

Contents

| | |
|---------------------------------------|-----------|
| 1 Init (Linux) | 1 |
| 1.1 vimrc | 1 |
| 1.2 template.cpp | 1 |
| 1.3 run.sh | 1 |
| 2 Reminder | 1 |
| 2.1 Observations and Tricks | 1 |
| 2.2 Bug List | 1 |
| 3 Basic | 1 |
| 3.1 template (optional) | 1 |
| 3.2 Stress | 2 |
| 3.3 PBDs | 2 |
| 3.4 Random | 2 |
| 4 Python | 2 |
| 4.1 I/O | 2 |
| 4.2 Decimal | 2 |
| 5 Data Structure | 2 |
| 5.1 Segment Tree | 2 |
| 5.2 Heavy Light Decomposition | 2 |
| 5.3 Skew Heap | 3 |
| 5.4 Leftist Heap | 3 |
| 5.5 Persistent Treap | 3 |
| 5.6 Li Chao Tree | 4 |
| 5.7 Time Segment Tree | 4 |
| 6 DP | 5 |
| 6.1 Aliens | 5 |
| 6.2 SOS DP | 5 |
| 7 Graph | 5 |
| 7.1 Tree Centroid | 5 |
| 7.2 Bellman-Ford + SPFA | 5 |
| 7.3 BCC - AP | 5 |
| 7.4 BCC - Bridge | 5 |
| 7.5 SCC - Tarjan | 5 |
| 7.6 Eulerian Path - Undir | 6 |
| 7.7 Eulerian Path - Dir | 6 |
| 7.8 Hamilton Path | 6 |
| 7.9 Kth Shortest Path | 6 |
| 7.10 System of Difference Constraints | 6 |
| 8 String | 8 |
| 8.1 Rolling Hash | 8 |
| 8.2 Trie | 8 |
| 8.3 KMP | 8 |
| 8.4 Z Value | 9 |
| 8.5 Manacher | 9 |
| 8.6 Suffix Array | 9 |
| 8.7 SA-IS | 9 |
| 8.8 Minimum Rotation | 9 |
| 8.9 Aho Corasick | 9 |
| 9 Geometry | 10 |
| 9.1 Basic Operations | 10 |
| 9.2 InPoly | 10 |
| 9.3 Sort by Angle | 10 |
| 9.4 Line Intersect Check | 10 |
| 9.5 Line Intersection | 10 |
| 9.6 Convex Hull | 10 |
| 9.7 Lower Concave Hull | 10 |
| 9.8 Polygon Area | 10 |
| 9.9 Pick's Theorem | 10 |
| 9.10 Minimum Enclosing Circle | 10 |
| 9.11 PolyUnion | 10 |
| 9.12 Minkowski Sum | 10 |
| 10 Number Theory | 11 |
| 10.1 Basic | 11 |
| 10.2 Prime Sieve and Defactor | 11 |
| 10.3 Harmonic Series | 11 |
| 10.4 Count Number of Divisors | 11 |
| 10.5 數論分塊 | 11 |
| 10.6 Pollard's rho | 11 |
| 10.7 Miller Rabin | 11 |
| 10.8 Fast Power | 11 |
| 10.9 Extend GCD | 11 |
| 10.10 Mu + Phi | 11 |
| 10.11 Other Formulas | 11 |
| 10.12 Polynomial | 11 |
| 11 Linear Algebra | 17 |
| 11.1 Gaussian-Jordan Elimination | 17 |
| 11.2 Determinant | 17 |
| 12 Flow / Matching | 17 |
| 12.1 Dinic | 17 |
| 12.2 ISAP | 17 |
| 12.3 MCMF | 17 |
| 12.4 Hopcroft-Karp | 17 |
| 12.5 Cover / Independent Set | 17 |
| 12.6 KM | 17 |
| 13 Combinatorics | 20 |
| 13.1 Catalan Number | 20 |
| 13.2 Burnside's Lemma | 20 |
| 14 Special Numbers | 20 |
| 14.1 Fibonacci Series | 20 |
| 14.2 Prime Numbers | 20 |

| | |
|---------------------------|-----------|
| 13 Combinatorics | 20 |
| 13.1 Catalan Number | 20 |
| 13.2 Burnside's Lemma | 20 |
| 14 Special Numbers | 20 |
| 14.1 Fibonacci Series | 20 |
| 14.2 Prime Numbers | 20 |

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

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

1.2 template.cpp

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

#define endl '\n'

typedef long long ll;
typedef pair<int, int> pii;

void input() {}
void solve() {}

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

1.3 run.sh

```
#!/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，例如矩陣快速幂

```

33
34 void _debug() {}
35 template<typename A, typename... B> void _debug(A a, B...
    b) { cerr << a << ' ', _debug(b...); }
36 #define debug(...) cerr<<GRAY<<#__VA_ARGS__<<" ": "<<
    COLOREND, _debug(__VA_ARGS__), cerr<<endl
37
38 void _output() {}
39 template<typename A, typename... B> void _output(A a, B
    ... b) { cout << a << ' ', _output(b...); }
40 #define output(...) _output(__VA_ARGS__), cout<<endl
41 /* ===== */
42 // BASIC ALGORITHM
43 string binary(ll x, int b = -1) {
44     if (b == -1) b = __lg(x) + 1;
45     string s = "";
46     for (int k = b - 1; k >= 0; k--) {
47         s.push_back((x & (1LL<<k)) ? '1' : '0');
48     }
49     return s;
50 }
51 /* ===== */
52 // CONSTANT
53 const int INF = 1.05e9;
54 const ll LINF = 4e18;
55 const int MOD = 1e9 + 7;
56 //const int MOD = 998244353;
57 const int maxn = 2e5 + 3;

```

3.2 Stress

```

1 g++ gen.cpp -o gen.out
2 g++ ac.cpp -o ac.out
3 g++ wa.cpp -o wa.out
4 for ((i=0;;i++))
5 do
6     echo "$i"
7     ./gen.out > in.txt
8     ./ac.out < in.txt > ac.txt
9     ./wa.out < in.txt > wa.txt
10    diff ac.txt wa.txt || break
11 done

```

3 Basic

3.1 template (optional)

```

1 typedef pair<ll, ll> pll;
2 typedef pair<int, ll> pil;
3 typedef pair<ll, int> pli;
4
5 /* ===== */
6 // STL and I/O
7 // pair
8 template<typename T1, typename T2>
9 ostream& operator<<(ostream& os, pair<T1, T2> p) {
10     return os << "(" << p.first << ", " << p.second <<
        " )";
11 }
12 template<typename T1, typename T2>
13 istream& operator>>(istream& is, pair<T1, T2>& p) {
14     return is >> p.first >> p.second; }
15
16 // vector
17 template<typename T>
18 istream& operator>>(istream& is, vector<T>& v) {
19     for (auto& x : v) is >> x;
20     return is;
21 }
22 template<typename T>
23 ostream& operator<<(ostream& os, const vector<T>& v) {
24     for (const auto& x : v) os << x << ' ';
25     return os;
26 }
27 /* ===== */
28 // debug(), output()
29 #define RED "\x1b[31m"
30 #define GREEN "\x1b[32m"
31 #define YELLOW "\x1b[33m"
32 #define GRAY "\x1b[90m"
33 #define COLOREND "\x1b[0m"

```

3.3 PBDS

```

1 #include <bits/extc++.h>
2 using namespace __gnu_pbds;
3
4 // map
5 tree<int, int, less<>, rb_tree_tag,
    tree_order_statistics_node_update> tr;
6 tr.order_of_key(element);
7 tr.find_by_order(rank);
8
9 // set
10 tree<int, null_type, less<>, rb_tree_tag,
    tree_order_statistics_node_update> tr;
11 tr.order_of_key(element);
12 tr.find_by_order(rank);
13
14 // priority queue
15 __gnu_pbds::priority_queue<int, less<int> > big_q; //
    Big First
16 __gnu_pbds::priority_queue<int, greater<int> > small_q;
    // Small First
17 q1.join(q2); // join

```

3.4 Random

```

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

```

4 Python

4.1 I/O

```

1 import sys
2 input = sys.stdin.readline
3
4 # Input
5 def readInt():
6     return int(input())
7 def readList():
8     return list(map(int, input().split()))
9 def readStr():
10    s = input()
11    return list(s[:len(s) - 1])
12 def readVars():
13    return map(int, input().split())
14
15 # Output
16 sys.stdout.write(string)
17
18 # faster
19 def main():
20     pass
21 main()

```

4.2 Decimal

```

1 from decimal import *
2 getcontext().prec = 2500000
3 getcontext().Emax = 2500000
4 a,b = Decimal(input()),Decimal(input())
5 a*=b
6 print(a)

```

5 Data Structure

5.1 Segment Tree

```

1 struct node {
2     ll sum, add, mod; int ln;
3     node(): sum(0), add(0), mod(0), ln(0) {}
4 };
5
6 struct segT {
7     int n;
8     vector<ll> ar;
9     vector<node> st;
10
11     void init(int _n) {
12         n = _n;
13         reset(ar, n, 0LL);
14         reset(st, n*4);
15     }
16     void pull(int cl, int cr, int i) {
17         st[i].sum = st[cl].sum + st[cr].sum;
18     }
19     void push(int cl, int cr, int i) {
20         ll md = st[i].mod, ad = st[i].add;
21         if (md) {
22             st[cl].sum = md * st[cl].ln, st[cr].sum =
23                 md * st[cr].ln;
24             st[cl].mod = md, st[cr].mod = md;
25             st[i].mod = 0;
26         }
27         if (ad) {
28             st[cl].sum += ad * st[cl].ln, st[cr].sum +=
29                 ad * st[cr].ln;
30             st[cl].add += ad, st[cr].add += ad;
31             st[i].add = 0;
32         }
33     }
34     void build(int l, int r, int i) {
35         if (l == r) {
36             st[i].sum = ar[l];
37             st[i].ln = 1;
38             return;
39         }
40         int mid = (l+r)>>1, cl = i<<1, cr = i<<1|1;
41         build(l, mid, cl);
42         build(mid + 1, r, cr);
43         pull(cl, cr, i);
44     }
45     void addval(int ql, int qr, ll val, int l, int r,
46         int i) {
47         if (qr < l || r < ql) return;
48         if (ql <= l && r <= qr) {
49             st[i].sum += val * st[i].ln;
50             st[i].add += val;
51             return;
52         }
53         int mid = (l+r)>>1, cl = i<<1, cr = i<<1|1;
54         push(cl, cr, i);
55         addval(ql, qr, val, l, mid, cl);
56         addval(ql, qr, val, mid + 1, r, cr);
57         pull(cl, cr, i);
58     }
59     void modify(int ql, int qr, ll val, int l, int r,
60         int i) {
61         if (qr < l || r < ql) return;
62         if (ql <= l && r <= qr) {
63             st[i].sum = val * st[i].ln;
64             st[i].add = 0;
65             st[i].mod = val;
66             return;
67         }
68         int mid = (l+r)>>1, cl = i<<1, cr = i<<1|1;
69         push(cl, cr, i);
70         modify(ql, qr, val, l, mid, cl);
71         modify(ql, qr, val, mid+1, r, cr);
72         pull(cl, cr, i);
73     }
74     ll query(int ql, int qr, int l, int r, int i) {
75         if (qr < l || r < ql) return 0;
76         if (ql <= l && r <= qr) return st[i].sum;
77         int mid = (l+r)>>1, cl = i<<1, cr = i<<1|1;
78         push(cl, cr, i);
79         return (query(ql, qr, l, mid, cl) +
80             query(ql, qr, mid+1, r, cr));
81     }
82 };

```

```

38     int mid = (l+r)>>1, cl = i<<1, cr = i<<1|1;
39     build(l, mid, cl);
40     build(mid + 1, r, cr);
41     pull(cl, cr, i);
42 }
43 void addval(int ql, int qr, ll val, int l, int r,
44     int i) {
45     if (qr < l || r < ql) return;
46     if (ql <= l && r <= qr) {
47         st[i].sum += val * st[i].ln;
48         st[i].add += val;
49         return;
50     }
51     int mid = (l+r)>>1, cl = i<<1, cr = i<<1|1;
52     push(cl, cr, i);
53     addval(ql, qr, val, l, mid, cl);
54     addval(ql, qr, val, mid + 1, r, cr);
55     pull(cl, cr, i);
56 }
57 void modify(int ql, int qr, ll val, int l, int r,
58     int i) {
59     if (qr < l || r < ql) return;
60     if (ql <= l && r <= qr) {
61         st[i].sum = val * st[i].ln;
62         st[i].add = 0;
63         st[i].mod = val;
64         return;
65     }
66     int mid = (l+r)>>1, cl = i<<1, cr = i<<1|1;
67     push(cl, cr, i);
68     modify(ql, qr, val, l, mid, cl);
69     modify(ql, qr, val, mid+1, r, cr);
70     pull(cl, cr, i);
71 }
72 ll query(int ql, int qr, int l, int r, int i) {
73     if (qr < l || r < ql) return 0;
74     if (ql <= l && r <= qr) return st[i].sum;
75     int mid = (l+r)>>1, cl = i<<1, cr = i<<1|1;
76     push(cl, cr, i);
77     return (query(ql, qr, l, mid, cl) +
78         query(ql, qr, mid+1, r, cr));
79 }
80 };

```

5.2 Heavy Light Decomposition

```

1 constexpr int maxn=2e5+5;
2 int arr[(maxn+1)<<2];
3 #define m ((l+r)>>1)
4 void build(V<int>& v, int i=1, int l=0, int r=maxn){
5     if((int)v.size()<=l) return;
6     if(r-l==1){arr[i]=v[l];return;}
7     build(v, i<<1, l, m), build(v, i<<1|1, m, r);
8     arr[i]=max(arr[i<<1], arr[i<<1|1]);
9 }
10 void modify(int p, int k, int i=1, int l=0, int r=maxn){
11     if(p<l||r<=p) return;
12     if(r-l==1){arr[i]=k;return;}
13     if(p<m) modify(p, k, i<<1, l, m);
14     else modify(p, k, i<<1|1, m, r);
15     arr[i]=max(arr[i<<1], arr[i<<1|1]);
16 }
17 int query(int ql, int qr, int i=1, int l=0, int r=maxn){
18     if(ql<=l||r<=qr) return 0;
19     if(ql<=l&&r<=qr) return arr[i];
20     if(ql<=m) return query(ql, qr, i<<1, l, m);
21     if(m<=qr) return query(ql, qr, i<<1|1, m, r);
22     return max(query(ql, qr, i<<1, l, m), query(ql, qr, i
23         <<1|1, m, r));
24 }
25 #undef m
26 inline void solve(){
27     int n,q;cin>>n>>q;
28     V<int> v(n);
29     for(auto& i:v)
30         cin>>i;
31     V<V<int>> e(n);
32     for(int i=1;i<n;i++){
33         int a,b;cin>>a>>b,a--,b--;
34         e[a].emplace_back(b);
35         e[b].emplace_back(a);
36     }

```

```

35     }
36     V<int> d(n,0), f(n,0), sz(n,1), son(n,-1);
37     F<void(int,int)> dfs1=
38     [&](int x, int pre){
39         for(auto i:e[x]) if(i!=pre){
40             d[i]=d[x]+1, f[i]=x;
41             dfs1(i,x), sz[x]+=sz[i];
42             if(!~son[x] || sz[son[x]]<sz[i])
43                 son[x]=i;
44         }
45     };dfs1(0,0);
46     V<int> top(n,0), dfn(n,-1), rnk(n,0);
47     F<void(int,int)> dfs2=
48     [&](int x, int t){
49         static int cnt=0;
50         dfn[x]=cnt++, rnk[dfn[x]]=x, top[x]=t;
51         if(!~son[x]) return;
52         dfs2(son[x],t);
53         for(auto i:e[x])
54             if(!~dfn[i]) dfs2(i,i);
55     };dfs2(0,0);
56     V<int> dfnv(n);
57     for(int i=0;i<n;i++)
58         dfnv[dfn[i]]=v[i];
59     build(dfnv);
60     while(q--){
61         int op,a,b;cin>>op>>a>>b;
62         switch(op){
63             case 1:{
64                 modify(dfn[a-1],b);
65             }break;
66             case 2:{
67                 a--,b--;
68                 int ans=0;
69                 while(top[a]!=top[b]){
70                     if(d[top[a]]>d[top[b]]) swap(a,b);
71                     ans=max(ans,query(dfn[top[b]],dfn[b]+1));
72                     b=f[top[b]];
73                 }
74                 if(dfn[a]>dfn[b]) swap(a,b);
75                 ans=max(ans,query(dfn[a],dfn[b]+1));
76                 cout<<ans<<endl;
77             }break;
78         }
79     }
80 }

```

5.3 Skew Heap

```

1 struct node{
2     node *l,*r;
3     int v;
4     node(int x):v(x){
5         l=r=nullptr;
6     }
7 };
8 node* merge(node* a,node* b){
9     if(!a||!b) return a?:b;
10    // min heap
11    if(a->v>b->v) swap(a,b);
12    a->r=merge(a->r,b);
13    swap(a->l,a->r);
14    return a;
15 }

```

5.4 Leftist Heap

```

1 struct node{
2     node *l,*r;
3     int d, v;
4     node(int x):d(1),v(x){
5         l=r=nullptr;
6     }
7 };
8 static inline int d(node* x){return x?x->d:0;}
9 node* merge(node* a,node* b){
10    if(!a||!b) return a?:b;
11    // min heap
12    if(a->v>b->v) swap(a,b);

```

```

13    a->r=merge(a->r,b);
14    if(d(a->l)<d(a->r))
15        swap(a->l,a->r);
16    a->d=d(a->r)+1;
17    return a;
18 }

```

5.5 Persistent Treap

```

1 struct node {
2     node *l, *r;
3     char c; int v, sz;
4     node(char x = '$'): c(x), v(mt()), sz(1) {
5         l = r = nullptr;
6     }
7     node(node* p) { *this = *p; }
8     void pull() {
9         sz = 1;
10        for (auto i : {l, r})
11            if (i) sz += i->sz;
12    }
13 } arr[maxn], *ptr = arr;
14 inline int size(node* p) {return p ? p->sz : 0;}
15 node* merge(node* a, node* b) {
16     if (!a || !b) return a ? : b;
17     if (a->v < b->v) {
18         node* ret = new(ptr++) node(a);
19         ret->r = merge(ret->r, b), ret->pull();
20         return ret;
21     }
22     else {
23         node* ret = new(ptr++) node(b);
24         ret->l = merge(a, ret->l), ret->pull();
25         return ret;
26     }
27 }
28 P<node*> split(node* p, int k) {
29     if (!p) return {nullptr, nullptr};
30     if (k >= size(p->l) + 1) {
31         auto [a, b] = split(p->r, k - size(p->l) - 1);
32         node* ret = new(ptr++) node(p);
33         ret->r = a, ret->pull();
34         return {ret, b};
35     }
36     else {
37         auto [a, b] = split(p->l, k);
38         node* ret = new(ptr++) node(p);
39         ret->l = b, ret->pull();
40         return {a, ret};
41     }
42 }

```

5.6 Li Chao Tree

```

1 constexpr int maxn = 5e4 + 5;
2 struct line {
3     ld a, b;
4     ld operator()(ld x) {return a * x + b;}
5 } arr[(maxn + 1) << 2];
6 bool operator<(line a, line b) {return a.a < b.a;}
7 #define m ((l+r)>>1)
8 void insert(line x, int i = 1, int l = 0, int r = maxn)
9 {
10    if (r - l == 1) {
11        if (x(l) > arr[i](l))
12            arr[i] = x;
13        return;
14    }
15    line a = max(arr[i], x), b = min(arr[i], x);
16    if (a(m) > b(m))
17        arr[i] = a, insert(b, i << 1, l, m);
18    else
19        arr[i] = b, insert(a, i << 1 | 1, m, r);
20 }
21 ld query(int x, int i = 1, int l = 0, int r = maxn) {
22     if (x < l || r <= x) return -numeric_limits<ld>::max();
23     if (r - l == 1) return arr[i](x);
24     return max({arr[i](x), query(x, i << 1, l, m), query(
25         x, i << 1 | 1, m, r)});

```

```
24 }
25 #undef m
```

5.7 Time Segment Tree

```
1 constexpr int maxn = 1e5 + 5;
2 V<P<int>> arr[(maxn + 1) << 2];
3 V<int> dsu, sz;
4 V<tuple<int, int, int>> his;
5 int cnt, q;
6 int find(int x) {
7     return x == dsu[x] ? x : find(dsu[x]);
8 };
9 inline bool merge(int x, int y) {
10     int a = find(x), b = find(y);
11     if (a == b) return false;
12     if (sz[a] > sz[b]) swap(a, b);
13     his.emplace_back(a, b, sz[b]), dsu[a] = b, sz[b] +=
        sz[a];
14     return true;
15 };
16 inline void undo() {
17     auto [a, b, s] = his.back(); his.pop_back();
18     dsu[a] = a, sz[b] = s;
19 }
20 #define m ((l + r) >> 1)
21 void insert(int ql, int qr, P<int> x, int i = 1, int l
    = 0, int r = q) {
22     // debug(ql, qr, x); return;
23     if (qr <= l || r <= ql) return;
24     if (ql <= l && r <= qr) {arr[i].push_back(x);
        return;}
25     if (qr <= m)
26         insert(ql, qr, x, i << 1, l, m);
27     else if (m <= ql)
28         insert(ql, qr, x, i << 1 | 1, m, r);
29     else {
30         insert(ql, qr, x, i << 1, l, m);
31         insert(ql, qr, x, i << 1 | 1, m, r);
32     }
33 }
34 void traversal(V<int>& ans, int i = 1, int l = 0, int r
    = q) {
35     int opcnt = 0;
36     // debug(i, l, r);
37     for (auto [a, b] : arr[i])
38         if (merge(a, b))
39             opcnt++, cnt--;
40     if (r - l == 1) ans[l] = cnt;
41     else {
42         traversal(ans, i << 1, l, m);
43         traversal(ans, i << 1 | 1, m, r);
44     }
45     while (opcnt--)
46         undo(), cnt++;
47     arr[i].clear();
48 }
49 #undef m
50 inline void solve() {
51     int n, m; cin >> n >> m >> q, q++;
52     dsu.resize(cnt = n), sz.assign(n, 1);
53     iota(dsu.begin(), dsu.end(), 0);
54     // a, b, time, operation
55     unordered_map<ll, V<int>> s;
56     for (int i = 0; i < m; i++) {
57         int a, b; cin >> a >> b;
58         if (a > b) swap(a, b);
59         s[(ll)a << 32 | b].emplace_back(0);
60     }
61     for (int i = 1; i < q; i++) {
62         int op, a, b;
63         cin >> op >> a >> b;
64         if (a > b) swap(a, b);
65         switch (op) {
66             case 1:
67                 s[(ll)a << 32 | b].push_back(i);
68                 break;
69             case 2:
70                 auto tmp = s[(ll)a << 32 | b].back();
71                 s[(ll)a << 32 | b].pop_back();
72                 insert(tmp, i, P<int> {a, b});
```

```
73     }
74 }
75 for (auto [p, v] : s) {
76     int a = p >> 32, b = p & -1;
77     while (v.size()) {
78         insert(v.back(), q, P<int> {a, b});
79         v.pop_back();
80     }
81 }
82 V<int> ans(q);
83 traversal(ans);
84 for (auto i : ans)
85     cout << i << ' ';
86 cout << endl;
87 }
```

6 DP

6.1 Aliens

```
1 int n; ll k;
2 vector<ll> a;
3 vector<pll> dp[2];
4 void init() {
5     cin >> n >> k;
6     Each(i, dp) i.clear(), i.resize(n);
7     a.clear(); a.resize(n);
8     Each(i, a) cin >> i;
9 }
10 pll calc(ll p) {
11     dp[0][0] = mp(0, 0);
12     dp[1][0] = mp(-a[0], 0);
13     FOR(i, 1, n, 1) {
14         if (dp[0][i-1].F > dp[1][i-1].F + a[i] - p) {
15             dp[0][i] = dp[0][i-1];
16         } else if (dp[0][i-1].F < dp[1][i-1].F + a[i] - p) {
17             dp[0][i] = mp(dp[1][i-1].F + a[i] - p, dp[1][i-1].S+1);
18         } else {
19             dp[0][i] = mp(dp[0][i-1].F, min(dp[0][i-1].S, dp[1][i-1].S+1));
20         }
21         if (dp[0][i-1].F - a[i] > dp[1][i-1].F) {
22             dp[1][i] = mp(dp[0][i-1].F - a[i], dp[0][i-1].S);
23         } else if (dp[0][i-1].F - a[i] < dp[1][i-1].F) {
24             dp[1][i] = dp[1][i-1];
25         } else {
26             dp[1][i] = mp(dp[1][i-1].F, min(dp[0][i-1].S, dp[1][i-1].S));
27         }
28     }
29     return dp[0][n-1];
30 }
31 void solve() {
32     ll l = 0, r = 1e7;
33     pll res = calc(0);
34     if (res.S <= k) return cout << res.F << endl, void();
35     while (l < r) {
36         ll mid = (l+r) >> 1;
37         res = calc(mid);
38         if (res.S <= k) r = mid;
39         else l = mid+1;
40     }
41     res = calc(l);
42     cout << res.F + k*l << endl;
43 }
```

6.2 SOS DP

```
1 for (int msk = 0; msk < (1<<n); msk++) {
2     for (int i = 1; i <= n; i++) {
3         if (msk & (1<<(i - 1))) {
4             // dp[msk][i] = dp[msk][i - 1] + dp[msk ^
                (1<<(i - 1))][i - 1];
5         } else {
6             // dp[msk][i] = dp[msk][i - 1];
7         }
8     }
```

```
9 | }
```

7 Graph

7.1 Tree Centroid

```
1 int n;
2 vector<vector<int>> G;
3
4 pii centroid;
5 vector<int> sz, mxcc; // mxcc[u]: max component size
6 // after removing u
7 void dfs(int u, int p) {
8     sz[u] = 1;
9     for (auto& v : G[u]) {
10         if (v == p) continue;
11         dfs(v, u);
12         sz[u] += sz[v];
13         mxcc[u] = max(mxcc[u], sz[v]);
14     }
15     mxcc[u] = max(mxcc[u], n - sz[u]);
16 }
17
18 void find_centroid() {
19     centroid = pii{-1, -1};
20     reset(sz, n + 1, 0);
21     reset(mxcc, n + 1, 0);
22     dfs(1, 1);
23     for (int u = 1; u <= n; u++) {
24         if (mxcc[u] <= n / 2) {
25             if (centroid.first != -1) centroid.second =
26                 u;
27             else centroid.first = u;
28         }
29     }
30 }
```

7.2 Bellman-Ford + SPFA

```
1 int n, m;
2
3 // Graph
4 vector<vector<pair<int, ll>>> g;
5 vector<ll> dis;
6 vector<bool> negCycle;
7
8 // SPFA
9 vector<int> rlx;
10 queue<int> q;
11 vector<bool> inq;
12 vector<int> pa;
13 void SPFA(vector<int>& src) {
14     dis.assign(n+1, LINF);
15     negCycle.assign(n+1, false);
16     rlx.assign(n+1, 0);
17     while (!q.empty()) q.pop();
18     inq.assign(n+1, false);
19     pa.assign(n+1, -1);
20
21     for (auto& s : src) {
22         dis[s] = 0;
23         q.push(s); inq[s] = true;
24     }
25
26     while (!q.empty()) {
27         int u = q.front();
28         q.pop(); inq[u] = false;
29         if (rlx[u] >= n) {
30             negCycle[u] = true;
31         }
32         else for (auto& e : g[u]) {
33             int v = e.first;
34             ll w = e.second;
35             if (dis[v] > dis[u] + w) {
36                 dis[v] = dis[u] + w;
37                 rlx[v] = rlx[u] + 1;
38                 pa[v] = u;
39                 if (!inq[v]) {
```

```
40                     q.push(v);
41                     inq[v] = true;
42                 } } } } }
43
44 // Bellman-Ford
45 queue<int> q;
46 vector<int> pa;
47 void BellmanFord(vector<int>& src) {
48     dis.assign(n+1, LINF);
49     negCycle.assign(n+1, false);
50     pa.assign(n+1, -1);
51
52     for (auto& s : src) dis[s] = 0;
53
54     for (int rlx = 1; rlx <= n; rlx++) {
55         for (int u = 1; u <= n; u++) {
56             if (dis[u] == LINF) continue; // Important
57             !!
58             for (auto& e : g[u]) {
59                 int v = e.first; ll w = e.second;
60                 if (dis[v] > dis[u] + w) {
61                     dis[v] = dis[u] + w;
62                     pa[v] = u;
63                     if (rlx == n) negCycle[v] = true;
64                 } } } } }
65
66 // Negative Cycle Detection
67 void NegCycleDetect() {
68     /* No Neg Cycle: NO
69     Exist Any Neg Cycle:
70     YES
71     v0 v1 v2 ... vk v0 */
72
73     vector<int> src;
74     for (int i = 1; i <= n; i++)
75         src.emplace_back(i);
76
77     SPFA(src);
78     // BellmanFord(src);
79
80     int ptr = -1;
81     for (int i = 1; i <= n; i++) if (negCycle[i])
82         { ptr = i; break; }
83
84     if (ptr == -1) { return cout << "NO" << endl, void
85         (); }
86
87     cout << "YES\n";
88     vector<int> ans;
89     vector<bool> vis(n+1, false);
90
91     while (true) {
92         ans.emplace_back(ptr);
93         if (vis[ptr]) break;
94         vis[ptr] = true;
95         ptr = pa[ptr];
96     }
97     reverse(ans.begin(), ans.end());
98
99     vis.assign(n+1, false);
100     for (auto& x : ans) {
101         cout << x << ' ';
102         if (vis[x]) break;
103         vis[x] = true;
104     }
105     cout << endl;
106 }
107
108 // Distance Calculation
109 void calcDis(int s) {
110     vector<int> src;
111     src.emplace_back(s);
112     SPFA(src);
113     // BellmanFord(src);
114
115     while (!q.empty()) q.pop();
116     for (int i = 1; i <= n; i++)
117         if (negCycle[i]) q.push(i);
118
119     while (!q.empty()) {
```



```

120     int u = q.front(); q.pop();
121     for (auto& e : g[u]) {
122         int v = e.first;
123         if (!negCycle[v]) {
124             q.push(v);
125             negCycle[v] = true;
126     } } } }

```

7.3 BCC - AP

```

1  int n, m;
2  int low[maxn], dfn[maxn], instp;
3  vector<int> E, g[maxn];
4  bitset<maxn> isap;
5  bitset<maxm> vis;
6  stack<int> stk;
7  int bccnt;
8  vector<int> bcc[maxn];
9  inline void popout(int u) {
10     bccnt++;
11     bcc[bccnt].emplace_back(u);
12     while (!stk.empty()) {
13         int v = stk.top();
14         if (u == v) break;
15         stk.pop();
16         bcc[bccnt].emplace_back(v);
17     }
18 }
19 void dfs(int u, bool rt = 0) {
20     stk.push(u);
21     low[u] = dfn[u] = ++instp;
22     int kid = 0;
23     Each(e, g[u]) {
24         if (vis[e]) continue;
25         vis[e] = true;
26         int v = E[e]^u;
27         if (!dfn[v]) {
28             // tree edge
29             kid++; dfs(v);
30             low[u] = min(low[u], low[v]);
31             if (!rt && low[v] >= dfn[u]) {
32                 // bcc found: u is ap
33                 isap[u] = true;
34                 popout(u);
35             }
36         } else {
37             // back edge
38             low[u] = min(low[u], dfn[v]);
39         }
40     }
41     // special case: root
42     if (rt) {
43         if (kid > 1) isap[u] = true;
44         popout(u);
45     }
46 }
47 void init() {
48     cin >> n >> m;
49     fill(low, low+maxn, INF);
50     REP(i, m) {
51         int u, v;
52         cin >> u >> v;
53         g[u].emplace_back(i);
54         g[v].emplace_back(i);
55         E.emplace_back(u^v);
56     }
57 }
58 void solve() {
59     FOR(i, 1, n+1, 1) {
60         if (!dfn[i]) dfs(i, true);
61     }
62     vector<int> ans;
63     int cnt = 0;
64     FOR(i, 1, n+1, 1) {
65         if (isap[i]) cnt++, ans.emplace_back(i);
66     }
67     cout << cnt << endl;
68     Each(i, ans) cout << i << ' ';
69     cout << endl;
70 }

```

7.4 BCC - Bridge

```

1  int n, m;
2  vector<int> g[maxn], E;
3  int low[maxn], dfn[maxn], instp;
4  int bccnt, bccid[maxn];
5  stack<int> stk;
6  bitset<maxm> vis, isbrg;
7  void init() {
8     cin >> n >> m;
9     REP(i, m) {
10         int u, v;
11         cin >> u >> v;
12         E.emplace_back(u^v);
13         g[u].emplace_back(i);
14         g[v].emplace_back(i);
15     }
16     fill(low, low+maxn, INF);
17 }
18 void popout(int u) {
19     bccnt++;
20     while (!stk.empty()) {
21         int v = stk.top();
22         if (v == u) break;
23         stk.pop();
24         bccid[v] = bccnt;
25     }
26 }
27 void dfs(int u) {
28     stk.push(u);
29     low[u] = dfn[u] = ++instp;
30
31     Each(e, g[u]) {
32         if (vis[e]) continue;
33         vis[e] = true;
34
35         int v = E[e]^u;
36         if (dfn[v]) {
37             // back edge
38             low[u] = min(low[u], dfn[v]);
39         } else {
40             // tree edge
41             dfs(v);
42             low[u] = min(low[u], low[v]);
43             if (low[v] == dfn[v]) {
44                 isbrg[e] = true;
45                 popout(u);
46             }
47         }
48     }
49 }
50 void solve() {
51     FOR(i, 1, n+1, 1) {
52         if (!dfn[i]) dfs(i);
53     }
54     vector<pii> ans;
55     vis.reset();
56     FOR(u, 1, n+1, 1) {
57         Each(e, g[u]) {
58             if (!isbrg[e] || vis[e]) continue;
59             vis[e] = true;
60             int v = E[e]^u;
61             ans.emplace_back(mp(u, v));
62         }
63     }
64     cout << (int)ans.size() << endl;
65     Each(e, ans) cout << e.F << ' ' << e.S << endl;
66 }

```

7.5 SCC - Tarjan

```

1  // 2-SAT
2  vector<int> E, g[maxn]; // 1~n, n+1~2n
3  int low[maxn], in[maxn], instp;
4  int sccnt, sccid[maxn];
5
6  stack<int> stk;
7  bitset<maxn> ins, vis;
8
9  int n, m;
10

```

```

11 void init() {
12     cin >> m >> n;
13     E.clear();
14     fill(g, g+maxn, vector<int>());
15     fill(low, low+maxn, INF);
16     memset(in, 0, sizeof(in));
17     instp = 1;
18     sccnt = 0;
19     memset(sccid, 0, sizeof(sccid));
20     ins.reset();
21     vis.reset();
22 }
23
24 inline int no(int u) {
25     return (u > n ? u-n : u+n);
26 }
27
28 int ecnt = 0;
29 inline void clause(int u, int v) {
30     E.eb(no(u)^v);
31     g[no(u)].eb(ecnt++);
32     E.eb(no(v)^u);
33     g[no(v)].eb(ecnt++);
34 }
35
36 void dfs(int u) {
37     in[u] = instp++;
38     low[u] = in[u];
39     stk.push(u);
40     ins[u] = true;
41
42     Each(e, g[u]) {
43         if (vis[e]) continue;
44         vis[e] = true;
45
46         int v = E[e]^u;
47         if (ins[v]) low[u] = min(low[u], in[v]);
48         else if (!in[v]) {
49             dfs(v);
50             low[u] = min(low[u], low[v]);
51         }
52     }
53
54     if (low[u] == in[u]) {
55         sccnt++;
56         while (!stk.empty()) {
57             int v = stk.top();
58             stk.pop();
59             ins[v] = false;
60             sccid[v] = sccnt;
61             if (u == v) break;
62         }
63     }
64 }
65
66 int main() {
67     WiWiHorz
68     init();
69
70     REP(i, m) {
71         char su, sv;
72         int u, v;
73         cin >> su >> u >> sv >> v;
74         if (su == '-') u = no(u);
75         if (sv == '-') v = no(v);
76         clause(u, v);
77     }
78
79     FOR(i, 1, 2*n+1, 1) {
80         if (!in[i]) dfs(i);
81     }
82
83     FOR(u, 1, n+1, 1) {
84         int du = no(u);
85         if (sccid[u] == sccid[du]) {
86             return cout << "IMPOSSIBLE\n", 0;
87         }
88     }
89
90     FOR(u, 1, n+1, 1) {
91         int du = no(u);

```

```

93         cout << (sccid[u] < sccid[du] ? '+' : '-') << '
94         ';
95     }
96     cout << endl;
97     return 0;
98 }

```

7.6 Eulerian Path - Undir

```

1 // from 1 to n
2 #define gg return cout << "IMPOSSIBLE\n", void();
3
4 int n, m;
5 vector<int> g[maxn];
6 bitset<maxn> inodd;
7
8 void init() {
9     cin >> n >> m;
10    inodd.reset();
11    for (int i = 0; i < m; i++) {
12        int u, v; cin >> u >> v;
13        inodd[u] = inodd[u] ^ true;
14        inodd[v] = inodd[v] ^ true;
15        g[u].emplace_back(v);
16        g[v].emplace_back(u);
17    }
18    stack<int> stk;
19    void dfs(int u) {
20        while (!g[u].empty()) {
21            int v = g[u].back();
22            g[u].pop_back();
23            dfs(v);
24        }
25        stk.push(u);

```

7.7 Eulerian Path - Dir

```

1 // from node 1 to node n
2 #define gg return cout << "IMPOSSIBLE\n", 0
3
4 int n, m;
5 vector<int> g[maxn];
6 stack<int> stk;
7 int in[maxn], out[maxn];
8
9 void init() {
10    cin >> n >> m;
11    for (int i = 0; i < m; i++) {
12        int u, v; cin >> u >> v;
13        g[u].emplace_back(v);
14        out[u]++, in[v]++;
15    }
16    for (int i = 1; i <= n; i++) {
17        if (i == 1 && out[i]-in[i] != 1) gg;
18        if (i == n && in[i]-out[i] != 1) gg;
19        if (i != 1 && i != n && in[i] != out[i]) gg;
20    }
21    void dfs(int u) {
22        while (!g[u].empty()) {
23            int v = g[u].back();
24            g[u].pop_back();
25            dfs(v);
26        }
27        stk.push(u);
28    }
29    void solve() {
30        dfs(1)
31        for (int i = 1; i <= n; i++)
32            if ((int)g[i].size()) gg;
33        while (!stk.empty()) {
34            int u = stk.top();
35            stk.pop();
36            cout << u << ' ';
37        }

```

7.8 Hamilton Path

```

1 // top down DP

```



```

2 // Be Aware Of Multiple Edges
3 int n, m;
4 ll dp[maxn][1<<maxn];
5 int adj[maxn][maxn];
6
7 void init() {
8     cin >> n >> m;
9     fill(dp[0], dp[maxn-1]+(1<<maxn), -1);
10 }
11
12 void DP(int i, int msk) {
13     if (dp[i][msk] != -1) return;
14     dp[i][msk] = 0;
15     REP(j, n) if (j != i && (msk & (1<<j)) && adj[j][i]) {
16         int sub = msk ^ (1<<i);
17         if (dp[j][sub] == -1) DP(j, sub);
18         dp[i][msk] += dp[j][sub] * adj[j][i];
19         if (dp[i][msk] >= MOD) dp[i][msk] %= MOD;
20     }
21 }
22
23 int main() {
24     WiWiHorz
25     init();
26
27     REP(i, m) {
28         int u, v;
29         cin >> u >> v;
30         if (u == v) continue;
31         adj[--u][--v]++;
32     }
33
34     dp[0][1] = 1;
35     FOR(i, 1, n, 1) {
36         dp[i][1] = 0;
37         dp[i][1|(1<<i)] = adj[0][i];
38     }
39     FOR(msk, 1, (1<<n), 1) {
40         if (msk == 1) continue;
41         dp[0][msk] = 0;
42     }
43
44     DP(n-1, (1<<n)-1);
45     cout << dp[n-1][(1<<n)-1] << endl;
46
47     return 0;
48 }

```

7.9 Kth Shortest Path

```

1 // time: O(|E| \lg |E|+|V| \lg |V|+K)
2 // memory: O(|E| \lg |E|+|V|)
3 struct KSP{ // 1-base
4     struct nd{
5         int u,v; ll d;
6         nd(int ui=0,int vi=0,ll di=INF){ u=ui; v=vi; d=di; }
7     };
8     struct heap{ nd* edge; int dep; heap* chd[4]; };
9     static int cmp(heap* a,heap* b)
10     { return a->edge->d > b->edge->d; }
11     struct node{
12         int v; ll d; heap* H; nd* E;
13         node(){
14             node(ll _d,int _v,nd* _E){ d=_d; v=_v; E=_E; }
15             node(heap* _H,ll _d){ H=_H; d=_d; }
16             friend bool operator<(node a,node b)
17             { return a.d>b.d; }
18         };
19     int n,k,s,t,dst[N]; nd *nxt[N];
20     vector<nd*> g[N],rg[N]; heap *nullNd,*head[N];
21     void init(int _n,int _k,int _s,int _t){
22         n=_n; k=_k; s=_s; t=_t;
23         for(int i=1;i<=n;i++){
24             g[i].clear(); rg[i].clear();
25             nxt[i]=NULL; head[i]=NULL; dst[i]=-1;
26         }
27     }

```

```

28 void addEdge(int ui,int vi,ll di){
29     nd* e=new nd(ui,vi,di);
30     g[ui].push_back(e); rg[vi].push_back(e);
31 }
32 queue<int> dfsQ;
33 void dijkstra(){
34     while(dfsQ.size()) dfsQ.pop();
35     priority_queue<node> Q; Q.push(node(0,t,NULL));
36     while (!Q.empty()){
37         node p=Q.top(); Q.pop(); if(dst[p.v]!=-1)continue;
38         dst[p.v]=p.d; nxt[p.v]=p.E; dfsQ.push(p.v);
39         for(auto e:rg[p.v]) Q.push(node(p.d+e->d,e->u,e));
40     }
41 }
42 heap* merge(heap* curNd,heap* newNd){
43     if(curNd==nullNd) return newNd;
44     heap* root=new heap; memcpy(root,curNd,sizeof(heap));
45     if(newNd->edge->d<curNd->edge->d){
46         root->edge=newNd->edge;
47         root->chd[2]=newNd->chd[2];
48         root->chd[3]=newNd->chd[3];
49         newNd->edge=curNd->edge;
50         newNd->chd[2]=curNd->chd[2];
51         newNd->chd[3]=curNd->chd[3];
52     }
53     if(root->chd[0]->dep<root->chd[1]->dep)
54         root->chd[0]=merge(root->chd[0],newNd);
55     else root->chd[1]=merge(root->chd[1],newNd);
56     root->dep=max(root->chd[0]->dep,
57                 root->chd[1]->dep)+1;
58     return root;
59 }
60 vector<heap*> V;
61 void build(){
62     nullNd=new heap; nullNd->dep=0; nullNd->edge=new nd;
63     fill(nullNd->chd,nullNd->chd+4,nullNd);
64     while(not dfsQ.empty()){
65         int u=dfsQ.front(); dfsQ.pop();
66         if(!nxt[u]) head[u]=nullNd;
67         else head[u]=head[nxt[u]->v];
68         V.clear();
69         for(auto&& e:g[u]){
70             int v=e->v;
71             if(dst[v]==-1) continue;
72             e->d+=dst[v]-dst[u];
73             if(nxt[u]!=e){
74                 heap* p=new heap; fill(p->chd,p->chd+4,nullNd);
75                 p->dep=1; p->edge=e; V.push_back(p);
76             }
77         }
78         if(V.empty()) continue;
79         make_heap(V.begin(),V.end(),cmp);
80 #define L(X) ((X<<1)+1)
81 #define R(X) ((X<<1)+2)
82         for(size_t i=0;i<V.size();i++){
83             if(L(i)<V.size()) V[i]->chd[2]=V[L(i)];
84             else V[i]->chd[2]=nullNd;
85             if(R(i)<V.size()) V[i]->chd[3]=V[R(i)];
86             else V[i]->chd[3]=nullNd;
87         }
88         head[u]=merge(head[u],V.front());
89     }
90 }
91 vector<ll> ans;
92 void first_K(){
93     ans.clear(); priority_queue<node> Q;
94     if(dst[s]==-1) return;
95     ans.push_back(dst[s]);
96     if(head[s]!=nullNd)
97         Q.push(node(head[s],dst[s]+head[s]->edge->d));
98     for(int _=1;_<k and not Q.empty();_++){
99         node p=Q.top(); Q.pop(); ans.push_back(p.d);
100         if(head[p.H->edge->v]!=nullNd){
101             q.H=head[p.H->edge->v]; q.d=p.d+q.H->edge->d;
102             Q.push(q);
103         }
104         for(int i=0;i<4;i++)

```

```

105     if(p.H->chd[i]!=nullNd){
106         q.H=p.H->chd[i];
107         q.d=p.d-p.H->edge->d+p.H->chd[i]->edge->d;
108         Q.push(q);
109     } }
110 void solve(){ // ans[i] stores the i-th shortest path
111     dijkstra(); build();
112     first_K(); // ans.size() might less than k
113 }
114 } solver;

```

7.10 System of Difference Constraints

```

1 vector<vector<pair<int, ll>>> G;
2 void add(int u, int v, ll w) {
3     G[u].emplace_back(make_pair(v, w));
4 }

```

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

8 String

8.1 Rolling Hash

```

1 const ll C = 27;
2 inline int id(char c) {return c-'a'+1;}
3 struct RollingHash {
4     string s; int n; ll mod;
5     vector<ll> Cexp, hs;
6     RollingHash(string& _s, ll _mod):
7         s(_s), n((int)_s.size()), mod(_mod)
8     {
9         Cexp.assign(n, 0);
10        hs.assign(n, 0);
11        Cexp[0] = 1;
12        for (int i = 1; i < n; i++) {
13            Cexp[i] = Cexp[i-1] * C;
14            if (Cexp[i] >= mod) Cexp[i] %= mod;
15        }
16        hs[0] = id(s[0]);
17        for (int i = 1; i < n; i++) {
18            hs[i] = hs[i-1] * C + id(s[i]);
19            if (hs[i] >= mod) hs[i] %= mod;
20        }
21        inline ll query(int l, int r) {
22            ll res = hs[r] - (l ? hs[l-1] * Cexp[r-l+1] : 0);
23            res = (res % mod + mod) % mod;
24            return res; }
25 };

```

8.2 Trie

```

1 struct node {
2     int c[26]; ll cnt;
3     node(): cnt(0) {memset(c, 0, sizeof(c));}
4     node(ll x): cnt(x) {memset(c, 0, sizeof(c));}
5 };
6 struct Trie {
7     vector<node> t;

```

```

8     void init() {
9         t.clear();
10        t.emplace_back(node());
11    }
12    void insert(string s) { int ptr = 0;
13        for (auto& i : s) {
14            if (!t[ptr].c[i-'a']) {
15                t.emplace_back(node());
16                t[ptr].c[i-'a'] = (int)t.size()-1; }
17            ptr = t[ptr].c[i-'a']; }
18        t[ptr].cnt++; }
19 } trie;

```

8.3 KMP

```

1 int n, m;
2 string s, p;
3 vector<int> f;
4 void build() {
5     f.clear(); f.resize(m, 0);
6     int ptr = 0; for (int i = 1; i < m; i++) {
7         while (ptr && p[i] != p[ptr]) ptr = f[ptr-1];
8         if (p[i] == p[ptr]) ptr++;
9         f[i] = ptr;
10    } }
11 void init() {
12     cin >> s >> p;
13     n = (int)s.size();
14     m = (int)p.size();
15     build(); }
16 void solve() {
17     int ans = 0, pi = 0;
18     for (int si = 0; si < n; si++) {
19         while (pi && s[si] != p[pi]) pi = f[pi-1];
20         if (s[si] == p[pi]) pi++;
21         if (pi == m) ans++, pi = f[pi-1];
22     }
23     cout << ans << endl; }

```

8.4 Z Value

```

1 string is, it, s;
2 int n; vector<int> z;
3 void init() {
4     cin >> is >> it;
5     s = it+'0'+is;
6     n = (int)s.size();
7     z.resize(n, 0); }
8 void solve() {
9     int ans = 0; z[0] = n;
10    for (int i = 1, l = 0, r = 0; i < n; i++) {
11        if (i <= r) z[i] = min(z[i-l], r-i+1);
12        while (i+z[i] < n && s[z[i]] == s[i+z[i]]) z[i]++;
13        if (i+z[i]-1 > r) l = i, r = i+z[i]-1;
14        if (z[i] == (int)it.size()) ans++;
15    }
16    cout << ans << endl; }

```

8.5 Manacher

```

1 int n; string S, s;
2 vector<int> m;
3 void manacher() {
4     s.clear(); s.resize(2*n+1, '.');
5     for (int i = 0, j = 1; i < n; i++, j += 2) s[j] = S[i];
6     m.clear(); m.resize(2*n+1, 0);
7     // m[i] := max k such that s[i-k, i+k] is palindrome
8     int mx = 0, mxk = 0;
9     for (int i = 1; i < 2*n+1; i++) {
10        if (mx-(i-mx) >= 0) m[i] = min(m[mx-(i-mx)], mx+mxk-i);
11        while (0 <= i-m[i]-1 && i+m[i]+1 < 2*n+1 &&
12            s[i-m[i]-1] == s[i+m[i]+1]) m[i]++;
13        if (i+m[i] > mx+mxk) mx = i, mxk = m[i];
14    } }
15 void init() { cin >> S; n = (int)S.size(); }
16 void solve() {
17     manacher();

```

```

18 int mx = 0, ptr = 0;
19 for (int i = 0; i < 2*n+1; i++) if (mx < m[i])
20 { mx = m[i]; ptr = i; }
21 for (int i = ptr-mx; i <= ptr+mx; i++)
22 if (s[i] != '.') cout << s[i];
23 cout << endl; }

```

8.6 Suffix Array

```

1 #define F first
2 #define S second
3 struct SuffixArray { // don't forget s += "$";
4     int n; string s;
5     vector<int> suf, lcp, rk;
6     vector<int> cnt, pos;
7     vector<pair<pii, int>> buc[2];
8     void init(string _s) {
9         s = _s; n = (int)s.size();
10    // resize(n): suf, rk, cnt, pos, lcp, buc[0~1]
11    }
12    void radix_sort() {
13        for (int t : {0, 1}) {
14            fill(cnt.begin(), cnt.end(), 0);
15            for (auto& i : buc[t]) cnt[ (t ? i.F.F : i.
16                F.S) ]++;
17            for (int i = 0; i < n; i++)
18                pos[i] = (i ? 0 : pos[i-1] + cnt[i-1])
19                ;
20            for (auto& i : buc[t])
21                buc[t^1][pos[ (t ? i.F.F : i.F.S) ]++]
22                = i;
23        }
24        bool fill_suf() {
25            bool end = true;
26            for (int i = 0; i < n; i++) suf[i] = buc[0][i].
27                S;
28            rk[suf[0]] = 0;
29            for (int i = 1; i < n; i++) {
30                int dif = (buc[0][i].F != buc[0][i-1].F);
31                end &= dif;
32                rk[suf[i]] = rk[suf[i-1]] + dif;
33            } return end;
34        }
35        void sa() {
36            for (int i = 0; i < n; i++)
37                buc[0][i] = make_pair(make_pair(s[i], s[i])
38                , i);
39            sort(buc[0].begin(), buc[0].end());
40            if (fill_suf()) return;
41            for (int k = 0; (1<k) < n; k++) {
42                for (int i = 0; i < n; i++)
43                    buc[0][i] = make_pair(make_pair(rk[i],
44                    rk[(i + (1<k)) % n]), i);
45                radix_sort();
46                if (fill_suf()) return;
47            }
48            void LCP() { int k = 0;
49                for (int i = 0; i < n-1; i++) {
50                    if (rk[i] == 0) continue;
51                    int pi = rk[i];
52                    int j = suf[pi-1];
53                    while (i+k < n && j+k < n && s[i+k] == s[j+
54                        k]) k++;
55                    lcp[pi] = k;
56                    k = max(k-1, 0);
57                }
58            }
59        };
60        SuffixArray suffixarray;

```

8.7 SA-IS

```

1 const int N=300010;
2 struct SA{
3     #define REP(i,n) for(int i=0;i<int(n);i++)
4     #define REP1(i,a,b) for(int i=(a);i<=int(b);i++)
5     bool _t[N*2]; int _s[N*2],_sa[N*2];
6     int _c[N*2],x[N],_p[N],_q[N*2],hei[N],r[N];
7     int operator [](int i){ return _sa[i]; }
8     void build(int *s,int n,int m){
9         memcpy(_s,s,sizeof(int)*n);

```

```

10     sais(_s,_sa,_p,_q,_t,_c,n,m); mkhei(n);
11 }
12 void mkhei(int n){
13     REP(i,n) r[_sa[i]]=i;
14     hei[0]=0;
15     REP(i,n) if(r[i]) {
16         int ans=i>0?max(hei[r[i-1]]-1,0):0;
17         while(_s[i+ans]==_s[_sa[r[i]-1]+ans]) ans++;
18         hei[r[i]]=ans;
19     }
20 }
21 void sais(int *s,int *sa,int *p,int *q,bool *t,int *c
22     ,int n,int z){
23     bool uniq=t[n-1]=true,neq;
24     int nn=0,nmxz=-1,*nsa=sa+n,*ns=s+n,lst=-1;
25     #define MS0(x,n) memset((x),0,n*sizeof(*(x)))
26     #define MAGIC(XD) MS0(sa,n);\
27     memcpy(x,c,sizeof(int)*z); XD;\
28     memcpy(x+1,c,sizeof(int)*(z-1));\
29     REP(i,n) if(sa[i]&&!t[sa[i]-1]) sa[x[s[sa[i]-1]]++]
30     =sa[i]-1;\
31     memcpy(x,c,sizeof(int)*z);\
32     for(int i=n-1;i>=0;i--) if(sa[i]&&t[sa[i]-1]) sa[--x[s[
33     sa[i]-1]]]=sa[i]-1;
34     MS0(c,z); REP(i,n) uniq&=++c[s[i]]<2;
35     REP(i,z-1) c[i+1]+=c[i];
36     if(uniq) { REP(i,n) sa[--c[s[i]]]=i; return; }
37     for(int i=n-2;i>=0;i--)
38         t[i]=(s[i]==s[i+1]?t[i+1]:s[i]<s[i+1]);
39     MAGIC(REP1(i,1,n-1) if(t[i]&&t[i-1]) sa[--x[s[i]
40     ]]=p[q[i]=nn++]=i);
41     REP(i,n) if(sa[i]&&t[sa[i]]&&t[sa[i]-1]){
42         neq=lst<0||memcmp(s+sa[i],s+lst,(p[q[sa[i]]+1]-sa
43         [i])*sizeof(int));
44         ns[q[lst=sa[i]]]=nmxz+=neq;
45     }
46     sais(ns,nsa,p+nn,q+n,t+n,c+z,nn,nmxz+1);
47     MAGIC(for(int i=nn-1;i>=0;i--) sa[--x[s[p[nsa[i]
48     ]]]]=p[nsa[i]]);
49 }
50 }sa;
51 int H[N],SA[N],RA[N];
52 void suffix_array(int* ip,int len){
53     // should padding a zero in the back
54     // ip is int array, len is array length
55     // ip[0..n-1] != 0, and ip[len]=0
56     ip[len++]=0; sa.build(ip,len,128);
57     memcpy(H,sa.hei+1,len<<2); memcpy(SA,sa._sa+1,len<<2)
58     ;
59     for(int i=0;i<len;i++) RA[i]=sa.r[i]-1;
60     // resulting height, sa array \in [0,len)
61 }

```

8.8 Minimum Rotation

```

1 //rotate(begin(s), begin(s)+minRotation(s), end(s))
2 int minRotation(string s) {
3     int a = 0, n = s.size(); s += s;
4     for(int b = 0; b < n; b++) for(int k = 0; k < n; k++) {
5         if(a + k == b || s[a + k] < s[b + k]) {
6             b += max(0, k - 1);
7             break; }
8         if(s[a + k] > s[b + k]) {
9             a = b;
10            break;
11        } }
12    return a; }

```

8.9 Aho Corasick

```

1 struct ACautomata{
2     struct Node{
3         int cnt;
4         Node *go[26], *fail, *dic;
5         Node (){
6             cnt = 0; fail = 0; dic=0;
7             memset(go,0,sizeof(go));
8         }
9     }pool[1048576],*root;
10    int nMem;

```

```

11 Node* new_Node(){
12     pool[nMem] = Node();
13     return &pool[nMem++];
14 }
15 void init() { nMem = 0; root = new_Node(); }
16 void add(const string &str) { insert(root, str, 0); }
17 void insert(Node *cur, const string &str, int pos){
18     for(int i=pos; i<str.size(); i++){
19         if(!cur->go[str[i]-'a'])
20             cur->go[str[i]-'a'] = new_Node();
21         cur=cur->go[str[i]-'a'];
22     }
23     cur->cnt++;
24 }
25 void make_fail(){
26     queue<Node*> que;
27     que.push(root);
28     while (!que.empty()){
29         Node* fr=que.front(); que.pop();
30         for (int i=0; i<26; i++){
31             if (fr->go[i]){
32                 Node *ptr = fr->fail;
33                 while (ptr && !ptr->go[i]) ptr = ptr->fail;
34                 fr->go[i]->fail=ptr=(ptr?ptr->go[i]:root);
35                 fr->go[i]->dic=(ptr->cnt?ptr:ptr->dic);
36                 que.push(fr->go[i]);
37             } } } }
38 }AC;

```

9 Geometry

9.1 Basic Operations

```

1 typedef long long T;
2 // typedef long double T;
3 const long double eps = 1e-8;
4
5 short sgn(T x) {
6     if (abs(x) < eps) return 0;
7     return x < 0 ? -1 : 1;
8 }
9
10 struct Pt {
11     T x, y;
12     Pt(T _x=0, T _y=0):x(_x), y(_y) {}
13     Pt operator+(Pt a) { return Pt(x+a.x, y+a.y); }
14     Pt operator-(Pt a) { return Pt(x-a.x, y-a.y); }
15     Pt operator*(T a) { return Pt(x*a, y*a); }
16     Pt operator/(T a) { return Pt(x/a, y/a); }
17     T operator*(Pt a) { return x*a.x + y*a.y; }
18     T operator^(Pt a) { return x*a.y - y*a.x; }
19     bool operator<(Pt a)
20     { return x < a.x || (x == a.x && y < a.y); }
21     //return sgn(x-a.x) < 0 || (sgn(x-a.x) == 0 && sgn(y-a.
22     y) < 0); }
23     bool operator==(Pt a)
24     { return sgn(x-a.x) == 0 && sgn(y-a.y) == 0; }
25 };
26
27 Pt mv(Pt a, Pt b) { return b-a; }
28 T len2(Pt a) { return a*a; }
29 T dis2(Pt a, Pt b) { return len2(b-a); }
30
31 short ori(Pt a, Pt b) { return ((a^b)>0) - ((a^b)<0); }
32 bool onseg(Pt p, Pt l1, Pt l2) {
33     Pt a = mv(p, l1), b = mv(p, l2);
34     return ((a^b) == 0) && ((a*b) <= 0);
35 }

```

9.2 InPoly

```

1 short inPoly(Pt p) {
2     // 0=Bound 1=In -1=Out
3     REP(i, n) if (onseg(p, E[i], E[(i+1)%n])) return 0;
4     int cnt = 0;
5     REP(i, n) if (banana(p, Pt(p.x+1, p.y+2e9),
6         E[i], E[(i+1)%n])) cnt ^= 1;
7     return (cnt ? 1 : -1);
8 }

```

9.3 Sort by Angle

```

1 int ud(Pt a) { // up or down half plane
2     if (a.y > 0) return 0;
3     if (a.y < 0) return 1;
4     return (a.x >= 0 ? 0 : 1);
5 }
6 sort(ALL(E), [&](const Pt& a, const Pt& b){
7     if (ud(a) != ud(b)) return ud(a) < ud(b);
8     return (a^b) > 0;
9 });

```

9.4 Line Intersect Check

```

1 inline bool banana(Pt p1, Pt p2, Pt q1, Pt q2) {
2     if (onseg(p1, q1, q2) || onseg(p2, q1, q2) ||
3         onseg(q1, p1, p2) || onseg(q2, p1, p2)) {
4         return true;
5     }
6     Pt p = mv(p1, p2), q = mv(q1, q2);
7     return (ori(p, mv(p1, q1)) * ori(p, mv(p1, q2)) < 0 &&
8         ori(q, mv(q1, p1)) * ori(q, mv(q1, p2)) < 0);
9 }

```

9.5 Line Intersection

```

1 // T: Long double
2 Pt bananaPoint(Pt p1, Pt p2, Pt q1, Pt q2) {
3     if (onseg(q1, p1, p2)) return q1;
4     if (onseg(q2, p1, p2)) return q2;
5     if (onseg(p1, q1, q2)) return p1;
6     if (onseg(p2, q1, q2)) return p2;
7     double s = abs(mv(p1, p2) ^ mv(p1, q1));
8     double t = abs(mv(p1, p2) ^ mv(p1, q2));
9     return q2 * (s/(s+t)) + q1 * (t/(s+t));
10 }

```

9.6 Convex Hull

```

1 vector<Pt> hull;
2 void convexHull() {
3     hull.clear(); sort(ALL(E));
4     REP(t, 2) {
5         int b = SZ(hull);
6         Each(ei, E) {
7             while (SZ(hull) - b >= 2 &&
8                 ori(mv(hull[SZ(hull)-2], hull.back()),
9                 mv(hull[SZ(hull)-2], ei)) == -1) {
10                 hull.pop_back();
11             }
12             hull.pb(ei);
13         }
14         hull.pop_back();
15         reverse(ALL(E));
16     } }

```

9.7 Lower Concave Hull

```

1 struct Line {
2     mutable ll m, b, p;
3     bool operator<(const Line& o) const { return m < o.m; }
4     bool operator<(ll x) const { return p < x; }
5 };
6
7 struct LineContainer : multiset<Line, less<>> {
8     // (for doubles, use inf = 1/.0, div(a,b) = a/b)
9     const ll inf = LLONG_MAX;
10     ll div(ll a, ll b) { // floored division
11         return a / b - ((a ^ b) < 0 && a % b); }
12     bool isect(iterator x, iterator y) {
13         if (y == end()) { x->p = inf; return false; }
14         if (x->m == y->m) x->p = x->b > y->b ? inf : -inf;
15         else x->p = div(y->b - x->b, x->m - y->m);
16         return x->p >= y->p;
17     }
18     void add(ll m, ll b) {
19         auto z = insert({m, b, 0}), y = z++, x = y;

```

```

20 while (isect(y, z)) z = erase(z);
21 if (x != begin() && isect(--x, y)) isect(x, y =
    erase(y));
22 while ((y = x) != begin() && (--x)->p >= y->p)
    isect(x, erase(y));
23 }
24 ll query(ll x) {
25     assert(!empty());
26     auto l = *lower_bound(x);
27     return l.m * x + l.b;
28 }
29 }
30 };

```

9.8 Polygon Area

```

1 T dbarea(vector<Pt>& e) {
2     ll res = 0;
3     REP(i, SZ(e)) res += e[i]^e[(i+1)%SZ(e)];
4     return abs(res);
5 }

```

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

```

1 Pt circumcenter(Pt A, Pt B, Pt C) {
2     // a1(x-A.x) + b1(y-A.y) = c1
3     // a2(x-A.x) + b2(y-A.y) = c2
4     // solve using Cramer's rule
5     T a1 = B.x-A.x, b1 = B.y-A.y, c1 = dis2(A, B)/2.0;
6     T a2 = C.x-A.x, b2 = C.y-A.y, c2 = dis2(A, C)/2.0;
7     T D = Pt(a1, b1) ^ Pt(a2, b2);
8     T Dx = Pt(c1, b1) ^ Pt(c2, b2);
9     T Dy = Pt(a1, c1) ^ Pt(a2, c2);
10    if (D == 0) return Pt(-INF, -INF);
11    return A + Pt(Dx/D, Dy/D);
12 }
13 Pt center; T r2;
14 void minEncloseCircle() {
15     mt19937 gen(chrono::steady_clock::now().
        time_since_epoch().count());
16     shuffle(ALL(E), gen);
17     center = E[0], r2 = 0;
18
19     for (int i = 0; i < n; i++) {
20         if (dis2(center, E[i]) <= r2) continue;
21         center = E[i], r2 = 0;
22         for (int j = 0; j < i; j++) {
23             if (dis2(center, E[j]) <= r2) continue;
24             center = (E[i] + E[j]) / 2.0;
25             r2 = dis2(center, E[i]);
26             for (int k = 0; k < j; k++) {
27                 if (dis2(center, E[k]) <= r2) continue;
28                 center = circumcenter(E[i], E[j], E[k]);
29                 r2 = dis2(center, E[i]);
30             }
31         }
32     }
33 }

```

9.11 PolyUnion

```

1 struct PY{
2     int n; Pt pt[5]; double area;
3     Pt& operator[](const int x){ return pt[x]; }
4     void init(){ //n,pt[0~n-1] must be filled
5         area=pt[n-1]^pt[0];
6         for(int i=0;i<n-1;i++) area+=pt[i]^pt[i+1];
7         if((area/=2)<0)reverse(pt,pt+n),area=-area;
8     }

```

```

9 };
10 PY py[500]; pair<double,int> c[5000];
11 inline double segP(Pt &p,Pt &p1,Pt &p2){
12     if(dcmp(p1.x-p2.x)==0) return (p.y-p1.y)/(p2.y-p1.y);
13     return (p.x-p1.x)/(p2.x-p1.x);
14 }
15 double polyUnion(int n){ //py[0~n-1] must be filled
16     int i,j,ii,jj,ta,tb,r,d; double z,w,s,sum=0,tc,td;
17     for(i=0;i<n;i++) py[i][py[i].n]=py[i][0];
18     for(i=0;i<n;i++){
19         for(ii=0;ii<py[i].n;ii++){
20             r=0;
21             c[r++]=make_pair(0.0,0); c[r++]=make_pair(1.0,0);
22             for(j=0;j<n;j++){
23                 if(i==j) continue;
24                 for(jj=0;jj<py[j].n;jj++){
25                     ta=dcmp(tri(py[i][ii],py[i][ii+1],py[j][jj]))
26                     ;
27                     tb=dcmp(tri(py[i][ii],py[i][ii+1],py[j][jj
28                         +1]));
29                     if(ta==0 && tb==0){
30                         if((py[j][jj+1]-py[j][jj])*(py[i][ii+1]-py[
31                             i][ii])>0&&j<i){
32                             c[r++]=make_pair(segP(py[j][jj],py[i][ii
33                                 ],py[i][ii+1]),1);
34                             c[r++]=make_pair(segP(py[j][jj+1],py[i][
35                                 ii],py[i][ii+1]),-1);
36                         }
37                     }else if(ta>0 && tb<0){
38                         tc=tri(py[j][jj],py[j][jj+1],py[i][ii]);
39                         td=tri(py[j][jj],py[j][jj+1],py[i][ii+1]);
40                         c[r++]=make_pair(tc/(tc-td),1);
41                     }else if(ta<0 && tb>0){
42                         tc=tri(py[j][jj],py[j][jj+1],py[i][ii]);
43                         td=tri(py[j][jj],py[j][jj+1],py[i][ii+1]);
44                         c[r++]=make_pair(tc/(tc-td),-1);
45                     }
46                 }
47             }
48             sort(c,c+r);
49             z=min(max(c[0].first,0.0),1.0); d=c[0].second; s
50             =0;
51             for(j=1;j<r;j++){
52                 w=min(max(c[j].first,0.0),1.0);
53                 if(!d) s+=w-z;
54                 d+=c[j].second; z=w;
55             }
56             sum+=(py[i][ii]^py[i][ii+1])*s;
57         }
58     }
59     return sum/2;
60 }

```

9.12 Minkowski Sum

```

1 /* convex hull Minkowski Sum*/
2 #define INF 1000000000000000LL
3 int pos(const Pt& tp){
4     if(tp.Y == 0) return tp.X > 0 ? 0 : 1;
5     return tp.Y > 0 ? 0 : 1;
6 }
7 #define N 300030
8 Pt pt[N], qt[N], rt[N];
9 LL Lx,Rx;
10 int dn,un;
11 inline bool cmp(Pt a, Pt b){
12     int pa=pos(a),pb=pos(b);
13     if(pa==pb) return (a^b)>0;
14     return pa<pb;
15 }
16 int minkowskiSum(int n,int m){
17     int i,j,r,p,q,fi,fj;
18     for(i=1,p=0;i<n;i++){
19         if(pt[i].Y<pt[p].Y ||
20            (pt[i].Y==pt[p].Y && pt[i].X<pt[p].X)) p=i; }
21     for(i=1,q=0;i<m;i++){
22         if(qt[i].Y<qt[q].Y ||
23            (qt[i].Y==qt[q].Y && qt[i].X<qt[q].X)) q=i; }
24     rt[0]=pt[p]+qt[q];
25     r=1; i=p; j=q; fi=fj=0;
26     while(1){
27         if((fj&&j==q) ||
28            (!fi||i==p) &&

```



```

29         cmp(pt[(p+1)%n]-pt[p],qt[(q+1)%m]-qt[q]) ) ){
30         rt[r]=rt[r-1]+pt[(p+1)%n]-pt[p];
31         p=(p+1)%n;
32         fi=1;
33     }else{
34         rt[r]=rt[r-1]+qt[(q+1)%m]-qt[q];
35         q=(q+1)%m;
36         fj=1;
37     }
38     if(r<=1 || ((rt[r]-rt[r-1])^(rt[r-1]-rt[r-2]))!=0)
39         r++;
40     else rt[r-1]=rt[r];
41     if(i==p && j==q) break;
42 }
43 return r-1;
44 }
45 void initInConvex(int n){
46     int i,p,q;
47     LL Ly,Ry;
48     Lx=INF; Rx=-INF;
49     for(i=0;i<n;i++){
50         if(pt[i].X<Lx) Lx=pt[i].X;
51         if(pt[i].X>Rx) Rx=pt[i].X;
52     }
53     Ly=Ry=INF;
54     for(i=0;i<n;i++){
55         if(pt[i].X==Lx && pt[i].Y<Ly){ Ly=pt[i].Y; p=i; }
56         if(pt[i].X==Rx && pt[i].Y>Ry){ Ry=pt[i].Y; q=i; }
57     }
58     for(dn=0,i=p;i!=q;i=(i+1)%n){ qt[dn++]=pt[i]; }
59     qt[dn]=pt[q]; Ly=Ry=-INF;
60     for(i=0;i<n;i++){
61         if(pt[i].X==Lx && pt[i].Y>Ly){ Ly=pt[i].Y; p=i; }
62         if(pt[i].X==Rx && pt[i].Y>Ry){ Ry=pt[i].Y; q=i; }
63     }
64     for(un=0,i=p;i!=q;i=(i+n-1)%n){ rt[un++]=pt[i]; }
65     rt[un]=pt[q];
66 }
67 inline int inConvex(Pt p){
68     int L,R,M;
69     if(p.X<Lx || p.X>Rx) return 0;
70     L=0;R=dn;
71     while(L<R-1){ M=(L+R)/2;
72         if(p.X<qt[M].X) R=M; else L=M; }
73     if(tri(qt[L],qt[R],p)<0) return 0;
74     L=0;R=un;
75     while(L<R-1){ M=(L+R)/2;
76         if(p.X<rt[M].X) R=M; else L=M; }
77     if(tri(rt[L],rt[R],p)>0) return 0;
78     return 1;
79 }
80 int main(){
81     int n,m,i;
82     Pt p;
83     scanf("%d",&n);
84     for(i=0;i<n;i++) scanf("%LLd%LLd",&pt[i].X,&pt[i].Y);
85     for(i=0;i<m;i++) scanf("%LLd%LLd",&qt[i].X,&qt[i].Y);
86     n=minkowskiSum(n,m);
87     for(i=0;i<n;i++) pt[i]=rt[i];
88     scanf("%d",&m);
89     for(i=0;i<m;i++) scanf("%LLd%LLd",&qt[i].X,&qt[i].Y);
90     n=minkowskiSum(n,m);
91     for(i=0;i<n;i++) pt[i]=rt[i];
92     initInConvex(n);
93     scanf("%d",&m);
94     for(i=0;i<m;i++){
95         scanf("%LLd %LLd",&p.X,&p.Y);
96         p.X*=3; p.Y*=3;
97         puts(inConvex(p)? "YES": "NO");
98     }
99 }

```

10 Number Theory

10.1 Basic

```

1 const int maxc = 5e5;
2 ll pw(ll a, ll n) {
3     ll res = 1;

```

```

4     while (n) {
5         if (n & 1) res = res * a % MOD;
6         a = a * a % MOD;
7         n >>= 1;
8     }
9     return res;
10 }
11
12 vector<ll> fac, ifac;
13 void build_fac() {
14     reset(fac, maxc + 1, 1LL);
15     reset(ifac, maxc + 1, 1LL);
16     for (int x = 2; x <= maxc; x++) {
17         fac[x] = x * fac[x - 1] % MOD;
18         ifac[x] = pw(fac[x], MOD - 2);
19     }
20 }
21
22 ll C(ll n, ll k) {
23     if (n < k) return 0LL;
24     return fac[n] * ifac[n - k] % MOD * ifac[k] % MOD;
25 }

```

10.2 Prime Seive and Defactor

```

1 const int maxc = 1e6 + 1;
2 vector<int> lpf;
3 vector<int> prime;
4
5 void seive() {
6     prime.clear();
7     lpf.resize(maxc, 1);
8     for (int i = 2; i < maxc; i++) {
9         if (lpf[i] == 1) {
10             lpf[i] = i;
11             prime.emplace_back(i);
12         }
13         for (auto& j : prime) {
14             if (i * j >= maxc) break;
15             lpf[i * j] = j;
16             if (j == lpf[i]) break;
17         }
18     }
19     vector<pii> fac;
20     void defactor(int u) {
21         fac.clear();
22         while (u > 1) {
23             int d = lpf[u];
24             fac.emplace_back(make_pair(d, 0));
25             while (u % d == 0) {
26                 u /= d;
27                 fac.back().second++;
28             }
29         }
30     }
31 }

```

10.3 Harmonic Series

```

1 // O(n Log n)
2 for (int i = 1; i <= n; i++) {
3     for (int j = i; j <= n; j += i) {
4         // O(1) code
5     }
6 }
7
8 // PIE
9 // given array a[0], a[1], ..., a[n - 1]
10 // calculate dp[x] = number of pairs (a[i], a[j]) such
11 // that
12 // gcd(a[i], a[j]) = x // (i < j)
13 // idea: let mc(x) = # of y s.t. x|y
14 // f(x) = # of pairs s.t. gcd(a[i], a[j]) >=
15 // x
16 // f(x) = C(mc(x), 2)
17 // dp[x] = f(x) - sum(dp[y], x < y and x|y)
18 const int maxc = 1e6;
19 vector<int> cnt(maxc + 1, 0), dp(maxc + 1, 0);
20 for (int i = 0; i < n; i++)
21     cnt[a[i]]++;
22 for (int x = maxc; x >= 1; x--) {
23     ll cnt_mul = 0; // number of multiples of x

```



```

24 for (int y = x; y <= maxc; y += x)
25     cnt_mul += cnt[y];

```

```

26
27 dp[x] = cnt_mul * (cnt_mul - 1) / 2; // number of
    pairs that are divisible by x
28 for (int y = x + x; y <= maxc; y += x)
29     dp[x] -= dp[y]; // PIE: subtract all dp[y] for
    y > x and x|y
30 }

```

10.4 Count Number of Divisors

```

1 // Count the number of divisors for all x <= 10^6
2 const int maxc = 1e6;
3 vector<int> facs;
4
5 void find_all_divisors() {
6     facs.clear(); facs.resize(maxc + 1, 0);
7     for (int x = 1; x <= maxc; x++) {
8         for (int y = x; y <= maxc; y += x) {
9             facs[y]++;
10        }
11    }
12 }

```

10.5 數論分塊

```

1 /*
2 n = 17
3 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 2 1 1 1 1 1 1 1 1
4         ^       ^
5         L(2)   R(2)
6
7 L(x) := Left bound for n/i = x
8 R(x) := right bound for n/i = x
9
10 ===== FORMULA =====
11 >>> R = n / (n/L) <<<
12 =====
13
14 Example: L(2) = 6
15           R(2) = 17 / (17 / 6)
16                = 17 / 2
17                = 8
18
19 */
20 // ===== CODE =====
21
22 for (ll l = 1, r = 1, q = n; l <= n; l = r + 1) {
23     q = n/l;
24     r = n/q;
25     // Process your code here
26 }
27 // q, L, r: 17 1 1
28 // q, L, r: 8 2 2
29 // q, L, r: 5 3 3
30 // q, L, r: 4 4 4
31 // q, L, r: 3 5 5
32 // q, L, r: 2 6 8
33 // q, L, r: 1 9 17

```

10.6 Pollard's rho

```

1 from itertools import count
2 from math import gcd
3 from sys import stdin
4
5 for s in stdin:
6     number, x = int(s), 2
7     break2 = False
8     for cycle in count(1):
9         y = x
10        if break2:
11            break
12        for i in range(1 << cycle):
13            x = (x * x + 1) % number
14            factor = gcd(x - y, number)
15            if factor > 1:
16                print(factor)

```

```

17 break2 = True
18 break

```

10.7 Miller Rabin

```

1 // n < 4,759,123,141      3 : 2, 7, 61
2 // n < 1,122,004,669,633 4 : 2, 13, 23, 1662803
3 // n < 3,474,749,660,383      6 : pimes <= 13
4 // n < 2^64              7 :
5 // 2, 325, 9375, 28178, 450775, 9780504, 1795265022
6 bool witness(ll a, ll n, ll u, int t){
7     if(!(a%n)) return 0;
8     ll x=mypow(a,u,n);
9     for(int i=0;i<t;i++){
10        ll nx=mul(x,x,n);
11        if(nx==1&&x!=1&&x!=n-1) return 1;
12        x=nx;
13    }
14    return x!=1;
15 }
16 bool miller_rabin(ll n, int s=100) {
17     // iterate s times of witness on n
18     // return 1 if prime, 0 otherwise
19     if(n<2) return 0;
20     if(!(n&1)) return n == 2;
21     ll u=n-1; int t=0;
22     while(!(u&1)) u>>=1, t++;
23     while(s--){
24         ll a=randll()%(n-1)+1;
25         if(witness(a,n,u,t)) return 0;
26     }
27     return 1;
28 }

```

10.8 Fast Power

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

10.9 Extend GCD

```

1 ll GCD;
2 pll extgcd(ll a, ll b) {
3     if (b == 0) {
4         GCD = a;
5         return pll{1, 0};
6     }
7     pll ans = extgcd(b, a % b);
8     return pll{ans.S, ans.F - a/b * ans.S};
9 }
10 pll bezout(ll a, ll b, ll c) {
11     bool negx = (a < 0), negy = (b < 0);
12     pll ans = extgcd(abs(a), abs(b));
13     if (c % GCD != 0) return pll{-LLINF, -LLINF};
14     return pll{ans.F * c/GCD * (negx ? -1 : 1),
15                ans.S * c/GCD * (negy ? -1 : 1)};
16 }
17 ll inv(ll a, ll p) {
18     if (p == 1) return -1;
19     pll ans = bezout(a % p, -p, 1);
20     if (ans == pll{-LLINF, -LLINF}) return -1;
21     return (ans.F % p + p) % p;
22 }

```

10.10 Mu + Phi

```

1 const int maxn = 1e6 + 5;
2 ll f[maxn];
3 vector<int> lpf, prime;
4 void build() {
5     lpf.clear(); lpf.resize(maxn, 1);
6     prime.clear();
7     f[1] = ...; /* mu[1] = 1, phi[1] = 1 */
8     for (int i = 2; i < maxn; i++) {
9         if (lpf[i] == 1) {
10            lpf[i] = i; prime.emplace_back(i);
11            f[i] = ...; /* mu[i] = 1, phi[i] = i-1 */
12        }
13        for (auto& j : prime) {
14            if (i*j >= maxn) break;

```

```

15     lpf[i*j] = j;
16     if (i % j == 0) f[i*j] = ...; /* 0, phi[i]*j
    */
17     else f[i*j] = ...; /* -mu[i], phi[i]*phi[j] */
18     if (j >= lpf[i]) break;
19 } } }

```

10.11 Other Formulas

- Inversion:

$$aa^{-1} \equiv 1 \pmod{m}. a^{-1} \text{ exists iff } \gcd(a, m) = 1.$$

- Linear inversion:

$$a^{-1} \equiv (m - \lfloor \frac{m}{a} \rfloor) \times (m \bmod a)^{-1} \pmod{m}$$

- Fermat's little theorem:

$$a^p \equiv a \pmod{p} \text{ if } p \text{ is prime.}$$

- Euler function:

$$\phi(n) = n \prod_{p|n} \frac{p-1}{p}$$

- Euler theorem:

$$a^{\phi(n)} \equiv 1 \pmod{n} \text{ if } \gcd(a, n) = 1.$$

- 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:

$$\sigma_x(n) = \sum_{d|n} d^x. n = \prod_{i=1}^r p_i^{a_i}.$$

$$\sigma_x(n) = \prod_{i=1}^r \frac{p_i^{(a_i+1)x} - 1}{p_i^x - 1} \text{ if } x \neq 0. \sigma_0(n) = \prod_{i=1}^r (a_i + 1).$$

- 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.

$$x \equiv m_1 p + a_1 \equiv m_2 q + a_2 \pmod{\text{lcm}(m_1, m_2)}$$

- Avoiding Overflow: $ca \bmod cb = c(a \bmod b)$

- Dirichlet Convolution: $(f * g)(n) = \sum_{d|n} f(d)g(n/d)$

- Important Multiplicative Functions + Properties:

$$1. \epsilon(n) = [n = 1]$$

$$2. 1(n) = 1$$

$$3. id(n) = n$$

$$4. \mu(n) = 0 \text{ if } n \text{ has squared prime factor}$$

$$5. \mu(n) = (-1)^k \text{ if } n = p_1 p_2 \cdots p_k$$

$$6. \epsilon = \mu * 1$$

$$7. \phi = \mu * id$$

$$8. [n = 1] = \sum_{d|n} \mu(d)$$

$$9. [gcd = 1] = \sum_{d|gcd} \mu(d)$$

- Möbius inversion: $f = g * 1 \Leftrightarrow g = f * \mu$

10.12 Polynomial

```

1  const int maxk = 20;
2  const int maxn = 1<<maxk;
3  const ll LINF = 1e18;
4
5  /* P = r*2^k + 1
6  P          r    k    g
7  998244353    119  23   3
8  1004535809    479  21   3
9
10 P          r    k    g

```

```

11 3          1    1    2
12 5          1    2    2
13 17         1    4    3
14 97         3    5    5
15 193        3    6    5
16 257        1    8    3
17 7681       15    9   17
18 12289      3   12   11
19 40961      5   13    3
20 65537      1   16    3
21 786433     3   18   10
22 5767169    11  19    3
23 7340033     7   20    3
24 23068673    11  21    3
25 104857601   25  22    3
26 167772161    5   25    3
27 469762049    7   26    3
28 1004535809  479  21    3
29 2013265921  15  27   31
30 2281701377   17  27    3
31 3221225473    3   30    5
32 75161927681  35  31    3
33 77309411329   9   33    7
34 206158430209  3   36   22
35 2061584302081 15  37    7
36 2748779069441  5   39    3
37 6597069766657  3   41    5
38 39582418599937  9   42    5
39 79164837199873  9   43    5
40 263882790666241 15  44    7
41 1231453023109121 35  45    3
42 1337006139375617 19  46    3
43 3799912185593857 27  47    5
44 4222124650659841 15  48   19
45 7881299347898369  7   50    6
46 31525197391593473  7   52    3
47 180143985094819841  5   55    6
48 194555039024054273 27  56    5
49 4179340454199820289 29  57    3
50 9097271247288401921 505 54    6 */
51
52 const int g = 3;
53 const ll MOD = 998244353;
54
55 ll pw(ll a, ll n) { /* fast pow */ }
56
57 #define siz(x) (int)x.size()
58
59 template<typename T>
60 vector<T>& operator+=(vector<T>& a, const vector<T>& b)
61 {
62     if (siz(a) < siz(b)) a.resize(siz(b));
63     for (int i = 0; i < min(siz(a), siz(b)); i++) {
64         a[i] += b[i];
65         a[i] -= a[i] >= MOD ? MOD : 0;
66     }
67     return a;
68 }
69
70 template<typename T>
71 vector<T>& operator-=(vector<T>& a, const vector<T>& b)
72 {
73     if (siz(a) < siz(b)) a.resize(siz(b));
74     for (int i = 0; i < min(siz(a), siz(b)); i++) {
75         a[i] -= b[i];
76         a[i] += a[i] < 0 ? MOD : 0;
77     }
78     return a;
79 }
80
81 template<typename T>
82 vector<T> operator-(const vector<T>& a) {
83     vector<T> ret(siz(a));
84     for (int i = 0; i < siz(a); i++) {
85         ret[i] = -a[i] < 0 ? -a[i] + MOD : -a[i];
86     }
87     return ret;
88 }
89
90 vector<ll> X, iX;
91 vector<int> rev;
92

```

```

91 void init_ntt() {
92     X.clear(); X.resize(maxn, 1); //  $x_1 = g^{((p-1)/n)}$ 
93     iX.clear(); iX.resize(maxn, 1);
94
95     ll u = pw(g, (MOD-1)/maxn);
96     ll iu = pw(u, MOD-2);
97
98     for (int i = 1; i < maxn; i++) {
99         X[i] = X[i-1] * u;
100         iX[i] = iX[i-1] * iu;
101         if (X[i] >= MOD) X[i] %= MOD;
102         if (iX[i] >= MOD) iX[i] %= MOD;
103     }
104
105     rev.clear(); rev.resize(maxn, 0);
106     for (int i = 1, hb = -1; i < maxn; i++) {
107         if (!(i & (i-1))) hb++;
108         rev[i] = rev[i ^ (1<<hb)] | (1<<(maxk-hb-1));
109     }
110
111     template<typename T>
112     void NTT(vector<T>& a, bool inv=false) {
113
114         int _n = (int)a.size();
115         int k = __lg(_n) + ((1<<__lg(_n)) != _n);
116         int n = 1<<k;
117         a.resize(n, 0);
118
119         short shift = maxk-k;
120         for (int i = 0; i < n; i++)
121             if (i > (rev[i]>>shift))
122                 swap(a[i], a[rev[i]>>shift]);
123
124         for (int len = 2, half = 1, div = maxn>>1; len <= n; len<=1, half<=1, div>>=1) {
125             for (int i = 0; i < n; i += len) {
126                 for (int j = 0; j < half; j++) {
127                     T u = a[i+j];
128                     T v = a[i+j+half] * (inv ? iX[j*div] : X[j*div]) % MOD;
129                     a[i+j] = (u+v >= MOD ? u+v-MOD : u+v);
130                     a[i+j+half] = (u-v < 0 ? u-v+MOD : u-v);
131                 }
132             }
133
134             if (inv) {
135                 T dn = pw(n, MOD-2);
136                 for (auto& x : a) {
137                     x *= dn;
138                     if (x >= MOD) x %= MOD;
139                 }
140             }
141
142             template<typename T>
143             inline void resize(vector<T>& a) {
144                 int cnt = (int)a.size();
145                 for (; cnt > 0; cnt--) if (a[cnt-1]) break;
146                 a.resize(max(cnt, 1));
147             }
148
149             template<typename T>
150             vector<T>& operator+=(vector<T>& a, vector<T> b) {
151                 int na = (int)a.size();
152                 int nb = (int)b.size();
153                 a.resize(na + nb - 1, 0);
154                 b.resize(na + nb - 1, 0);
155
156                 NTT(a); NTT(b);
157                 for (int i = 0; i < (int)a.size(); i++) {
158                     a[i] *= b[i];
159                     if (a[i] >= MOD) a[i] %= MOD;
160                 }
161                 NTT(a, true);
162
163                 resize(a);
164                 return a;
165             }
166
167             template<typename T>
168             void inv(vector<T>& ia, int N) {
169                 vector<T> _a(move(ia));
170                 ia.resize(1, pw(_a[0], MOD-2));
171                 vector<T> a(1, -_a[0] + (-_a[0] < 0 ? MOD : 0));
172
173                 for (int n = 1; n < N; n<=1) {
174                     //  $n \rightarrow 2*n$ 
175                     //  $ia' = ia(2-a*ia)$ 
176
177                     for (int i = n; i < min(siz(_a), (n<<1)); i++)
178                         a.emplace_back(-_a[i] + (-_a[i] < 0 ? MOD : 0));
179
180                     vector<T> tmp = ia;
181                     ia *= a;
182                     ia.resize(n<<1);
183                     ia[0] = ia[0] + 2 >= MOD ? ia[0] + 2 - MOD : ia[0] + 2;
184                     ia *= tmp;
185                     ia.resize(n<<1);
186                 }
187                 ia.resize(N);
188
189                 template<typename T>
190                 void mod(vector<T>& a, vector<T>& b) {
191                     int n = (int)a.size()-1, m = (int)b.size()-1;
192                     if (n < m) return;
193
194                     vector<T> ra = a, rb = b;
195                     reverse(ra.begin(), ra.end()); ra.resize(min(n+1, n-m+1));
196                     reverse(rb.begin(), rb.end()); rb.resize(min(m+1, n-m+1));
197
198                     inv(rb, n-m+1);
199
200                     vector<T> q = move(ra);
201                     q *= rb;
202                     q.resize(n-m+1);
203                     reverse(q.begin(), q.end());
204
205                     q *= b;
206                     a -= q;
207                     resize(a);
208                 }
209
210                 /* Kitamasa Method (Fast Linear Recurrence):
211                 Find  $a[K]$  (Given  $a[j] = c[0]a[j-N] + \dots + c[N-1]a[j-1]$ )
212                 Let  $B(x) = x^N - c[N-1]x^{N-1} - \dots - c[1]x^1 - c[0]$ 
213                 Let  $R(x) = x^K \bmod B(x)$  (get  $x^K$  using fast pow and use poly mod to get  $R(x)$ )
214                 Let  $r[i]$  = the coefficient of  $x^i$  in  $R(x)$ 
215                  $\Rightarrow a[K] = a[0]r[0] + a[1]r[1] + \dots + a[N-1]r[N-1]$  */
216
217                 template<typename T>
218                 void gauss(vector<vector<ll>>& v) {
219                     int n;
220                     for (int i = 0; i < n; i++) {
221                         bool ok = false;
222                         for (int j = r; j < n; j++) {
223                             if (v[j][i] == 0) continue;
224                             swap(v[j], v[r]);
225                             ok = true; break;
226                         }
227                         if (!ok) continue;
228                         ll div = inv(v[r][i]);
229                         for (int j = 0; j < n+1; j++) {
230                             v[r][j] *= div;
231                             if (v[r][j] >= MOD) v[r][j] %= MOD;
232                         }
233
234                         for (int j = 0; j < n; j++) {
235                             if (j == r) continue;
236                             ll t = v[j][i];
237                             for (int k = 0; k < n+1; k++) {
238                                 v[j][k] -= v[r][k] * t % MOD;
239                                 if (v[j][k] < 0) v[j][k] += MOD;
240                             }
241                         }
242                     }
243                     r++;
244                 }
245
246                 for (int n = 1; n < N; n<=1) {
247                     //  $n \rightarrow 2*n$ 
248                     //  $ia' = ia(2-a*ia)$ 
249
250                     for (int i = n; i < min(siz(_a), (n<<1)); i++)
251                         a.emplace_back(-_a[i] + (-_a[i] < 0 ? MOD : 0));
252
253                     vector<T> tmp = ia;
254                     ia *= a;
255                     ia.resize(n<<1);
256                     ia[0] = ia[0] + 2 >= MOD ? ia[0] + 2 - MOD : ia[0] + 2;
257                     ia *= tmp;
258                     ia.resize(n<<1);
259                 }
260                 ia.resize(N);
261
262                 template<typename T>
263                 void mod(vector<T>& a, vector<T>& b) {
264                     int n = (int)a.size()-1, m = (int)b.size()-1;
265                     if (n < m) return;
266
267                     vector<T> ra = a, rb = b;
268                     reverse(ra.begin(), ra.end()); ra.resize(min(n+1, n-m+1));
269                     reverse(rb.begin(), rb.end()); rb.resize(min(m+1, n-m+1));
270
271                     inv(rb, n-m+1);
272
273                     vector<T> q = move(ra);
274                     q *= rb;
275                     q.resize(n-m+1);
276                     reverse(q.begin(), q.end());
277
278                     q *= b;
279                     a -= q;
280                     resize(a);
281                 }
282
283                 /* Kitamasa Method (Fast Linear Recurrence):
284                 Find  $a[K]$  (Given  $a[j] = c[0]a[j-N] + \dots + c[N-1]a[j-1]$ )
285                 Let  $B(x) = x^N - c[N-1]x^{N-1} - \dots - c[1]x^1 - c[0]$ 
286                 Let  $R(x) = x^K \bmod B(x)$  (get  $x^K$  using fast pow and use poly mod to get  $R(x)$ )
287                 Let  $r[i]$  = the coefficient of  $x^i$  in  $R(x)$ 
288                  $\Rightarrow a[K] = a[0]r[0] + a[1]r[1] + \dots + a[N-1]r[N-1]$  */
289
290                 template<typename T>
291                 void gauss(vector<vector<ll>>& v) {
292                     int n;
293                     for (int i = 0; i < n; i++) {
294                         bool ok = false;
295                         for (int j = r; j < n; j++) {
296                             if (v[j][i] == 0) continue;
297                             swap(v[j], v[r]);
298                             ok = true; break;
299                         }
300                         if (!ok) continue;
301                         ll div = inv(v[r][i]);
302                         for (int j = 0; j < n+1; j++) {
303                             v[r][j] *= div;
304                             if (v[r][j] >= MOD) v[r][j] %= MOD;
305                         }
306
307                         for (int j = 0; j < n; j++) {
308                             if (j == r) continue;
309                             ll t = v[j][i];
310                             for (int k = 0; k < n+1; k++) {
311                                 v[j][k] -= v[r][k] * t % MOD;
312                                 if (v[j][k] < 0) v[j][k] += MOD;
313                             }
314                         }
315                     }
316                     r++;
317                 }
318
319                 for (int n = 1; n < N; n<=1) {
320                     //  $n \rightarrow 2*n$ 
321                     //  $ia' = ia(2-a*ia)$ 
322
323                     for (int i = n; i < min(siz(_a), (n<<1)); i++)
324                         a.emplace_back(-_a[i] + (-_a[i] < 0 ? MOD : 0));
325
326                     vector<T> tmp = ia;
327                     ia *= a;
328                     ia.resize(n<<1);
329                     ia[0] = ia[0] + 2 >= MOD ? ia[0] + 2 - MOD : ia[0] + 2;
330                     ia *= tmp;
331                     ia.resize(n<<1);
332                 }
333                 ia.resize(N);
334
335                 template<typename T>
336                 void mod(vector<T>& a, vector<T>& b) {
337                     int n = (int)a.size()-1, m = (int)b.size()-1;
338                     if (n < m) return;
339
340                     vector<T> ra = a, rb = b;
341                     reverse(ra.begin(), ra.end()); ra.resize(min(n+1, n-m+1));
342                     reverse(rb.begin(), rb.end()); rb.resize(min(m+1, n-m+1));
343
344                     inv(rb, n-m+1);
345
346                     vector<T> q = move(ra);
347                     q *= rb;
348                     q.resize(n-m+1);
349                     reverse(q.begin(), q.end());
350
351                     q *= b;
352                     a -= q;
353                     resize(a);
354                 }
355
356                 /* Kitamasa Method (Fast Linear Recurrence):
357                 Find  $a[K]$  (Given  $a[j] = c[0]a[j-N] + \dots + c[N-1]a[j-1]$ )
358                 Let  $B(x) = x^N - c[N-1]x^{N-1} - \dots - c[1]x^1 - c[0]$ 
359                 Let  $R(x) = x^K \bmod B(x)$  (get  $x^K$  using fast pow and use poly mod to get  $R(x)$ )
360                 Let  $r[i]$  = the coefficient of  $x^i$  in  $R(x)$ 
361                  $\Rightarrow a[K] = a[0]r[0] + a[1]r[1] + \dots + a[N-1]r[N-1]$  */
362
363                 template<typename T>
364                 void gauss(vector<vector<ll>>& v) {
365                     int n;
366                     for (int i = 0; i < n; i++) {
367                         bool ok = false;
368                         for (int j = r; j < n; j++) {
369                             if (v[j][i] == 0) continue;
370                             swap(v[j], v[r]);
371                             ok = true; break;
372                         }
373                         if (!ok) continue;
374                         ll div = inv(v[r][i]);
375                         for (int j = 0; j < n+1; j++) {
376                             v[r][j] *= div;
377                             if (v[r][j] >= MOD) v[r][j] %= MOD;
378                         }
379
380                         for (int j = 0; j < n; j++) {
381                             if (j == r) continue;
382                             ll t = v[j][i];
383                             for (int k = 0; k < n+1; k++) {
384                                 v[j][k] -= v[r][k] * t % MOD;
385                                 if (v[j][k] < 0) v[j][k] += MOD;
386                             }
387                         }
388                     }
389                     r++;
390                 }
391
392                 for (int n = 1; n < N; n<=1) {
393                     //  $n \rightarrow 2*n$ 
394                     //  $ia' = ia(2-a*ia)$ 
395
396                     for (int i = n; i < min(siz(_a), (n<<1)); i++)
397                         a.emplace_back(-_a[i] + (-_a[i] < 0 ? MOD : 0));
398
399                     vector<T> tmp = ia;
400                     ia *= a;
401                     ia.resize(n<<1);
402                     ia[0] = ia[0] + 2 >= MOD ? ia[0] + 2 - MOD : ia[0] + 2;
403                     ia *= tmp;
404                     ia.resize(n<<1);
405                 }
406                 ia.resize(N);
407
408                 template<typename T>
409                 void mod(vector<T>& a, vector<T>& b) {
410                     int n = (int)a.size()-1, m = (int)b.size()-1;
411                     if (n < m) return;
412
413                     vector<T> ra = a, rb = b;
414                     reverse(ra.begin(), ra.end()); ra.resize(min(n+1, n-m+1));
415                     reverse(rb.begin(), rb.end()); rb.resize(min(m+1, n-m+1));
416
417                     inv(rb, n-m+1);
418
419                     vector<T> q = move(ra);
420                     q *= rb;
421                     q.resize(n-m+1);
422                     reverse(q.begin(), q.end());
423
424                     q *= b;
425                     a -= q;
426                     resize(a);
427                 }
428
429                 /* Kitamasa Method (Fast Linear Recurrence):
430                 Find  $a[K]$  (Given  $a[j] = c[0]a[j-N] + \dots + c[N-1]a[j-1]$ )
431                 Let  $B(x) = x^N - c[N-1]x^{N-1} - \dots - c[1]x^1 - c[0]$ 
432                 Let  $R(x) = x^K \bmod B(x)$  (get  $x^K$  using fast pow and use poly mod to get  $R(x)$ )
433                 Let  $r[i]$  = the coefficient of  $x^i$  in  $R(x)$ 
434                  $\Rightarrow a[K] = a[0]r[0] + a[1]r[1] + \dots + a[N-1]r[N-1]$  */
435
436                 template<typename T>
437                 void gauss(vector<vector<ll>>& v) {
438                     int n;
439                     for (int i = 0; i < n; i++) {
440                         bool ok = false;
441                         for (int j = r; j < n; j++) {
442                             if (v[j][i] == 0) continue;
443                             swap(v[j], v[r]);
444                             ok = true; break;
445                         }
446                         if (!ok) continue;
447                         ll div = inv(v[r][i]);
448                         for (int j = 0; j < n+1; j++) {
449                             v[r][j] *= div;
450                             if (v[r][j] >= MOD) v[r][j] %= MOD;
451                         }
452
453                         for (int j = 0; j < n; j++) {
454                             if (j == r) continue;
455                             ll t = v[j][i];
456                             for (int k = 0; k < n+1; k++) {
457                                 v[j][k] -= v[r][k] * t % MOD;
458                                 if (v[j][k] < 0) v[j][k] += MOD;
459                             }
460                         }
461                     }
462                     r++;
463                 }
464
465                 for (int n = 1; n < N; n<=1) {
466                     //  $n \rightarrow 2*n$ 
467                     //  $ia' = ia(2-a*ia)$ 
468
469                     for (int i = n; i < min(siz(_a), (n<<1)); i++)
470                         a.emplace_back(-_a[i] + (-_a[i] < 0 ? MOD : 0));
471
472                     vector<T> tmp = ia;
473                     ia *= a;
474                     ia.resize(n<<1);
475                     ia[0] = ia[0] + 2 >= MOD ? ia[0] + 2 - MOD : ia[0] + 2;
476                     ia *= tmp;
477                     ia.resize(n<<1);
478                 }
479                 ia.resize(N);
480
481                 template<typename T>
482                 void mod(vector<T>& a, vector<T>& b) {
483                     int n = (int)a.size()-1, m = (int)b.size()-1;
484                     if (n < m) return;
485
486                     vector<T> ra = a, rb = b;
487                     reverse(ra.begin(), ra.end()); ra.resize(min(n+1, n-m+1));
488                     reverse(rb.begin(), rb.end()); rb.resize(min(m+1, n-m+1));
489
490                     inv(rb, n-m+1);
491
492                     vector<T> q = move(ra);
493                     q *= rb;
494                     q.resize(n-m+1);
495                     reverse(q.begin(), q.end());
496
497                     q *= b;
498                     a -= q;
499                     resize(a);
500                 }
501
502                 /* Kitamasa Method (Fast Linear Recurrence):
503                 Find  $a[K]$  (Given  $a[j] = c[0]a[j-N] + \dots + c[N-1]a[j-1]$ )
504                 Let  $B(x) = x^N - c[N-1]x^{N-1} - \dots - c[1]x^1 - c[0]$ 
505                 Let  $R(x) = x^K \bmod B(x)$  (get  $x^K$  using fast pow and use poly mod to get  $R(x)$ )
506                 Let  $r[i]$  = the coefficient of  $x^i$  in  $R(x)$ 
507                  $\Rightarrow a[K] = a[0]r[0] + a[1]r[1] + \dots + a[N-1]r[N-1]$  */
508
509                 template<typename T>
510                 void gauss(vector<vector<ll>>& v) {
511                     int n;
512                     for (int i = 0; i < n; i++) {
513                         bool ok = false;
514                         for (int j = r; j < n; j++) {
515                             if (v[j][i] == 0) continue;
516                             swap(v[j], v[r]);
517                             ok = true; break;
518                         }
519                         if (!ok) continue;
520                         ll div = inv(v[r][i]);
521                         for (int j = 0; j < n+1; j++) {
522                             v[r][j] *= div;
523                             if (v[r][j] >= MOD) v[r][j] %= MOD;
524                         }
525
526                         for (int j = 0; j < n; j++) {
527                             if (j == r) continue;
528                             ll t = v[j][i];
529                             for (int k = 0; k < n+1; k++) {
530                                 v[j][k] -= v[r][k] * t % MOD;
531                                 if (v[j][k] < 0) v[j][k] += MOD;
532                             }
533                         }
534                     }
535                     r++;
536                 }
537
538                 for (int n = 1; n < N; n<=1) {
539                     //  $n \rightarrow 2*n$ 
540                     //  $ia' = ia(2-a*ia)$ 
541
542                     for (int i = n; i < min(siz(_a), (n<<1)); i++)
543                         a.emplace_back(-_a[i] + (-_a[i] < 0 ? MOD : 0));
544
545                     vector<T> tmp = ia;
546                     ia *= a;
547                     ia.resize(n<<1);
548                     ia[0] = ia[0] + 2 >= MOD ? ia[0] + 2 - MOD : ia[0] + 2;
549                     ia *= tmp;
550                     ia.resize(n<<1);
551                 }
552                 ia.resize(N);
553
554                 template<typename T>
555                 void mod(vector<T>& a, vector<T>& b) {
556                     int n = (int)a.size()-1, m = (int)b.size()-1;
557                     if (n < m) return;
558
559                     vector<T> ra = a, rb = b;
560                     reverse(ra.begin(), ra.end()); ra.resize(min(n+1, n-m+1));
561                     reverse(rb.begin(), rb.end()); rb.resize(min(m+1, n-m+1));
562
563                     inv(rb, n-m+1);
564
565                     vector<T> q = move(ra);
566                     q *= rb;
567                     q.resize(n-m+1);
568                     reverse(q.begin(), q.end());
569
570                     q *= b;
571                     a -= q;
572                     resize(a);
573                 }
574
575                 /* Kitamasa Method (Fast Linear Recurrence):
576                 Find  $a[K]$  (Given  $a[j] = c[0]a[j-N] + \dots + c[N-1]a[j-1]$ )
577                 Let  $B(x) = x^N - c[N-1]x^{N-1} - \dots - c[1]x^1 - c[0]$ 
578                 Let  $R(x) = x^K \bmod B(x)$  (get  $x^K$  using fast pow and use poly mod to get  $R(x)$ )
579                 Let  $r[i]$  = the coefficient of  $x^i$  in  $R(x)$ 
580                  $\Rightarrow a[K] = a[0]r[0] + a[1]r[1] + \dots + a[N-1]r[N-1]$  */
581
582                 template<typename T>
583                 void gauss(vector<vector<ll>>& v) {
584                     int n;
585                     for (int i = 0; i < n; i++) {
586                         bool ok = false;
587                         for (int j = r; j < n; j++) {
588                             if (v[j][i] == 0) continue;
589                             swap(v[j], v[r]);
590                             ok = true; break;
591                         }
592                         if (!ok) continue;
593                         ll div = inv(v[r][i]);
594                         for (int j = 0; j < n+1; j++) {
595                             v[r][j] *= div;
596                             if (v[r][j] >= MOD) v[r][j] %= MOD;
597                         }
598
599                         for (int j = 0; j < n; j++) {
600                             if (j == r) continue;
601                             ll t = v[j][i];
602                             for (int k = 0; k < n+1; k++) {
603                                 v[j][k] -= v[r][k] * t % MOD;
604                                 if (v[j][k] < 0) v[j][k] += MOD;
605                             }
606                         }
607                     }
608                     r++;
609                 }
610
611                 for (int n = 1; n < N; n<=1) {
612                     //  $n \rightarrow 2*n$ 
613                     //  $ia' = ia(2-a*ia)$ 
614
615                     for (int i = n; i < min(siz(_a), (n<<1)); i++)
616                         a.emplace_back(-_a[i] + (-_a[i] < 0 ? MOD : 0));
617
618                     vector<T> tmp = ia;
619                     ia *= a;
620                     ia.resize(n<<1);
621                     ia[0] = ia[0] + 2 >= MOD ? ia[0] + 2 - MOD : ia[0] + 2;
622                     ia *= tmp;
623                     ia.resize(n<<1);
624                 }
625                 ia.resize(N);
626
627                 template<typename T>
628                 void mod(vector<T>& a, vector<T>& b) {
629                     int n = (int)a.size()-1, m = (int)b.size()-1;
630                     if (n < m) return;
631
632                     vector<T> ra = a, rb = b;
633                     reverse(ra.begin(), ra.end()); ra.resize(min(n+1, n-m+1));
634                     reverse(rb.begin(), rb.end()); rb.resize(min(m+1, n-m+1));
635
636                     inv(rb, n-m+1);
637
638                     vector<T> q = move(ra);
639                     q *= rb;
640                     q.resize(n-m+1);
641                     reverse(q.begin(), q.end());
642
643                     q *= b;
644                     a -= q;
645                     resize(a);
646                 }
647
648                 /* Kitamasa Method (Fast Linear Recurrence):
649                 Find  $a[K]$  (Given  $a[j] = c[0]a[j-N] + \dots + c[N-1]a[j-1]$ )
650                 Let  $B(x) = x^N - c[N-1]x^{N-1} - \dots - c[1]x^1 - c[0]$ 
651                 Let  $R(x) = x^K \bmod B(x)$  (get  $x^K$  using fast pow and use poly mod to get  $R(x)$ )
652                 Let  $r[i]$  = the coefficient of  $x^i$  in  $R(x)$ 
653                  $\Rightarrow a[K] = a[0]r[0] + a[1]r[1] + \dots + a[N-1]r[N-1]$  */
654
655                 template<typename T>
656                 void gauss(vector<vector<ll>>& v) {
657                     int n;
658                     for (int i = 0; i < n; i++) {
659                         bool ok = false;
660                         for (int j = r; j < n; j++) {
661                             if (v[j][i] == 0) continue;
662                             swap(v[j], v[r]);
663                             ok = true; break;
664                         }
665                         if (!ok) continue;
666                         ll div = inv(v[r][i]);
667                         for (int j = 0; j < n+1; j++) {
668                             v[r][j] *= div;
669                             if (v[r][j] >= MOD) v[r][j] %= MOD;
670                         }
671
672                         for (int j = 0; j < n; j++) {
673                             if (j == r) continue;
674                             ll t = v[j][i];
675                             for (int k = 0; k < n+1; k++) {
676                                 v[j][k] -= v[r][k] * t % MOD;
677                                 if (v[j][k] < 0) v[j][k] += MOD;
678                             }
679                         }
680                     }
681                     r++;
682                 }
683
684                 for (int n = 1; n < N; n<=1) {
685                     //  $n \rightarrow 2*n$ 
686                     //  $ia' = ia(2-a*ia)$ 
687
688                     for (int i = n; i < min(siz(_a), (n<<1)); i++)
689                         a.emplace_back(-_a[i] + (-_a[i] < 0 ? MOD : 0));
690
691                     vector<T> tmp = ia;
692                     ia *= a;
693                     ia.resize(n<<1);
694                     ia[0] = ia[0] + 2 >= MOD ? ia[0] + 2 - MOD : ia[0] + 2;
695                     ia *= tmp;
696                     ia.resize(n<<1);
697                 }
698                 ia.resize(N);
699
700                 template<typename T>
701                 void mod(vector<T>& a, vector<T>& b) {
702                     int n = (int)a.size()-1, m = (int)b.size()-1;
703                     if (n < m) return;
704
705                     vector<T> ra = a, rb = b;
706                     reverse(ra.begin(), ra.end()); ra.resize(min(n+1, n-m+1));
707                     reverse(rb.begin(), rb.end()); rb.resize(min(m+1, n-m+1));
708
709                     inv(rb, n-m+1);
710
711                     vector<T> q = move(ra);
712                     q *= rb;
713                     q.resize(n-m+1);
714                     reverse(q.begin(), q.end());
715
716                     q *= b;
717                     a -= q;
718                     resize(a);
719                 }
720
721                 /* Kitamasa Method (Fast Linear Recurrence):
722                 Find  $a[K]$  (Given  $a[j] = c[0]a[j-N] + \dots + c[N-1]a[j-1]$ )
723                 Let  $B(x) = x^N - c[N-1]x^{N-1} - \dots - c[1]x^1 - c[0]$ 
724                 Let  $R(x) = x^K \bmod B(x)$  (get  $x^K$  using fast pow and use poly mod to get  $R(x)$ )
725                 Let  $r[i]$  = the coefficient of  $x^i$  in  $R(x)$ 
726                  $\Rightarrow a[K] = a[0]r[0] + a[1]r[1] + \dots + a[N-1]r[N-1]$  */
727
728                 template<typename T>
729                 void gauss(vector<vector<ll>>& v) {
730                     int n;
731                     for (int i = 0; i < n; i++) {
732                         bool ok = false;
733                         for (int j = r; j < n; j++) {
734                             if (v[j][i] == 0) continue;
735                             swap(v[j], v[r]);
736                             ok = true; break;
737                         }
738                         if (!ok) continue;
739                         ll div = inv(v[r][i]);
740                         for (int j = 0; j < n+1; j++) {
741                             v[r][j] *= div;
742                             if (v[r][j] >= MOD) v[r][j] %= MOD;
743                         }
744
745                         for (int j = 0; j < n; j++) {
746                             if (j == r) continue;
747                             ll t = v[j][i];
748                             for (int k = 0; k < n+1; k++) {
749                                 v[j][k] -= v[r][k] * t % MOD;
750                                 if (v[j][k] < 0) v[j][k] += MOD;
751                             }
752                         }
753                     }
754                     r++;
755                 }
756
757                 for (int n = 1; n < N; n<=1) {
758                     //  $n \rightarrow 2*n$ 
759                     //  $ia' = ia(2-a*ia)$ 
760
761                     for (int i = n; i < min(siz(_a), (n<<1)); i++)
762                         a.emplace_back(-_a[i] + (-_a[i] < 0 ? MOD : 0));
763
764                     vector<T> tmp = ia;
765                     ia *= a;
766                     ia.resize(n<<1);
767                     ia[0] = ia[0] + 2 >= MOD ? ia[0] + 2 - MOD : ia[0] + 2;
768                     ia *= tmp;
769                     ia.resize(n<<1);
770                 }
771                 ia.resize(N);
772
773                 template<typename T>
774                 void mod(vector<T>& a, vector<T>& b) {
775                     int n = (int)a.size()-1, m = (int)b.size()-1;
776                     if (n < m) return;
777
778                     vector<T> ra = a, rb = b;
779                     reverse(ra.begin(), ra.end()); ra.resize(min(n+1, n-m+1));
780                     reverse(rb.begin(), rb.end()); rb.resize(min(m+1, n-m+1));
781
782                     inv(rb, n-m+1);
783
784                     vector<T> q = move(ra);
785                     q *= rb;
786                     q.resize(n-m+1);
787                     reverse(q.begin(), q.end());
788
789                     q *= b;
790                     a -= q;
791                     resize(a);
792                 }
793
794                 /* Kitamasa Method (Fast Linear Recurrence):
795                 Find  $a[K]$  (Given  $a[j] = c[0]a[j-N] + \dots + c[N-1]a[j-1]$ )
796                 Let  $B(x) = x^N - c[N-1]x^{N-1} - \dots - c[1]x^1 - c[0]$ 
797                 Let  $R(x) = x^K \bmod B(x)$  (get  $x^K$  using fast pow and use poly mod to get  $R(x)$ )
798                 Let  $r[i]$  = the coefficient of  $x^i$  in  $R(x)$ 
799                  $\Rightarrow a[K] = a[0]r[0] + a[1]r[1] + \dots + a[N-1]r[N-1]$  */
800
801                 template<typename T>
802                 void gauss(vector<vector<ll>>& v) {
803                     int n;
804                     for (int i = 0; i < n; i++) {
805                         bool ok = false;
806                         for (int j = r; j < n; j++) {
807                             if (v[j][i] == 0) continue;
808                             swap(v[j], v[r]);
809                             ok = true; break;
810                         }
811                         if (!ok) continue;
812                         ll div = inv(v[r][i]);
813                         for (int j = 0; j < n+1; j++) {
814                             v[r][j] *= div;
815                             if (v[r][j] >= MOD) v[r][j] %= MOD;
816                         }
817
818                         for (int j = 0; j < n; j++) {
819                             if (j == r) continue;
820                             ll t = v[j][i];
821                             for (int k = 0; k < n+1; k++) {
822                                 v[j][k] -= v[r][k] * t % MOD;
823                                 if (v[j][k] < 0) v[j][k] += MOD;
824                             }
825                         }
826                     }
827                     r++;
828                 }
829
830                 for (int n = 1; n < N; n<=1) {
831                     //  $n \rightarrow 2*n$ 
832                     //  $ia' = ia(2-a*ia)$ 
833
834                     for (int i = n; i < min(siz(_a), (n<<1)); i++)
835                         a.emplace_back(-_a[i] + (-_a[i] < 0 ? MOD : 0));
836
837                     vector<T> tmp = ia;
838                     ia *= a;
839                     ia.resize(n<<1);
840                     ia[0] = ia[0] + 2 >= MOD ? ia[0] + 2 - MOD : ia[0] + 2;
841                     ia *= tmp;
842                     ia.resize(n<<1);
843                 }
844                 ia.resize(N);
845
846                 template<typename T>
847                 void mod(vector<T>& a, vector<T>& b) {
848                     int n = (int)a.size()-1, m = (int)b.size()-1;
849                     if (n < m) return;
850
851                     vector<T> ra = a, rb = b;
852                     reverse(ra.begin(), ra.end()); ra.resize(min(n+1, n-m+1));
853                     reverse(rb.begin(), rb.end()); rb.resize(min(m+1, n-m+1));
854
855                     inv(rb, n-m+1);
856
857                     vector<T> q = move(ra);
858                     q *= rb;
859                     q.resize(n-m+1);
860                     reverse(q.begin(), q.end());
861
862                     q *= b;
863                     a -= q;
864                     resize(a);
865                 }
866
867                 /* Kitamasa Method (Fast Linear Recurrence):
868                 Find  $a[K]$  (Given  $a[j] = c[0]a[j-N] + \dots + c[N-1]a[j-1]$ )
869                 Let  $B(x) = x^N - c[N-1]x^{N-1} - \dots - c[1]x^1 - c[0]$ 
870                 Let  $R(x) = x^K \bmod B(x)$  (get  $x^K$  using fast pow and use poly mod to get  $R(x)$ )
871                 Let  $r[i]$  = the coefficient of  $x^i$ 
```

```
25 | } }
```

11.2 Determinant

1. Use GJ Elimination, if there's any row consists of only 0, then $\det = 0$, otherwise $\det = \text{product of diagonal elements}$.

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 : Unchanged

12 Flow / Matching

12.1 Dinic

```
1 struct Dinic {
2     struct Edge {
3         int t, c, r;
4         Edge() {}
5         Edge(int _t, int _c, int _r):
6             t(_t), c(_c), r(_r) {}
7     };
8     vector<vector<Edge>> G;
9     vector<int> dis, iter;
10    int s, t;
11    void init(int n) {
12        G.resize(n), dis.resize(n), iter.resize(n);
13        for(int i = 0; i < n; ++i)
14            G[i].clear();
15    }
16    void add(int a, int b, int c) {
17        G[a].eb(b, c, G[b].size());
18        G[b].eb(a, 0, G[a].size() - 1);
19    }
20    bool bfs() {
21        fill(ALL(dis), -1);
22        dis[s] = 0;
23        queue<int> que;
24        que.push(s);
25        while(!que.empty()) {
26            int u = que.front(); que.pop();
27            for(auto& e : G[u]) {
28                if(e.c > 0 && dis[e.t] == -1) {
29                    dis[e.t] = dis[u] + 1;
30                    que.push(e.t);
31                }
32            }
33        }
34        return dis[t] != -1;
35    }
36    int dfs(int u, int cur) {
37        if(u == t) return cur;
38        for(int &i = iter[u]; i < (int)G[u].size(); ++i) {
39            auto& e = G[u][i];
40            if(e.c > 0 && dis[u] + 1 == dis[e.t]) {
41                int ans = dfs(e.t, min(cur, e.c));
42                if(ans > 0) {
43                    G[e.t][e.r].c += ans;
44                    e.c -= ans;
45                    return ans;
46                }
47            }
48        }
49        return 0;
50    }
51
52    int flow(int a, int b) {
53        s = a, t = b;
54        int ans = 0;
55        while(bfs()) {
56            fill(ALL(iter), 0);
57            int tmp;
```

```
58        while((tmp = dfs(s, INF)) > 0)
59            ans += tmp;
60    }
61    return ans;
62 }
63 };

12.2 ISAP

1 #define SZ(c) ((int)(c).size())
2 struct Maxflow{
3     static const int MAXV=50010;
4     static const int INF =1000000;
5     struct Edge{
6         int v,c,r;
7         Edge(int _v,int _c,int _r):v(_v),c(_c),r(_r){}
8     };
9     int s,t; vector<Edge> G[MAXV];
10    int iter[MAXV],d[MAXV],gap[MAXV],tot;
11    void init(int n,int _s,int _t){
12        tot=n,s=_s,t=_t;
13        for(int i=0;i<tot;i++){
14            G[i].clear(); iter[i]=d[i]=gap[i]=0;
15        }
16    }
17    void addEdge(int u,int v,int c){
18        G[u].push_back(Edge(v,c,SZ(G[v])));
19        G[v].push_back(Edge(u,0,SZ(G[u])-1));
20    }
21    int DFS(int p,int flow){
22        if(p==t) return flow;
23        for(int &i=iter[p];i<SZ(G[p]);i++){
24            Edge &e=G[p][i];
25            if(e.c>0&&d[p]==d[e.v]+1){
26                int f=DFS(e.v,min(flow,e.c));
27                if(f){ e.c-=f; G[e.v][e.r].c+=f; return f; }
28            }
29        }
30        if(--gap[d[p]]==0) d[s]=tot;
31        else{ d[p]++; iter[p]=0; ++gap[d[p]]; }
32        return 0;
33    }
34    int flow(){
35        int res=0;
36        for(res=0,gap[0]=tot;d[s]<tot;res+=DFS(s,INF));
37        return res;
38    } // reset: set iter,d,gap to 0
39 } flow;
```

12.3 MCMF

```
1 struct MCMF {
2     struct Edge {
3         int to, cap, rev;
4         ll cost;
5         Edge() {}
6         Edge(int _to, int _cap, int _rev, ll _cost) :
7             to(_to), cap(_cap), rev(_rev), cost(_cost)
8         {}
9     };
10    static const int N = 2000;
11    vector<Edge> G[N];
12    int n, s, t;
13    void init(int _n, int _s, int _t) {
14        n = _n, s = _s, t = _t;
15        for(int i = 0; i <= n; ++i)
16            G[i].clear();
17    }
18    void add_edge(int from, int to, int cap, ll cost) {
19        G[from].eb(to, cap, (int)G[to].size(), cost);
20        G[to].eb(from, 0, (int)G[from].size() - 1, -cost);
21    }
22
23    bool vis[N];
24    int iter[N];
25    ll dis[N];
26    bool SPFA() {
27        for(int i = 0; i <= n; ++i)
28            vis[i] = 0, dis[i] = LINF;
```

```

28     dis[s] = 0; vis[s] = 1;
29     queue<int> que; que.push(s);
30     while(!que.empty()) {
31         int u = que.front(); que.pop();
32         vis[u] = 0;
33         for(auto& e : G[u]) if(e.cap > 0 && dis[e.
34             to] > dis[u] + e.cost) {
35             dis[e.to] = dis[u] + e.cost;
36             if(!vis[e.to]) {
37                 que.push(e.to);
38                 vis[e.to] = 1;
39             }
40         }
41     }
42     return dis[t] != LINF;
43 }
44
45 int dfs(int u, int cur) {
46     if(u == t) return cur;
47     int ret = 0; vis[u] = 1;
48     for(int &i = iter[u]; i < (int)G[u].size(); ++i
49         ) {
50         auto &e = G[u][i];
51         if(e.cap > 0 && dis[e.to] == dis[u] + e.
52             cost && !vis[e.to]) {
53             int tmp = dfs(e.to, min(cur, e.cap));
54             e.cap -= tmp;
55             G[e.to][e.rev].cap += tmp;
56             cur -= tmp;
57             ret += tmp;
58             if(cur == 0) {
59                 vis[u] = 0;
60                 return ret;
61             }
62         }
63         vis[u] = 0;
64         return ret;
65     }
66     pair<int, ll> flow() {
67         int flow = 0; ll cost = 0;
68         while(SPFA()) {
69             memset(iter, 0, sizeof(iter));
70             int tmp = dfs(s, INF);
71             flow += tmp, cost += tmp * dis[t];
72         }
73         return {flow, cost};
74     }
75 };

```

12.4 Hopcroft-Karp

```

1 struct HopcroftKarp {
2     // id: X = [1, nx], Y = [nx+1, nx+ny]
3     int n, nx, ny, m, MXCNT;
4     vector<vector<int>> g;
5     vector<int> mx, my, dis, vis;
6     void init(int nnx, int nny, int mm) {
7         nx = nnx, ny = nny, m = mm;
8         n = nx + ny + 1;
9         g.clear(); g.resize(n);
10    }
11    void add(int x, int y) {
12        g[x].emplace_back(y);
13        g[y].emplace_back(x);
14    }
15    bool dfs(int x) {
16        vis[x] = true;
17        Each(y, g[x]) {
18            int px = my[y];
19            if (px == -1 ||
20                (dis[px] == dis[x]+1 &&
21                 !vis[px] && dfs(px))) {
22                mx[x] = y;
23                my[y] = x;
24                return true;
25            }
26        }
27        return false;
28    }

```

```

29 void get() {
30     mx.clear(); mx.resize(n, -1);
31     my.clear(); my.resize(n, -1);
32
33     while (true) {
34         queue<int> q;
35         dis.clear(); dis.resize(n, -1);
36         for (int x = 1; x <= nx; x++){
37             if (mx[x] == -1) {
38                 dis[x] = 0;
39                 q.push(x);
40             }
41         }
42         while (!q.empty()) {
43             int x = q.front(); q.pop();
44             Each(y, g[x]) {
45                 if (my[y] != -1 && dis[my[y]] ==
46                     -1) {
47                     dis[my[y]] = dis[x] + 1;
48                     q.push(my[y]);
49                 }
50             }
51         }
52         bool brk = true;
53         vis.clear(); vis.resize(n, 0);
54         for (int x = 1; x <= nx; x++)
55             if (mx[x] == -1 && dfs(x))
56                 brk = false;
57         if (brk) break;
58     }
59     MXCNT = 0;
60     for (int x = 1; x <= nx; x++) if (mx[x] != -1)
61         MXCNT++;
62 }
63 } hk;

```

12.5 Cover / Independent Set

```

1 V(E) Cover: choose some V(E) to cover all E(V)
2 V(E) Independ: set of V(E) not adj to each other
3
4 M = Max Matching
5 Cv = Min V Cover
6 Ce = Min E Cover
7 Iv = Max V Ind
8 Ie = Max E Ind (equiv to M)
9
10 M = Cv (Konig Theorem)
11 Iv = V \ Cv
12 Ce = V - M
13
14 Construct Cv:
15 1. Run Dinic
16 2. Find s-t min cut
17 3. Cv = {X in T} + {Y in S}

```

12.6 KM

```

1 #include <bits/stdc++.h>
2
3 using namespace std;
4 const int inf = 1e9;
5
6 struct KuhnMunkres {
7     int n;
8     vector<vector<int>> g;
9     vector<int> lx, ly, slack;
10    vector<int> match, visx, visy;
11    KuhnMunkres(int n) : n(n), g(n, vector<int>(n)),
12        lx(n), ly(n), slack(n), match(n), visx(n), visy
13        (n)) {}
14    vector<int> & operator[](int i) { return g[i]; }
15    bool dfs(int i, bool aug) { // aug = true 表示要更
16        新 match
17        if(visx[i]) return false;
18        visx[i] = true;
19        for(int j = 0; j < n; j++) {
20            if(visy[j]) continue;

```

```

19 // 一邊擴增交錯樹、尋找增廣路徑
20 // 一邊更新slack：樹上的點跟樹外的點所造成
    的最小權重
21 int d = lx[i] + ly[j] - g[i][j];
22 if(d == 0) {
23     visy[j] = true;
24     if(match[j] == -1 || dfs(match[j], aug)
        ) {
25         if(aug)
26             match[j] = i;
27         return true;
28     }
29 } else {
30     slack[j] = min(slack[j], d);
31 }
32 }
33 return false;
34 }
35 bool augment() { // 回傳是否有增廣路
36     for(int j = 0; j < n; j++) if(!visy[j] && slack
        [j] == 0) {
37         visy[j] = true;
38         if(match[j] == -1 || dfs(match[j], false))
        {
39             return true;
40         }
41     }
42     return false;
43 }
44 void relabel() {
45     int delta = inf;
46     for(int j = 0; j < n; j++) if(!visy[j]) delta =
        min(delta, slack[j]);
47     for(int i = 0; i < n; i++) if(visx[i]) lx[i] -=
        delta;
48     for(int j = 0; j < n; j++) {
49         if(visy[j]) ly[j] += delta;
50         else slack[j] -= delta;
51     }
52 }
53 int solve() {
54     for(int i = 0; i < n; i++) {
55         lx[i] = 0;
56         for(int j = 0; j < n; j++) lx[i] = max(lx[i]
            , g[i][j]);
57     }
58     fill(ly.begin(), ly.end(), 0);
59     fill(match.begin(), match.end(), -1);
60     for(int i = 0; i < n; i++) {
61         // slack 在每一輪都要初始化
62         fill(slack.begin(), slack.end(), inf);
63         fill(visx.begin(), visx.end(), false);
64         fill(visy.begin(), visy.end(), false);
65         if(dfs(i, true)) continue;
66         // 重複調整頂標直到找到增廣路徑
67         while(!augment()) relabel();
68         fill(visx.begin(), visx.end(), false);
69         fill(visy.begin(), visy.end(), false);
70         dfs(i, true);
71     }
72     int ans = 0;
73     for(int j = 0; j < n; j++) if(match[j] != -1)
        ans += g[match[j]][j];
74     return ans;
75 }
76 };
77 signed main() {
78     ios_base::sync_with_stdio(0), cin.tie(0);
79     int n;
80     while(cin >> n && n) {
81         KuhnMunkres KM(n);
82         for(int i = 0; i < n; i++) {
83             for(int j = 0; j < n; j++) {
84                 int c;
85                 cin >> c;
86                 if(c > 0)
87                     KM[i][j] = c;
88             }
89         }
90         cout << KM.solve() << '\n';
91     }

```

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 $\leq n \approx n/((\ln n) - 1)$

| |
|---|
| $\pi(100) = 25, \pi(200) = 46$ |
| $\pi(500) = 95, \pi(1000) = 168$ |
| $\pi(2000) = 303, \pi(4000) = 550$ |
| $\pi(10^4) = 1229, \pi(10^5) = 9592$ |
| $\pi(10^6) = 78498, \pi(10^7) = 664579$ |