

Contents

1	Reminder	1
1.1	Observations and Tricks	1
1.2	Bug List	1
2	Init	1
2.1	vimrc	1
2.2	template.cpp	1
2.3	run.sh	1
2.4	init.sh	1
3	Basic	2
3.1	template (optional)	2
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	3
5.3	Skew Heap	4
5.4	Leftist Heap	4
5.5	Persistent Treap	4
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	6
7.4	BCC - Bridge	7
7.5	SCC - Tarjan	7
7.6	Eulerian Path - Undir	8
7.7	Eulerian Path - Dir	8
7.8	Hamilton Path	8
7.9	Kth Shortest Path	9
7.10	System of Difference Constraints	9
8	String	9
8.1	Rolling Hash	9
8.2	Trie	10
8.3	KMP	10
8.4	Z Value	10
8.5	Manacher	10
8.6	Suffix Array	10
8.7	SA-IS	11
8.8	Minimum Rotation	11
8.9	Aho Corasick	11
9	Geometry	11
9.1	Basic Operations	11
9.2	InPoly	12
9.3	Sort by Angle	12
9.4	Line Intersect Check	12
9.5	Line Intersection	12
9.6	Convex Hull	12
9.7	Lower Concave Hull	12
9.8	Polygon Area	12
9.9	Pick's Theorem	12
9.10	Minimum Enclosing Circle	12
9.11	PolyUnion	13
9.12	Minkowski Sum	13
10	Number Theory	14
10.1	Basic	14
10.2	Prime Seive and Defactor	14
10.3	Harmonic Series	14
10.4	Count Number of Divisors	14
10.5	數論分塊	14
10.6	Pollard's rho	15
10.7	Miller Rabin	15
10.8	Fast Power	15
10.9	Extend GCD	15
10.10	$\mu + \Phi$	15
10.11	Other Formulas	15
10.12	Polynomial	16
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	18
12.4	Hopcroft-Karp	18
12.5	Cover / Independent Set	19
12.6	KM	19

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 Reminder

1.1 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

1.2 Bug List

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

2 Init

開場流程：

1. 打好 ./vimrc
2. 新資料夾 contest，打好 template.cpp、run.sh，
chmod +x run.sh
3. 打好 init.sh，chmod +x init.sh
4. ./init.sh
5. vim */*.cpp

2.1 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 <C-k> 4<C-y>4k
no <C-j> 4<C-e>4j

```

```

17 no ; :
18
19 inoremap { {}<Left>
20 inoremap {<CR> {}<CR>}<Esc>kO
21
22 :command! Run :!clear && %:~>run.sh %:~>CR>
23 :command! Stdio :!clear && %:~>run_stdio.sh %:~>p -
    stdio<CR>

```

2.2 template.cpp

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 #define endl '\n'
5 #define iter(v) v.begin(), v.end()
6
7 typedef long long ll;
8 typedef pair<int, int> pii;
9 typedef pair<ll, ll> pll;
10
11 /* ===== */
12 // debug(), output()
13 #define GRAY "\x1b[90m"
14 #define COLOREND "\x1b[0m"
15
16 void _debug() {}
17 template<typename A, typename... B> void _debug(A a, B...
    b) { cerr << a << ' ', _debug(b...); }
18 #define debug(...) cerr<<GRAY<<#__VA_ARGS__<<": "<<
    COLOREND, _debug(__VA_ARGS__), cerr<<endl
19
20 const int INF = 1.05e9;
21 const ll LINF = 4e18;
22
23 /* ===== */
24
25 void init(int &TEST) {
26     // 如果有 t 筆測資的話，取消底下的註解
27     // cin >> TEST;
28 }
29
30 void input() {}
31 void solve(int kase) {}
32
33 /* ===== */
34 int main() {
35     ios_base::sync_with_stdio(false); cin.tie(0);
36     int TEST = 1; init(TEST);
37     for (int kase = 1; kase <= TEST; kase++) {
38         input();
39         solve(kase);
40     }
41     return 0;
42 }

```

2.3 run.sh

```

1 #!/bin/bash
2
3 g++ -std=c++17 -O2 -g -fsanitize=undefined,address "$1"
    && echo DONE COMPILE || exit 1
4
5 if [[ $2 == "-stdio" ]]; then
6     echo ===== RUN =====
7     ./a.out
8 else
9     for input in testcases/*.in; do
10         id=${input%.in}
11         echo ===== INPUT =====
12         cat $input
13         echo ===== STDERR =====
14         ./a.out <$input >$id.ot
15         echo ===== OUTPUT =====
16         cat $id.ot
17     done
18 fi

```

2.4 init.sh

```

1 #!/bin/bash
2
3 for c in {A..P}; do
4     mkdir -p $c/testcases
5     if [ ! -s $c/$c.cpp ]; then
6         cp template.cpp $c/$c.cpp
7         cp run.sh $c/run.sh
8     fi
9 done

```

3 Basic

3.1 template (optional)

```

1 typedef pair<int, ll> pil;
2 typedef pair<ll, int> pli;
3
4 /* ===== */
5 // STL and I/O
6 // pair
7 template<typename T1, typename T2>
8 ostream& operator<<(ostream& os, pair<T1, T2> p) {
9     return os << "(" << p.first << ", " << p.second <<
    ")";
10 }
11 template<typename T1, typename T2>
12 istream& operator>>(istream& is, pair<T1, T2>& p) {
13     return is >> p.first >> p.second;
14 }
15 // vector
16 template<typename T>
17 istream& operator>>(istream& is, vector<T>& v) {
18     for (auto& x : v) is >> x;
19     return is;
20 }
21 template<typename T>
22 ostream& operator<<(ostream& os, const vector<T>& v) {
23     for (const auto& x : v) os << x << ' ';
24     return os;
25 }
26 /* ===== */
27 // debug(), output()
28 #define RED "\x1b[31m"
29 #define GREEN "\x1b[32m"
30 #define YELLOW "\x1b[33m"
31
32 void _output() {}
33 template<typename A, typename... B> void _output(A a, B
    ... b) { cout << a << ' ', _output(b...); }
34 #define output(...) _output(__VA_ARGS__), cout<<endl
35 /* ===== */
36 // BASIC ALGORITHM
37 string binary(ll x, int b = -1) {
38     if (b == -1) b = __lg(x) + 1;
39     string s = "";
40     for (int k = b - 1; k >= 0; k--) {
41         s.push_back((x & (1LL<<k)) ? '1' : '0');
42     }
43     return s;
44 }
45 /* ===== */
46 // CONSTANT
47 const int MOD = 1e9 + 7;
48 //const int MOD = 998244353;
49 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

```

3.3 PBDS

```

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

```

3.4 Random

```

1 mt19937 gen(chrono::steady_clock::now().
2     time_since_epoch().count());
3 #define RANDINT(a, b) uniform_int_distribution<int> (a,
4     b)(rng) // inclusive
5 #define RANDLL(a, b) uniform_int_distribution<long long>
6     >(a, b)(rng) // inclusive
7 #define RANDFLOAT(a, b) uniform_real_distribution<float>
8     >(a, b)(rng) // exclusive
9 #define RANDDOUBLE(a, b) uniform_real_distribution<
10     double>(a, b)(rng) // exclusive
11 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 };

```

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(qr<=l||r<=ql) return 0;
19     if(ql<=l&&r<=qr) return arr[i];
20     if(qr<=m) return query(ql, qr, i<<1, l, m);
21     if(m<=ql) return query(ql, qr, i<<1|1, m, r);
22     return max(query(ql, qr, i<<1, l, m), query(ql, qr, i
        <<1|1, m, r));
23 }
24 #undef m
25 inline void solve(){
26     int n, q; cin>>n>>q;
27     V<int> v(n);
28     for(auto& i:v)
29         cin>>i;
30     V<V<int>> e(n);
31     for(int i=1; i<n; i++){
32         int a, b; cin>>a>>b, a--, b--;
33         e[a].emplace_back(b);
34         e[b].emplace_back(a);
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     }

```

```

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     if (r - l == 1) {
10         if (x(l) > arr[i](l))
11             arr[i] = x;
12         return;
13     }
14     line a = max(arr[i], x), b = min(arr[i], x);
15     if (a(m) > b(m))
16         arr[i] = a, insert(b, i << 1, l, m);
17     else
18         arr[i] = b, insert(a, i << 1 | 1, m, r);
19 }
20 ld query(int x, int i = 1, int l = 0, int r = maxn) {
21     if (x < l || r <= x) return -numeric_limits<ld>::max();
22     if (r - l == 1) return arr[i](x);
23     return max({arr[i](x), query(x, i << 1, l, m), query(
24         x, i << 1 | 1, m, r)});
25 }
26 #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] +=
14         sz[a];
15     return true;
16 };
17 inline void undo() {
18     auto [a, b, s] = his.back(); his.pop_back();
19     dsu[a] = a, sz[b] = s;
20 }
21 #define m ((l + r) >> 1)
22 void insert(int ql, int qr, P<int> x, int i = 1, int l
23     = 0, int r = q) {
24     // debug(ql, qr, x); return;
25     if (qr <= l || r <= ql) return;
26     if (ql <= l && r <= qr) {arr[i].push_back(x);
27         return;}
28     if (qr <= m)
29         insert(ql, qr, x, i << 1, l, m);
30     else if (ql <= m)
31         insert(ql, qr, x, i << 1 | 1, m, r);
32     else {
33         insert(ql, qr, x, i << 1, l, m);
34         insert(ql, qr, x, i << 1 | 1, m, r);
35     }
36 }
37 void traversal(V<int>& ans, int i = 1, int l = 0, int r
38     = q) {
39     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     }
22 }

```

```

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
   after removing u
6
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 = u;
26             else centroid.first = u;
27         }
28     }
29 }

```

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         }
45     }
46
47 // Bellman-Ford
48 queue<int> q;
49 vector<int> pa;
50 void BellmanFord(vector<int>& src) {
51     dis.assign(n+1, LINF);
52     negCycle.assign(n+1, false);
53     pa.assign(n+1, -1);
54
55     for (auto& s : src) dis[s] = 0;
56
57     for (int rlx = 1; rlx <= n; rlx++) {
58         for (int u = 1; u <= n; u++) {
59             if (dis[u] == LINF) continue; // Important
60             !!
61             for (auto& e : g[u]) {
62                 int v = e.first; ll w = e.second;
63                 if (dis[v] > dis[u] + w) {
64                     dis[v] = dis[u] + w;
65                     pa[v] = u;
66                     if (rlx == n) negCycle[v] = true;
67                 }
68             }
69         }
70     }
71
72 // Negative Cycle Detection
73 void NegCycleDetect() {
74     /* No Neg Cycle: NO
75     Exist Any Neg Cycle:
76     YES
77     v0 v1 v2 ... vk v0 */
78
79     vector<int> src;
80     for (int i = 1; i <= n; i++)
81         src.emplace_back(i);
82
83     SPFA(src);
84     // BellmanFord(src);
85
86     int ptr = -1;
87     for (int i = 1; i <= n; i++) if (negCycle[i])
88         { ptr = i; break; }
89 }

```



```

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
109void 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            }
127        }
128    }
129}

```

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

```

```

45
46     DP(n-1, (1<<n)-1);
47     cout << dp[n-1][(1<<n)-1] << endl;
48
49     return 0;
50 }

```

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     };
9     struct heap{ nd* edge; int dep; heap* chd[4]; };
10     static int cmp(heap* a,heap* b)
11     { return a->edge->d > b->edge->d; }
12     struct node{
13         int v; ll d; heap* H; nd* E;
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             ;
64         fill(nullNd->chd,nullNd->chd+4,nullNd);
65         while(not dfsQ.empty()){
66             int u=dfsQ.front(); dfsQ.pop();
67             if(!nxt[u]) head[u]=nullNd;
68             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         }
111     }
112     void solve(){ // ans[i] stores the i-th shortest path
113         dijkstra(); build();
114         first_K(); // ans.size() might less than k
115     }
116 } 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 }

```

- $x_u - x_v \leq c \Rightarrow \text{add}(v, u, c)$
- $x_u - x_v \geq c \Rightarrow \text{add}(u, v, -c)$
- $x_u - x_v = c \Rightarrow \text{add}(v, u, c), \text{add}(u, v, -c)$
- $x_u \geq c \Rightarrow \text{add super vertex } x_0 = 0, \text{ then } x_u - x_0 \geq c \Rightarrow \text{add}(u, 0, -c)$
- 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] :
23                0);
24            res = (res % mod + mod) % mod;
25            return res; }
26    };

```

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            ++;
14        if (i+z[i]-1 > r) l = i, r = i+z[i]-1;
15        if (z[i] == (int)it.size()) ans++;
16    }
17    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            );
12        while (0 <= i-m[i]-1 && i+m[i]+1 < 2*n+1 &&
13            s[i-m[i]-1] == s[i+m[i]+1]) m[i]++;
14        if (i+m[i] > mx+mxk) mx = i, mxk = m[i];
15    } }
16 void init() { cin >> S; n = (int)S.size(); }
17 void solve() {
18     manacher();
19     int mx = 0, ptr = 0;
20     for (int i = 0; i < 2*n+1; i++) if (mx < m[i])
21         { mx = m[i]; ptr = i; }
22     for (int i = ptr-mx; i <= ptr+mx; i++)
23         if (s[i] != '.') cout << s[i];
24     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.F.S)]++;
16            for (int i = 0; i < n; i++)
17                pos[i] = (!i ? 0 : pos[i-1] + cnt[i-1]);
18            for (auto& i : buc[t])
19                buc[t][pos[(t ? i.F.F : i.F.S)]] = i;
20        }
21    }
22    bool fill_suf() {
23        bool end = true;
24        for (int i = 0; i < n; i++) suf[i] = buc[0][i].S;
25        rk[suf[0]] = 0;
26        for (int i = 1; i < n; i++) {
27            int dif = (buc[0][i].F != buc[0][i-1].F);
28            end &= dif;
29            rk[suf[i]] = rk[suf[i-1]] + dif;
30        }
31        return end;
32    }
33    void sa() {

```

```

34        for (int i = 0; i < n; i++)
35            buc[0][i] = make_pair(make_pair(s[i], s[i]), i);
36        sort(buc[0].begin(), buc[0].end());
37        if (fill_suf()) return;
38        for (int k = 0; (1<<k) < n; k++) {
39            for (int i = 0; i < n; i++)
40                buc[0][i] = make_pair(make_pair(rk[i],
41                    rk[(i + (1<<k)) % n]), i);
42            radix_sort();
43            if (fill_suf()) return;
44        }
45        void LCP() { int k = 0;
46            for (int i = 0; i < n-1; i++) {
47                if (rk[i] == 0) continue;
48                int pi = rk[i];
49                int j = suf[pi-1];
50                while (i+k < n && j+k < n && s[i+k] == s[j+k]) k++;
51                lcp[pi] = k;
52                k = max(k-1, 0);
53            }
54        }
55    };
56    SuffixArray suffixarray;

```

8.7 SA-IS

```

1 const int N=300010;
2 struct SA{
3     #define REP(i,n) for(int i=0;i<(n);i++)
4     #define REP1(i,a,b) for(int i=(a);i<=(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, nmzx=-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[sa[i]-1]]+=sa[i]-1;\
30        memcpy(x,c,sizeof(int)*z);\
31        for(int i=n-1;i>=0;i--) if(sa[i]&&t[sa[i]-1]) sa[--x[sa[i]-1]]+=sa[i]-1;\
32        MS0(c,z); REP(i,n) uniq&=++c[s[i]]<2;
33        REP(i,z-1) c[i+1]+=c[i];
34        if(uniq) { REP(i,n) sa[--c[s[i]]]=i; return; }
35        for(int i=n-2;i>=0;i--){
36            t[i]=(s[i]==s[i+1]?t[i+1]:s[i]<s[i+1]);
37            MAGIC(REP1(i,1,n-1) if(t[i]&&t[i-1]) sa[--x[s[i]]]=p[q[i]=nn++]=i);
38            REP(i,n) if(sa[i]&&t[sa[i]]&&t[sa[i]-1]){
39                neq=lst<0||memcmp(s+sa[i],s+lst,(p[q[sa[i]]+1]-sa[i])*sizeof(int));
40                ns[q[lst=sa[i]]]=nmzx+=neq;
41            }
42            sais(ns,nsa,p+nn,q+n,t+n,c+z,nn,nmzx+1);
43            MAGIC(for(int i=nn-1;i>=0;i--) sa[--x[s[p[nsa[i]]]]]=p[nsa[i]]);
44        }
45    }sa;
46    int H[N], SA[N], RA[N];
47    void suffix_array(int* ip, int len){
48        // should padding a zero in the back
49        // ip is int array, len is array length

```

```

49 // ip[0..n-1] != 0, and ip[len]=0
50 ip[len++]=0; sa.build(ip,len,128);
51 memcpy(H,sa.hei+1,len<<2); memcpy(SA,sa._sa+1,len<<2);
52 ;
53 for(int i=0;i<len;i++) RA[i]=sa.r[i]-1;
54 // resulting height, sa array \in [0,len)

```

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 Pt mv(Pt a, Pt b) { return b-a; }
27 T len2(Pt a) { return a*a; }
28 T dis2(Pt a, Pt b) { return len2(b-a); }
29
30 short ori(Pt a, Pt b) { return ((a^b)>0) - ((a^b)<0); }
31 bool onseg(Pt p, Pt l1, Pt l2) {
32     Pt a = mv(p, l1), b = mv(p, l2);
33     return ((a^b) == 0) && ((a*b) <= 0);
34 }

```

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 =
22            erase(y));
23        while ((y = x) != begin() && (--x)->p >= y->p)
24            isect(x, erase(y));
25    }
26    ll query(ll x) {
27        assert(!empty());
28        auto l = *lower_bound(x);
29        return l.m * x + l.b;
30    }
31 };

```

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().
16         time_since_epoch().count());
17     shuffle(ALL(E), gen);
18     center = E[0], r2 = 0;
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     } }

```

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[i][ii])>0&&j<i){
31                             c[r++]=make_pair(segP(py[j][jj],py[i][ii],py[i][ii+1]),1);
32                             c[r++]=make_pair(segP(py[j][jj+1],py[i][ii+1],py[i][ii]),-1);
33                         }
34                     }else if(ta>0 && tb<0){
35                         tc=tri(py[j][jj],py[j][jj+1],py[i][ii]);
36                         td=tri(py[j][jj],py[j][jj+1],py[i][ii+1]);
37                         c[r++]=make_pair(tc/(tc+td),1);
38                     }else if(ta<0 && tb>0){
39                         tc=tri(py[j][jj],py[j][jj+1],py[i][ii]);
40                         td=tri(py[j][jj],py[j][jj+1],py[i][ii+1]);
41                         c[r++]=make_pair(tc/(tc+td),-1);
42                     }
43                 }
44             }
45             sort(c,c+r);
46             z=min(max(c[0].first,0.0),1.0); d=c[0].second; s
47                 =0;
48             for(j=1;j<r;j++){
49                 w=min(max(c[j].first,0.0),1.0);

```



```

45     if(!d) s+=w-z;
46     d+=c[j].second; z=w;
47 }
48 sum+=(py[i][ii]^py[i][ii+1])*s;
49 }
50 }
51 return sum/2;
52 }

```

9.12 Minkowski Sum

```

1  /* convex hull Minkowski Sum */
2  #define INF 100000000000000LL
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;

```

```

70 while(L<R-1){ M=(L+R)/2;
71     if(p.X<qt[M].X) R=M; else L=M; }
72     if(tri(qt[L],qt[R],p)<0) return 0;
73     L=0;R=un;
74     while(L<R-1){ M=(L+R)/2;
75         if(p.X<rt[M].X) R=M; else L=M; }
76         if(tri(rt[L],rt[R],p)>0) return 0;
77         return 1;
78     }
79 int main(){
80     int n,m,i;
81     Pt p;
82     scanf("%d",&n);
83     for(i=0;i<n;i++) scanf("%Lld%Lld",&pt[i].X,&pt[i].Y);
84     scanf("%d",&m);
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 vector<pii> fac;
19 void defactor(int u) {
20     fac.clear();
21     while (u > 1) {
22         int d = lpf[u];
23         fac.emplace_back(make_pair(d, 0));
24         while (u % d == 0) {
25             u /= d;
26             fac.back().second++;
27     } } }

```

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
    that
11 // gcd(a[i], a[j]) = x // (i < j)
12 //
13 // idea: let mc(x) = # of y s.t. x|y
14 // f(x) = # of pairs s.t. gcd(a[i], a[j]) >=
    x
15 // f(x) = C(mc(x), 2)
16 // dp[x] = f(x) - sum(dp[y], x < y and x|y)
17 const int maxc = 1e6;
18 vector<int> cnt(maxc + 1, 0), dp(maxc + 1, 0);
19 for (int i = 0; i < n; i++)
20     cnt[a[i]]++;
21
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
4  n/i: 17  8  5  4  3  2  2  2  1  1  1  1  1  1  1  1  1
5           ^      ^
6         L(2)  R(2)
7
8  L(x) := Left bound for n/i = x
9  R(x) := right bound for n/i = x
10
11  ===== FORMULA =====
12  >>> R = n / (n/L) <<<
13  =====

```

```

14
15 Example: L(2) = 6
16           R(2) = 17 / (17 / 6)
17              = 17 / 2
18              = 8
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         }
20     }
21 }

```

10.11 Other Formulas

- Inversion:
 $aa^{-1} \equiv 1 \pmod{m}$. a^{-1} exists iff $\gcd(a, m) = 1$.
- Linear inversion:
 $a^{-1} \equiv (m - \lfloor \frac{m}{a} \rfloor) \times (m \bmod a)^{-1} \pmod{m}$
- Fermat's little theorem:
 $a^p \equiv a \pmod{p}$ if p is prime.
- Euler function:
 $\phi(n) = n \prod_{p|n} \frac{p-1}{p}$
- Euler theorem:
 $a^{\phi(n)} \equiv 1 \pmod{n}$ if $\gcd(a, n) = 1$.
- Extended Euclidean algorithm:
 $ax + by = \gcd(a, b) = \gcd(b, a \bmod b) = \gcd(b, a - \lfloor \frac{a}{b} \rfloor b)$
 $\lfloor \frac{a}{b} \rfloor b = bx_1 + (a - \lfloor \frac{a}{b} \rfloor b)y_1 = ay_1 + b(x_1 - \lfloor \frac{a}{b} \rfloor y_1)$
- Divisor function:
 $\sigma_x(n) = \sum_{d|n} d^x$. $n = \prod_{i=1}^r p_i^{a_i}$.
 $\sigma_x(n) = \prod_{i=1}^r \frac{p_i^{(a_i+1)x} - 1}{p_i^x - 1}$ if $x \neq 0$. $\sigma_0(n) = \prod_{i=1}^r (a_i + 1)$.
- Chinese remainder theorem (Coprime Moduli):
 $x \equiv a_i \pmod{m_i}$.
 $M = \prod m_i$. $M_i = M/m_i$. $t_i = M_i^{-1}$.
 $x = kM + \sum a_i t_i M_i, k \in \mathbb{Z}$.

- Chinese remainder theorem:

$$x \equiv a_1 \pmod{m_1}, x \equiv a_2 \pmod{m_2} \Rightarrow x = m_1 p + a_1 = m_2 q + a_2 \Rightarrow m_1 p - m_2 q = a_2 - a_1$$

Solve for (p, q) using ExtGCD.

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

- 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. $\mu(n) = 0$ if n has squared prime factor
5. $\mu(n) = (-1)^k$ if $n = p_1 p_2 \cdots p_k$
6. $\epsilon = \mu * 1$
7. $\phi = \mu * id$
8. $[n = 1] = \sum_{d|n} \mu(d)$
9. $[gcd = 1] = \sum_{d|gcd} \mu(d)$

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

10.12 Polynomial

```

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 1945555039024054273 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
93 void init_ntt() {
94     X.clear(); X.resize(maxn, 1); // x1 = g^((p-1)/n)
95     iX.clear(); iX.resize(maxn, 1);
96
97     ll u = pw(g, (MOD-1)/maxn);
98     ll iu = pw(u, MOD-2);
99
100     for (int i = 1; i < maxn; i++) {
101         X[i] = X[i-1] * u;
102         iX[i] = iX[i-1] * iu;
103         if (X[i] >= MOD) X[i] %= MOD;
104         if (iX[i] >= MOD) iX[i] %= MOD;
105     }
106
107     rev.clear(); rev.resize(maxn, 0);
108     for (int i = 1, hb = -1; i < maxn; i++) {
109         if (!(i & (i-1))) hb++;
110         rev[i] = rev[i ^ (1<<hb)] | (1<<(maxk-hb-1));
111     }
112 }
113
114 template<typename T>
115 void NTT(vector<T>& a, bool inv=false) {
116     int _n = (int)a.size();
117     int k = __lg(_n) + ((1<<__lg(_n)) != _n);
118     int n = 1<<k;
119     a.resize(n, 0);
120
121     short shift = maxk-k;
122     for (int i = 0; i < n; i++)
123         if (i > (rev[i]>>shift))
124             swap(a[i], a[rev[i]>>shift]);
125
126     for (int len = 2, half = 1, div = maxn>>1; len <= n; len<<=1, half<<=1, div>>=1) {
127         for (int i = 0; i < n; i += len) {
128             for (int j = 0; j < half; j++) {
129                 T u = a[i+j];
130                 T v = a[i+j+half] * (inv ? iX[j*div] : X[j*div]) % MOD;
131                 a[i+j] = (u+v >= MOD ? u+v-MOD : u+v);
132                 a[i+j+half] = (u-v < 0 ? u-v+MOD : u-v);
133             }
134         }
135     }
136
137     if (inv) {
138         T dn = pw(n, MOD-2);
139         for (auto& x : a) {
140             x *= dn;
141             if (x >= MOD) x %= MOD;
142         }
143     }
144 }
145
146 template<typename T>
147 inline void resize(vector<T>& a) {
148     int cnt = (int)a.size();
149     for (; cnt > 0; cnt--) if (a[cnt-1]) break;
150     a.resize(max(cnt, 1));
151 }
152
153 template<typename T>
154 vector<T>& operator*=(vector<T>& a, vector<T> b) {
155     int na = (int)a.size();
156     int nb = (int)b.size();
157     a.resize(na + nb - 1, 0);
158     b.resize(na + nb - 1, 0);
159
160     NTT(a); NTT(b);
161     for (int i = 0; i < (int)a.size(); i++) {
162         a[i] *= b[i];
163         if (a[i] >= MOD) a[i] %= MOD;
164     }
165     NTT(a, true);
166     resize(a);
167     return a;
168 }
169
170 template<typename T>
171 void inv(vector<T>& ia, int N) {
172     vector<T> _a(move(ia));
173     ia.resize(1, pw(_a[0], MOD-2));
174     vector<T> a(1, -_a[0] + (-_a[0] < 0 ? MOD : 0));
175
176     for (int n = 1; n < N; n<<=1) {
177         // n -> 2*n
178         // ia' = ia(2-a*ia);
179
180         for (int i = n; i < min(siz(_a), (n<<1)); i++)
181             a.emplace_back(-_a[i] + (-_a[i] < 0 ? MOD : 0));
182
183         vector<T> tmp = ia;
184         ia *= a;
185         ia.resize(n<<1);
186         ia[0] = ia[0] + 2 >= MOD ? ia[0] + 2 - MOD : ia[0] + 2;
187         ia *= tmp;
188         ia.resize(n<<1);
189     }
190     ia.resize(N);
191 }
192
193 template<typename T>
194 void mod(vector<T>& a, vector<T>& b) {
195     int n = (int)a.size()-1, m = (int)b.size()-1;
196     if (n < m) return;
197
198     vector<T> ra = a, rb = b;
199     reverse(ra.begin(), ra.end()); ra.resize(min(n+1, n-m+1));
200     reverse(rb.begin(), rb.end()); rb.resize(min(m+1, n-m+1));
201
202     inv(rb, n-m+1);
203
204     vector<T> q = move(ra);
205     q *= rb;
206     q.resize(n-m+1);
207     reverse(q.begin(), q.end());
208
209     q *= b;
210     a -= q;
211     resize(a);

```

```

207 }
208
209 /* Kitamasa Method (Fast Linear Recurrence):
210 Find a[K] (Given a[j] = c[0]a[j-N] + ... + c[N-1]a[j
    -1])
211 Let B(x) = x^N - c[N-1]x^(N-1) - ... - c[1]x^1 - c[0]
212 Let R(x) = x^K mod B(x) (get x^K using fast pow and
    use poly mod to get R(x))
213 Let r[i] = the coefficient of x^i in R(x)
214 => a[K] = a[0]r[0] + a[1]r[1] + ... + a[N-1]r[N-1] */

```

11 Linear Algebra

11.1 Gaussian-Jordan Elimination

```

1 int n; vector<vector<ll>> > v;
2 void gauss(vector<vector<ll>>& v) {
3     int r = 0;
4     for (int i = 0; i < n; i++) {
5         bool ok = false;
6         for (int j = r; j < n; j++) {
7             if (v[j][i] == 0) continue;
8             swap(v[j], v[r]);
9             ok = true; break;
10        }
11        if (!ok) continue;
12        ll div = inv(v[r][i]);
13        for (int j = 0; j < n+1; j++) {
14            v[r][j] *= div;
15            if (v[r][j] >= MOD) v[r][j] %= MOD;
16        }
17        for (int j = 0; j < n; j++) {
18            if (j == r) continue;
19            ll t = v[j][i];
20            for (int k = 0; k < n+1; k++) {
21                v[j][k] -= v[r][k] * t % MOD;
22                if (v[j][k] < 0) v[j][k] += MOD;
23            }
24            r++;
25        }
26    }
27 }

```

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) {

```

```

G[a].eb(b, c, G[b].size());
G[b].eb(a, 0, G[a].size() - 1);
}
bool bfs() {
    fill(ALL(dis), -1);
    dis[s] = 0;
    queue<int> que;
    que.push(s);
    while (!que.empty()) {
        int u = que.front(); que.pop();
        for (auto& e : G[u]) {
            if (e.c > 0 && dis[e.t] == -1) {
                dis[e.t] = dis[u] + 1;
                que.push(e.t);
            }
        }
    }
    return dis[t] != -1;
}
int dfs(int u, int cur) {
    if (u == t) return cur;
    for (int &i = iter[u]; i < (int)G[u].size(); ++i) {
        auto& e = G[u][i];
        if (e.c > 0 && dis[u] + 1 == dis[e.t]) {
            int ans = dfs(e.t, min(cur, e.c));
            if (ans > 0) {
                G[e.t][e.r].c += ans;
                e.c -= ans;
                return ans;
            }
        }
    }
    return 0;
}
int flow(int a, int b) {
    s = a, t = b;
    int ans = 0;
    while (bfs()) {
        fill(ALL(iter), 0);
        int tmp;
        while ((tmp = dfs(s, INF)) > 0)
            ans += tmp;
    }
    return ans;
}
};

```

12.2 ISAP

```

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;

```

```

65 pair<int, ll> flow() {
66     int flow = 0; ll cost = 0;
67     while(SPFA()) {
68         memset(iter, 0, sizeof(iter));
69         int tmp = dfs(s, INF);
70         flow += tmp, cost += tmp * dis[t];
71     }
72     return {flow, cost};
73 }
74 };

```

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    bool vis[N];
23    int iter[N];
24    ll dis[N];
25    bool SPFA() {
26        for(int i = 0; i <= n; ++i)
27            vis[i] = 0, dis[i] = LINF;
28
29        dis[s] = 0; vis[s] = 1;
30        queue<int> que; que.push(s);
31        while(!que.empty()) {
32            int u = que.front(); que.pop();
33            vis[u] = 0;
34            for(auto& e : G[u]) if(e.cap > 0 && dis[e.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.cost && !vis[e.to]) {
52                    int tmp = dfs(e.to, min(cur, e.cap));
53                    e.cap -= tmp;
54                    G[e.to][e.rev].cap += tmp;
55                    cur -= tmp;
56                    ret += tmp;
57                    if(cur == 0) {
58                        vis[u] = 0;
59                        return ret;
60                    }
61                }
62            }
63        vis[u] = 0;
64        return ret;
65    }
66 }

```

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]] == -1) {
46                        dis[my[y]] = dis[x] + 1;
47                        q.push(my[y]);
48                    }
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
58            if (brk) break;
59        }
60        MXCNT = 0;
61        for (int x = 1; x <= nx; x++) if (mx[x] != -1)
62            MXCNT++;
63    } hk;
64 }

```

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(n) {}
13    vector<int> & operator[](int i) { return g[i]; }
14    bool dfs(int i, bool aug) { // aug = true 表示要更新 match
15        // 新 match
16        if(visx[i]) return false;
17        visx[i] = true;
18        for(int j = 0; j < n; j++) {
19            if(visy[j]) continue;
20            // 一邊擴增交錯樹、尋找增廣路徑
21            // 一邊更新 slack: 樹上的點跟樹外的點所造成的最小權重
22            int d = lx[i] + ly[j] - g[i][j];
23            if(d == 0) {
24                visy[j] = true;
25                if(match[j] == -1 || dfs(match[j], aug)) {
26                    if(aug) 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) KM[i][j] = c;
87             }
88         }
89         cout << KM.solve() << '\n';
90     }
91 }
92

```

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

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