str.c_str());
```

2.2 Int to string

```
to_string(num);
```

2.3 Lexicographically minimum string rotation

```
#define rep(b,a,N) for(int b=a;b<N;b++)
int minRotation(string s) { //Lexicographically minimum
     string rotation
  //rotate後使字典序最小
  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;
}
```

2.4 Hash

```
/*if we want to get Hash of str[l,r]
     H(l,r)=[H(r)-H(l-1)*p^(r-l+1)]%mod
     ex:H(1,3)=H(0,3)-H(0,0)*p^3 %mod
*/
const ll P = 75577;
const ll MOD = 998244353;
ll Hash[MXN];    //Hash[i] is the hash value of str[0,i
    ]
void build(const string& s) {
    int val = 0;
    for (int i = 0; i < s.size(); i++) {
        val = (val * P + s[i]) % MOD;
        Hash[i] = val;
    }
}
/*prime:271,601,977,2221,5113,9151,12251,83357,115249,
859433,1398296,6972503,16769023,43112609,73939133*/
```

2.5 Trie

```
struct trie {
    trie* nxt[26];
    int cnt;    //how many string end at this node
    int sz;     //how many string's prefix include this
         node
    trie() :cnt(0), sz(0) {
        memset(nxt, 0, sizeof(nxt));
    }
};
//create new trie
trie* root = new trie();
// O(|s|)
void insert(string& s) {
    trie* now = root;  //start at root everytime
    for (auto i : s) {
        now->sz++;
        if (now->nxt[i - 'a'] == NULL) {
            now->nxt[i - 'a'] = new trie();
        }
        now = now->nxt[i - 'a'];  //go to next letter
    }
    now->cnt++;
    now->sz++;
}
// O(|s|)
int query_prefix(string& s) {  //search how many prefix
     is s
    trie* now = root;     //start at root everytime
    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) {  //search how many times s
     appear
    trie* now = root;     //start at root everytime
    for (auto i : s) {
        if (now->nxt[i - 'a'] == NULL) {
            return 0;
        }
        now = now->nxt[i - 'a'];
    }
    return now->cnt;
}
```

2.6 Bitwise Trie

```
struct trie {
    trie* nxt[2];
    int cnt;    //how many number end at this node
    int sz;     //how many number's prefix include this
         node
    trie() :cnt(0), sz(0) {
        memset(nxt, 0, sizeof(nxt));
    }
};
//create new trie
trie* root = new trie();
void insert(int x){
    trie *now = root;  //start at root everytime
    for(int i=30;i>=0;i--){
        now->sz++;
        if(now->nxt[x>>i&1] == NULL){
            now->nxt[x>>i&1] = new trie();
        }
        now = now->nxt[x>>i&1]; / go to next
    }
    now->cnt++;
    now->sz++;
}
```

2.7 Double Hash

```
const ll P1 = 75577;
const ll P2 = 12721;
const ll MOD = 998244353;
pair<ll, ll> Hash[MXN];
void build(const string& s) {
    pair<ll, ll> val = make_pair(0, 0);
    for (int i = 0; i < s.size(); i++) {
        val.first = (val.first * P1 + s[i]) % MOD;
        val.second = (val.second * P2 + s[i]) % MOD;
        Hash[i] = val;
    }
}
```

2.8 Suffix Array

```
//S = “babd”
//而 SA[i] 存的就是第i小的後綴是第幾個字元開始的字串

//SA[0] = 1 ( “abd” )
//SA[1] = 0 ( “babd” )
//SA[2] = 2 ( “bd” )
//SA[3] = 3 ( “d” )

//Height 數組
//裡面儲存的值為第 i 小的字串
//跟第 i-1 小的字串共同前綴長度(LCP)

//H[0] = 0 (第0個字串沒有前一個)
//H[1] = 0
//H[2] = 1 ( “b” )

//應用:LCS最長共同子字串
//S = A + '@' + B
//就看i , i+1 字串為不同字串, 取 H[i+1] 的 max

//應用:相異字串數量
//全部子字串數量 - H[0~s.size()]即為答案
const int N = 300010; //字串的2倍大小
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 ];
//ip[i]=(int)str[i]
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)
}
int ip[N];
for (int i = 0; i < str.size(); i++) {
    ip[i] = int(str[i]);
  }
suffix_array(ip,str.size());
```

2.9 KMP

```
/*len - failure[k]:
在k結尾的情況下，這個子字串可以由開頭
長度為(len - failure[k])的部分重複出現來表達
failure[k] :
    failure[k]為次長相同前綴後綴
    如果我們不只想求最多，而且以0 - base做為考量
    ，那可能的長度由大到小會是
    failuer[k]、failure[failuer[k] - 1]
    、failure[failure[failuer[k] - 1] - 1]..
    直到有值為0為止*/
int failure[MXN];
void KMP(string& t, string& p)
{
    if (p.size() > t.size()) return;
    for (int i = 1, j = failure[0] = -1; i < p.size();
        ++i)
    {
        while (j >= 0 && p[j + 1] != p[i])
            j = failure[j];
        if (p[j + 1] == p[i]) j++;
        failure[i] = j;
    }
    for (int i = 0, j = -1; i < t.size(); ++i)
    {
        while (j >= 0 && p[j + 1] != t[i])
            j = failure[j];
        if (p[j + 1] == t[i]) j++;
        if (j == p.size() - 1)
        {
            cout << i - p.size() + 1 << " ";
            j = failure[j];
        }
    }
}
```

2.10 Manacher

```
void z_value_pal(string s, int len, int *z) { //z陣列儲
    存以每個字元為中心的回文半徑+1 s的size要是原字串
    *2+1
  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];
  }
}
//s = “@a@b@a@a@c@”
//z = [12141232121]
//原字串為a,要傳a+a+'0'進去,不然會re.
```

2.11 Palindromic-Tree 回文樹

```
// len[state[i]]是對應的回文長度  len[fail[i]]
// num[state[i]]是有幾個回文後綴
// cnt[state[i]]是這個回文子字串在整個字串中的出現次數
// fail[i]是他長度次長的回文後綴，aba的fail是a
// state[i] 以i為結尾的最長回文
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 string _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;
```

3 STL

3.1 Vector

```
vec.insert(vec.begin()+i,a);  //insert a in vec[i]
//erase vec[i]~vec[j-1]
vec.erase(vec.begin()+i,vec.begin()+j);
{
  vector<int>::iterator low, up;
  low = std::lower_bound(v.begin(), v.end(), 20);
  up = std::upper_bound(v.begin(), v.end(), 20);
}
//erase repeat number
{
  sort(vec.begin(), vec.end());
  vec.erase(unique(vec.begin(), vec.end()), iter, vec.
      end())
}
```

3.2 Set

```
//sort in small to big,O(lg n)
*st.begin()   //get first value
*st.rbegin()  //get last value
st.count(x)   //return 0 represent not found
st.find(x)    //return st.end() represent not found
st.erase(x)   //x can be value or iterator
st.clear()  st.empty()  st.size()
st.insert(st2.begin(),st2.end())  //insert st2[begin,
    end) to st
/*dont have operator[]*/

//customize compare
struct cmp {
  bool operator()(int lhs, int rhs)/*const*/{
    return lhs > rhs;
  }
}
set<int, cmp> st;
```

3.3 Map

```
//almost same to set,O(lg n)
map<key,value>
mp[i]   //can get the value ,whitch set dont have
mp[i]=2 //if map dont have key=i,then it will insert({i
    ,2})
mp.insert({i,2})  //if key=i is already exist,it will
    fail
mp.count()  //return 0 represent not found
mp.find()   //return end() represent not found
mp.erase(mp.begin(),mp.end())
```

3.4 Priority queue

```
priority_queue<int>  //number from big to small
 //number from small to big
priority_queue<int, vector<int>, greater<int> >num;
//function
empty  size  push  pop  top
```

3.5 Queue

```
empty  size  front  back  push  pop
```

3.6 Stack

```
empty  size  back  push  pop  top
```

3.7 Next permutation

```
//ex:123->132->213->231->312->321
int a[] = { 0,1,2,3,4,5,6,7,8,9 };
while (next_permutation(a, a + 10)){}
```

4 Divide and conquer

4.1 Merge sort

```
vector<int> merge_sort(vector<int> vec) {
  if (vec.size() == 1)return vec;
  vector<int> left(vec.begin(), vec.begin() + vec.size
      ()/2),
    right(vec.begin() + vec.size()/2, vec.end());
  left = merge_sort(left);
  right = merge_sort(right);
  vector<int> re;
  int i, j;
  for (i = 0,j=0; i < left.size() && j<right.size();) {
    if (left[i] < right[j]) re.push_back(left[i++]);
    else if(left[i] > right[j]) re.push_back(right[j
        ++]);
    else {
      re.push_back(left[i++]);
      re.push_back(right[j++]);
    }
  }
  while (i < left.size())re.push_back(left[i++]);
  while (j < right.size())re.push_back(right[j++]);
  return re;
}
```

4.2 Quick sort

```
vector<int> quick_sort(vector<int> vec) {
  if (vec.size() <= 1)return vec;
  srand(time(NULL));
  int pivot =rand() % vec.size();
  vector<int> left, right;
  for (int i = 0; i < vec.size(); i++) {
    if (i == pivot)continue;
    if (vec[i] > vec[pivot])right.push_back(vec[i]);
    else left.push_back(vec[i]);
  }
  left = quick_sort(left);
  right = quick_sort(right);
  left.push_back(vec[pivot]);
  for (int i = 0; i < right.size(); i++)left.push_back(
      right[i]);
  return left;
}
```

4.3 Closest pair

```
struct Node{
  long int x, y;
};
bool cmp_y(Node a,Node b){
  if (a.y != b.y)
    return a.y < b.y;
  else
    return a.x < b.x;
}
bool cmp_x(Node a, Node b) {
  if (a.x != b.x)
    return a.x < b.x;
  else
    return a.y < b.y;
}
void update_dis(Node a,Node b){
  mindis = min(mindis, sqrt((a.x - b.x) * (a.x - b.x) +
       (a.y - b.y) * (a.y - b.y)));
}
vector<Node> node;
void closest(int l,int r) {
  if (r-l <= 3) {//點少暴力拆解
    for (int i = l; i < r; i++) {
      for (int j = i + 1; j < r ; j++) {
        update_dis(node[i], node[j]);
      }
    }
    return;
  }
  //點不少就拆2邊(分,治)
  int mid = (l + r) >> 1, midx = node[mid].x;
  closest(l,mid); closest(mid,r);
  //合
  vector<Node> all;
  int k = mid;
  while (midx - node[k].x < mindis) {
    all.push_back(node[k]);
    k--;
    if (k < l)break;
  }
  k = mid + 1;
  while (node[k].x - midx < mindis) {
    all.push_back(node[k]);
    k++;
    if (k >= r)break;
  }
  sort(all.begin(), all.end(), cmp_y);
  for (int i = 0; i < all.size(); i++) {
    for (int j = i+1; j < all.size(); j++) {
      if (all[j].y - all[i].y > mindis)break;
      if ((all[i].x >= node[mid].x && all[j].x <= node[
          mid].x) || (all[j].x >= node[mid].x && all[i
          ].x <= node[mid].x)) {
        update_dis(all[i], all[j]);
      }
    }
  }
}
int main() {
  ios::sync_with_stdio(0);
  cin.tie(0);
  cout.tie(0);
  int n;
  cin >> n;
  node.resize(n);
  for (int i = 0; i < n; i++) {
    cin >> node[i].x >> node[i].y;
  }
  sort(node.begin(), node.end(), cmp_x);
  closest(0,n);
  cout << fixed << setprecision(4) << mindis << "\n";
}
```

5 DP

5.1 Coin

```
void findway() {
  way[0] = 1;
  int price[5] = { 1,5,10,25,50 };
  for (int i = 0; i < 5; i++) {
    for (int j = price[i]; j < MXN;j++) {
      way[j] += way[j - price[i]];
    }
  }
}
```

5.2 Knapsack (01backpack)

```
void knapsack(int n, int w){        //--2dim
    memset(c, 0, sizeof(c));
    for (int i = 0; i < n; ++i)
        for (int j = 0; j <= w; ++j)
            if (j - weight[i] < 0)
                c[i+1][j] = c[i][j];
            else
                c[i+1][j] = max(c[i][j],c[i][j - weight
                    [i]] + cost[i]);
    cout << "max value:" << c[n][w];
}

void knapsack(int n, int w) {     //--1dim
    memset(c, 0, sizeof(c));
    for (int i = 0; i < n; ++i)
        for (int j = w; j - weight[i] >= 0; --j)
            c[j] = max(c[j], c[j - weight[i]] + cost[i
                ]);
    cout << "max value:" << c[w];
}
```

5.3 LCS

```
dp[i][j]=dp[i-1][j-1]+1 (s[i]==t[j])
         max(dp[i-1][j],dp[i][j-1]) (s[i]!=t[j])
init:dp[i][-1]=dp[-1][j]=0
```

5.4 LIS

```
int LIS(const vector<int>& arr){
    vector<int> lis;
    for(auto i:arr){
        if(lis.empty() || lis.back() < i)   lis.
            push_back(i);
        else    *lower_bound(lis.begin(),lis.end(),i) =
             i;
    }
    return lis.size();
}
```

5.5 Max-sum of sub-array

```
dp[i]=max(dp[i-1],0)+a[i]
```

6 Flow

6.1 Flow

```
#define PB push_back
#define SZ(x) (int)x.size()
using namespace std;
const int MXN = 10000;
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;
```

6.2 KM

```
//perfect match,return max value
struct KM{   //找最小值時，代入-w
  static const int MXN = 2001; // 1-based
  static const ll INF = 0x3f3f3f3f;
  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) {
    n = _n;
    for(int i=1; i<=n; i++)
  fill(g[i], g[i]+n+1, 0);//找最小值時，0變-INF
  }
  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;
```

6.3 MinCostFlow

```
struct MinCostMaxFlow {
   typedef int Tcost;
   static const int MAXV = 20010;
   static const int INFf = 1000000;
   static const Tcost INFc = 1e9;
   struct Edge {
       int v, cap;
       Tcost w;
       int rev;
       Edge() {}
       Edge(int t2, int t3, Tcost t4, int t5)
           : v(t2), cap(t3), w(t4), rev(t5) {}
   };
   int V, s, t;
   vector<Edge> g[MAXV];
                //sum, start, end
   void init(int n, int _s, int _t) {
       V = n; s = _s; t = _t;
       for (int i = 0; i <= V; i++) g[i].clear();
   }
                       //start,end,capacity,cost
   void addEdge(int a, int b, int cap, Tcost w) {
       g[a].push_back(Edge(b, cap, w, (int)g[b].size())
           );
       g[b].push_back(Edge(a, 0, -w, (int)g[a].size() -
            1));
   }
   Tcost d[MAXV];
   int id[MAXV], mom[MAXV];
   bool inqu[MAXV];
   queue<int> q;
   pair<int, Tcost> solve() {
       int mxf = 0; Tcost mnc = 0;
       while (1) {
           fill(d, d + 1 + V, INFc);
           fill(inqu, inqu + 1 + V, 0);
           fill(mom, mom + 1 + V, -1);
           mom[s] = s;
           d[s] = 0;
           q.push(s); inqu[s] = 1;
           while (q.size()) {
               int u = q.front(); q.pop();
               inqu[u] = 0;
               for (int i = 0; i < (int)g[u].size(); i
                   ++) {
                   Edge& e = g[u][i];
                   int v = e.v;
                   if (e.cap > 0 && d[v] > d[u] + e.w)
                       {
                       d[v] = d[u] + e.w;
                       mom[v] = u;
                       id[v] = i;
                       if (!inqu[v]) q.push(v), inqu[v]
                            = 1;
                   }
               }
           }
           if (mom[t] == -1) break;
           int df = INFf;
           for (int u = t; u != s; u = mom[u])
               df = min(df, g[mom[u]][id[u]].cap);
           for (int u = t; u != s; u = mom[u]) {
               Edge& e = g[mom[u]][id[u]];
               e.cap -= df;
               g[e.v][e.rev].cap += df;
           }
           mxf += df;
           mnc += df * d[t];
       }
       return { mxf,mnc };
   }
}flow;
```

7 Graph

7.1 BCC

```
//return a 2dim array,if size=2 represent this is a
    bridge
//else if size>=3 represent this is a group
#define MXN (int)(1e5+5)
#define PB push_back
#define REP(i,x) for(int i=0;i<x;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) {
    n = _n; nScc = step = 0;
    for (int i=0; i<n; i++) E[i].clear();
  }
  void addEdge(int u, int v)
  { 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() {
    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;
```

7.2 Max Clique

```
struct MaxClique{
  static const int MAXN=105;
  int N,ans;
  int g[MAXN][MAXN],dp[MAXN],stk[MAXN][MAXN];
  int sol[MAXN],tmp[MAXN];  //sol[0~ans-1]is answer
  void init(int n){
    N=n;    //0-base
    memset(g,0,sizeof(g));
  }
  void add_edge(int u,int v){
    g[u][v]=g[v][u]=1;
  }
  int dfs(int ns,int dep){
    if(!ns){
      if(dep>ans){
        ans=dep;
        memcpy(sol,tmp,sizeof tmp);
        return 1;
      }else return 0;
    }
    for(int i=0;i<ns;++i){
      if(dep+ns-i<=ans)return 0;
      int u=stk[dep][i],cnt=0;
      if(dep+dp[u]<=ans)return 0;
      for(int j=i+1;j<ns;++j){
        int v=stk[dep][j];
        if(g[u][v])stk[dep+1][cnt++]=v;
      }
      tmp[dep]=u;
      if(dfs(cnt,dep+1))return 1;
    }
    return 0;
  }
  int clique(){
    int u,v,ns;
    for(ans=0,u=N-1;u>=0;--u){
      for(ns=0,tmp[0]=u,v=u+1;v<N;++v)
        if(g[u][v])stk[1][ns++]=v;
      dfs(ns,1),dp[u]=ans;
    }
    return ans; //max num make Complete Graph
  }
};
```

7.3 SCC

```
#define PB push_back
#define MXN 10000
#define FZ(x) memset(x,0,sizeof(0))
struct Scc {
  int n, nScc, vst[MXN], bln[MXN]; // 最後每個點所屬的
      連通分量存在bln陣列
  vector<int> E[MXN], rE[MXN], vec;
  void init(int _n) { //先初始化點的數量
    n = _n;
    for (int i = 0; i < MXN; i++)
      E[i].clear(), rE[i].clear();
  }
  void addEdge(int u, int v) {  // 加有向邊
    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() {  // 跑 kosaraju
    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++;
      }
  }
}scc;
```

7.4 SPFA 差分約束

```
//判斷負環 , 差分約束
//差分約束 :
//xj - xi <= k
//連接一條邊 連接一條邊(i,j), 權重為 k
//最後再設置一個起點 s,連向所有邊邊權為 0
//從起點 s,跑 SPFA,若出現負環則代表這組不等式無解

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

```
                len[i.first]=len[u]+1;
                if(len[i.first]>n)  return 1;
                if(inq[i.first])  continue;
                if(!dq.empty()&&dis[dq.front()]>dis[i.
                    first])
                    dq.push_front(i.first);
                else
                    dq.push_back(i.first);
                inq[i.first]=1;
    } } }
    return 0;
}
```

7.5 歐拉路徑-迴路

```
//存在歐拉的條件

//歐拉迴路
//無向圖: 所有點的度數為偶數
//有向圖: 所有點入度等於出度

//歐拉路徑
//無向圖: 度數為奇數的點數量不超過2
//有向圖: 全部點的入度出度一樣,
//        或剛好一個點出度-1=入度,
//        另一點入度-1=出度,
//        其他點入度等於出度
//  且圖連通!!!! <反例> a->b b->a c->d d->c
vector<int> path;
void dfs(int x){
    while(!edge[x].empty()){
        int u = edge[x].back();
        edge[x].pop_back();
        dfs(u);
    }
    path.push_back(x);
}

int main(){
    build();
    dfs(st);
    reverse(path.begin(),path.end());
    for(int i:path)    cout<<i<<' ';
    cout<<endl;
}
```

7.6 MinimumMeanCycle

```
#include<cfloat> //for DBL_MAX
int dp[MAXN][MAXN]; // 1-base,O(NM)
vector<tuple<int, int, int>> edge;
double mmc(int n) {//allow negative weight
  const int INF = 0x3f3f3f3f;
  for (int t = 0; t < n; ++t) {
    memset(dp[t + 1], 0x3f, sizeof(dp[t + 1]));
    for (const auto& e : edge) {
      int u, v, w;
      tie(u, v, w) = e;
      dp[t + 1][v] = min(dp[t + 1][v], dp[t][u] + w);
    }
  }
  double res = DBL_MAX;
  for (int u = 1; u <= n; ++u) {
    if (dp[n][u] == INF) continue;
    double val = -DBL_MAX;
    for (int t = 0; t < n; ++t)
      val = max(val, (dp[n][u] - dp[t][u]) * 1.0 / (n -
          t));
    res = min(res, val);
  }
  return res; //if there are no cycle return DBL_MAX
}
```

7.7 Minimum Spanning Tree(kruskal)

```
struct Edge{
    int u,v,w;
    friend bool operator<(const Edge& lhs,const Edge&
        rhs){
        return lhs.w<rhs.w;
    }
};
vector<Edge> graph;
void kruskal(){
    int sum=0;
    sort(graph.begin(),graph.end());
    for(auto i:graph){
        if(Find(i.u)!=Find(i.v)){
            Union(find(i.u),find(i.v));
            sum+=i.w;
        }
    }
    cout<<sum<<endl;
}
```

7.8 Minimum Spanning Tree(prim)

```
void prim(){
    v.clear();v.resize(n);
    priority_queue<pair<ll,int>,vector<pair<ll,int>>,
        greater<pair<ll,int>>> pq;
    pq.push({0,0});
    ll sum=0;
    while(!pq.empty()){
        auto u=pq.top();pq.pop();
        if(v[u.second]) continue;
        v[u.second]=1;
        sum+=u.first;
        for(auto i:edge[u.second]){
            if(!v[i.first]){
                pq.push({i.second,i.first});
            }
        }
    }
    cout<<sum<<endl;
}
```

7.9 Single Source Shortest Paths(dijkstra)

```
void dijkstra(int startPoint,int endPoint){
    priority_queue<pair<ll,int>,vector<pair<ll,int>>,
        greater<pair<ll,int>>> pq;
    v.clear();v.resize(n);
    dis.clear();dis.resize(n,INF);
    dis[startPoint]=0;
    pq.push({dis[startPoint],startPoint});//push
        startpoint into pq
    while(!pq.empty()){//if pq is not empty then
        continue
       auto u=pq.top();pq.pop();  //pop the point that
           is closet to startpoint everytime
       if(v[u.second]) continue; //if the point is
           visited represent there already have shorter
            path and dont have to walk again
       v[u.second]=1;       //set the point visited
       for(auto i:edge[u.second]){
          if(dis[i.first]>u.first+i.second){//determine
              whether it can relax
             dis[i.first]=u.first+i.second;
             pq.push({dis[i.first],i.first});//connect
                 the path that can relax
          }
       }
    }
    cout<<dis[endPoint]<<endl;
}
```

7.10 Union find

```
void init(){
    for(int i=0;i<n;i++)
        f[i]=i;
}
int find(int x){
    if(f[x] == x)    return x;
    f[x] = find(f[x]);    //make root as x's father
    return f[x];
}
void merge(int x,int y){
    x=find(x),y=find(y);
    if(x!=y)    f[y]=x;
}
```

7.11 Maxium General Weighted Matching

```
//一般圖帶權匹配
//滿足最大匹配情況下最大化權重
struct WeightGraph {
  static const int INF = INT_MAX;
  static const int N = 514;
  struct edge{
    int u,v,w; edge(){}
    edge(int ui,int vi,int wi)
      :u(ui),v(vi),w(wi){}
  };
  int n,n_x;
  edge g[N*2][N*2];
  int lab[N*2];
  int match[N*2],slack[N*2],st[N*2],pa[N*2];
  int flo_from[N*2][N+1],S[N*2],vis[N*2];
  vector<int> flo[N*2];
  queue<int> q;
  int e_delta(const edge &e){
    return lab[e.u]+lab[e.v]-g[e.u][e.v].w*2;
  }
  void update_slack(int u,int x){
    if(!slack[x]||e_delta(g[u][x])<e_delta(g[slack[x]][
        x]))slack[x]=u;
  }
  void set_slack(int x){
    slack[x]=0;
    for(int u=1;u<=n;++u)
      if(g[u][x].w>0&&st[u]!=x&&S[st[u]]==0)
        update_slack(u,x);
  }
  void q_push(int x){
    if(x<=n)q.push(x);
    else for(size_t i=0;i<flo[x].size();i++)
      q_push(flo[x][i]);
  }
  void set_st(int x,int b){
    st[x]=b;
    if(x>n)for(size_t i=0;i<flo[x].size();++i)
      set_st(flo[x][i],b);
  }
  int get_pr(int b,int xr){
    int pr=find(flo[b].begin(),flo[b].end(),xr)-flo[b].
        begin();
    if(pr%2==1){
      reverse(flo[b].begin()+1,flo[b].end());
      return (int)flo[b].size()-pr;
    }else return pr;
  }
  void set_match(int u,int v){
    match[u]=g[u][v].v;
    if(u<=n) return;
    edge e=g[u][v];
    int xr=flo_from[u][e.u],pr=get_pr(u,xr);
    for(int i=0;i<pr;++i)set_match(flo[u][i],flo[u][i
        ^1]);
    set_match(xr,v);
    rotate(flo[u].begin(),flo[u].begin()+pr,flo[u].end
        ());
  }
  void augment(int u,int v){
    for(;;){
      int xnv=st[match[u]];
      set_match(u,v);
      if(!xnv)return;
      set_match(xnv,st[pa[xnv]]);
      u=st[pa[xnv]],v=xnv;
  } }
  int get_lca(int u,int v){
    static int t=0;
    for(++t;u||v;swap(u,v)){
      if(u==0)continue;
      if(vis[u]==t)return u;
      vis[u]=t;
      u=st[match[u]];
      if(u)u=st[pa[u]];
    }
    return 0;
  }
  void add_blossom(int u,int lca,int v){
    int b=n+1;
    while(b<=n_x&&st[b])++b;
    if(b>n_x)++n_x;
    lab[b]=0,S[b]=0;
    match[b]=match[lca];
    flo[b].clear();
    flo[b].push_back(lca);
    for(int x=u,y;x!=lca;x=st[pa[y]])
      flo[b].push_back(x),flo[b].push_back(y=st[match[x
          ]]),q_push(y);
    reverse(flo[b].begin()+1,flo[b].end());
    for(int x=v,y;x!=lca;x=st[pa[y]])
      flo[b].push_back(x),flo[b].push_back(y=st[match[x
          ]]),q_push(y);
    set_st(b,b);
    for(int x=1;x<=n_x;++x)g[b][x].w=g[x][b].w=0;
    for(int x=1;x<=n;++x)flo_from[b][x]=0;
    for(size_t i=0;i<flo[b].size();++i){
      int xs=flo[b][i];
      for(int x=1;x<=n_x;++x)
        if(g[b][x].w==0||e_delta(g[xs][x])<e_delta(g[b
            ][x]))
          g[b][x]=g[xs][x],g[x][b]=g[x][xs];
      for(int x=1;x<=n;++x)
        if(flo_from[xs][x])flo_from[b][x]=xs;
    }
    set_slack(b);
  }
  void expand_blossom(int b){
    for(size_t i=0;i<flo[b].size();++i)
      set_st(flo[b][i],flo[b][i]);
    int xr=flo_from[b][g[b][pa[b]].u],pr=get_pr(b,xr);
    for(int i=0;i<pr;i+=2){
      int xs=flo[b][i],xns=flo[b][i+1];
      pa[xs]=g[xns][xs].u;
      S[xs]=1,S[xns]=0;
      slack[xs]=0,set_slack(xns);
      q_push(xns);
    }
    S[xr]=1,pa[xr]=pa[b];
    for(size_t i=pr+1;i<flo[b].size();++i){
      int xs=flo[b][i];
      S[xs]=-1,set_slack(xs);
    }
    st[b]=0;
  }
  bool on_found_edge(const edge &e){
    int u=st[e.u],v=st[e.v];
    if(S[v]==-1){
      pa[v]=e.u,S[v]=1;
      int nu=st[match[v]];
      slack[v]=slack[nu]=0;
      S[nu]=0,q_push(nu);
    }else if(S[v]==0){
      int lca=get_lca(u,v);
      if(!lca)return augment(u,v),augment(v,u),true;
      else add_blossom(u,lca,v);
    }
    return false;
  }
  bool matching(){
    memset(S+1,-1,sizeof(int)*n_x);
    memset(slack+1,0,sizeof(int)*n_x);
    q=queue<int>();
    for(int x=1;x<=n_x;++x)
      if(st[x]==x&&!match[x])pa[x]=0,S[x]=0,q_push(x);
    if(q.empty())return false;
    for(;;){
      while(q.size()){
        int u=q.front();q.pop();
        if(S[st[u]]==1)continue;
        for(int v=1;v<=n;++v)
          if(g[u][v].w>0&&st[u]!=st[v]){
            if(e_delta(g[u][v])==0){
              if(on_found_edge(g[u][v]))return true;
            }else update_slack(u,st[v]);
      }   }
      int d=INF;
      for(int b=n+1;b<=n_x;++b)
        if(st[b]==b&&S[b]==1)d=min(d,lab[b]/2);
      for(int x=1;x<=n_x;++x)
        if(st[x]==x&&slack[x]){
          if(S[x]==-1)d=min(d,e_delta(g[slack[x]][x]));
```

```
          else if(S[x]==0)d=min(d,e_delta(g[slack[x]][x
              ])/2);
        }
      for(int u=1;u<=n;++u){
        if(S[st[u]]==0){
          if(lab[u]<=d)return 0;
          lab[u]-=d;
        }else if(S[st[u]]==1)lab[u]+=d;
      }
      for(int b=n+1;b<=n_x;++b)
        if(st[b]==b){
          if(S[st[b]]==0)lab[b]+=d*2;
          else if(S[st[b]]==1)lab[b]-=d*2;
        }
      q=queue<int>();
      for(int x=1;x<=n_x;++x)
        if(st[x]==x&&slack[x]&&st[slack[x]]!=x&&e_delta
            (g[slack[x]][x])==0)
          if(on_found_edge(g[slack[x]][x]))return true;
      for(int b=n+1;b<=n_x;++b)
        if(st[b]==b&&S[b]==1&&lab[b]==0)expand_blossom(
            b);
    }
    return false;
  }
  pair<long long,int> solve(){
    memset(match+1,0,sizeof(int)*n);
    n_x=n;
    int n_matches=0;
    long long tot_weight=0;
    for(int u=0;u<=n;++u)st[u]=u,flo[u].clear();
    int w_max=0;
    for(int u=1;u<=n;++u)
      for(int v=1;v<=n;++v){
        flo_from[u][v]=(u==v?u:0);
        w_max=max(w_max,g[u][v].w);
      }
    for(int u=1;u<=n;++u)lab[u]=w_max;
    while(matching())++n_matches;
    for(int u=1;u<=n;++u)
      if(match[u]&&match[u]<u)
        tot_weight+=g[u][match[u]].w;
    return make_pair(tot_weight,n_matches);
  }
  void add_edge( int ui , int vi , int wi ){ //無向圖
    g[ui][vi].w = g[vi][ui].w = wi;
  }
  void init( int _n ){  //1-base
    n = _n;
    for(int u=1;u<=n;++u)
      for(int v=1;v<=n;++v)
        g[u][v]=edge(u,v,0);
  }
} graph;
```

7.12 Maxium General graph Matching

```
//一般圖匹配 用在無向圖
//<cf> flow 只能用在2分匹配
// should shuffle vertices and edges  要打亂輸入的邊和
    點

mt19937 gen(chrono::steady_clock::now().
    time_since_epoch().count());

for(int i=1;i<=n;i++)
  shuffle(edge[i].begin(),edge[i].end(),gen);

把點打亂: shuffle(ind.begin(),ind.end(),gen);   //ind是
    1~n
edge[ind[a]].push_back(ind[b]);
edge[ind[b]].push_back(ind[a]);

const int N=100005,E=(2e5)*2+40;
struct Graph{ // 1-based; match: i <-> lnk[i]
  int to[E],bro[E],head[N],e,lnk[N],vis[N],stp,n;
  void init(int _n){
    stp=0; e=1; n=_n;
    for(int i=1;i<=n;i++) head[i]=lnk[i]=vis[i]=0;
  }
  void add_edge(int u,int v){
    to[e]=v,bro[e]=head[u],head[u]=e++;
    to[e]=u,bro[e]=head[v],head[v]=e++;
  }
  bool dfs(int x){
    vis[x]=stp;
    for(int i=head[x];i;i=bro[i]){
      int v=to[i];
      if(!lnk[v]){ lnk[x]=v,lnk[v]=x; return true; }
    }
    for(int i=head[x];i;i=bro[i]){
      int v=to[i];
      if(vis[lnk[v]]<stp){
        int w=lnk[v]; lnk[x]=v,lnk[v]=x,lnk[w]=0;
        if(dfs(w)) return true;
        lnk[w]=v,lnk[v]=w,lnk[x]=0;
      }
    }
    return false;
  }
  int solve(){
    int ans=0;
    for(int i=1;i<=n;i++) if(!lnk[i]) stp++,ans+=dfs(i)
        ;
    return ans;
  }
}graph;
```

8 Data structure

8.1 Point Modify

```
#define cl(x) (x<<1)+1
#define cr(x) (x<<1)+2

int arr[MXN];
int tree[MXN*4];


void build(int index,int left,int right){
    if( left == right ){
        tree[index] = arr[left];
        return;
    }
    int mid=(left + right)/2;
    build(cl(index),left,mid);
    build(cr(index),mid+1,right);
    tree[index] = max(tree[cl(x)] ,tree[cr(x)]);
}
int query(int index,int left,int right,int query_left,
    int query_right){
    if( query_left <= left && right <= query_right){
        return tree[index];
    }
    int mid=(left + right)/2;
    int ans=-INF;
    if(query_left <= mid){
        ans = max(ans, query(cl(index),left,mid,
            query_left,query_right));
    }
    if(query_right > mid){
        ans = max(ans, query(cr(index),mid+1,right,
            query_left,query_right));
    }
    return ans;
}
void update(int index,int left,int right,int position,
    int value){
    if(left == right){
        tree[index] = value;
        return;
    }
    int mid=(left+right)/2;
    if(position <= mid){
        update(cl(index),left,mid,position,value);
    }
    esle{
        update(cr(index),mid+1,right,position,value);
    }
    tree[index] = max(tree[cl(index)],tree[cr(index)]);
}

int main(){
    build(0,0,n-1);
```

```
    query(0,0,n-1,2,7);
}
```

8.2 Interval Modify

```
#define cl(x) (x<<1)+1
#define cr(x) (x<<1)+2
#define INF 1e9
struct seg_tree {//0-base
  static const int MXN = 1e5 + 5, NO_TAG = 0;  //to be
      set
  ll a[MXN], val[MXN * 4], tag[MXN * 4], v;
  int n, ql, qr;
  void push(int i, int l, int r) {
    if (tag[i] != NO_TAG) {
      val[i] += tag[i]; //update by tag
      if (l != r) {
        tag[cl(i)] += tag[i]; //push
        tag[cr(i)] += tag[i]; //push
      }
      tag[i] = NO_TAG;
    }
  }
  void pull(int i, int l, int r) {
    int mid = (l + r) >> 1;
    push(cl(i), l, mid); push(cr(i), mid + 1, r);
    val[i] = max(val[cl(i)], val[cr(i)]); //pull
  }
  void build(int i, int l, int r) {
    if (l == r) {
      val[i] = a[l]; //set value
      return;
    }
    int mid = (l + r) >> 1;
    build(cl(i), l, mid); build(cr(i), mid + 1, r);
    pull(i, l, r);
  }
  void update(int i, int l, int r) {
    push(i, l, r);
    if (ql == l && r == qr) {
      tag[i] += v; //update tag
      return;
    }
    int mid = (l + r) >> 1;
    if (ql <= mid) update(cl(i), l, mid);
    if (qr > mid) update(cr(i), mid + 1, r);
    pull(i, l, r);
  }
  void query(int i, int l, int r) {
    push(i, l, r);
    if (ql <= l && r <= qr) {
      v = max(v, val[i]); //update answer
      return;
    }
    int mid = (l + r) >> 1;
    if (ql <= mid) query(cl(i), l, mid);
    if (qr > mid) query(cr(i), mid + 1, r);
  }
  int Query(int _ql, int _qr) { //傳 入 詢 問 區 間
    v = -INF, ql = _ql, qr = _qr;
    query(0, 0, n - 1);
    return v;
  }
  void Update(int _v, int _ql, int _qr) { //傳 入 更 新
      值,區間
    v = _v, ql = _ql, qr = _qr;
    update(0, 0, n - 1);
  }
  void init() {
    memset(tag, 0, sizeof(tag));
    memset(val, 0, sizeof(val));
  }
}tree;
```

8.3 BIT

```
void update(int x){
  for (;x < MXN;x+= lowbit(x)) {
    BIT[x] += 1;
  }
}
ll query(int x){
  ll ans = 0;
  for (;x > 0; x -= lowbit(x)) {
    ans += BIT[x];
  }
  return ans;
}
map<ll, int> v_idx;   //點對應的idx(離散化)
for (int i = 1; i <= n; i++)
  {
    cin >> arr[i];
    v_idx[arr[i]] = 1;    //建點
    v_idx[k * arr[i]] = 1;
  }
  map<ll, int>::iterator iter;
  int idx = 0;
  for (iter = v_idx.begin(); iter != v_idx.end(); iter
      ++) //儲存idx
    iter->second = ++idx;
  for (int i = 1; i <= n; i ++)
  {
    ans += query(idx) - query(v_idx[k*arr[i]]);//所有已
        加進去的-符合條件的=不符合的
    //或ans += i-1 - query(v_idx[k*arr[i]]);
    update(v_idx[arr[i]]);
  }
```

9 Tree

9.1 LCA

```
vector<vector<int>> tree;
vector<vector<int>> anc;
vector<int> timeIn, timeOut;
int ti = 0;
void build(int x, int fa) {
    anc[x].resize(__lg(n)+10);//tle改+1
    for (int i = 0; i < __lg(n) + 10; i++) {
        anc[x][i] = fa;
        fa = anc[fa][i];
    }
}
void dfs(int x, int fa) {
    timeIn[x] = ti++;
    build(x, fa);
    for (int i = 0; i < tree[x].size(); i++) {
        if (tree[x][i] == fa) continue;
        dfs(tree[x][i], x);
    }
    timeOut[x] = ti++;
}
bool isAnc(int a, int b) {
    if (timeIn[a] <= timeIn[b] && timeOut[a] >= timeOut
        [b])return 1; //a是祖先
    return 0;
}
int query(int a, int b) {
    if (isAnc(a, b))return a;
    if (isAnc(b, a))return b;
    for (int i = __lg(n) + 10 - 1; i >= 0; i--) {
        if (!isAnc(anc[a][i], b)) a = anc[a][i];
    }
    return anc[a][0];
}
int nlca(int x, int y) {   //x's yth anc
    if (y == 0)return x;
    int tp = log2(y);
    if (y == (1 << tp)) return anc[x][tp];
    else return nlca(anc[x][tp], y - (1 << tp));
}
```

9.2 Persistent Segment Tree 持久化

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

vector<node *> version;    //用一個vector紀錄全部版本的
    根節點

//線段樹
```

```
void build(node *now_version, l, r);
ll query(node *now_version, l, r, ql, qr);
node *update_version(node *pre_version,int l,int r,int
    pos,int v); //回傳新建的節點

void add_version(int x,int v){    //修改位置 x 的值為 v
    version.push_back(update_version(version.back(), 0,
        n-1, x, v));
}

node *update_version(node *pre_version,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(pre_version->l, l, mid, pos, v);
            //左邊節點連向新節點
        x->r = pre->version->r;
            //右邊連到原本的右邊
    }
    else{ //更新右邊
        x->l = pre->version->l;
                                    //左邊連到原本的左
            邊
        x->r = update(pre_version->r, r, mid, pos, v);
             //右邊節點連向新節點
    }
    x->val = x->l->val + x->r->val;
    return x;
}
//並查集
void build(node* now, int left, int right) {
  if (left == right) {
    now->fa = left;
    now->sz = 1;
    return;
  }
  int mid = (left + right) / 2;
  now->cl = new node;
  now->cr = new node;
  build(now->cl, left, mid);
  build(now->cr, mid + 1, right);
}
node* update_fa(node* pre, int left, int right, int pos
    , int val) {
  node* x = new node;
  if (left == right) {
    x->fa = val;
    x->sz = 1;
    return x;
  }
  int mid = (left + right) >> 1;
  if (pos <= mid) {
    x->cl = update_fa(pre->cl, left, mid, pos, val);
    x->cr = pre->cr;
  }
  else {
    x->cl = pre->cl;
    x->cr = update_fa(pre->cr, mid + 1, right, pos, val
        );
  }
  return x;
}
void update_sz(node* now, int left, int right, int pos,
     int val) {
  if (left == right) {
    now->sz += val;
    return;
  }
  int mid = (left + right) >> 1;
  if (pos <= mid) update_sz(now->cl, left, mid, pos,
      val);
  else update_sz(now->cr, mid + 1, right, pos, val);
}
pair<int,int> query(node* now, int left, int right, int
     pos) {
  if (left == right) return { now->fa,now->sz};
  int mid = (left + right) >> 1;
  if (pos <= mid) return query(now->cl, left, mid, pos)
      ;
  else return query(now->cr, mid+1, right, pos);
}
pair<int, int> find_(node* now_version,int m,int x) {
  pair<int,int> tp = query(now_version, 1, m, x);
  if (x == tp.first) return tp;
  else return find_(now_version, m, tp.first);
}
void union_(node* now_version,int new_,int n,int x,int
    y) {
  pair<int, int> X = find_(now_version, n, x);
  pair<int, int> Y = find_(now_version, n, y);
  if (X.first != Y.first) {
    if (X.second < Y.second) {
      version[new_] = update_fa(now_version, 1, n, X.
          first, Y.first);
      update_sz(version[new_], 1, n, Y.first, X.second)
          ;
    }
    else {
      version[new_] = update_fa(now_version, 1, n, Y.
          first, X.first);
      update_sz(version[new_], 1, n, X.first, Y.second)
          ;
    }
  }else version[new_] = now_version;

}
```

9.3 Treap

```
struct Treap {
  int key, pri, sz, tag;
  Treap* cl, * cr;   //左右子樹
  Treap() {}
  Treap(int key_) {
    key = key_;
    sz = 1;
    tag = 0;
    pri = rand();
    cl = cr = nullptr;
  }
};
int Size(Treap* x) { return x ? x->sz : 0; }
void pull(Treap* x) { x->sz = Size(x->cl) + Size(x->cr)
     + 1; }
Treap* merge(Treap* a, Treap* b) {
  push(a);
  push(b);
  if (!a || !b)return a ? a : b; //其中一個子樹為空則回
      傳另一個
  if (a->pri > b->pri) {          //如果a的pri比較大則a
      比較上面
    push(a->cr);
    push(b);
    a->cr = merge(a->cr, b);  //將a的右子樹跟b合併
    pull(a);
    return a;
  }
  else {
    push(b->cl);
    push(a);
    b->cl = merge(a, b->cl);   //如果b的pri比較大則b比
        較上面
    pull(b);                   //將b的左子樹根a合併
    return b;
  }
}
void splitByKey(Treap* x, int k, Treap*& a, Treap*& b)
    {
  //將一棵Treap分成兩棵，
  //key小於等於k的分到左邊那棵a，其他分到右邊那棵b
  push(x);
  if (!x) { a = b = nullptr; }
  else if (x->key <= k) {
    a = x;
    splitByKey(x->cr, k, a->cr, b);
    pull(a);
  }
  else {
    b = x;
```

```
    splitByKey(x->cl, k, a, b->cl);
    pull(b);
  }
}
void splitByKth(Treap* x, int k, Treap*& a, Treap*& b)
    {
  //將一棵Treap分成兩棵，
  //左邊那棵a的節點數有k個，右邊那棵b節點數為n - k個
  push(x);
  if (!x) { a = b = nullptr; }
  else if (Size(x->cl) + 1 <= k) {
    a = x;  //如果左子樹+自己的size小於等於k則左子樹跟
        自己為k個以內
    splitByKth(x->cr, k - Size(x->cl) - 1, a->cr, b);
    pull(a);
  }
  else {
    b = x;  //如果左子樹+自己的size大於k個則右子樹跟自
        己會分到右邊
    splitByKth(x->cl, k, a, b->cl);
    pull(b);
  }
}
void insert(int val){              //新增一個值為val的元
    素
    Treap *x = new Treap(val);     //設一個treap節點
    Treap *l,*r;
    splitByKey(root, val, l, r);  //找到新節點要放的位
        置
    root = merge(merge(l,x),r);    //合併到原本的treap裡
}
void erase(int val){               //移除所有值為val的元
    素
    Treap *l,*mid,*r;
    splitByKey(root, val, l, r);  //把小於等於val的丟到
        l
    splitByKey(l, val-1, l, mid); //小於val的丟到l,等於
        val的就會在mid裡
    root = merge(l,r);             //將除了val以外的值合
        併
}
//翻轉
void push(Treap* x) {
  if (!x)return;
  if (x->tag) {
    swap(x->cl, x->cr);
    if (x->cl)x->cl->tag ^= 1;
    if (x->cr)x->cr->tag ^= 1;
    x->tag ^= 1;
  }
}
```

9.4 樹壓平

```
vector<vector<int>>edge;
vector<pair<int,int>>times;
int ti = 0;
void dfs(int x,int fa){
  times[x].fir = times[x].sec = ti++;
  for(int i:edge[x]){
    if(i == fa)continue;
    dfs(i,x);
    times[x].sec = max(times[i].sec,times[x].sec);
  }
}
```

9.5 樹上全點對距離總和

```
vector<vector<int>>edge;   //要建雙向邊
vector<int>dp,sz;
void dfs1(int fa,int x){
    sz[x] = 1;
    for(int i=0;i<edge[x].size();i++){
        if(edge[x][i] == fa)continue;
        dfs1(x,edge[x][i]);
        dp[x] += dp[edge[x][i]]+sz[edge[x][i]];
        sz[x] += sz[edge[x][i]];
    }
}
ll ans = 0;
void dfs2(int fa,int x,ll sum){
    ans += (sum + dp[x]);
    for(int i=0;i<edge[x].size();i++){
        if(edge[x][i] == fa) continue;
        ll tp = sum + dp[x] - (dp[edge[x][i]] + sz[edge
            [x][i]]) + (n-sz[edge[x][i]]); //從x到edge[
            x][i]的距離總和
        dfs2(x,edge[x][i],tp);
    }
}
dfs1(1,1);
dfs2(1,1,0);  //答案是ans/2
```

10 Geometry

10.1 Convex hull 凸包

```
struct Pt{
    int x,y;
    Pt(){}
    Pt(int _x,int _y){
        x=_x,y=_y;
    }
    friend bool operator<(const Pt& lhs,const Pt& rhs){
        return lhs.x==rhs.x?lhs.y<rhs.y:lhs.x<rhs.x;
    }
    friend Pt operator-(const Pt& lhs,const Pt& rhs){
        return Pt(rhs.x-lhs.x,rhs.y-lhs.y);
    }
    friend int cross(const Pt& o,const Pt& a,const Pt&
        b){
        Pt lhs = o-a, rhs = o-b;
        return lhs.x*rhs.y - lhs.y*rhs.x;
    }
};
vector<Pt> convex_hull(vector<Pt> hull){ //回傳凸包陣列
    sort(hull.begin(),hull.end());
    int top=0;
    vector<Pt> stk;
    for(int i=0;i<hull.size();i++){
        while(top>=2&&cross(stk[top-2],stk[top-1],hull[
            i])<=0)
            stk.pop_back(),top--;
        stk.push_back(hull[i]);
        top++;
    }
    for(int i=hull.size()-2,t=top+1;i>=0;i--){
        while(top>=t&&cross(stk[top-2],stk[top-1],hull[
            i])<=0)
            stk.pop_back(),top--;
        stk.push_back(hull[i]);
        top++;
    }
    stk.pop_back();
    return stk;
}
double FarthestPair(vector<Pt> arr) { //回傳最遠點對
    double ret = 0;
    for (int i = 0, j = i + 1; i < arr.size(); i++) {
        while (distance(i, j) < distance(i, (j + 1) %
            arr.size())) {
            j = (j + 1) % arr.size();
        }
        ret = max(ret, distance(i, j));
    }
    return ret;
}
vector<Pt> tui;
double distance(int i, int j) {
    double a = (tui[i].x - tui[j].x) * (tui[i].x - tui[
        j].x),
           b = (tui[i].y - tui[j].y) * (tui[i].y - tui[
               j].y);
    return sqrt(a + b);
}
```

10.2 Stack Square Area

```
#include<bits/stdc++.h>
#include<unordered_map>
using namespace std;
int c, n, m, min_area, head, tail, mid, f, area;
bool check;
unordered_map<int, int>done;
```

```
vector<vector<int>>G, h;
vector<int>element;
vector<pair<int, int>>Stack;
pair<long long, long long>ans;
int main() {
  ios::sync_with_stdio(0);
  cin.tie(0);
  cout.tie(0);
  cin >> c;
  while (c--) {
    h.clear();
    G.clear();
    element.clear();
    cin >> n >> m >> min_area;
    G.resize(n);
    done.clear();
    for (int i = 0; i < n; i++) {
      G[i].resize(m);
      for (int j = 0; j < m; j++) {
        cin >> G[i][j];
        element.push_back(G[i][j]);
      }
    }
    sort(element.begin(), element.end());
    ans.first = *element.begin() - 1;
    head = mid = 0;
    tail = element.size();
    while (mid!=((head+tail)>>1)||!done.count(element[
        mid])) {
      mid = (head + tail) >> 1;
      if (!done.count(element[mid])) {
        area = 0;
        h.clear();
        h.resize(n);
        for (int i = 0; i < n; i++) {
          h[i].resize(m);
          for (int j = 0; j < m; j++) {
            h[i][j] = (G[i][j] >= element[mid] ? 1 : 0)
                ;
            if (i != 0 && h[i][j] != 0) h[i][j] += h[i
                - 1][j];
          }
        }
        for (int i = 0; i < n; i++) {
          Stack.clear();
          for (int j = 0; j < m; j++) {
            f = j;
            while (!Stack.empty() && h[i][j] < Stack.
                back().second) {
              area = max(area, (j - Stack.back().first)
                  *Stack.back().second);
              f = Stack.back().first;
              Stack.pop_back();
            }
            if (h[i][j] != 0 && (Stack.empty() || h[i][
                j] != Stack.back().second))Stack.
                push_back(make_pair(f, h[i][j]));
          }
          while (!Stack.empty()) {
            area = max(area, (m - Stack.back().first)*
                Stack.back().second);
            Stack.pop_back();
          }
        }
      }
      if (done.count(element[mid])) area = done[element
          [mid]];
      if (area < min_area) tail = mid;
      else {
        done[element[mid]] = area;
        ans.first = element[mid];
        ans.second = area;
        head = mid;
      }
    }
    cout << ans.first << " " << ans.second << "\n";
  }
  return 0;
}
```

11 Sqrt

11.1 分塊

```
struct blk{
    vector<int> local;    //每塊的全部元素
    int global;           //儲存每塊的總和
    int tag;              //儲存整塊一起更新的值
    blk(){                //初始化
        local.clear();    //清空區間元素
        tag = global = 0; //將區間總和先設為0
    }
};
vector<blk> b;
void build(){
    int len=sqrt(n),num=(n+len-1)/len;
    for(int i=0;i<n;i++){    //第i個元素分在第 i/len 塊
        cin>>x;
        //存入區間中
        b[i/len].local.push_back(x);
        //更新區間總和
        b[i/len].global += x;
    }
}
void update(int ql,int qr,int v){
    int blk_l=ql/len,blk_r=qr/len,ret=0;
    if(blk_l == blk_r){    //如果都在同一塊直接一個一個
        跑過去就好
        for(int i=ql;i<=qr;i++)
            b[blk_l].local[i%len]+=v;
        b[blk_l].global+=(qr-ql+1)*v;
        return;
    }
    for(int i=ql;i<(blk_l+1)*len;i++){  //最左的那一塊
        b[blk_l].local[i%len]+=v;
        b[blk_l].global+=v;
    }
    for(int i=blk_l+1;i<blk_r;i++){  //中間每塊
        b[i].tag+=v;
        b[i].global+=v*len;
    }
    for(int i=blk_r*len;i<=qr;i++){  //最右的那一塊
        b[blk_r].local[i%len]+=v;
        b[blk_r].global+=v;
    }
}
int query(int ql,int qr){
    int blk_l=ql/len,blk_r=qr/len,ret=0;
    if(blk_l == blk_r){    //如果都在同一塊直接一個一個
        跑過去就好
        for(int i=ql;i<=qr;i++)
            ret+=b[blk_l].local[i%len]+b[blk_l].tag;
        return ret;
    }
    for(int i=ql;i<(blk_l+1)*len;i++)    //最左的那一塊
        ret+=b[blk_l].local[i%len]+b[blk_l].tag;
    for(int i=blk_l+1;i<blk_r;i++)    //中間每塊的總和
        ret+=b[i].global;
    for(int i=blk_r*len;i<=qr;i++)    //最右的那一塊
        ret+=b[blk_r].local[i%len]+b[blk_r].tag;
    return ret;
}
```

11.2 Mo's 莫隊

```
int n,k = sqrt(n);    //每塊大小為k
struct query{
    int l,r,id;       //詢問的左界右界 以及 第幾筆詢問
    friend bool operator<(const query& lhs,const query&
         rhs){
        return lhs.l/k==rhs.l/k ? lhs.r<rhs.r : lhs.l<
            rhs.l;
    }    //先判斷是不是在同一塊 不同塊的話就比較塊的順
        序，否則比較右界r
};
  int num = 0;
int cnt[1'000'005], ans[30005];
vector<query> q;
void add(int index){ ... }    //新增元素到區間內
void sub(int index){ ... }    //從區間內移除元素
void solve(){
```

```
    sort(q.begin(),q.end());
    for(int i=0,l=-1,r=0;i<n;i++){
        while(l>q[i].l)    add(--l);
        while(r<q[i].r)    add(++r);    //記得要先做新增元素的
        while(l<q[i].l)    sub(l++);    //再做移除元素的
        while(r>q[i].r)    sub(r--);
        ans[q[i].id] = num;             //移到區間後儲存答案
}    }
```

12 Math

12.1 Miller rabin 找質數

```
// 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.
//<ex> magic[3] ={2,7,61};
#define ull unsigned long long
ull magic[]={}
ull mul(ull a, ull b, ull c) {  //快速乘
  ull ans = 0;
  while (b) {
    if (b & 1) ans = (ans + a) % c;
    a = (a + a) % c;
    b >>= 1;
  }
  return ans;
}
ull mypow(ull a, ull u, ull n) {
  ull ans = 1;
  while (u) {
    if (u & 1) ans = mul(ans,a,n);
    a = mul(a,a,n);
    u >>= 1;
  }
  return ans;
}
bool witness(ull a, ull n, ull u, ull t) {
  if (!a) return 0;
  ull x = mypow(a, u, n); //a^u % n
  for (int i = 0; i < t; i++) {
    ull nx = mul(x, x, n);
    if (nx == 1 && x != 1 && x != n-1) return 1;
    x = nx;
  }
  return x != 1;
}
bool miller_rabin(ull n) {
  int s = (magic numbers size);
    // iterate s times of witness on n
    if(n < 2) return 0;
  if (!(n & 1)) return n == 2;
  ull u = n-1; ull t = 0;
  // n-1 = u*2^t
  while (!(u & 1)) u >>= 1, t++;
  while (s--) {
    ull a = magic[s] % n;
    if (witness(a, n, u, t)) return 0;
  }
  return 1;
}
```

12.2 Chinese remainder theorem

```
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;
}

/*   x ≡ 2 (mod 5)
     x ≡ 3 (mod 7)
     x ≡ 4 (mod 11)

     x[3] = {2,3,5};
     m[3] = {5,7,11};
     n = 3
*/
```

12.3 快速冪

```
ll mypow(ll x,ll y,ll m){    //x^y % m
    ll ret = 1 % m;
    while(y){
        if(y&1)    ret = ret * x % m;
        x = x * x % m
        y >>= 1;
    }
    return ret;
}
```

12.4 矩陣快速冪

```
#define MOD 1'000'000'007
#define ll long long

vector<vector<ll>> operator*(const vector<vector<ll>>&
    lhs,const vector<vector<ll>>& rhs){
    vector<vector<ll>> ret(lhs.size(),vector<ll>(rhs
        [0].size(),0));
    for(int i=0;i<lhs.size();i++){
        for(int j=0;j<rhs[0].size();j++){
            for(int k=0;k<rhs.size();k++){
                ret[i][j] += lhs[i][k] * rhs[k][j] %
                    MOD;
                ret[i][j] %= MOD;
            }
        }
    }
    return ret;
}

vector<vector<ll>> init_value={{1},{0}};  //第0,1項
vector<vector<ll>> base={{1,1},{1,0}};    //費式數列轉移式
vector<vector<ll>> matrix={{1,0},{0,1}};  //單位矩陣
// x^y
while(y){
    if(y&1){
        matrix = matrix * base;
    }
    base = base * base;
    y >>= 1;
}
matrix = matrix * init_value;
cout<< matrix[0][0] << endl;
```

12.5 模逆元

```
ll f[MXN+1] = { 0 }, inv[MXN+1] = {0};
ll mypow(ll x, ll y, ll m) {    //x^y % m
  ll ret = 1 % m;
  while (y) {
    if (y & 1)    ret = ret * x % m;
    x = x * x % m;
    y >>= 1;
  }
  return ret;
}
void init() {
  f[0] = 1;    // 0! = 1
  for (int i = 1; i <= MXN; i++) { //1!到n!
    f[i] = f[i - 1] * i % MOD;
```

```
  }
  inv[MXN] = mypow(f[MXN], (MOD - 2), MOD);
  for (long long i = MXN - 1; i >= 0; i--) { //模逆元
    inv[i] = inv[i + 1] * (i + 1) % MOD;
  }
}
ll C(int n, int m) { //用途再算C幾取幾 % MOD
  return f[n] * inv[m] % MOD * inv[n - m] % MOD;
}
```

12.6 判斷線交叉

```
double Direction(point Pi, point Pj, point Pk){
  return (Pj.x-Pi.x)*(Pk.y-Pi.y)-(Pk.x-Pi.x)*(Pj.y-Pi.y
      );
}

bool isIntersect(line p, line q){
  double d1, d2, d3, d4;
  d1 = Direction(p.a, p.b, q.a);
  d2 = Direction(p.a, p.b, q.b);
  d3 = Direction(q.a, q.b, p.a);
  d4 = Direction(q.a, q.b, p.b);
  if(d1*d2<0 && d3*d4<0)  { return true;}  //規範相交
  //非規範相交
  else return false;
}
```

12.7 Theorem

- Wilson's theorem :
  $(p-1)! \equiv -1 (mod\ p)$ (p is a prime)
- Fermat's little theorem :
  $a^p \equiv a (mod\ p)$ (p is a prime and gcd(a,p)=1)
  $\frac{1}{a} (mod\ p) \equiv a^{p-2}$ (with fast pow)