Contents

1 Init (Linux)

開場流程：

```
vim ~/.vimrc
mkdir contest && cd contest

vim template.cpp
for c in {A..P}; do
    cp template.cpp $c.cpp
done

vim run.sh && chmod 777 run.sh
```

1.1 vimrc

```
syn on
set nu rnu ru cul mouse=a
set cin et ts=4 sw=4 sts=4
set autochdir
set clipboard=unnamedplus

colo koehler

no <C-h> ^
no <C-l> $
no ; :

inoremap {<CR> {<CR>}<Esc>ko
```

1.2 template.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

void solve() {
}

int main() {
    ios_base::sync_with_stdio(false); cin.tie(0);
    int TEST = 1;
    //cin >> TEST;
    while (TEST--) solve();
    return 0;
}
```

1.3 run.sh

```bash
#!/bin/bash

g++ -std=c++17 -O2 -g -fsanitize=undefined,address $1 && echo DONE COMPILE || exit 1
./a.out
```

2 Reminder

2.1 Observations and Tricks

- Contribution Technique
- 二分圖/Spanning Tree/DFS Tree
- 行、列操作互相獨立
- 奇偶性
- 當 s,t 遞增並且 $t=f(s)$，對 s 二分搜不好做，可以改成對 t 二分搜，再算 $f(t)$
- 啟發式合併
- Permutation Normalization（做一些平移對齊兩個 permutation）

- 枚舉 $a_1 \sim a_n$ 再枚舉 $a_n \sim a_1$ 可以包在一個迴圈
- 兩個凸型函數相加還是凸型函數，相減不一定

2.2 Bug List

- 沒開 long long
- 陣列戳出界／陣列開不夠大
- 寫好的函式忘記呼叫
- 0-base / 1-base
- 忘記初始化
- == 打成 =
- <= 打成 <+
- dp[i] 從 dp[i-1] 轉移時忘記特判 i > 0
- std::sort 比較運算子寫成 < 或是讓 = 的情況為 true
- 漏 case
- 線段樹改值懶標初始值不能設為 0
- DFS 的時候不小心覆寫到全域變數
- 浮點數誤差
- unsigned int128
- 多筆測資不能沒讀完直接 return
- 記得刪 cerr
- vector 超級肥，小 vector 請用 array，例如矩陣快速冪

3 Basic

3.1 template (optional)

```cpp
#define F first
#define S second
#define ep emplace
#define eb emplace_back
#define endl '\n'

template<class T> using V=vector<T>;
typedef long long ll;
typedef pair<int, int> pii;
typedef pair<ll, ll> pll;
typedef pair<int, ll> pil;
typedef pair<ll, int> pli;

/* ============================================= */
// STL and I/O
// pair
template<typename T1, typename T2>
ostream& operator<<(ostream& os, pair<T1, T2> p) {
    return os << "(" << p.first << ", " << p.second << ")";
}
template<typename T1, typename T2>
istream& operator>>(istream& is, pair<T1, T2>& p) { return is >> p.first >> p.second; }

// vector
template<typename T>
istream& operator>>(istream& is, vector<T>& v) {
    for (auto& x : v) is >> x;
    return is;
}
template<typename T>
ostream& operator<<(ostream& os, const vector<T>& v) {
    for (const auto& x : v) os << x << ' ';
    return os;
}
/* ============================================= */
// debug(), output()
#define RED        "\x1b[31m"
#define GREEN      "\x1b[32m"
#define YELLOW     "\x1b[33m"
#define GRAY       "\x1b[90m"
#define COLOREND   "\x1b[0m"

void _debug() {}
template<typename A,typename... B> void _debug(A a,B... b) { cerr << a << ' ', _debug(b...); }
#define debug(...) cerr<<GRAY<<#__VA_ARGS__<<": "<<COLOREND,_debug(__VA_ARGS__),cerr<<endl

void _output() {}
template<typename A,typename... B> void _output(A a,B... b) { cout << a << ' ', _output(b...); }
#define output(...) _output(__VA_ARGS__),cout<<endl
/* ============================================= */
// BASIC ALGORITHM
string binary(ll x, int b = -1) {
    if (b == -1) b = __lg(x) + 1;
    string s = "";
    for (int k = b - 1; k >= 0; k--) {
        s.push_back((x & (1LL<<k)) ? '1' : '0');
    }
    return s;
}
/* ============================================= */
// CONSTANT
const int INF = 1.05e9;
const ll LINF = 4e18;
const int MOD = 1e9 + 7;
//const int MOD = 998244353;
const int maxn = 2e5 + 3;
```

3.2 Stress

```bash
g++ gen.cpp -o gen.out
g++ ac.cpp -o ac.out
g++ wa.cpp -o wa.out
for ((i=0;;i++))
do
    echo "$i"
    ./gen.out > in.txt
    ./ac.out < in.txt > ac.txt
    ./wa.out < in.txt > wa.txt
    diff ac.txt wa.txt || break
done
```

3.3 PBDS

```cpp
#include <bits/extc++.h>
using namespace __gnu_pbds;

// map
tree<int, int, less<>, rb_tree_tag, tree_order_statistics_node_update> tr;
tr.order_of_key(element);
tr.find_by_order(rank);

// set
tree<int, null_type, less<>, rb_tree_tag, tree_order_statistics_node_update> tr;
tr.order_of_key(element);
tr.find_by_order(rank);

// priority queue
__gnu_pbds::priority_queue<int, less<int> > big_q;  // Big First
__gnu_pbds::priority_queue<int, greater<int> > small_q;  // Small First
q1.join(q2);  // join
```

3.4 Random

```cpp
mt19937 gen(chrono::steady_clock::now().time_since_epoch().count());
#define RANDINT(a, b) uniform_int_distribution<int> (a, b)(rng) // inclusive
#define RANDLL(a, b) uniform_int_distribution<long long>(a, b)(rng)  // inclusive
#define RANDFLOAT(a, b) uniform_real_distribution<float>(a, b)(rng)  // exclusive
#define RANDDOUBLE(a, b) uniform_real_distribution<double>(a, b)(rng)  // exclusive
shuffle(v.begin(), v.end(), gen);
```

4 Python

4.1 I/O

```python
import sys
input = sys.stdin.readline

# Input
def readInt():
    return int(input())
def readList():
    return list(map(int,input().split()))
def readStr():
    s = input()
    return list(s[:len(s) - 1])
def readVars():
    return map(int,input().split())

# Output
sys.stdout.write(string)

# faster
def main():
    pass
main()
```

4.2 Decimal

```python
from decimal import *
getcontext().prec = 2500000
getcontext().Emax = 2500000
a,b = Decimal(input()),Decimal(input())
a*=b
print(a)
```

5 Data Structure

5.1 Segment Tree

```cpp
struct node {
    ll sum, add, mod; int ln;
    node(): sum(0), add(0), mod(0), ln(0) {}
};

struct segT {
    int n;
    vector<ll> ar;
    vector<node> st;

    void init(int _n) {
        n = _n;
        reset(ar, n, 0LL);
        reset(st, n*4);
    }
    void pull(int cl, int cr, int i) {
        st[i].sum = st[cl].sum + st[cr].sum;
    }
    void push(int cl, int cr, int i) {
        ll md = st[i].mod, ad = st[i].add;
        if (md) {
            st[cl].sum = md * st[cl].ln, st[cr].sum =
                md * st[cr].ln;
            st[cl].mod = md, st[cr].mod = md;
            st[i].mod = 0;
        }
        if (ad) {
            st[cl].sum += ad * st[cl].ln, st[cr].sum +=
                 ad * st[cr].ln;
            st[cl].add += ad, st[cr].add += ad;
            st[i].add = 0;
        }
    }
    void build(int l, int r, int i) {
        if (l == r) {
            st[i].sum = ar[l];
            st[i].ln = 1;
            return;
        }
        int mid = (l+r)>>1, cl = i<<1, cr = i<<1|1;
        build(l, mid, cl);
        build(mid + 1, r, cr);
        pull(cl, cr, i);
    }
    void addval(int ql, int qr, ll val, int l, int r,
         int i) {
        if (qr < l || r < ql) return;
        if (ql <= l && r <= qr) {
            st[i].sum += val * st[i].ln;
            st[i].add += val;
            return;
        }
        int mid = (l+r)>>1, cl = i<<1, cr = i<<1|1;
        push(cl, cr, i);
        addval(ql, qr, val, l, mid, cl);
        addval(ql, qr, val, mid + 1, r, cr);
        pull(cl, cr, i);
    }
    void modify(int ql, int qr, ll val, int l, int r,
         int i) {
        if (qr < l || r < ql) return;
        if (ql <= l && r <= qr) {
            st[i].sum = val * st[i].ln;
            st[i].add = 0;
            st[i].mod = val;
            return;
        }
        int mid = (l+r)>>1, cl = i<<1, cr = i<<1|1;
        push(cl, cr, i);
        modify(ql, qr, val, l, mid, cl);
        modify(ql, qr, val, mid+1, r, cr);
        pull(cl, cr, i);
    }
    ll query(int ql, int qr, int l, int r, int i) {
        if (qr < l || r < ql) return 0;
        if (ql <= l && r <= qr) return st[i].sum;
        int mid = (l+r)>>1, cl = i<<1, cr = i<<1|1;
        push(cl, cr, i);
        return (query(ql, qr, l, mid, cl) +
                query(ql, qr, mid+1, r, cr));
    }
};
```

5.2 Heavy Light Decomposition

```cpp
constexpr int maxn=2e5+5;
int arr[(maxn+1)<<2];
#define m ((l+r)>>1)
void build(V<int>& v,int i=1,int l=0,int r=maxn){
    if((int)v.size()<=l) return;
    if(r-l==1){arr[i]=v[l];return;}
    build(v,i<<1,l,m),build(v,i<<1|1,m,r);
    arr[i]=max(arr[i<<1],arr[i<<1|1]);
}
void modify(int p,int k,int i=1,int l=0,int r=maxn){
    if(p<l||r<=p) return;
    if(r-l==1){arr[i]=k;return;}
    if(p<m) modify(p,k,i<<1,l,m);
    else modify(p,k,i<<1|1,m,r);
    arr[i]=max(arr[i<<1],arr[i<<1|1]);
}
int query(int ql,int qr,int i=1,int l=0,int r=maxn){
    if(qr<=l||r<=ql) return 0;
    if(ql<=l&&r<=qr) return arr[i];
    if(qr<=m) return query(ql,qr,i<<1,l,m);
    if(m<=ql) return query(ql,qr,i<<1|1,m,r);
    return max(query(ql,qr,i<<1,l,m),query(ql,qr,i
        <<1|1,m,r));
}
#undef m
inline void solve(){
    int n,q;cin>>n>>q;
    V<int> v(n);
    for(auto& i:v)
        cin>>i;
    V<V<int>> e(n);
    for(int i=1;i<n;i++){
        int a,b;cin>>a>>b,a--,b--;
        e[a].emplace_back(b);
        e[b].emplace_back(a);
    }
    V<int> d(n,0),f(n,0),sz(n,1),son(n,-1);
    F<void(int,int)> dfs1=
    [&](int x,int pre){
        for(auto i:e[x]) if(i!=pre){
            d[i]=d[x]+1,f[i]=x;
            dfs1(i,x),sz[x]+=sz[i];
            if(!~son[x]||sz[son[x]]<sz[i])
                son[x]=i;
        }
    };dfs1(0,0);
    V<int> top(n,0),dfn(n,-1),rnk(n,0);
    F<void(int,int)> dfs2=
    [&](int x,int t){
        static int cnt=0;
        dfn[x]=cnt++,rnk[dfn[x]]=x,top[x]=t;
        if(!~son[x]) return;
        dfs2(son[x],t);
        for(auto i:e[x])
            if(!~dfn[i]) dfs2(i,i);
    };dfs2(0,0);
    V<int> dfnv(n);
    for(int i=0;i<n;i++)
        dfnv[dfn[i]]=v[i];
    build(dfnv);
    while(q--){
        int op,a,b;cin>>op>>a>>b;
        switch(op){
        case 1:{
            modify(dfn[a-1],b);
        }break;
        case 2:{
            a--,b--;
            int ans=0;
            while(top[a]!=top[b]){
                if(d[top[a]]>d[top[b]]) swap(a,b);
                ans=max(ans,query(dfn[top[b]],dfn[b]+1));
                b=f[top[b]];
            }
            if(dfn[a]>dfn[b]) swap(a,b);
            ans=max(ans,query(dfn[a],dfn[b]+1));
            cout<<ans<<endl;
        }break;
        }
    }
}
```

5.3 Skew Heap

```
struct node{
    node *l,*r;
    int v;
    node(int x):v(x){
        l=r=nullptr;
    }
};
node* merge(node* a,node* b){
    if(!a||!b) return a?:b;
//  min heap
    if(a->v>b->v) swap(a,b);
    a->r=merge(a->r,b);
    swap(a->l,a->r);
    return a;
}
```

5.4 Leftist Heap

```
struct node{
    node *l,*r;
    int d, v;
    node(int x):d(1),v(x){
        l=r=nullptr;
    }
};
static inline int d(node* x){return x?x->d:0;}
node* merge(node* a,node* b){
    if(!a||!b) return a?:b;
//  min heap
    if(a->v>b->v) swap(a,b);
    a->r=merge(a->r,b);
    if(d(a->l)<d(a->r))
        swap(a->l,a->r);
    a->d=d(a->r)+1;
    return a;
}
```

5.5 Persistent Treap

```
struct node {
  node *l, *r;
  char c; int v, sz;
  node(char x = '$'): c(x), v(mt()), sz(1) {
    l = r = nullptr;
  }
  node(node* p) {*this = *p;}
  void pull() {
    sz = 1;
    for (auto i : {l, r})
      if (i) sz += i->sz;
  }
} arr[maxn], *ptr = arr;
inline int size(node* p) {return p ? p->sz : 0;}
node* merge(node* a, node* b) {
  if (!a || !b) return a ? : b;
  if (a->v < b->v) {
    node* ret = new(ptr++) node(a);
    ret->r = merge(ret->r, b), ret->pull();
    return ret;
  }
  else {
    node* ret = new(ptr++) node(b);
    ret->l = merge(a, ret->l), ret->pull();
    return ret;
  }
}
P<node*> split(node* p, int k) {
  if (!p) return {nullptr, nullptr};
  if (k >= size(p->l) + 1) {
    auto [a, b] = split(p->r, k - size(p->l) - 1);
    node* ret = new(ptr++) node(p);
    ret->r = a, ret->pull();
    return {ret, b};
  }
  else {
    auto [a, b] = split(p->l, k);
    node* ret = new(ptr++) node(p);
    ret->l = b, ret->pull();
    return {a, ret};
  }
}
```

5.6 Li Chao Tree

```
constexpr int maxn = 5e4 + 5;
struct line {
  ld a, b;
  ld operator()(ld x) {return a * x + b;}
} arr[(maxn + 1) << 2];
bool operator<(line a, line b) {return a.a < b.a;}
#define m ((l+r)>>1)
void insert(line x, int i = 1, int l = 0, int r = maxn) {
  if (r - l == 1) {
    if (x(l) > arr[i](l))
      arr[i] = x;
    return;
  }
  line a = max(arr[i], x), b = min(arr[i], x);
  if (a(m) > b(m))
    arr[i] = a, insert(b, i << 1, l, m);
  else
    arr[i] = b, insert(a, i << 1 | 1, m, r);
}
ld query(int x, int i = 1, int l = 0, int r = maxn) {
  if (x < l || r <= x) return -numeric_limits<ld>::max();
  if (r - l == 1) return arr[i](x);
  return max({arr[i](x), query(x, i << 1, l, m), query(x, i << 1 | 1, m, r)});
}
#undef m
```

5.7 Time Segment Tree

```cpp
constexpr int maxn = 1e5 + 5;
V<P<int>> arr[(maxn + 1) << 2];
V<int> dsu, sz;
V<tuple<int, int, int>> his;
int cnt, q;
int find(int x) {
    return x == dsu[x] ? x : find(dsu[x]);
};
inline bool merge(int x, int y) {
    int a = find(x), b = find(y);
    if (a == b) return false;
    if (sz[a] > sz[b]) swap(a, b);
    his.emplace_back(a, b, sz[b]), dsu[a] = b, sz[b] += sz[a];
    return true;
};
inline void undo() {
    auto [a, b, s] = his.back(); his.pop_back();
    dsu[a] = a, sz[b] = s;
}
#define m ((l + r) >> 1)
void insert(int ql, int qr, P<int> x, int i = 1, int l = 0, int r = q) {
    // debug(ql, qr, x); return;
    if (qr <= l || r <= ql) return;
    if (ql <= l && r <= qr) {arr[i].push_back(x); return;}
    if (qr <= m)
        insert(ql, qr, x, i << 1, l, m);
    else if (m <= ql)
        insert(ql, qr, x, i << 1 | 1, m, r);
    else {
        insert(ql, qr, x, i << 1, l, m);
        insert(ql, qr, x, i << 1 | 1, m, r);
    }
}
void traversal(V<int>& ans, int i = 1, int l = 0, int r = q) {
    int opcnt = 0;
    // debug(i, l, r);
    for (auto [a, b] : arr[i])
        if (merge(a, b))
            opcnt++, cnt--;
    if (r - l == 1) ans[l] = cnt;
    else {
        traversal(ans, i << 1, l, m);
        traversal(ans, i << 1 | 1, m, r);
    }
    while (opcnt--)
        undo(), cnt++;
    arr[i].clear();
}
#undef m
inline void solve() {
    int n, m; cin>>n>>m>>q,q++;
    dsu.resize(cnt = n), sz.assign(n, 1);
    iota(dsu.begin(), dsu.end(), 0);
    // a, b, time, operation
    unordered_map<ll, V<int>> s;
    for (int i = 0; i < m; i++) {
        int a, b; cin>>a>>b;
        if (a > b) swap(a, b);
        s[((ll)a << 32) | b].emplace_back(0);
    }
    for (int i = 1; i < q; i++) {
        int op,a, b;
        cin>>op>>a>>b;
        if (a > b) swap(a, b);
        switch (op) {
        case 1:
            s[((ll)a << 32) | b].push_back(i);
            break;
        case 2:
            auto tmp = s[((ll)a << 32) | b].back();
            s[((ll)a << 32) | b].pop_back();
            insert(tmp, i, P<int> {a, b});
        }
    }
    for (auto [p, v] : s) {
        int a = p >> 32, b = p & -1;
        while (v.size()) {
            insert(v.back(), q, P<int> {a, b});
            v.pop_back();
        }
    }
    V<int> ans(q);
    traversal(ans);
    for (auto i : ans)
        cout<<i<<' ';
    cout<<endl;
}
```

6 DP

6.1 Aliens

```cpp
int n; ll k;
vector<ll> a;
vector<pll> dp[2];
void init() {
  cin >> n >> k;
  Each(i, dp) i.clear(), i.resize(n);
  a.clear(); a.resize(n);
  Each(i, a) cin >> i;
}
pll calc(ll p) {
  dp[0][0] = mp(0, 0);
  dp[1][0] = mp(-a[0], 0);
  FOR(i, 1, n, 1) {
    if (dp[0][i-1].F > dp[1][i-1].F + a[i] - p) {
      dp[0][i] = dp[0][i-1];
    } else if (dp[0][i-1].F < dp[1][i-1].F + a[i] - p) {
      dp[0][i] = mp(dp[1][i-1].F + a[i] - p, dp[1][i-1].S+1);
    } else {
      dp[0][i] = mp(dp[0][i-1].F, min(dp[0][i-1].S, dp[1][i-1].S+1));
    }
    if (dp[0][i-1].F - a[i] > dp[1][i-1].F) {
      dp[1][i] = mp(dp[0][i-1].F - a[i], dp[0][i-1].S);
    } else if (dp[0][i-1].F - a[i] < dp[1][i-1].F) {
      dp[1][i] = dp[1][i-1];
    } else {
      dp[1][i] = mp(dp[1][i-1].F, min(dp[0][i-1].S, dp[1][i-1].S));
    }
  }
  return dp[0][n-1];
}
void solve() {
  ll l = 0, r = 1e7;
  pll res = calc(0);
  if (res.S <= k) return cout << res.F << endl, void();
  while (l < r) {
    ll mid = (l+r)>>1;
    res = calc(mid);
    if (res.S <= k) r = mid;
    else l = mid+1;
  }
  res = calc(l);
  cout << res.F + k*l << endl;
}
```

6.2 SOS DP

```cpp
for (int msk = 0; msk < (1<<n); msk++) {
    for (int i = 1; i <= n; i++) {
        if (msk & (1<<(i - 1))) {
            // dp[msk][i] = dp[msk][i - 1] + dp[msk ^ (1<<(i - 1))][i - 1];
        } else {
            // dp[msk][i] = dp[msk][i - 1];
        }
    }
}
```

7 Graph

7.1 Tree Centroid

```cpp
int n;
vector<vector<int>> G;

pii centroid;
vector<int> sz, mxcc;  // mxcc[u]: max component size after removing u

void dfs(int u, int p) {
    sz[u] = 1;
    for (auto& v : G[u]) {
        if (v == p) continue;
        dfs(v, u);
        sz[u] += sz[v];
        mxcc[u] = max(mxcc[u], sz[v]);
    }
    mxcc[u] = max(mxcc[u], n - sz[u]);
}

void find_centroid() {
    centroid = pii{-1, -1};
    reset(sz, n + 1, 0);
    reset(mxcc, n + 1, 0);
    dfs(1, 1);
    for (int u = 1; u <= n; u++) {
        if (mxcc[u] <= n / 2) {
            if (centroid.first != -1) centroid.second = u;
            else centroid.first = u;
        }
    }
}
```

7.2 Bellman-Ford + SPFA

```cpp
int n, m;

// Graph
vector<vector<pair<int, ll> > > g;
vector<ll> dis;
vector<bool> negCycle;

// SPFA
vector<int> rlx;
queue<int> q;
vector<bool> inq;
vector<int> pa;
void SPFA(vector<int>& src) {
    dis.assign(n+1, LINF);
    negCycle.assign(n+1, false);
    rlx.assign(n+1, 0);
    while (!q.empty()) q.pop();
    inq.assign(n+1, false);
    pa.assign(n+1, -1);

    for (auto& s : src) {
        dis[s] = 0;
        q.push(s); inq[s] = true;
    }

    while (!q.empty()) {
        int u = q.front();
        q.pop(); inq[u] = false;
        if (rlx[u] >= n) {
            negCycle[u] = true;
        }
        else for (auto& e : g[u]) {
            int v = e.first;
            ll w = e.second;
            if (dis[v] > dis[u] + w) {
                dis[v] = dis[u] + w;
                rlx[v] = rlx[u] + 1;
                pa[v] = u;
                if (!inq[v]) {
                    q.push(v);
                    inq[v] = true;
} } } } }


// Bellman-Ford
queue<int> q;
vector<int> pa;
void BellmanFord(vector<int>& src) {
    dis.assign(n+1, LINF);
    negCycle.assign(n+1, false);
    pa.assign(n+1, -1);

    for (auto& s : src) dis[s] = 0;

    for (int rlx = 1; rlx <= n; rlx++) {
        for (int u = 1; u <= n; u++) {
            if (dis[u] == LINF) continue;  // Important !!
            for (auto& e : g[u]) {
                int v = e.first; ll w = e.second;
                if (dis[v] > dis[u] + w) {
                    dis[v] = dis[u] + w;
                    pa[v] = u;
                    if (rlx == n) negCycle[v] = true;
} } } } }


// Negative Cycle Detection
void NegCycleDetect() {
/* No Neg Cycle: NO
Exist Any Neg Cycle:
YES
v0 v1 v2 ... vk v0 */

    vector<int> src;
    for (int i = 1; i <= n; i++)
        src.emplace_back(i);

    SPFA(src);
    // BellmanFord(src);

    int ptr = -1;
    for (int i = 1; i <= n; i++) if (negCycle[i])
        { ptr = i; break; }

    if (ptr == -1) { return cout << "NO" << endl, void(); }

    cout << "YES\n";
    vector<int> ans;
    vector<bool> vis(n+1, false);

    while (true) {
        ans.emplace_back(ptr);
        if (vis[ptr]) break;
        vis[ptr] = true;
        ptr = pa[ptr];
    }
    reverse(ans.begin(), ans.end());

    vis.assign(n+1, false);
    for (auto& x : ans) {
        cout << x << ' ';
        if (vis[x]) break;
        vis[x] = true;
    }
    cout << endl;
}

// Distance Calculation
void calcDis(int s) {
    vector<int> src;
    src.emplace_back(s);
    SPFA(src);
    // BellmanFord(src);

    while (!q.empty()) q.pop();
    for (int i = 1; i <= n; i++)
        if (negCycle[i]) q.push(i);

    while (!q.empty()) {
        int u = q.front(); q.pop();
        for (auto& e : g[u]) {
            int v = e.first;
            if (!negCycle[v]) {
                q.push(v);
                negCycle[v] = true;
} } } }
```

7.3 BCC - AP

```
int n, m;
int low[maxn], dfn[maxn], instp;
vector<int> E, g[maxn];
bitset<maxn> isap;
bitset<maxm> vis;
stack<int> stk;
int bccnt;
vector<int> bcc[maxn];
inline void popout(int u) {
  bccnt++;
  bcc[bccnt].emplace_back(u);
  while (!stk.empty()) {
    int v = stk.top();
    if (u == v) break;
    stk.pop();
    bcc[bccnt].emplace_back(v);
  }
}
void dfs(int u, bool rt = 0) {
  stk.push(u);
  low[u] = dfn[u] = ++instp;
  int kid = 0;
  Each(e, g[u]) {
    if (vis[e]) continue;
    vis[e] = true;
    int v = E[e]^u;
    if (!dfn[v]) {
      // tree edge
      kid++; dfs(v);
      low[u] = min(low[u], low[v]);
      if (!rt && low[v] >= dfn[u]) {
        // bcc found: u is ap
        isap[u] = true;
        popout(u);
      }
    } else {
      // back edge
      low[u] = min(low[u], dfn[v]);
    }
  }
  // special case: root
  if (rt) {
    if (kid > 1) isap[u] = true;
    popout(u);
  }
}
void init() {
  cin >> n >> m;
  fill(low, low+maxn, INF);
  REP(i, m) {
    int u, v;
    cin >> u >> v;
    g[u].emplace_back(i);
    g[v].emplace_back(i);
    E.emplace_back(u^v);
  }
}
void solve() {
  FOR(i, 1, n+1, 1) {
    if (!dfn[i]) dfs(i, true);
  }
  vector<int> ans;
  int cnt = 0;
  FOR(i, 1, n+1, 1) {
    if (isap[i]) cnt++, ans.emplace_back(i);
  }
  cout << cnt << endl;
  Each(i, ans) cout << i << ' ';
  cout << endl;
}
```

7.4 BCC - Bridge

```
int n, m;
vector<int> g[maxn], E;
int low[maxn], dfn[maxn], instp;
int bccnt, bccid[maxn];
stack<int> stk;
bitset<maxm> vis, isbrg;
void init() {
  cin >> n >> m;
  REP(i, m) {
    int u, v;
    cin >> u >> v;
    E.emplace_back(u^v);
    g[u].emplace_back(i);
    g[v].emplace_back(i);
  }
  fill(low, low+maxn, INF);
}
void popout(int u) {
  bccnt++;
  while (!stk.empty()) {
    int v = stk.top();
    if (v == u) break;
    stk.pop();
    bccid[v] = bccnt;
  }
}
void dfs(int u) {
  stk.push(u);
  low[u] = dfn[u] = ++instp;

  Each(e, g[u]) {
    if (vis[e]) continue;
    vis[e] = true;

    int v = E[e]^u;
    if (dfn[v]) {
      // back edge
      low[u] = min(low[u], dfn[v]);
    } else {
      // tree edge
      dfs(v);
      low[u] = min(low[u], low[v]);
      if (low[v] == dfn[v]) {
        isbrg[e] = true;
        popout(u);
      }
    }
  }
}
void solve() {
  FOR(i, 1, n+1, 1) {
    if (!dfn[i]) dfs(i);
  }
  vector<pii> ans;
  vis.reset();
  FOR(u, 1, n+1, 1) {
    Each(e, g[u]) {
      if (!isbrg[e] || vis[e]) continue;
      vis[e] = true;
      int v = E[e]^u;
      ans.emplace_back(mp(u, v));
    }
  }
  cout << (int)ans.size() << endl;
  Each(e, ans) cout << e.F << ' ' << e.S << endl;
}
```

7.5 SCC - Tarjan

```
// 2-SAT
vector<int> E, g[maxn];  // 1~n, n+1~2n
int low[maxn], in[maxn], instp;
int sccnt, sccid[maxn];

stack<int> stk;
bitset<maxn> ins, vis;

int n, m;

void init() {
    cin >> m >> n;
    E.clear();
```

```cpp
    fill(g, g+maxn, vector<int>());
    fill(low, low+maxn, INF);
    memset(in, 0, sizeof(in));
    instp = 1;
    sccnt = 0;
    memset(sccid, 0, sizeof(sccid));
    ins.reset();
    vis.reset();
}

inline int no(int u) {
    return (u > n ? u-n : u+n);
}

int ecnt = 0;
inline void clause(int u, int v) {
    E.eb(no(u)^v);
    g[no(u)].eb(ecnt++);
    E.eb(no(v)^u);
    g[no(v)].eb(ecnt++);
}

void dfs(int u) {
    in[u] = instp++;
    low[u] = in[u];
    stk.push(u);
    ins[u] = true;

    Each(e, g[u]) {
        if (vis[e]) continue;
        vis[e] = true;

        int v = E[e]^u;
        if (ins[v]) low[u] = min(low[u], in[v]);
        else if (!in[v]) {
            dfs(v);
            low[u] = min(low[u], low[v]);
        }
    }

    if (low[u] == in[u]) {
        sccnt++;
        while (!stk.empty()) {
            int v = stk.top();
            stk.pop();
            ins[v] = false;
            sccid[v] = sccnt;
            if (u == v) break;
        }
    }
}


int main() {
    WiwiHorz
    init();

    REP(i, m) {
        char su, sv;
        int u, v;
        cin >> su >> u >> sv >> v;
        if (su == '-') u = no(u);
        if (sv == '-') v = no(v);
        clause(u, v);
    }

    FOR(i, 1, 2*n+1, 1) {
        if (!in[i]) dfs(i);
    }

    FOR(u, 1, n+1, 1) {
        int du = no(u);
        if (sccid[u] == sccid[du]) {
            return cout << "IMPOSSIBLE\n", 0;
        }
    }

    FOR(u, 1, n+1, 1) {
        int du = no(u);
        cout << (sccid[u] < sccid[du] ? '+' : '-') << ' ';
    }
    cout << endl;

    return 0;
}
```

7.6 Eulerian Path - Undir

```cpp
// from 1 to n
#define gg return cout << "IMPOSSIBLE\n", void();

int n, m;
vector<int> g[maxn];
bitset<maxn> inodd;

void init() {
cin >> n >> m;
inodd.reset();
for (int i = 0; i < m; i++) {
  int u, v; cin >> u >> v;
  inodd[u] = inodd[u] ^ true;
  inodd[v] = inodd[v] ^ true;
  g[u].emplace_back(v);
  g[v].emplace_back(u);
} }
stack<int> stk;
void dfs(int u) {
    while (!g[u].empty()) {
        int v = g[u].back();
        g[u].pop_back();
        dfs(v);
    }
stk.push(u);}
```

7.7 Eulerian Path - Dir

```cpp
// from node 1 to node n
#define gg return cout << "IMPOSSIBLE\n", 0

int n, m;
vector<int> g[maxn];
stack<int> stk;
int in[maxn], out[maxn];

void init() {
cin >> n >> m;
for (int i = 0; i < m; i++) {
  int u, v; cin >> u >> v;
  g[u].emplace_back(v);
  out[u]++, in[v]++;
}
for (int i = 1; i <= n; i++) {
  if (i == 1 && out[i]-in[i] != 1) gg;
  if (i == n && in[i]-out[i] != 1) gg;
  if (i != 1 && i != n && in[i] != out[i]) gg;
} }
void dfs(int u) {
    while (!g[u].empty()) {
        int v = g[u].back();
        g[u].pop_back();
        dfs(v);
    }
    stk.push(u);
}
void solve() {
  dfs(1)
    for (int i = 1; i <= n; i++)
        if ((int)g[i].size()) gg;
    while (!stk.empty()) {
        int u = stk.top();
        stk.pop();
        cout << u << ' ';
} }
```

7.8 Hamilton Path

```cpp
// top down DP
// Be Aware Of Multiple Edges
int n, m;
ll dp[maxn][1<<maxn];
```

```
int adj[maxn][maxn];

void init() {
    cin >> n >> m;
    fill(dp[0], dp[maxn-1]+(1<<maxn), -1);
}

void DP(int i, int msk) {
    if (dp[i][msk] != -1) return;
    dp[i][msk] = 0;
    REP(j, n) if (j != i && (msk & (1<<j)) && adj[j][i]) {
        int sub = msk ^ (1<<i);
        if (dp[j][sub] == -1) DP(j, sub);
        dp[i][msk] += dp[j][sub] * adj[j][i];
        if (dp[i][msk] >= MOD) dp[i][msk] %= MOD;
    }
}


int main() {
    WiwiHorz
    init();

    REP(i, m) {
        int u, v;
        cin >> u >> v;
        if (u == v) continue;
        adj[--u][--v]++;
    }

    dp[0][1] = 1;
    FOR(i, 1, n, 1) {
        dp[i][1] = 0;
        dp[i][1|(1<<i)] = adj[0][i];
    }
    FOR(msk, 1, (1<<n), 1) {
        if (msk == 1) continue;
        dp[0][msk] = 0;
    }


    DP(n-1, (1<<n)-1);
    cout << dp[n-1][(1<<n)-1] << endl;

    return 0;
}
```

7.9 Kth 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,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;
```

7.10 System of Difference Constraints

```
vector<vector<pair<int, ll>>> G;
void add(int u, int v, ll w) {
    G[u].emplace_back(make_pair(v, w));
}
```

- $x_u - x_v \le c \Rightarrow$ `add(v, u, c)`
- $x_u - x_v \ge c \Rightarrow$ `add(u, v, -c)`
- $x_u - x_v = c \Rightarrow$ `add(v, u, c), add(u, v -c)`
- $x_u \ge c \Rightarrow$ add super vertex $x_0 = 0$, then $x_u - x_0 \ge c \Rightarrow$ `add(u, 0, -c)`
- Don’t for get non-negative constraints for every variable if specified implicitly.
- Interval sum $\Rightarrow$ Use prefix sum to transform into differential constraints. Don’t for get $S_{i+1} - S_i \ge 0$ if x_i needs to be non-negative.
- $\frac{x_u}{x_v} \le c \Rightarrow \log x_u - \log x_v \le \log c$

8 String

8.1 Rolling Hash

```
const ll C = 27;
inline int id(char c) {return c-'a'+1;}
struct RollingHash {
    string s; int n; ll mod;
    vector<ll> Cexp, hs;
    RollingHash(string& _s, ll _mod):
        s(_s), n((int)_s.size()), mod(_mod)
    {
        Cexp.assign(n, 0);
        hs.assign(n, 0);
        Cexp[0] = 1;
        for (int i = 1; i < n; i++) {
            Cexp[i] = Cexp[i-1] * C;
            if (Cexp[i] >= mod) Cexp[i] %= mod;
        }
        hs[0] = id(s[0]);
        for (int i = 1; i < n; i++) {
            hs[i] = hs[i-1] * C + id(s[i]);
            if (hs[i] >= mod) hs[i] %= mod;
    } }
    inline ll query(int l, int r) {
        ll res = hs[r] - (l ? hs[l-1] * Cexp[r-l+1] : 0);
        res = (res % mod + mod) % mod;
        return res; }
};
```

8.2 Trie

```
struct node {
    int c[26]; ll cnt;
    node(): cnt(0) {memset(c, 0, sizeof(c));}
    node(ll x): cnt(x) {memset(c, 0, sizeof(c));}
};
struct Trie {
    vector<node> t;
    void init() {
        t.clear();
        t.emplace_back(node());
    }
    void insert(string s) { int ptr = 0;
        for (auto& i : s) {
            if (!t[ptr].c[i-'a']) {
                t.emplace_back(node());
                t[ptr].c[i-'a'] = (int)t.size()-1; }
            ptr = t[ptr].c[i-'a']; }
        t[ptr].cnt++; }
} trie;
```

8.3 KMP

```
int n, m;
string s, p;
vector<int> f;
void build() {
  f.clear(); f.resize(m, 0);
  int ptr = 0; for (int i = 1; i < m; i++) {
    while (ptr && p[i] != p[ptr]) ptr = f[ptr-1];
    if (p[i] == p[ptr]) ptr++;
    f[i] = ptr;
}}
void init() {
  cin >> s >> p;
  n = (int)s.size();
  m = (int)p.size();
  build(); }
void solve() {
  int ans = 0, pi = 0;
  for (int si = 0; si < n; si++) {
    while (pi && s[si] != p[pi]) pi = f[pi-1];
    if (s[si] == p[pi]) pi++;
    if (pi == m) ans++, pi = f[pi-1];
  }
cout << ans << endl; }
```

8.4 Z Value

```
string is, it, s;
int n; vector<int> z;
void init() {
    cin >> is >> it;
    s = it+'0'+is;
    n = (int)s.size();
    z.resize(n, 0); }
void solve() {
    int ans = 0; z[0] = n;
    for (int i = 1, l = 0, r = 0; i < n; i++) {
        if (i <= r) z[i] = min(z[i-l], r-i+1);
        while (i+z[i] < n && s[z[i]] == s[i+z[i]]) z[i]++;
        if (i+z[i]-1 > r) l = i, r = i+z[i]-1;
        if (z[i] == (int)it.size()) ans++;
    }
    cout << ans << endl; }
```

8.5 Manacher

```
int n; string S, s;
vector<int> m;
void manacher() {
s.clear(); s.resize(2*n+1, '.');
for (int i = 0, j = 1; i < n; i++, j += 2) s[j] = S[i];
m.clear(); m.resize(2*n+1, 0);
// m[i] := max k such that s[i-k, i+k] is palindrome
int mx = 0, mxk = 0;
for (int i = 1; i < 2*n+1; i++) {
  if (mx-(i-mx) >= 0) m[i] = min(m[mx-(i-mx)], mx+mxk-i);
  while (0 <= i-m[i]-1 && i+m[i]+1 < 2*n+1 &&
       s[i-m[i]-1] == s[i+m[i]+1]) m[i]++;
  if (i+m[i] > mx+mxk) mx = i, mxk = m[i];
} }
void init() { cin >> S; n = (int)S.size(); }
void solve() {
  manacher();
  int mx = 0, ptr = 0;
  for (int i = 0; i < 2*n+1; i++) if (mx < m[i])
    { mx = m[i]; ptr = i; }
    for (int i = ptr-mx; i <= ptr+mx; i++)
        if (s[i] != '.') cout << s[i];
cout << endl; }
```

8.6 Suffix Array

```
#define F first
#define S second
struct SuffixArray {  // don't forget s += "$";
    int n; string s;
    vector<int> suf, lcp, rk;
    vector<int> cnt, pos;
    vector<pair<pii, int> > buc[2];
    void init(string _s) {
        s = _s; n = (int)s.size();
// resize(n): suf, rk, cnt, pos, lcp, buc[0~1]
    }
    void radix_sort() {
        for (int t : {0, 1}) {
            fill(cnt.begin(), cnt.end(), 0);
            for (auto& i : buc[t]) cnt[ (t ? i.F.F : i.F.S) ]++;
            for (int i = 0; i < n; i++)
                pos[i] = (!i ? 0 : pos[i-1] + cnt[i-1]);
            for (auto& i : buc[t])
                buc[t^1][pos[ (t ? i.F.F : i.F.S) ]++] = i;
    }}
    bool fill_suf() {
        bool end = true;
        for (int i = 0; i < n; i++) suf[i] = buc[0][i].S;
        rk[suf[0]] = 0;
        for (int i = 1; i < n; i++) {
            int dif = (buc[0][i].F != buc[0][i-1].F);
            end &= dif;
            rk[suf[i]] = rk[suf[i-1]] + dif;
        } return end;
    }
    void sa() {
        for (int i = 0; i < n; i++)
            buc[0][i] = make_pair(make_pair(s[i], s[i]), i);
        sort(buc[0].begin(), buc[0].end());
        if (fill_suf()) return;
        for (int k = 0; (1<<k) < n; k++) {
            for (int i = 0; i < n; i++)
                buc[0][i] = make_pair(make_pair(rk[i], rk[(i + (1<<k)) % n]), i);
            radix_sort();
            if (fill_suf()) return;
    }}
    void LCP() {  int k = 0;
        for (int i = 0; i < n-1; i++) {
            if (rk[i] == 0) continue;
            int pi = rk[i];
            int j = suf[pi-1];
            while (i+k < n && j+k < n && s[i+k] == s[j+k]) k++;
            lcp[pi] = k;
            k = max(k-1, 0);
    }}
};
SuffixArray suffixarray;
```

8.7 SA-IS

```
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];
  int _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],RA[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);
  memcpy(H,sa.hei+1,len<<2); memcpy(SA,sa._sa+1,len<<2);
  for(int i=0;i<len;i++) RA[i]=sa.r[i]-1;
  // resulting height, sa array \in [0,len)
}
```

8.8 Minimum Rotation

```
//rotate(begin(s), begin(s)+minRotation(s), end(s))
int minRotation(string s) {
int a = 0, n = s.size(); s += s;
for(int b = 0; b < n; b++) for(int k = 0; k < n; k++) {
    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; }
```

8.9 Aho Corasick

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

9 Geometry

9.1 Basic Operations

```
typedef long long T;
// typedef long double T;
const long double eps = 1e-8;

short sgn(T x) {
    if (abs(x) < eps) return 0;
    return x < 0 ? -1 : 1;
}

struct Pt {
T x, y;
Pt(T _x=0, T _y=0):x(_x), y(_y) {}
Pt operator+(Pt a) { return Pt(x+a.x, y+a.y); }
Pt operator-(Pt a) { return Pt(x-a.x, y-a.y); }
Pt operator*(T a)  { return Pt(x*a, y*a); }
Pt operator/(T a)  { return Pt(x/a, y/a); }
T operator*(Pt a)  { return x*a.x + y*a.y; }
T operator^(Pt a)  { return x*a.y - y*a.x; }
bool operator<(Pt a)
    { return x < a.x || (x == a.x && y < a.y); }
//return sgn(x-a.x) < 0 || (sgn(x-a.x) == 0 && sgn(y-a.y) < 0); }
bool operator==(Pt a)
    { return sgn(x-a.x) == 0 && sgn(y-a.y) == 0; }
};

Pt mv(Pt a, Pt b) { return b-a; }
T len2(Pt a) { return a*a; }
T dis2(Pt a, Pt b) { return len2(b-a); }

short ori(Pt a, Pt b) { return ((a^b)>0) - ((a^b)<0); }
bool onseg(Pt p, Pt l1, Pt l2) {
    Pt a = mv(p, l1), b = mv(p, l2);
    return ((a^b) == 0) && ((a*b) <= 0);
}
```

9.2 InPoly

```
short inPoly(Pt p) {
// 0=Bound 1=In -1=Out
REP(i, n) if (onseg(p, E[i], E[(i+1)%n])) return 0;
int cnt = 0;
REP(i, n) if (banana(p, Pt(p.x+1, p.y+2e9),
                     E[i], E[(i+1)%n])) cnt ^= 1;
return (cnt ? 1 : -1);
}
```

9.3 Sort by Angle

```
int ud(Pt a) {  // up or down half plane
    if (a.y > 0) return 0;
    if (a.y < 0) return 1;
    return (a.x >= 0 ? 0 : 1);
}
sort(ALL(E), [&](const Pt& a, const Pt& b){
    if (ud(a) != ud(b)) return ud(a) < ud(b);
    return (a^b) > 0;
});
```

9.4 Line Intersect Check

```
inline bool banana(Pt p1, Pt p2, Pt q1, Pt q2) {
if (onseg(p1, q1, q2) || onseg(p2, q1, q2) ||
    onseg(q1, p1, p2) || onseg(q2, p1, p2)) {
    return true;
}
Pt p = mv(p1, p2), q = mv(q1, q2);
return (ori(p, mv(p1, q1)) * ori(p, mv(p1, q2)) < 0 &&
        ori(q, mv(q1, p1)) * ori(q, mv(q1, p2)) < 0);
}
```

9.5 Line Intersection

```
// T: long double
Pt bananaPoint(Pt p1, Pt p2, Pt q1, Pt q2) {
if (onseg(q1, p1, p2)) return q1;
if (onseg(q2, p1, p2)) return q2;
if (onseg(p1, q1, q2)) return p1;
if (onseg(p2, q1, q2)) return p2;
double s = abs(mv(p1, p2) ^ mv(p1, q1));
double t = abs(mv(p1, p2) ^ mv(p1, q2));
return q2 * (s/(s+t)) + q1 * (t/(s+t));
}
```

9.6 Convex Hull

```
vector<Pt> hull;
void convexHull() {
hull.clear(); sort(ALL(E));
REP(t, 2) {
    int b = SZ(hull);
    Each(ei, E) {
        while (SZ(hull) - b >= 2 &&
               ori(mv(hull[SZ(hull)-2], hull.back()),
                   mv(hull[SZ(hull)-2], ei)) == -1) {
            hull.pop_back();
        }
        hull.eb(ei);
    }
    hull.pop_back();
    reverse(ALL(E));
} }
```

9.7 Lower Concave Hull

```
struct Line {
  mutable ll m, b, p;
  bool operator<(const Line& o) const { return m < o.m; }
  bool operator<(ll x) const { return p < x; }
};

struct LineContainer : multiset<Line, less<>> {
  // (for doubles, use inf = 1/.0, div(a,b) = a/b)
  const ll inf = LLONG_MAX;
  ll div(ll a, ll b) { // floored division
    return a / b - ((a ^ b) < 0 && a % b); }
  bool isect(iterator x, iterator y) {
    if (y == end()) { x->p = inf; return false; }
    if (x->m == y->m) x->p = x->b > y->b ? inf : -inf;
    else x->p = div(y->b - x->b, x->m - y->m);
    return x->p >= y->p;
  }
  void add(ll m, ll b) {
    auto z = insert({m, b, 0}), y = z++, x = y;
    while (isect(y, z)) z = erase(z);
    if (x != begin() && isect(--x, y)) isect(x, y = erase(y));
    while ((y = x) != begin() && (--x)->p >= y->p)
      isect(x, erase(y));
  }
  ll query(ll x) {
    assert(!empty());
    auto l = *lower_bound(x);
    return l.m * x + l.b;
  }
};
```

9.8 Polygon Area

```
T dbarea(vector<Pt>& e) {
ll res = 0;
REP(i, SZ(e)) res += e[i]^e[(i+1)%SZ(e)];
return abs(res);
}
```

9.9 Pick's Theorem

Consider a polygon which vertices are all lattice points.

Let i = number of points inside the polygon.

Let b = number of points on the boundary of the polygon.

Then we have the following formula:

$$Area = i + \frac{b}{2} - 1$$

9.10 Minimum Enclosing Circle

```
Pt circumcenter(Pt A, Pt B, Pt C) {
// a1(x-A.x) + b1(y-A.y) = c1
// a2(x-A.x) + b2(y-A.y) = c2
// solve using Cramer's rule
T a1 = B.x-A.x, b1 = B.y-A.y, c1 = dis2(A, B)/2.0;
T a2 = C.x-A.x, b2 = C.y-A.y, c2 = dis2(A, C)/2.0;
T D  = Pt(a1, b1) ^ Pt(a2, b2);
T Dx = Pt(c1, b1) ^ Pt(c2, b2);
T Dy = Pt(a1, c1) ^ Pt(a2, c2);
if (D == 0) return Pt(-INF, -INF);
return A + Pt(Dx/D, Dy/D);
}
Pt center; T r2;
void minEncloseCircle() {
mt19937 gen(chrono::steady_clock::now().time_since_epoch().count());
shuffle(ALL(E), gen);
center = E[0], r2 = 0;

for (int i = 0; i < n; i++) {
    if (dis2(center, E[i]) <= r2) continue;
    center = E[i], r2 = 0;
    for (int j = 0; j < i; j++) {
        if (dis2(center, E[j]) <= r2) continue;
        center = (E[i] + E[j]) / 2.0;
        r2 = dis2(center, E[i]);
        for (int k = 0; k < j; k++) {
            if (dis2(center, E[k]) <= r2) continue;
            center = circumcenter(E[i], E[j], E[k]);
            r2 = dis2(center, E[i]);
        }
    }
} }
```

9.11 PolyUnion

```
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;
}
```

9.12 Minkowski Sum

```
/* convex hull Minkowski Sum*/
#define INF 100000000000000LL
int pos( const Pt& tp ){
  if( tp.Y == 0 ) return tp.X > 0 ? 0 : 1;
  return tp.Y > 0 ? 0 : 1;
}
#define N 300030
Pt pt[ N ], qt[ N ], rt[ N ];
LL Lx,Rx;
int dn,un;
inline bool cmp( Pt a, Pt b ){
  int pa=pos( a ),pb=pos( b );
  if(pa==pb) return (a^b)>0;
  return pa<pb;
}
int minkowskiSum(int n,int m){
  int i,j,r,p,q,fi,fj;
  for(i=1,p=0;i<n;i++){
    if( pt[i].Y<pt[p].Y ||
        (pt[i].Y==pt[p].Y && pt[i].X<pt[p].X) ) p=i; }
  for(i=1,q=0;i<m;i++){
    if( qt[i].Y<qt[q].Y ||
        (qt[i].Y==qt[q].Y && qt[i].X<qt[q].X) ) q=i; }
  rt[0]=pt[p]+qt[q];
  r=1; i=p; j=q; fi=fj=0;
  while(1){
    if((fj&&j==q) ||
       ( (!fi||i!=p) &&
         cmp(pt[(p+1)%n]-pt[p],qt[(q+1)%m]-qt[q]) ) ){
      rt[r]=rt[r-1]+pt[(p+1)%n]-pt[p];
      p=(p+1)%n;
      fi=1;
    }else{
      rt[r]=rt[r-1]+qt[(q+1)%m]-qt[q];
      q=(q+1)%m;
      fj=1;
    }
    if(r<=1 || ((rt[r]-rt[r-1])^(rt[r-1]-rt[r-2]))!=0) r++;
    else rt[r-1]=rt[r];
    if(i==p && j==q) break;
  }
  return r-1;
}
void initInConvex(int n){
  int i,p,q;
  LL Ly,Ry;
  Lx=INF; Rx=-INF;
  for(i=0;i<n;i++){
    if(pt[i].X<Lx) Lx=pt[i].X;
    if(pt[i].X>Rx) Rx=pt[i].X;
  }
  Ly=Ry=INF;
  for(i=0;i<n;i++){
    if(pt[i].X==Lx && pt[i].Y<Ly){ Ly=pt[i].Y; p=i; }
    if(pt[i].X==Rx && pt[i].Y<Ry){ Ry=pt[i].Y; q=i; }
  }
  for(dn=0,i=p;i!=q;i=(i+1)%n){ qt[dn++]=pt[i]; }
  qt[dn]=pt[q]; Ly=Ry=-INF;
  for(i=0;i<n;i++){
    if(pt[i].X==Lx && pt[i].Y>Ly){ Ly=pt[i].Y; p=i; }
    if(pt[i].X==Rx && pt[i].Y>Ry){ Ry=pt[i].Y; q=i; }
  }
  for(un=0,i=p;i!=q;i=(i+n-1)%n){ rt[un++]=pt[i]; }
  rt[un]=pt[q];
}
inline int inConvex(Pt p){
  int L,R,M;
  if(p.X<Lx || p.X>Rx) return 0;
  L=0;R=dn;
  while(L<R-1){ M=(L+R)/2;
    if(p.X<qt[M].X) R=M; else L=M; }
    if(tri(qt[L],qt[R],p)<0) return 0;
    L=0;R=un;
    while(L<R-1){ M=(L+R)/2;
      if(p.X<rt[M].X) R=M; else L=M; }
      if(tri(rt[L],rt[R],p)>0) return 0;
      return 1;
}
int main(){
  int n,m,i;
  Pt p;
  scanf("%d",&n);
  for(i=0;i<n;i++) scanf("%lld%lld",&pt[i].X,&pt[i].Y);
  scanf("%d",&m);
  for(i=0;i<m;i++) scanf("%lld%lld",&qt[i].X,&qt[i].Y);
  n=minkowskiSum(n,m);
  for(i=0;i<n;i++) pt[i]=rt[i];
  scanf("%d",&m);
  for(i=0;i<m;i++) scanf("%lld%lld",&qt[i].X,&qt[i].Y);
  n=minkowskiSum(n,m);
  for(i=0;i<n;i++) pt[i]=rt[i];
  initInConvex(n);
  scanf("%d",&m);
  for(i=0;i<m;i++){
    scanf("%lld %lld",&p.X,&p.Y);
    p.X*=3; p.Y*=3;
    puts(inConvex(p)?"YES":"NO");
  }
}
```

10 Number Theory

10.1 Basic

```
const int maxc = 5e5;
ll pw(ll a, ll n) {
    ll res = 1;
    while (n) {
        if (n & 1) res = res * a % MOD;
        a = a * a % MOD;
        n >>= 1;
    }
    return res;
}

vector<ll> fac, ifac;
void build_fac() {
    reset(fac, maxc + 1, 1LL);
    reset(ifac, maxc + 1, 1LL);
    for (int x = 2; x <= maxc; x++) {
        fac[x] = x * fac[x - 1] % MOD;
        ifac[x] = pw(fac[x], MOD - 2);
    }
}

ll C(ll n, ll k) {
    if (n < k) return 0LL;
    return fac[n] * ifac[n - k] % MOD * ifac[k] % MOD;
}
```

10.2 Prime Seive and Defactor

```
const int maxc = 1e6 + 1;
vector<int> lpf;
vector<int> prime;

void seive() {
    prime.clear();
    lpf.resize(maxc, 1);
    for (int i = 2; i < maxc; i++) {
        if (lpf[i] == 1) {
            lpf[i] = i;
            prime.emplace_back(i);
        }
        for (auto& j : prime) {
            if (i * j >= maxc) break;
            lpf[i * j] = j;
            if (j == lpf[i]) break;
} } }
vector<pii> fac;
void defactor(int u) {
    fac.clear();
    while (u > 1) {
        int d = lpf[u];
        fac.emplace_back(make_pair(d, 0));
        while (u % d == 0) {
            u /= d;
            fac.back().second++;
} } }
```

10.3 Harmonic Series

```
// O(n log n)
for (int i = 1; i <= n; i++) {
    for (int j = i; j <= n; j += i) {
        // O(1) code
    }
}

// PIE
// given array a[0], a[1], ..., a[n - 1]
// calculate dp[x] = number of pairs (a[i], a[j]) such that
//                   gcd(a[i], a[j]) = x   // (i < j)
//
// idea: let mc(x) = # of y s.t. x|y
//            f(x) = # of pairs s.t. gcd(a[i], a[j]) >= x
//            f(x) = C(mc(x), 2)
//           dp[x] = f(x) - sum(dp[y], x < y and x|y)
const int maxc = 1e6;
vector<int> cnt(maxc + 1, 0), dp(maxc + 1, 0);
for (int i = 0; i < n; i++)
    cnt[a[i]]++;

for (int x = maxc; x >= 1; x--) {
    ll cnt_mul = 0;  // number of multiples of x
    for (int y = x; y <= maxc; y += x)
        cnt_mul += cnt[y];

```

```cpp
    dp[x] = cnt_mul * (cnt_mul - 1) / 2;  // number of pairs that are divisible by x
    for (int y = x + x; y <= maxc; y += x)
        dp[x] -= dp[y];  // PIE: subtract all dp[y] for y > x and x|y
}
```

10.4 Count Number of Divisors

```cpp
// Count the number of divisors for all  x <= 10^6
const int maxc = 1e6;
vector<int> facs;

void find_all_divisors() {
    facs.clear(); facs.resize(maxc + 1, 0);
    for (int x = 1; x <= maxc; x++) {
        for (int y = x; y <= maxc; y += x) {
            facs[y]++;
        }
    }
}
```

10.5 數論分塊

```cpp
/*
n = 17
  i:  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17
n/i: 17  8  5  4  3  2  2  2  1  1  1  1  1  1  1  1  1
                     ^     ^
                   L(2)   R(2)

L(x) :=  left bound for n/i = x
R(x) := right bound for n/i = x

====== FORMULA ======
>>> R = n / (n/L) <<<
=====================

Example: L(2) = 6
         R(2) = 17 / (17 / 6)
              = 17 / 2
              = 8
*/
// ======= CODE ========

for (ll l = 1, r = 1, q = n; l <= n; l = r + 1) {
    q = n/l;
    r = n/q;
    // Process your code here
}
// q, l, r: 17 1 1
// q, l, r: 8 2 2
// q, l, r: 5 3 3
// q, l, r: 4 4 4
// q, l, r: 3 5 5
// q, l, r: 2 6 8
// q, l, r: 1 9 17
```

10.6 Pollard's rho

```python
from itertools import count
from math import gcd
from sys import stdin

for s in stdin:
    number, x = int(s), 2
    break2 = False
    for cycle in count(1):
        y = x
        if break2:
            break
        for i in range(1 << cycle):
            x = (x * x + 1) % number
            factor = gcd(x - y, number)
            if factor > 1:
                print(factor)
                break2 = True
                break
```

10.7 Miller Rabin

```cpp
// 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
bool witness(ll a,ll n,ll u,int t){
  if(!(a%=n)) 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=100) {
  // iterate s times of witness on n
  // return 1 if prime, 0 otherwise
  if(n<2) return 0;
  if(!(n&1)) return n == 2;
  ll u=n-1; int t=0;
  while(!(u&1)) u>>=1, t++;
  while(s--){
    ll a=randll()%(n-1)+1;
    if(witness(a,n,u,t)) return 0;
  }
  return 1;
}
```

10.8 Fast Power

Note: $a^n \equiv a^{(n \ mod \ (p-1))} (mod \ p)$

10.9 Extend GCD

```cpp
ll GCD;
pll extgcd(ll a, ll b) {
    if (b == 0) {
        GCD = a;
        return pll{1, 0};
    }
    pll ans = extgcd(b, a % b);
    return pll{ans.S, ans.F - a/b * ans.S};
}
pll bezout(ll a, ll b, ll c) {
    bool negx = (a < 0), negy = (b < 0);
    pll ans = extgcd(abs(a), abs(b));
    if (c % GCD != 0) return pll{-LLINF, -LLINF};
    return pll{ans.F * c/GCD * (negx ? -1 : 1),
               ans.S * c/GCD * (negy ? -1 : 1)};
}
ll inv(ll a, ll p) {
    if (p == 1) return -1;
    pll ans = bezout(a % p, -p, 1);
    if (ans == pll{-LLINF, -LLINF}) return -1;
    return (ans.F % p + p) % p;
}
```

10.10 Mu + Phi

```cpp
const int maxn = 1e6 + 5;
ll f[maxn];
vector<int> lpf, prime;
void build() {
lpf.clear(); lpf.resize(maxn, 1);
prime.clear();
f[1] = ...;  /* mu[1] = 1, phi[1] = 1 */
for (int i = 2; i < maxn; i++) {
    if (lpf[i] == 1) {
        lpf[i] = i; prime.emplace_back(i);
        f[i] = ...;  /* mu[i] = 1, phi[i] = i-1 */
    }
    for (auto& j : prime) {
        if (i*j >= maxn) break;
        lpf[i*j] = j;
        if (i % j == 0) f[i*j] = ...;  /* 0, phi[i]*j */
        else f[i*j] = ...;  /* -mu[i], phi[i]*phi[j] */
        if (j >= lpf[i]) break;
} } }
```

10.11 Other Formulas

- Inversion:
  $aa^{-1} \equiv 1 \pmod{m}$. a^{-1} exists iff $\gcd(a, m) = 1$.
- Linear inversion:
  $a^{-1} \equiv (m - \lfloor \frac{m}{a} \rfloor) \times (m \bmod a)^{-1} \pmod{m}$
- Fermat's little theorem:
  $a^p \equiv a \pmod{p}$ if p is prime.
- Euler function:
  $\phi(n) = n \prod_{p|n} \frac{p-1}{p}$
- Euler theorem:
  $a^{\phi(n)} \equiv 1 \pmod{n}$ if $\gcd(a, n) = 1$.
- Extended Euclidean algorithm:
  $ax + by = \gcd(a, b) = \gcd(b, a \bmod b) = \gcd(b, a - \lfloor \frac{a}{b} \rfloor b) = bx_1 + (a - \lfloor \frac{a}{b} \rfloor b)y_1 = ay_1 + b(x_1 - \lfloor \frac{a}{b} \rfloor y_1)$
- Divisor function:
  $\sigma_x(n) = \sum_{d|n} d^x$. $n = \prod_{i=1}^{r} p_i^{a_i}$.
  $\sigma_x(n) = \prod_{i=1}^{r} \frac{p_i^{(a_i+1)x} - 1}{p_i^x - 1}$ if $x \neq 0$. $\sigma_0(n) = \prod_{i=1}^{r}(a_i + 1)$.
- Chinese remainder theorem (Coprime Moduli):
  $x \equiv a_i \pmod{m_i}$.
  $M = \prod m_i$. $M_i = M/m_i$. $t_i = M_i^{-1}$.
  $x = kM + \sum a_i t_i M_i$, $k \in \mathbb{Z}$.
- Chinese remainder theorem:
  $x \equiv a_1 \pmod{m_1}, x \equiv a_2 \pmod{m_2} \Rightarrow x = m_1 p + a_1 = m_2 q + a_2 \Rightarrow m_1 p - m_2 q = a_2 - a_1$
  Solve for (p, q) using ExtGCD.
  $x \equiv m_1 p + a_1 \equiv m_2 q + a_2 \pmod{lcm(m_1, m_2)}$
- Avoiding Overflow: $ca \mod cb = c(a \mod b)$
- Dirichlet Convolution: $(f * g)(n) = \sum_{d|n} f(n) g(n/d)$
- Important Multiplicative Functions + Proterties:
  1. $\epsilon(n) = [n = 1]$
  2. $1(n) = 1$
  3. $id(n) = n$
  4. $\mu(n) = 0$ if n has squared prime factor
  5. $\mu(n) = (-1)^k$ if $n = p_1 p_2 \cdots p_k$
  6. $\epsilon = \mu * 1$
  7. $\phi = \mu * id$
  8. $[n = 1] = \sum_{d|n} \mu(d)$
  9. $[gcd = 1] = \sum_{d|gcd} \mu(d)$
- Möbius inversion: $f = g * 1 \Leftrightarrow g = f * \mu$

10.12 Polynomial

```
const int maxk = 20;
const int maxn = 1<<maxk;
const ll LINF = 1e18;

/* P = r*2^k + 1
P                   r   k   g
998244353           119 23  3
1004535809          479 21  3

P                   r   k   g
3                   1   1   2
5                   1   2   2
17                  1   4   3
97                  3   5   5
193                 3   6   5
257                 1   8   3
7681                15  9   17
12289               3   12  11
40961               5   13  3
65537               1   16  3
786433              3   18  10
5767169             11  19  3
7340033             7   20  3
23068673            11  21  3
104857601           25  22  3
167772161           5   25  3
469762049           7   26  3
1004535809          479 21  3
2013265921          15  27  31
2281701377          17  27  3
3221225473          3   30  5
75161927681         35  31  3
77309411329         9   33  7
206158430209        3   36  22
2061584302081       15  37  7
2748779069441       5   39  3
6597069766657       3   41  5
39582418599937      9   42  5
79164837199873      9   43  5
263882790666241     15  44  7
1231453023109121    35  45  3
1337006139375617    19  46  3
3799912185593857    27  47  5
4222124650659841    15  48  19
7881299347898369    7   50  6
31525197391593473   7   52  3
180143985094819841  5   55  6
1945555039024054273 27  56  5
4179340454199820289 29  57  3
9097271247288401921 505 54  6 */

const int g = 3;
const ll MOD = 998244353;

ll pw(ll a, ll n) { /* fast pow */ }

#define siz(x) (int)x.size()

template<typename T>
vector<T>& operator+=(vector<T>& a, const vector<T>& b) {
    if (siz(a) < siz(b)) a.resize(siz(b));
    for (int i = 0; i < min(siz(a), siz(b)); i++) {
        a[i] += b[i];
        a[i] -= a[i] >= MOD ? MOD : 0;
    }
    return a;
}

template<typename T>
vector<T>& operator-=(vector<T>& a, const vector<T>& b) {
    if (siz(a) < siz(b)) a.resize(siz(b));
    for (int i = 0; i < min(siz(a), siz(b)); i++) {
        a[i] -= b[i];
        a[i] += a[i] < 0 ? MOD : 0;
    }
    return a;
}

template<typename T>
vector<T> operator-(const vector<T>& a) {
    vector<T> ret(siz(a));
    for (int i = 0; i < siz(a); i++) {
        ret[i] = -a[i] < 0 ? -a[i] + MOD : -a[i];
    }
    return ret;
}

vector<ll> X, iX;
vector<int> rev;

void init_ntt() {
    X.clear(); X.resize(maxn, 1);  // x1 = g^((p-1)/n)
    iX.clear(); iX.resize(maxn, 1);
```

```cpp
    ll u = pw(g, (MOD-1)/maxn);
    ll iu = pw(u, MOD-2);

    for (int i = 1; i < maxn; i++) {
        X[i] = X[i-1] * u;
        iX[i] = iX[i-1] * iu;
        if (X[i] >= MOD) X[i] %= MOD;
        if (iX[i] >= MOD) iX[i] %= MOD;
    }

    rev.clear(); rev.resize(maxn, 0);
    for (int i = 1, hb = -1; i < maxn; i++) {
        if (!(i & (i-1))) hb++;
        rev[i] = rev[i ^ (1<<hb)] | (1<<(maxk-hb-1));
} }

template<typename T>
void NTT(vector<T>& a, bool inv=false) {

    int _n = (int)a.size();
    int k = __lg(_n) + ((1<<__lg(_n)) != _n);
    int n = 1<<k;
    a.resize(n, 0);

    short shift = maxk-k;
    for (int i = 0; i < n; i++)
        if (i > (rev[i]>>shift))
            swap(a[i], a[rev[i]>>shift]);

    for (int len = 2, half = 1, div = maxn>>1; len <= n; len<<=1, half<<=1, div>>=1) {
        for (int i = 0; i < n; i += len) {
            for (int j = 0; j < half; j++) {
                T u = a[i+j];
                T v = a[i+j+half] * (inv ? iX[j*div] : X[j*div]) % MOD;
                a[i+j] = (u+v >= MOD ? u+v-MOD : u+v);
                a[i+j+half] = (u-v < 0 ? u-v+MOD : u-v);
    } } }

    if (inv) {
        T dn = pw(n, MOD-2);
        for (auto& x : a) {
            x *= dn;
            if (x >= MOD) x %= MOD;
} } }

template<typename T>
inline void resize(vector<T>& a) {
    int cnt = (int)a.size();
    for (; cnt > 0; cnt--) if (a[cnt-1]) break;
    a.resize(max(cnt, 1));
}

template<typename T>
vector<T>& operator*=(vector<T>& a, vector<T> b) {
    int na = (int)a.size();
    int nb = (int)b.size();
    a.resize(na + nb - 1, 0);
    b.resize(na + nb - 1, 0);

    NTT(a); NTT(b);
    for (int i = 0; i < (int)a.size(); i++) {
        a[i] *= b[i];
        if (a[i] >= MOD) a[i] %= MOD;
    }
    NTT(a, true);

    resize(a);
    return a;
}

template<typename T>
void inv(vector<T>& ia, int N) {
    vector<T> _a(move(ia));
    ia.resize(1, pw(_a[0], MOD-2));
    vector<T> a(1, -_a[0] + (-_a[0] < 0 ? MOD : 0));

    for (int n = 1; n < N; n<<=1) {
        // n -> 2*n
        // ia' = ia(2-a*ia);

        for (int i = n; i < min(siz(_a), (n<<1)); i++)
            a.emplace_back(-_a[i] + (-_a[i] < 0 ? MOD : 0));

        vector<T> tmp = ia;
        ia *= a;
        ia.resize(n<<1);
        ia[0] = ia[0] + 2 >= MOD ? ia[0] + 2 - MOD : ia[0] + 2;
        ia *= tmp;
        ia.resize(n<<1);
    }
    ia.resize(N);
}

template<typename T>
void mod(vector<T>& a, vector<T>& b) {
    int n = (int)a.size()-1, m = (int)b.size()-1;
    if (n < m) return;

    vector<T> ra = a, rb = b;
    reverse(ra.begin(), ra.end()); ra.resize(min(n+1, n-m+1));
    reverse(rb.begin(), rb.end()); rb.resize(min(m+1, n-m+1));

    inv(rb, n-m+1);

    vector<T> q = move(ra);
    q *= rb;
    q.resize(n-m+1);
    reverse(q.begin(), q.end());

    q *= b;
    a -= q;
    resize(a);
}

/* Kitamasa Method (Fast Linear Recurrence):
Find a[K] (Given a[j] = c[0]a[j-N] + ... + c[N-1]a[j-1])
Let B(x) = x^N - c[N-1]x^(N-1) - ... - c[1]x^1 - c[0]
Let R(x) = x^K mod B(x)    (get x^K using fast pow and use poly mod to get R(x))
Let r[i] = the coefficient of x^i in R(x)
=> a[K] = a[0]r[0] + a[1]r[1] + ... + a[N-1]r[N-1] */
```

11 Linear Algebra

11.1 Gaussian-Jordan Elimination

```cpp
int n; vector<vector<ll> > v;
void gauss(vector<vector<ll>>& v) {
int r = 0;
for (int i = 0; i < n; i++) {
    bool ok = false;
    for (int j = r; j < n; j++) {
        if (v[j][i] == 0) continue;
        swap(v[j], v[r]);
        ok = true; break;
    }
    if (!ok) continue;
    ll div = inv(v[r][i]);
    for (int j = 0; j < n+1; j++) {
        v[r][j] *= div;
        if (v[r][j] >= MOD) v[r][j] %= MOD;
    }
    for (int j = 0; j < n; j++) {
        if (j == r) continue;
        ll t = v[j][i];
        for (int k = 0; k < n+1; k++) {
            v[j][k] -= v[r][k] * t % MOD;
            if (v[j][k] < 0) v[j][k] += MOD;
    } }
    r++;
} }
```

11.2 Determinant

1. Use GJ Elimination, if there's any row consists of only 0, then det = 0, otherwise det = product of diagonal elements.
2. Properties of det:
   - Transpose: Unchanged
   - Row Operation 1 - Swap 2 rows: $-det$
   - Row Operation 2 - $k\vec{r_i}$: $k \times det$
   - Row Operation 3 - $k\vec{r_i}$ add to $\vec{r_j}$: Unchaged

12 Flow / Matching

12.1 Dinic

```cpp
struct Dinic {
    struct Edge {
        int t, c, r;
        Edge() {}
        Edge(int _t, int _c, int _r):
            t(_t), c(_c), r(_r) {}
    };
    vector<vector<Edge>> G;
    vector<int> dis, iter;
    int s, t;
    void init(int n) {
        G.resize(n), dis.resize(n), iter.resize(n);
        for(int i = 0; i < n; ++i)
            G[i].clear();
    }
    void add(int a, int b, int c) {
        G[a].eb(b, c, G[b].size());
        G[b].eb(a, 0, G[a].size() - 1);
    }
    bool bfs() {
        fill(ALL(dis), -1);
        dis[s] = 0;
        queue<int> que;
        que.push(s);
        while(!que.empty()) {
            int u = que.front(); que.pop();
            for(auto& e : G[u]) {
                if(e.c > 0 && dis[e.t] == -1) {
                    dis[e.t] = dis[u] + 1;
                    que.push(e.t);
                }
            }
        }
        return dis[t] != -1;
    }
    int dfs(int u, int cur) {
        if(u == t) return cur;
        for(int &i = iter[u]; i < (int)G[u].size(); ++i ) {
            auto& e = G[u][i];
            if(e.c > 0 && dis[u] + 1 == dis[e.t]) {
                int ans = dfs(e.t, min(cur, e.c));
                if(ans > 0) {
                    G[e.t][e.r].c += ans;
                    e.c -= ans;
                    return ans;
                }
            }
        }
        return 0;
    }

    int flow(int a, int b) {
        s = a, t = b;
        int ans = 0;
        while(bfs()) {
            fill(ALL(iter), 0);
            int tmp;
            while((tmp = dfs(s, INF)) > 0)
                ans += tmp;
        }
        return ans;
    }
};
```

12.2 ISAP

```cpp
#define SZ(c) ((int)(c).size())
struct Maxflow{
  static const int MAXV=50010;
  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];
  int iter[MAXV],d[MAXV],gap[MAXV],tot;
  void init(int n,int _s,int _t){
    tot=n,s=_s,t=_t;
    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 flow(){
    int res=0;
    for(res=0,gap[0]=tot;d[s]<tot;res+=DFS(s,INF));
    return res;
  } // reset: set iter,d,gap to 0
} flow;
```

12.3 MCMF

```cpp
struct MCMF {
    struct Edge {
        int to, cap, rev;
        ll cost;
        Edge() {}
        Edge(int _to, int _cap, int _rev, ll _cost) :
            to(_to), cap(_cap), rev(_rev), cost(_cost) {}
    };
    static const int N = 2000;
    vector<Edge> G[N];
    int n, s, t;
    void init(int _n, int _s, int _t) {
        n = _n, s = _s, t = _t;
        for(int i = 0; i <= n; ++i)
            G[i].clear();
    }
    void add_edge(int from, int to, int cap, ll cost) {
        G[from].eb(to, cap, (int)G[to].size(), cost);
        G[to].eb(from, 0, (int)G[from].size() - 1, -cost);
    }

    bool vis[N];
    int iter[N];
    ll dis[N];
    bool SPFA() {
        for(int i = 0; i <= n; ++i)
            vis[i] = 0, dis[i] = LINF;

        dis[s] = 0; vis[s] = 1;
        queue<int> que; que.push(s);
        while(!que.empty()) {
            int u = que.front(); que.pop();
            vis[u] = 0;
            for(auto& e : G[u]) if(e.cap > 0 && dis[e.to] > dis[u] + e.cost) {
                dis[e.to] = dis[u] + e.cost;
                if(!vis[e.to]) {
                    que.push(e.to);
                    vis[e.to] = 1;
                }
            }
        }
        return dis[t] != LINF;
    }

    int dfs(int u, int cur) {
        if(u == t) return cur;
        int ret = 0; vis[u] = 1;
        for(int &i = iter[u]; i < (int)G[u].size(); ++i) {
            auto &e = G[u][i];
            if(e.cap > 0 && dis[e.to] == dis[u] + e.cost && !vis[e.to]) {
                int tmp = dfs(e.to, min(cur, e.cap));
                e.cap -= tmp;
                G[e.to][e.rev].cap += tmp;
                cur -= tmp;
                ret += tmp;
                if(cur == 0) {
                    vis[u] = 0;
                    return ret;
                }
            }
        }
        vis[u] = 0;
        return ret;
    }
    pair<int, ll> flow() {
        int flow = 0; ll cost = 0;
        while(SPFA()) {
            memset(iter, 0, sizeof(iter));
            int tmp = dfs(s, INF);
            flow += tmp, cost += tmp * dis[t];
        }
        return {flow, cost};
    }
};
```

12.4 Hopcroft-Karp

```
struct HopcroftKarp {
    // id: X = [1, nx], Y = [nx+1, nx+ny]
    int n, nx, ny, m, MXCNT;
    vector<vector<int> > g;
    vector<int> mx, my, dis, vis;
    void init(int nnx, int nny, int mm) {
        nx = nnx, ny = nny, m = mm;
        n = nx + ny + 1;
        g.clear(); g.resize(n);
    }
    void add(int x, int y) {
        g[x].emplace_back(y);
        g[y].emplace_back(x);
    }
    bool dfs(int x) {
        vis[x] = true;
        Each(y, g[x]) {
            int px = my[y];
            if (px == -1 ||
                (dis[px] == dis[x]+1 &&
                !vis[px] && dfs(px))) {
                mx[x] = y;
                my[y] = x;
                return true;
            }
        }
        return false;
    }
    void get() {
        mx.clear(); mx.resize(n, -1);
        my.clear(); my.resize(n, -1);

        while (true) {
            queue<int> q;
            dis.clear(); dis.resize(n, -1);
            for (int x = 1; x <= nx; x++){
                if (mx[x] == -1) {
                    dis[x] = 0;
                    q.push(x);
                }
            }
            while (!q.empty()) {
                int x = q.front(); q.pop();
                Each(y, g[x]) {
                    if (my[y] != -1 && dis[my[y]] == -1) {
                        dis[my[y]] = dis[x] + 1;
                        q.push(my[y]);
                    }
                }
            }

            bool brk = true;
            vis.clear(); vis.resize(n, 0);
            for (int x = 1; x <= nx; x++)
                if (mx[x] == -1 && dfs(x))
                    brk = false;

            if (brk) break;
        }
        MXCNT = 0;
        for (int x = 1; x <= nx; x++) if (mx[x] != -1) MXCNT++;
    }
} hk;
```

12.5 Cover / Independent Set

```
V(E) Cover: choose some V(E) to cover all E(V)
V(E) Independ: set of V(E) not adj to each other

M = Max Matching
Cv = Min V Cover
Ce = Min E Cover
Iv = Max V Ind
Ie = Max E Ind (equiv to M)

M = Cv (Konig Theorem)
Iv = V \ Cv
Ce = V - M

Construct Cv:
1. Run Dinic
2. Find s-t min cut
3. Cv = {X in T} + {Y in S}
```

12.6 KM

```
#include <bits/stdc++.h>

using namespace std;
const int inf = 1e9;

struct KuhnMunkres {
    int n;
    vector<vector<int>> g;
    vector<int> lx, ly, slack;
    vector<int> match, visx, visy;
    KuhnMunkres(int n) : n(n), g(n, vector<int>(n)),
        lx(n), ly(n), slack(n), match(n), visx(n), visy(n) {}
    vector<int> & operator[](int i) { return g[i]; }
    bool dfs(int i, bool aug) { // aug = true 表示要更新 match
        if(visx[i]) return false;
        visx[i] = true;
        for(int j = 0; j < n; j++) {
            if(visy[j]) continue;
            // 一邊擴增交錯樹、尋找增廣路徑
            // 一邊更新slack：樹上的點跟樹外的點所造成的最小權重
            int d = lx[i] + ly[j] - g[i][j];
            if(d == 0) {
                visy[j] = true;
                if(match[j] == -1 || dfs(match[j], aug)
                    ) {
                    if(aug)
                        match[j] = i;
                    return true;
                }
            } else {
                slack[j] = min(slack[j], d);
            }
        }
        return false;
    }
    bool augment() { // 回傳是否有增廣路
        for(int j = 0; j < n; j++) if(!visy[j] && slack
            [j] == 0) {
            visy[j] = true;
            if(match[j] == -1 || dfs(match[j], false))
                {
                return true;
            }
        }
        return false;
    }
    void relabel() {
        int delta = inf;
        for(int j = 0; j < n; j++) if(!visy[j]) delta =
             min(delta, slack[j]);
        for(int i = 0; i < n; i++) if(visx[i]) lx[i] -=
             delta;
        for(int j = 0; j < n; j++) {
            if(visy[j]) ly[j] += delta;
            else slack[j] -= delta;
        }
    }
    int solve() {
        for(int i = 0; i < n; i++) {
            lx[i] = 0;
            for(int j = 0; j < n; j++) lx[i] = max(lx[i
                ], g[i][j]);
        }
        fill(ly.begin(), ly.end(), 0);
        fill(match.begin(), match.end(), -1);
        for(int i = 0; i < n; i++) {
            // slack 在每一輪都要初始化
            fill(slack.begin(), slack.end(), inf);
            fill(visx.begin(), visx.end(), false);
            fill(visy.begin(), visy.end(), false);
            if(dfs(i, true)) continue;
            // 重複調整頂標直到找到增廣路徑
            while(!augment()) relabel();
            fill(visx.begin(), visx.end(), false);
            fill(visy.begin(), visy.end(), false);
            dfs(i, true);
        }
        int ans = 0;
        for(int j = 0; j < n; j++) if(match[j] != -1)
            ans += g[match[j]][j];
        return ans;
    }
};
signed main() {
    ios_base::sync_with_stdio(0), cin.tie(0);
    int n;
    while(cin >> n && n) {
        KuhnMunkres KM(n);
        for(int i = 0; i < n; i++) {
            for(int j = 0; j < n; j++) {
                int c;
                cin >> c;
                if(c > 0)
                    KM[i][j] = c;
            }
        }
        cout << KM.solve() << '\n';
    }
}
```

13 Combinatorics

13.1 Catalan Number

$$C_0 = 1, C_n = \sum_{i=0}^{n-1} C_i C_{n-1-i}, C_n = C_n^{2n} - C_{n-1}^{2n}$$

0	1	1	2	5
4	14	42	132	429
8	1430	4862	16796	58786
12	208012	742900	2674440	9694845

13.2 Burnside's Lemma

Let X be the original set.
Let G be the group of operations acting on X.
Let X^g be the set of x not affected by g.
Let X/G be the set of orbits.
Then the following equation holds:

$$|X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g|$$

14 Special Numbers

14.1 Fibonacci Series

1	1	1	2	3
5	5	8	13	21
9	34	55	89	144
13	233	377	610	987
17	1597	2584	4181	6765
21	10946	17711	28657	46368
25	75025	121393	196418	317811
29	514229	832040	1346269	2178309
33	3524578	5702887	9227465	14930352

$f(45) \approx 10^9, f(88) \approx 10^{18}$

14.2 Prime Numbers

- First 50 prime numbers:

1	2	3	5	7	11
6	13	17	19	23	29
11	31	37	41	43	47
16	53	59	61	67	71
21	73	79	83	89	97
26	101	103	107	109	113
31	127	131	137	139	149
36	151	157	163	167	173
41	179	181	191	193	197
46	199	211	223	227	229

- Very large prime numbers:
  1000001333 1000500889 2500001909
  2000000659 900004151 850001359

- $\pi(n) \equiv$ Number of primes $\leq n \approx n/((\ln n) - 1)$
  $\pi(100) = 25, \pi(200) = 46$
  $\pi(500) = 95, \pi(1000) = 168$
  $\pi(2000) = 303, \pi(4000) = 550$
  $\pi(10^4) = 1229, \pi(10^5) = 9592$
  $\pi(10^6) = 78498, \pi(10^7) = 664579$