

Contents

1 Init (Linux)

1.1 vimrc	1
1.2 template.cpp	1
1.3 run.sh	1

2 Reminder

2.1 Observations and Tricks	1
2.2 Bug List	1

3 Basic

3.1 template (optional)	1
3.2 Stress	2
3.3 PBDS	2
3.4 Random	2

4 Python

4.1 I/O	2
4.2 Decimal	2

5 Data Structure

5.1 Segment Tree	2
5.2 Heavy Light Decomposition (Benson)	3
5.3 Heavy Light Decomposition (Ian)	3
5.4 Heavy Light Decomposition (Gino)	4
5.5 Skew Heap	5
5.6 Leftist Heap	6
5.7 Persistent Treap	7
5.8 Li Chao Tree	8
5.9 Time Segment Tree	9

6 DP

6.1 Aliens	6
6.2 SOS DP	6

7 Graph

7.1 Tree Centroid	6
7.2 Bellman-Ford + SPFA	6
7.3 BCC - AP	7
7.4 BCC - Bridge	8
7.5 SCC - Tarjan	8
7.6 Eulerian Path - Undir	9
7.7 Eulerian Path - Dir	9
7.8 Hamilton Path	10
7.9 Kth Shortest Path	10
7.10 System of Difference Constraints	11

8 String

8.1 Rolling Hash	11
8.2 Trie	11
8.3 KMP	11
8.4 Z Value	11
8.5 Manacher	11
8.6 Suffix Array	11
8.7 SA-IS	12
8.8 Minimum Rotation	12
8.9 Aho Corasick	12

9 Geometry

9.1 Basic Operations	12
9.2 InPoly	12
9.3 Sort by Angle	13
9.4 Line Intersection Check	13
9.5 Line Intersection	13
9.6 Convex Hull	13
9.7 Lower Concave Hull	13
9.8 Polygon Area	13
9.9 Pick's Theorem	14
9.10 Minimum Enclosing Circle	14
9.11 PolyUnion	14
9.12 Minkowski Sum	14

10 Number Theory

10.1 Basic	15
10.2 Prime Seive and Defactor	15
10.3 Harmonic Series	15
10.4 Count Number of Divisors	15
10.5 數論分塊	16
10.6 Pollard's rho	16
10.7 Miller Rabin	16
10.8 Fast Power	16
10.9 Extend GCD	16
10.10 Mu + Phi	16
10.11 Other Formulas	17
10.12 Polynomial	17

11 Linear Algebra

11.1 Gaussian-Jordan Elimination	18
11.2 Determinant	18

12 Flow / Matching

12.1 Dinic	19
12.2 ISAP	19
12.3 MCMF	19
12.4 Hopcroft-Karp	20
12.5 Cover / Independent Set	20
12.6 KM	21

13 Combinatorics

13.1 Catalan Number	21
13.2 Bertrand's Ballot Theorem	21
13.3 Burnside's Lemma	21

14 Special Numbers

14.1 Fibonacci Series	21
14.2 Prime Numbers	22

1 Init (Linux)

開場流程：

```
1 vim ~/.vimrc
2 mkdir contest && cd contest
3
4 vim template.cpp
5 for c in {A..P}; do
6     cp template.cpp $c.cpp
7 done
8
9 vim run.sh && chmod 777 run.sh
```

1.1 vimrc

```
6 1 syn on
6 2 set nu rnu ru cul mouse=a
6 3 set cin et ts=4 sw=4 sts=4
7 4 set autochdir
8 5 set clipboard=unnamedplus
8 6
9 7 colo koehler
9 8
9 9 no <C-h> ^
10 10 no <C-l> $
10 11 no ; :
11 12
11 13 inoremap {<CR> {<CR><Esc>k0
```

1.2 template.cpp

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 void solve() {
5 }
6
7 int main() {
8     ios_base::sync_with_stdio(false); cin.tie(0);
9     int TEST = 1;
10    //cin >> TEST;
11    while (TEST--) solve();
12    return 0;
13 }
```

1.3 run.sh

```
15 1 #!/bin/bash
15 2
15 3 g++ -std=c++17 -O2 -g -fsanitize=undefined,address $1
15 4 && echo DONE COMPILE || exit 1
16 5 ./a.out
```

2 Reminder

2.1 Observations and Tricks

- Contribution Technique
- 二分圖/Spanning Tree/DFS Tree
- 行、列操作互相獨立

- 奇偶性
- 當 s, t 遞增並且 $t = f(s)$ ，對 s 二分搜不好做，可以改成對 t 二分搜，再算 $f(t)$
- 啟發式合併
- Permutation Normalization (做一些平移對齊兩個 permutation)
- 枚舉 $a_1 \sim a_n$ 再枚舉 $a_n \sim a_1$ 可以包在一個迴圈
- 兩個凸型函數相加還是凸型函數，相減不一定

2.2 Bug List

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

3 Basic

3.1 template (optional)

```

1 #define F first
2 #define S second
3 #define ep emplace
4 #define eb emplace_back
5 #define endl '\n'
6
7 template<class T> using V=vector<T>;
8 typedef long long ll;
9 typedef pair<int, int> pii;
10 typedef pair<ll, ll> pll;
11 typedef pair<int, ll> pil;
12 typedef pair<ll, int> pli;
13
14 /* ===== */
15 // STL and I/O
16 // pair
17 template<typename T1, typename T2>
18 ostream& operator<<(ostream& os, pair<T1, T2> p) {
19     return os << "(" << p.first << ", " << p.second <<
20         " ";
21 }
22 template<typename T1, typename T2>
23 istream& operator>>(istream& is, pair<T1, T2>& p) {
24     return is >> p.first >> p.second;
25 }
26 // vector
27 template<typename T>
28 istream& operator>>(istream& is, vector<T>& v) {
29     for (auto& x : v) is >> x;
30     return is;
31 }
32 template<typename T>
33 ostream& operator<<(ostream& os, const vector<T>& v) {
34     for (const auto& x : v) os << x << ' ';
35     return os;
36 }
37 /* ===== */
38 // debug(), output()
39 #define RED "\x1b[31m"
40 #define GREEN "\x1b[32m"
41 #define YELLOW "\x1b[33m"

```

```

40 #define GRAY "\x1b[90m"
41 #define COLOREND "\x1b[0m"
42
43 void _debug() {}
44 template<typename A, typename... B> void _debug(A a, B...
45     b) { cerr << a << ' ', _debug(b...); }
46 #define debug(...) cerr<<GRAY<<#__VA_ARGS__<<": "<<
47     COLOREND, _debug(__VA_ARGS__), cerr<<endl
48
49 void _output() {}
50 template<typename A, typename... B> void _output(A a, B
51     ... b) { cout << a << ' ', _output(b...); }
52 #define output(...) _output(__VA_ARGS__), cout<<endl
53 /* ===== */
54 // BASIC ALGORITHM
55 string binary(ll x, int b = -1) {
56     if (b == -1) b = __lg(x) + 1;
57     string s = "";
58     for (int k = b - 1; k >= 0; k--) {
59         s.push_back((x & (1LL<<k)) ? '1' : '0');
60     }
61     return s;
62 }
63 /* ===== */
64 // CONSTANT
65 const int INF = 1.05e9;
66 const ll LINF = 4e18;
67 const int MOD = 1e9 + 7;
68 //const int MOD = 998244353;
69 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

```

```

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 // Author: Gino
2 struct node {
3     ll sum, add, mod; int ln;
4     node(): sum(0), add(0), mod(0), ln(0) {}
5 };
6
7 struct segT {
8     int n;
9     vector<ll> ar;
10    vector<node> st;
11
12    void init(int _n) {
13        n = _n;
14        reset(ar, n, 0LL);
15        reset(st, n*4);
16    }
17    void pull(int cl, int cr, int i) {
18        st[i].sum = st[cl].sum + st[cr].sum;
19    }
20    void push(int cl, int cr, int i) {
21        ll md = st[i].mod, ad = st[i].add;
22        if (md) {
23            st[cl].sum = md * st[cl].ln, st[cr].sum =
24                md * st[cr].ln;
25            st[cl].mod = md, st[cr].mod = md;
26            st[i].mod = 0;
27        }
28        if (ad) {
29            st[cl].sum += ad * st[cl].ln, st[cr].sum +=
30                ad * st[cr].ln;
31            st[cl].add += ad, st[cr].add += ad;
32            st[i].add = 0;
33        }
34    }
35 };

```

```

33 void build(int l, int r, int i) {
34     if (l == r) {
35         st[i].sum = ar[l];
36         st[i].ln = 1;
37         return;
38     }
39     int mid = (l+r)>>1, cl = i<<1, cr = i<<1|1;
40     build(l, mid, cl);
41     build(mid + 1, r, cr);
42     pull(cl, cr, i);
43     // DONT FORGET THIS
44     st[i].ln = st[cl].ln + st[cr].ln;
45 }
46 void addval(int ql, int qr, ll val, int l, int r,
47     int i) {
48     if (qr < l || r < ql) return;
49     if (ql <= l && r <= qr) {
50         st[i].sum += val * st[i].ln;
51         st[i].add += val;
52         return;
53     }
54     int mid = (l+r)>>1, cl = i<<1, cr = i<<1|1;
55     push(cl, cr, i);
56     addval(ql, qr, val, l, mid, cl);
57     addval(ql, qr, val, mid + 1, r, cr);
58     pull(cl, cr, i);
59 }
60 void modify(int ql, int qr, ll val, int l, int r,
61     int i) {
62     if (qr < l || r < ql) return;
63     if (ql <= l && r <= qr) {
64         st[i].sum = val * st[i].ln;
65         st[i].add = 0;
66         st[i].mod = val;
67         return;
68     }
69     int mid = (l+r)>>1, cl = i<<1, cr = i<<1|1;
70     push(cl, cr, i);
71     modify(ql, qr, val, l, mid, cl);
72     modify(ql, qr, val, mid+1, r, cr);
73     pull(cl, cr, i);
74 }
75 ll query(int ql, int qr, int l, int r, int i) {
76     if (qr < l || r < ql) return 0;
77     if (ql <= l && r <= qr) return st[i].sum;
78     int mid = (l+r)>>1, cl = i<<1, cr = i<<1|1;
79     push(cl, cr, i);
80     return (query(ql, qr, l, mid, cl) +
81         query(ql, qr, mid+1, r, cr));
82 }

```

5.2 Heavy Light Decomposition (Benson)

```

1 // Author: Benson
2 const int N = 2e5+5;
3 vector<int> adj[N];
4
5 int dsu[N], sz[N], head[N], heavy[N], arr[N], num[N],
6     dep[N], par[N], visited[N], val[N], t;
7 int tr[N<<2], tag[N<<2];
8
9 int combine(int a, int b){
10     return max(a,b);
11 }
12
13 void push(int idx){
14     if(tag[idx]){
15         tr[idx<<1] = tag[idx];
16         tr[idx<<1|1] = tag[idx];
17         tag[idx<<1] = tag[idx];
18         tag[idx<<1|1] = tag[idx];
19         tag[idx] = 0;
20     }
21 }
22
23 void modify(int ml, int mr, int val, int idx, int l,
24     int r){
25     if(ml > mr) swap(ml,mr);
26     if(l==r) push(idx);

```

```

26     if(ml <= l && r <= mr){
27         tr[idx] = val;
28         tag[idx] = val;
29         return;
30     }
31     int mid = l+r>>1;
32     if(ml <= mid) modify(ml,mr,val,idx<<1,l,mid);
33     if(mr > mid) modify(ml,mr,val,idx<<1|1,mid+1,r);
34     tr[idx] = combine(tr[idx<<1],tr[idx<<1|1]);
35 }
36
37 int query(int ml, int mr, int idx, int l, int r){
38     if(ml > mr) swap(ml,mr);
39     if(l==r) push(idx);
40     if(ml <= l && r <= mr){
41         return tr[idx];
42     }
43     int mid = l+r>>1;
44     if(mr <= mid) return query(ml,mr,idx<<1,l,mid);
45     if(ml > mid) return query(ml,mr,idx<<1|1,mid+1,r);
46     return combine(query(ml,mr,idx<<1,l,mid),query(ml,
47         mr,idx<<1|1,mid+1,r));
48 }
49 void dfs(int u, int p){
50     int mxsz = 0, idx = 0;
51     sz[u] = 1;
52     visited[u] = 1;
53     for(auto v : adj[u]){
54         if(v==p) continue;
55         dep[v] = dep[u]+1;
56         par[v] = u;
57         dfs(v,u);
58         if(sz[v] > mxsz) mxsz = sz[v], idx = v;
59         sz[u] += sz[v];
60     }
61     heavy[u] = idx;
62 }
63
64 void decompose(int u, int h){
65     head[u] = h;
66     arr[u] = ++t;
67     visited[u] = 1;
68     if(heavy[u]) decompose(heavy[u],h);
69
70     for(auto v : adj[u]){
71         if(v==par[u]||v==heavy[u]) continue;
72         decompose(v,v);
73     }
74 }
75
76 int hld_query(int u, int v){
77     int res = 0;
78     while(head[u]!=head[v]){
79         if(dep[head[u]] < dep[head[v]]) swap(u,v);
80         res = max(res,query(arr[head[u]],arr[u],1,1,t));
81         u = par[head[u]];
82     }
83     if(dep[u] > dep[v]) swap(u,v);
84     res = max(res,query(arr[u],arr[v],1,1,t));
85     return res;
86 }
87
88 void hld_modify(int u, int v, int val){
89     if(dep[u] > dep[v]) swap(u,v);
90     while(head[u]!=head[v]){
91         if(dep[head[u]] < dep[head[v]]) swap(u,v);
92         modify(arr[head[u]],arr[u],val,1,1,t);
93         u = par[head[u]];
94     }
95     if(dep[u] > dep[v]) swap(u,v);
96     modify(arr[u],arr[v],val,1,1,t);
97 }
98

```

5.3 Heavy Light Decomposition (Ian)

```

1 // Author: Ian
2 // TODO: (2025/08/26) Debug Lazy Segment Tree
3 constexpr int maxn=2e5+5;

```

```

4 int arr[(maxn+1)<<2];
5 #define m ((l+r)>>1)
6 void build(V<int>& v, int i=1, int l=0, int r=maxn){
7     if((int)v.size()<=l) return;
8     if(r-l==1){arr[i]=v[l];return;}
9     build(v,i<<1,l,m),build(v,i<<1|1,m,r);
10    arr[i]=max(arr[i<<1],arr[i<<1|1]);
11 }
12 void modify(int p, int k, int i=1, int l=0, int r=maxn){
13     if(p<l||r<=p) return;
14     if(r-l==1){arr[i]=k;return;}
15     if(p<m) modify(p,k,i<<1,l,m);
16     else modify(p,k,i<<1|1,m,r);
17     arr[i]=max(arr[i<<1],arr[i<<1|1]);
18 }
19 int query(int ql, int qr, int i=1, int l=0, int r=maxn){
20     if(qr<=l||r<=ql) return 0;
21     if(ql<=l&&r<=qr) return arr[i];
22     if(qr<=m) return query(ql,qr,i<<1,l,m);
23     if(m<=ql) return query(ql,qr,i<<1|1,m,r);
24     return max(query(ql,qr,i<<1,l,m),query(ql,qr,i
25         <<1|1,m,r));
26 }
27 #undef m
28 inline void solve(){
29     int n,q;cin>>n>>q;
30     V<int> v(n);
31     for(auto& i:v) cin>>i;
32     V<V<int>> e(n);
33     for(int i=1;i<n;i++){
34         int a,b;cin>>a>>b,a--,b--;
35         e[a].emplace_back(b);
36         e[b].emplace_back(a);
37     }
38     V<int> d(n,0),f(n,0),sz(n,1),son(n,-1);
39     F<void(int,int)> dfs1=
40     [&](int x,int pre){
41         for(auto i:e[x]) if(i!=pre){
42             d[i]=d[x]+1,f[i]=x;
43             dfs1(i,x),sz[x]+=sz[i];
44             if(!~son[x]||sz[son[x]]<sz[i])
45                 son[x]=i;
46         }
47     };dfs1(0,0);
48     V<int> top(n,0),dfn(n,-1),rnk(n,0);
49     F<void(int,int)> dfs2=
50     [&](int x,int t){
51         static int cnt=0;
52         dfn[x]=cnt++,rnk[dfn[x]]=x,top[x]=t;
53         if(!~son[x]) return;
54         dfs2(son[x],t);
55         for(auto i:e[x])
56             if(!~dfn[i]) dfs2(i,i);
57     };dfs2(0,0);
58     V<int> dfnv(n);
59     for(int i=0;i<n;i++)
60         dfnv[dfn[i]]=v[i];
61     build(dfnv);
62     while(q--){
63         int op,a,b;cin>>op>>a>>b;
64         switch(op){
65             case 1:{
66                 modify(dfn[a-1],b);
67             }break;
68             case 2:{
69                 a--,b--;
70                 int ans=0;
71                 while(top[a]!=top[b]){
72                     if(d[top[a]]>d[top[b]]) swap(a,b);
73                     ans=max(ans,query(dfn[top[b]],dfn[b]+1));
74                     b=f[top[b]];
75                 }
76                 if(dfn[a]>dfn[b]) swap(a,b);
77                 ans=max(ans,query(dfn[a],dfn[b]+1));
78                 cout<<ans<<endl;
79             }break;
80             }
81     }
82 }

```

5.4 Heavy Light Decomposition (Gino)

```

1 // Author: Gino
2 // usage: call HLD() and modify code in query()
3 // when apply update to subtree, maintain mxi (max id
  of u's subtree),
4 // but don't forget Leverage mxi[mxkid[u]]!!! this is
  tricky!!!
5 int n;
6 vector<vector<int>> G;
7 vector<int> sz; // size of subtree
8 vector<int> mxkid; // kid that has max sz
9 vector<int> dep, pa;
10
11 vector<int> id, top; // id: node id in ds
12 // position in data structure (e.g. segment tree) for
  every vertex
13 int pos_in_ds;
14
15 void dfs(int u, int p, int h) {
16     pa[u] = p, dep[u] = h;
17     for (auto& v : G[u]) {
18         if (v == p) continue;
19         dfs(v, u, h + 1);
20         sz[u] += sz[v];
21         if (mxkid[u] == -1 || sz[v] > sz[mxkid[u]])
22             mxkid[u] = v;
23     }
24 }
25
26 // t stands for top vertex
27 void hld(int u, int t) {
28     // 0-base or 1-base depends on your data structure
29     id[u] = pos_in_ds++;
30     top[u] = t;
31     if (mxkid[u] == -1) return; // if this is a leaf
32
33     // extend current chain with heavy kid
34     hld(mxkid[u], t);
35     // light kids: open a new chain
36     for (auto& v : G[u]) {
37         if (v == pa[u] || v == mxkid[u]) continue;
38         hld(v, v);
39     }
40 }
41
42 void HLD() {
43     sz.assign(n + 1, 1);
44     mxkid.assign(n + 1, -1);
45     dep.assign(n + 1, 0);
46     pa.assign(n + 1, 0);
47     pos_in_ds = 0;
48
49     dfs(1, 1, 0);
50     hld(1, 1);
51
52     /* [CODE] initialize data structure */
53     // remember to use tree id
54 }
55
56 void query(int a, int b) {
57     while (top[a] != top[b]) {
58         // W.L.O.G. a is deeper than b
59         if (dep[top[a]] < dep[top[b]]) swap(a, b);
60         /* [CODE] some operations on interval [id[ta],
          id[a]] */
61         // Leverage "a" to above chain
62         a = pa[top[a]];
63     }
64     if (dep[a] < dep[b]) swap(a, b);
65     /* [CODE] some operations on interval [id[b], id[a]] */
66 }

```

5.5 Skew Heap

```

1 // Author: Ian
2 // Function: 插入、删除最小值、合併兩個左偏樹都能 O(Log
  n)
3 struct node{
4     node *l,*r;

```

```

5     int v;
6     node(int x):v(x){
7         l=r=nullptr;
8     }
9 };
10 node* merge(node* a,node* b){
11     if(!a||!b) return a?:b;
12     // min heap
13     if(a->v>b->v) swap(a,b);
14     a->r=merge(a->r,b);
15     swap(a->l,a->r);
16     return a;
17 }

```

5.6 Leftist Heap

```

1 // Author: Unknown
2 // Function: 插入、删除最小值、合併兩個左偏樹都能 O(Log
  n)
3 struct node{
4     node *l,*r;
5     int d, v;
6     node(int x):d(1),v(x){
7         l=r=nullptr;
8     }
9 };
10 static inline int d(node* x){return x?x->d:0;}
11 node* merge(node* a,node* b){
12     if(!a||!b) return a?:b;
13     // min heap
14     if(a->v>b->v) swap(a,b);
15     a->r=merge(a->r,b);
16     if(d(a->l)<d(a->r))
17         swap(a->l,a->r);
18     a->d=d(a->r)+1;
19     return a;
20 }

```

5.7 Persistent Treap

```

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

```

```

41     return {a, ret};
42 }
43 }

```

5.8 Li Chao Tree

```

1 // Author: Unknown
2 // Function: Query maximum value of L_i(x), L_i is the
   i-th line.
3 typedef long double ld;
4 constexpr int maxn = 5e4 + 5;
5 struct line {
6     ld a, b;
7     ld operator()(ld x) {return a * x + b;}
8 } arr[(maxn + 1) << 2];
9 bool operator<(line a, line b) {return a.a < b.a;}
10 #define m ((l+r)>>1)
11 void insert(line x, int i = 1, int l = 0, int r = maxn) {
12     if (r - l == 1) {
13         if (x(l) > arr[i](l))
14             arr[i] = x;
15         return;
16     }
17     line a = max(arr[i], x), b = min(arr[i], x);
18     if (a(m) > b(m))
19         arr[i] = a, insert(b, i << 1, l, m);
20     else
21         arr[i] = b, insert(a, i << 1 | 1, m, r);
22 }
23 ld query(int x, int i = 1, int l = 0, int r = maxn) {
24     if (x < l || r <= x) return -numeric_limits<ld>::max();
25     if (r - l == 1) return arr[i](x);
26     return max({arr[i](x), query(x, i << 1, l, m), query(
27         x, i << 1 | 1, m, r)});
28 #undef m

```

5.9 Time Segment Tree

```

1 // Author: Ian
2 constexpr int maxn = 1e5 + 5;
3 V<P<int>> arr[(maxn + 1) << 2];
4 V<int> dsu, sz;
5 V<tuple<int, int, int>> his;
6 int cnt, q;
7 int find(int x) {
8     return x == dsu[x] ? x : find(dsu[x]);
9 };
10 inline bool merge(int x, int y) {
11     int a = find(x), b = find(y);
12     if (a == b) return false;
13     if (sz[a] > sz[b]) swap(a, b);
14     his.emplace_back(a, b, sz[b]), dsu[a] = b, sz[b] +=
15         sz[a];
16     return true;
17 };
18 inline void undo() {
19     auto [a, b, s] = his.back(); his.pop_back();
20     dsu[a] = a, sz[b] = s;
21 };
22 #define m ((l + r) >> 1)
23 void insert(int ql, int qr, P<int> x, int i = 1, int l
24     = 0, int r = q) {
25     // debug(ql, qr, x); return;
26     if (qr <= l || r <= ql) return;
27     if (ql <= l && r <= qr) {arr[i].push_back(x);
28         return;}
29     if (qr <= m)
30         insert(ql, qr, x, i << 1, l, m);
31     else if (m <= ql)
32         insert(ql, qr, x, i << 1 | 1, m, r);
33     else {
34         insert(ql, qr, x, i << 1, l, m);
35         insert(ql, qr, x, i << 1 | 1, m, r);
36     }
37 }
38 void traversal(V<int>& ans, int i = 1, int l = 0, int r
39     = q) {

```

```

36 int opcnt = 0;
37 // debug(i, l, r);
38 for (auto [a, b] : arr[i])
39     if (merge(a, b))
40         opcnt++, cnt--;
41 if (r - l == 1) ans[l] = cnt;
42 else {
43     traversal(ans, i << 1, l, m);
44     traversal(ans, i << 1 | 1, m, r);
45 }
46 while (opcnt--)
47     undo(), cnt++;
48 arr[i].clear();
49 }
50 #undef m
51 inline void solve() {
52     int n, m; cin >> n >> m >> q, q++;
53     dsu.resize(cnt = n), sz.assign(n, 1);
54     iota(dsu.begin(), dsu.end(), 0);
55     // a, b, time, operation
56     unordered_map<ll, V<int>> s;
57     for (int i = 0; i < m; i++) {
58         int a, b; cin >> a >> b;
59         if (a > b) swap(a, b);
60         s[(((ll)a << 32) | b).emplace_back(0);
61     }
62     for (int i = 1; i < q; i++) {
63         int op, a, b;
64         cin >> op >> a >> b;
65         if (a > b) swap(a, b);
66         switch (op) {
67             case 1:
68                 s[(((ll)a << 32) | b).push_back(i);
69                 break;
70             case 2:
71                 auto tmp = s[(((ll)a << 32) | b).back();
72                 s[(((ll)a << 32) | b).pop_back();
73                 insert(tmp, i, P<int> {a, b});
74         }
75     }
76     for (auto [p, v] : s) {
77         int a = p >> 32, b = p & -1;
78         while (v.size()) {
79             insert(v.back(), q, P<int> {a, b});
80             v.pop_back();
81         }
82     }
83     V<int> ans(q);
84     traversal(ans);
85     for (auto i : ans)
86         cout << i << ' ';
87     cout << endl;
88 }

```

6 DP

6.1 Aliens

```

1 // Author: Gino
2 // Function: TODO
3 int n; ll k;
4 vector<ll> a;
5 vector<pll> dp[2];
6 void init() {
7     cin >> n >> k;
8     for (auto& d : dp) d.clear(), d.resize(n);
9     a.clear(); a.resize(n);
10    for (auto& i : a) cin >> i;
11 }
12 pll calc(ll p) {
13     dp[0][0] = make_pair(0, 0);
14     dp[1][0] = make_pair(-a[0], 0);
15     for (int i = 1; i < n; i++) {
16         if (dp[0][i-1].first > dp[1][i-1].first + a[i] - p)
17             {
18                 dp[0][i] = dp[0][i-1];
19             }
20         else if (dp[0][i-1].first < dp[1][i-1].first + a[
21             i] - p) {
22             dp[0][i] = make_pair(dp[1][i-1].first + a[i] - p,
23                 dp[1][i-1].second+1);
24         }
25     }
26 }

```



```

20     } else {
21         dp[0][i] = make_pair(dp[0][i-1].first, min(dp[0][
            i-1].second, dp[1][i-1].second+1));
22     }
23     if (dp[0][i-1].first - a[i] > dp[1][i-1].first) {
24         dp[1][i] = make_pair(dp[0][i-1].first - a[i], dp
            [0][i-1].second);
25     } else if (dp[0][i-1].first - a[i] < dp[1][i-1].
        first) {
26         dp[1][i] = dp[1][i-1];
27     } else {
28         dp[1][i] = make_pair(dp[1][i-1].first, min(dp[0][
            i-1].second, dp[1][i-1].second));
29     }
30 }
31 return dp[0][n-1];
32 }
33 void solve() {
34     ll l = 0, r = 1e7;
35     pll res = calc(0);
36     if (res.second <= k) return cout << res.first << endl
        , void();
37     while (l < r) {
38         ll mid = (l+r)>>1;
39         res = calc(mid);
40         if (res.second <= k) r = mid;
41         else l = mid+1;
42     }
43     res = calc(l);
44     cout << res.first + k*l << endl;
45 }

```

6.2 SOS DP

```

1 // Author: Gino
2 // Function: Solve problems that enumerates subsets of
    subsets ( $3^n \Rightarrow n \cdot 2^n$ )
3 for (int msk = 0; msk < (1<<n); msk++) {
4     for (int i = 1; i <= n; i++) {
5         if (msk & (1<<(i - 1))) {
6             // dp[msk][i] = dp[msk][i - 1] + dp[msk ^
                (1<<(i - 1))][i - 1];
7         } else {
8             // dp[msk][i] = dp[msk][i - 1];
9         }
10    }
11 }

```

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;

```

```

        else centroid.first = u;
    }
}

```

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 */

```

```

74 vector<int> src;
75 for (int i = 1; i <= n; i++)
76     src.emplace_back(i);
77
78 SPFA(src);
79 // BellmanFord(src);
80
81 int ptr = -1;
82 for (int i = 1; i <= n; i++) if (negCycle[i])
83     { ptr = i; break; }
84
85 if (ptr == -1) { return cout << "NO" << endl, void(); }
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             }
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() > 0) 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
24 int main() {
25     WiWiHorz
26     init();
27
28     REP(i, m) {
29         int u, v;
30         cin >> u >> v;
31         if (u == v) continue;
32         adj[--u][--v]++;
33     }
34

```

```

35     dp[0][1] = 1;
36     FOR(i, 1, n, 1) {
37         dp[i][1] = 0;
38         dp[i][1|(1<<i)] = adj[0][i];
39     }
40     FOR(msk, 1, (1<<n), 1) {
41         if (msk == 1) continue;
42         dp[0][msk] = 0;
43     }
44
45     DP(n-1, (1<<n)-1);
46     cout << dp[n-1][(1<<n)-1] << endl;
47
48     return 0;
49 }
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     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; 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        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.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^1][pos[(t ? i.F.F : i.F.S) ]++] = i;
20        } }

```

```

21 bool fill_suf() {
22     bool end = true;
23     for (int i = 0; i < n; i++) suf[i] = buc[0][i].S;
24     rk[suf[0]] = 0;
25     for (int i = 1; i < n; i++) {
26         int dif = (buc[0][i].F != buc[0][i-1].F);
27         end &= dif;
28         rk[suf[i]] = rk[suf[i-1]] + dif;
29     } return end;
30 }
31 void sa() {
32     for (int i = 0; i < n; i++)
33         buc[0][i] = make_pair(make_pair(s[i], s[i]), i);
34     sort(buc[0].begin(), buc[0].end());
35     if (fill_suf()) return;
36     for (int k = 0; (1<<k) < n; k++) {
37         for (int i = 0; i < n; i++)
38             buc[0][i] = make_pair(make_pair(rk[i], rk[(i + (1<<k)) % n]), i);
39         radix_sort();
40         if (fill_suf()) return;
41     } }
42 void LCP() { int k = 0;
43     for (int i = 0; i < n-1; i++) {
44         if (rk[i] == 0) continue;
45         int pi = rk[i];
46         int j = suf[pi-1];
47         while (i+k < n && j+k < n && s[i+k] == s[j+k]) k++;
48         lcp[pi] = k;
49         k = max(k-1, 0);
50     } }
51 };
52 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[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]){

```

```

38     neq=lst<0||memcmp(s+sa[i],s+lst,(p[q[sa[i]]+1]-sa
        [i])*sizeof(int));
39     ns[q[lst=sa[i]]]=nmxz+neq;
40 }
41 sais(ns,nsa,p+nn,q+n,t+n,c+z,nn,nmxz+1);
42 MAGIC(for(int i=nn-1;i>=0;i--) sa[--x[s[p[nsa[i]
        ]]]]=p[nsa[i]]);
43 }
44 }sa;
45 int H[N],SA[N],RA[N];
46 void suffix_array(int* ip,int len){
47     // should padding a zero in the back
48     // 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)
55 }

```

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 // Author: Gino
2 typedef long long T;
3 // typedef long double T;
4 const long double eps = 1e-8;
5
6 short sgn(T x) {
7     if (abs(x) < eps) return 0;
8     return x < 0 ? -1 : 1;
9 }
10
11 struct Pt {
12     T x, y;
13     Pt(T _x=0, T _y=0):x(_x), y(_y) {}
14     Pt operator+(Pt a) { return Pt(x+a.x, y+a.y); }
15     Pt operator-(Pt a) { return Pt(x-a.x, y-a.y); }
16     Pt operator*(T a) { return Pt(x*a, y*a); }
17     Pt operator/(T a) { return Pt(x/a, y/a); }
18     T operator*(Pt a) { return x*a.x + y*a.y; }
19     T operator^(Pt a) { return x*a.y - y*a.x; } // 不要打反
20     bool operator<(Pt a)
21         { return x < a.x || (x == a.x && y < a.y); }
22     //return sgn(x-a.x) < 0 || (sgn(x-a.x) == 0 && sgn(y-a.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 // Author: Gino
2 // Function: Check if a point P sits in a polygon (
3 // doesn't have to be convex hull)
4 // 0 = Bound, 1 = In, -1 = Out
5 short inPoly(Pt p) {
6     for (int i = 0; i < n; i++)
7         if (onseg(p, E[i], E[(i+1)%n])) return 0;
8     int cnt = 0;
9     for (int i = 0; i < n; i++)
10        if (banana(p, Pt(p.x+1, p.y+2e9), E[i], E[(i+1)%n]))
11            cnt ^= 1;
12 return (cnt ? 1 : -1);

```

9.3 Sort by Angle

```

1 // Author: Gino
2 int ud(Pt a) { // up or down half plane
3     if (a.y > 0) return 0;
4     if (a.y < 0) return 1;
5     return (a.x >= 0 ? 0 : 1);
6 }
7 sort(ALL(E), [&](const Pt& a, const Pt& b){
8     if (ud(a) != ud(b)) return ud(a) < ud(b);
9     return (a^b) > 0;
10 });

```

9.4 Line Intersect Check

```

1 // Author: Gino
2 // Function: check if (p1---p2) (q1---q2) banana
3 inline bool banana(Pt p1, Pt p2, Pt q1, Pt q2) {
4     if (onseg(p1, q1, q2) || onseg(p2, q1, q2) ||
5         onseg(q1, p1, p2) || onseg(q2, p1, p2)) {

```

```

6     return true;
7 }
8 Pt p = mv(p1, p2), q = mv(q1, q2);
9 return (ori(p, mv(p1, q1)) * ori(p, mv(p1, q2)) < 0 &&
10        ori(q, mv(q1, p1)) * ori(q, mv(q1, p2)) < 0);
11 }

```

9.5 Line Intersection

```

1 // Author: Gino
2 // T: Long double
3 Pt bananaPoint(Pt p1, Pt p2, Pt q1, Pt q2) {
4     if (onseg(q1, p1, p2)) return q1;
5     if (onseg(q2, p1, p2)) return q2;
6     if (onseg(p1, q1, q2)) return p1;
7     if (onseg(p2, q1, q2)) return p2;
8     double s = abs(mv(p1, p2) ^ mv(p1, q1));
9     double t = abs(mv(p1, p2) ^ mv(p1, q2));
10    return q2 * (s/(s+t)) + q1 * (t/(s+t));
11 }

```

9.6 Convex Hull

```

1 // Author: Gino
2 vector<Pt> hull;
3 void convexHull() {
4     hull.clear(); sort(E.begin(), E.end());
5     for (int t : {0, 1}) {
6         int b = (int)hull.size();
7         for (auto& ei : E) {
8             while ((int)hull.size() - b >= 2 &&
9                    ori(mv(hull[(int)hull.size()-2], hull.
10                        back()),
11                        mv(hull[(int)hull.size()-2], ei)) ==
12                        -1) {
13                 hull.pop_back();
14             }
15             hull.emplace_back(ei);
16         }
17         hull.pop_back();
18         reverse(E.begin(), E.end());
19     } }

```

9.7 Lower Concave Hull

```

1 // Author: Unknown
2 struct Line {
3     mutable ll m, b, p;
4     bool operator<(const Line& o) const { return m < o.m; }
5     bool operator<(ll x) const { return p < x; }
6 };
7
8 struct LineContainer : multiset<Line, less<>> {
9     // (for doubles, use inf = 1/.0, div(a,b) = a/b)
10    const ll inf = LLONG_MAX;
11    ll div(ll a, ll b) { // floored division
12        return a / b - ((a ^ b) < 0 && a % b); }
13    bool isect(iterator x, iterator y) {
14        if (y == end()) { x->p = inf; return false; }
15        if (x->m == y->m) x->p = x->b > y->b ? inf : -inf;
16        else x->p = div(y->b - x->b, x->m - y->m);
17        return x->p >= y->p;
18    }
19    void add(ll m, ll b) {
20        auto z = insert({m, b, 0}), y = z++, x = y;
21        while (isect(y, z)) z = erase(z);
22        if (x != begin() && isect(--x, y)) isect(x, y =
23            erase(y));
24        while ((y = x) != begin() && (--x)->p >= y->p)
25            isect(x, erase(y));
26    }
27    ll query(ll x) {
28        assert(!empty());
29        auto l = *lower_bound(x);
30        return l.m * x + l.b;
31    }
32 };

```

9.8 Polygon Area

```

1 // Author: Gino
2 // Function: Return doubled area of a polygon
3 T dbarea(vector<Pt>& e) {
4     ll res = 0;
5     for (int i = 0; i < (int)e.size(); i++)
6         res += e[i]^e[(i+1)%SZ(e)];
7     return abs(res);
8 }

```

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 // Author: Gino
2 // Function: Find Min Enclosing Circle using Randomized
3 // O(n) Algorithm
4 Pt circumcenter(Pt A, Pt B, Pt C) {
5     // a1(x-A.x) + b1(y-A.y) = c1
6     // a2(x-A.x) + b2(y-A.y) = c2
7     // solve using Cramer's rule
8     T a1 = B.x-A.x, b1 = B.y-A.y, c1 = dis2(A, B)/2.0;
9     T a2 = C.x-A.x, b2 = C.y-A.y, c2 = dis2(A, C)/2.0;
10    T D = Pt(a1, b1) ^ Pt(a2, b2);
11    T Dx = Pt(c1, b1) ^ Pt(c2, b2);
12    T Dy = Pt(a1, c1) ^ Pt(a2, c2);
13    if (D == 0) return Pt(-INF, -INF);
14    return A + Pt(Dx/D, Dy/D);
15 }
16 Pt center; T r2;
17
18 void minEncloseCircle() {
19     mt19937 gen(chrono::steady_clock::now().
20         time_since_epoch().count());
21     shuffle(ALL(E), gen);
22     center = E[0], r2 = 0;
23
24     for (int i = 0; i < n; i++) {
25         if (dis2(center, E[i]) <= r2) continue;
26         center = E[i], r2 = 0;
27         for (int j = 0; j < i; j++) {
28             if (dis2(center, E[j]) <= r2) continue;
29             center = (E[i] + E[j]) / 2.0;
30             r2 = dis2(center, E[i]);
31             for (int k = 0; k < j; k++) {
32                 if (dis2(center, E[k]) <= r2) continue;
33                 center = circumcenter(E[i], E[j], E[k]);
34                 r2 = dis2(center, E[i]);
35             }
36         }
37     }
38 }

```

9.11 PolyUnion

```

1 // Author: Unknown
2 struct PY {
3     int n; Pt pt[5]; double area;
4     Pt& operator[](const int x) { return pt[x]; }
5     void init() { //n,pt[0~n-1] must be filled
6         area=pt[n-1]^pt[0];
7         for(int i=0;i<n-1;i++) area+=pt[i]^pt[i+1];
8         if((area/=2)<0)reverse(pt,pt+n),area=-area;
9     }
10 };
11 PY py[500]; pair<double,int> c[5000];
12 inline double segP(Pt &p,Pt &p1,Pt &p2){
13     if(dcmp(p1.x-p2.x)==0) return (p.y-p1.y)/(p2.y-p1.y);
14     return (p.x-p1.x)/(p2.x-p1.x);
15 }

```



```

16 double polyUnion(int n){ //py[0~n-1] must be filled
17     int i,j,ii,jj,ta,tb,r,d; double z,w,s,sum=0,tc,td;
18     for(i=0;i<n;i++) py[i][py[i].n]=py[i][0];
19     for(i=0;i<n;i++){
20         for(ii=0;ii<py[i].n;ii++){
21             r=0;
22             c[r++]=make_pair(0.0,0); c[r++]=make_pair(1.0,0);
23             for(j=0;j<n;j++){
24                 if(i==j) continue;
25                 for(jj=0;jj<py[j].n;jj++){
26                     ta=dcmp(tri(py[i][ii],py[i][ii+1],py[j][jj]))
27                     ;
28                     tb=dcmp(tri(py[i][ii],py[i][ii+1],py[j][jj
29                     +1]));
30                     if(ta==0 && tb==0){
31                         if((py[j][jj+1]-py[j][jj])*(py[i][ii+1]-py[
32                         i][ii])>0&&j<i){
33                             c[r++]=make_pair(segP(py[j][jj],py[i][ii
34                             ],py[i][ii+1]),1);
35                             c[r++]=make_pair(segP(py[j][jj+1],py[i][
36                             ii],py[i][ii+1]),-1);
37                         }
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                     }else if(ta<0 && tb>0){
43                         tc=tri(py[j][jj],py[j][jj+1],py[i][ii]);
44                         td=tri(py[j][jj],py[j][jj+1],py[i][ii+1]);
45                         c[r++]=make_pair(tc/(tc-td),-1);
46                     } } }
47                     sort(c,c+r);
48                     z=min(max(c[0].first,0.0),1.0); d=c[0].second; s
49                     =0;
50                     for(j=1;j<r;j++){
51                         w=min(max(c[j].first,0.0),1.0);
52                         if(!d) s+=w-z;
53                         d+=c[j].second; z=w;
54                     }
55                     sum+=(py[i][ii]^py[i][ii+1])*s;
56                 }
57             }
58             return sum/2;
59         }
60     }
61 }
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

9.12 Minkowski Sum

```

1 // Author: Unknown
2 /* convex hull Minkowski Sum*/
3 #define INF 1000000000000000LL
4 int pos( const Pt& tp ) {
5     if( tp.Y == 0 ) return tp.X > 0 ? 0 : 1;
6     return tp.Y > 0 ? 0 : 1;
7 }
8 #define N 300030
9 Pt pt[ N ], qt[ N ], rt[ N ];
10 LL Lx,Rx;
11 int dn,un;
12 inline bool cmp( Pt a, Pt b ) {
13     int pa=pos( a ),pb=pos( b );
14     if(pa==pb) return (a^b)>0;
15     return pa<pb;
16 }
17 int minkowskiSum(int n,int m){
18     int i,j,r,p,q,fi,fj;
19     for(i=1,p=0;i<n;i++){
20         if( pt[i].Y<pt[p].Y ||
21             (pt[i].Y==pt[p].Y && pt[i].X<pt[p].X) ) p=i; }
22     for(i=1,q=0;i<m;i++){
23         if( qt[i].Y<qt[q].Y ||
24             (qt[i].Y==qt[q].Y && qt[i].X<qt[q].X) ) q=i; }
25     rt[0]=pt[p]+qt[q];
26     r=1; i=p; j=q; fi=fj=0;
27     while(1){
28         if((fj&&j==q) ||
29             ( !fi || i==p) &&
30             cmp(pt[(p+1)%n]-pt[p],qt[(q+1)%m]-qt[q]) ) {
31             rt[r]=rt[r-1]+pt[(p+1)%n]-pt[p];
32             p=(p+1)%n;
33             fi=1;
34         }else{
35             rt[r]=rt[r-1]+qt[(q+1)%m]-qt[q];
36             q=(q+1)%m;
37             fj=1;
38         }
39         r++;
40         if(i==p && j==q) break;
41     }
42     return r-1;
43 }
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
68 inline int inConvex(Pt p){
69     int L,R,M;
70     if(p.X<Lx || p.X>Rx) return 0;
71     L=0;R=dn;
72     while(L<R-1){ M=(L+R)/2;
73         if(p.X<qt[M].X) R=M; else L=M; }
74     if(tri(qt[L],qt[R],p)<0) return 0;
75     L=0;R=un;
76     while(L<R-1){ M=(L+R)/2;
77         if(p.X<rt[M].X) R=M; else L=M; }
78     if(tri(rt[L],rt[R],p)>0) return 0;
79     return 1;
80 }
81
82 int main(){
83     int n,m,i;
84     Pt p;
85     scanf("%d",&n);
86     for(i=0;i<n;i++) scanf("%Lld%Lld",&pt[i].X,&pt[i].Y);
87     scanf("%d",&m);
88     for(i=0;i<m;i++) scanf("%Lld%Lld",&qt[i].X,&qt[i].Y);
89     n=minkowskiSum(n,m);
90     for(i=0;i<n;i++) pt[i]=rt[i];
91     scanf("%d",&m);
92     for(i=0;i<m;i++) scanf("%Lld%Lld",&qt[i].X,&qt[i].Y);
93     n=minkowskiSum(n,m);
94     for(i=0;i<n;i++) pt[i]=rt[i];
95     initInConvex(n);
96     scanf("%d",&m);
97     for(i=0;i<m;i++){
98         scanf("%Lld %Lld",&p.X,&p.Y);
99         p.X*=3; p.Y*=3;
100         puts(inConvex(p)? "YES": "NO");
101     }
102 }

```

10 Number Theory

10.1 Basic

```

1 // Author: Gino
2 const int maxc = 5e5;
3 ll pw(ll a, ll n) {
4     ll res = 1;
5     while (n) {
6         if (n & 1) res = res * a % MOD;
7         a = a * a % MOD;
8         n >>= 1;
9     }
10 }

```

```

9     }
10    return res;
11 }
12
13 vector<ll> fac, ifac;
14 void build_fac() {
15     reset(fac, maxc + 1, 1LL);
16     reset(ifac, maxc + 1, 1LL);
17     for (int x = 2; x <= maxc; x++) {
18         fac[x] = x * fac[x - 1] % MOD;
19         ifac[x] = pw(fac[x], MOD - 2);
20     }
21 }
22
23 ll C(ll n, ll k) {
24     if (n < k) return 0LL;
25     return fac[n] * ifac[n - k] % MOD * ifac[k] % MOD;
26 }

```

10.2 Prime Seive and Defactor

```

1 // Author: Gino
2 const int maxc = 1e6 + 1;
3 vector<int> lpf;
4 vector<int> prime;
5
6 void seive() {
7     prime.clear();
8     lpf.resize(maxc, 1);
9     for (int i = 2; i < maxc; i++) {
10         if (lpf[i] == 1) {
11             lpf[i] = i;
12             prime.emplace_back(i);
13         }
14         for (auto& j : prime) {
15             if (i * j >= maxc) break;
16             lpf[i * j] = j;
17             if (j == lpf[i]) break;
18         }
19     }
20     vector<pii> fac;
21     void defactor(int u) {
22         fac.clear();
23         while (u > 1) {
24             int d = lpf[u];
25             fac.emplace_back(make_pair(d, 0));
26             while (u % d == 0) {
27                 u /= d;
28                 fac.back().second++;
29             }
30         }
31     }
32 }

```

10.3 Harmonic Series

```

1 // Author: Gino
2 // O(n log n)
3 for (int i = 1; i <= n; i++) {
4     for (int j = i; j <= n; j += i) {
5         // O(1) code
6     }
7 }
8
9 // PIE
10 // given array a[0], a[1], ..., a[n - 1]
11 // calculate dp[x] = number of pairs (a[i], a[j]) such
12 // that gcd(a[i], a[j]) = x // (i < j)
13 //
14 // idea: Let mc(x) = # of y s.t. x|y
15 // f(x) = # of pairs s.t. gcd(a[i], a[j]) >= 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
23 for (int x = maxc; x >= 1; x--) {
24     ll cnt_mul = 0; // number of multiples of x
25     for (int y = x; y <= maxc; y += x)
26         cnt_mul += cnt[y];
27 }

```

```

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

```

10.4 Count Number of Divisors

```

1 // Author: Gino
2 // Function: Count the number of divisors for all x <=
3 // 10^6 using harmonic series
4 const int maxc = 1e6;
5 vector<int> facs;
6
7 void find_all_divisors() {
8     facs.clear(); facs.resize(maxc + 1, 0);
9     for (int x = 1; x <= maxc; x++) {
10         for (int y = x; y <= maxc; y += x) {
11             facs[y]++;
12         }
13     }
14 }

```

10.5 數論分塊

```

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

```

10.6 Pollard's rho

```

1 // Author: Unknown
2 // Function: Find a non-trivial factor of a big number
3 // in O(n^(1/4) log^2(n))
4
5 ll find_factor(ll number) {
6     __int128 x = 2;
7     for (__int128 cycle = 1; ; cycle++) {
8         __int128 y = x;
9         for (int i = 0; i < (1 << cycle); i++) {
10             x = (x * x + 1) % number;
11             __int128 factor = __gcd(x - y, number);
12             if (factor > 1)
13                 return factor;
14         }
15     }
16 }

```

```

1 # Author: Unknown
2 # Function: Find a non-trivial factor of a big number
  in O(n^(1/4) log^2(n))
3 from itertools import count
4 from math import gcd
5 from sys import stdin
6
7 for s in stdin:
8     number, x = int(s), 2
9     brk = False
10    for cycle in count(1):
11        y = x
12        if brk:
13            break
14        for i in range(1 << cycle):
15            x = (x * x + 1) % number
16            factor = gcd(x - y, number)
17            if factor > 1:
18                print(factor)
19                brk = True
20                break

```

10.7 Miller Rabin

```

1 // Author: Unknown
2 // Function: Check if a number is a prime in O(100 *
  Log^2(n))
3 // miller_rabin(): return 1 if prime, 0 otherwise
4
5 // n < 4,759,123,141      3 : 2, 7, 61
6 // n < 1,122,004,669,633  4 : 2, 13, 23, 1662803
7 // n < 3,474,749,660,383  6 : pimes <= 13
8 // n < 2^64              7 :
9 // 2, 325, 9375, 28178, 450775, 9780504, 1795265022
10 bool witness(ll a, ll n, ll u, int t){
11     if(!(a%n)) return 0;
12     ll x=mypow(a,u,n);
13     for(int i=0;i<t;i++) {
14         ll nx=mul(x,x,n);
15         if(nx==1&&x!=1&&x!=n-1) return 1;
16         x=nx;
17     }
18     return x!=1;
19 }
20 bool miller_rabin(ll n, int s=100) {
21     // iterate s times of witness on n
22     if(n<2) return 0;
23     if(!(n&1)) return n == 2;
24     ll u=n-1; int t=0;
25     while(!(u&1)) u>>=1, t++;
26     while(s--){
27         ll a=randll()%(n-1)+1;
28         if(witness(a,n,u,t)) return 0;
29     }
30     return 1;
31 }

```

10.8 Fast Power

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

10.9 Extend GCD

```

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

```

```

18 ll inv(ll a, ll p) {
19     if (p == 1) return -1;
20     pll ans = bezout(a % p, -p, 1);
21     if (ans == pll{-LLINF, -LLINF}) return -1;
22     return (ans.F % p + p) % p;
23 }

```

10.10 Mu + Phi

```

1 // Author: Gino
2 const int maxn = 1e6 + 5;
3 ll f[maxn];
4 vector<int> lpf, prime;
5 void build() {
6     lpf.clear(); lpf.resize(maxn, 1);
7     prime.clear();
8     f[1] = ...; /* mu[1] = 1, phi[1] = 1 */
9     for (int i = 2; i < maxn; i++) {
10         if (lpf[i] == 1) {
11             lpf[i] = i; prime.emplace_back(i);
12             f[i] = ...; /* mu[i] = 1, phi[i] = i-1 */
13         }
14         for (auto& j : prime) {
15             if (i*j >= maxn) break;
16             lpf[i*j] = j;
17             if (i % j == 0) f[i*j] = ...; /* 0, phi[i]*j */
18             else f[i*j] = ...; /* -mu[i], phi[i]*phi[j] */
19             if (j >= lpf[i]) break;
20         } } }

```

10.11 Other Formulas

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

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 // Author: Gino
2 const int maxk = 20;
3 const int maxn = 1<<maxk;
4 const ll LINF = 1e18;
5
6 /* P = r*2^k + 1
7 P          r    k    g
8 998244353  119  23    3
9 1004535809 479  21    3
10
11 P          r    k    g
12 3          1    1    2
13 5          1    2    2
14 17         1    4    3
15 97         3    5    5
16 193        3    6    5
17 257        1    8    3
18 7681       15    9   17
19 12289      3   12   11
20 40961      5   13    3
21 65537      1   16    3
22 786433     3   18   10
23 5767169    11  19    3
24 7340033     7  20    3
25 23068673   11  21    3
26 104857601  25  22    3
27 167772161   5  25    3
28 469762049   7  26    3
29 1004535809 479  21    3
30 2013265921  15  27   31
31 2281701377  17  27    3
32 3221225473   3  30    5
33 75161927681 35  31    3
34 77309411329   9  33    7
35 206158430209 3  36  22
36 2061584302081 15  37    7
37 2748779069441 5  39    3
38 6597069766657 3  41    5
39 39582418599937 9  42    5
40 79164837199873 9  43    5
41 263882790666241 15 44    7
42 1231453023109121 35 45    3
43 1337006139375617 19 46    3
44 3799912185593857 27 47    5
45 4222124650659841 15 48   19
46 7881299347898369 7  50    6
47 31525197391593473 7  52    3
48 180143985094819841 5  55    6
49 1945555039024054273 27 56    5
50 4179340454199820289 29 57    3
51 9097271247288401921 505 54    6 */
52
53 const int g = 3;
54 const ll MOD = 998244353;
55
56 ll pw(ll a, ll n) { /* fast pow */ }
57
58 #define siz(x) (int)x.size()
59
60 template<typename T>
61 vector<T>& operator+=(vector<T>& a, const vector<T>& b)
62 {
63     if (siz(a) < siz(b)) a.resize(siz(b));
64     for (int i = 0; i < min(siz(a), siz(b)); i++) {
65         a[i] += b[i];
66         a[i] -= a[i] >= MOD ? MOD : 0;
67     }
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 template<typename T>
114 void NTT(vector<T>& a, bool inv=false) {
115
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         if (inv) {
137             T dn = pw(n, MOD-2);
138             for (auto& x : a) {
139                 x *= dn;
140                 if (x >= MOD) x %= MOD;
141             }
142         }
143     }
144
145 template<typename T>
146 inline void resize(vector<T>& a) {
147     int cnt = (int)a.size();

```

```

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 template<typename T>
114 void NTT(vector<T>& a, bool inv=false) {
115
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         if (inv) {
137             T dn = pw(n, MOD-2);
138             for (auto& x : a) {
139                 x *= dn;
140                 if (x >= MOD) x %= MOD;
141             }
142         }
143     }
144
145 template<typename T>
146 inline void resize(vector<T>& a) {
147     int cnt = (int)a.size();

```

```

144     for (; cnt > 0; cnt--) if (a[cnt-1]) break;
145     a.resize(max(cnt, 1));
146 }
147
148 template<typename T>
149 vector<T>& operator*=(vector<T>& a, vector<T> b) {
150     int na = (int)a.size();
151     int nb = (int)b.size();
152     a.resize(na + nb - 1, 0);
153     b.resize(na + nb - 1, 0);
154
155     NTT(a); NTT(b);
156     for (int i = 0; i < (int)a.size(); i++) {
157         a[i] *= b[i];
158         if (a[i] >= MOD) a[i] %= MOD;
159     }
160     NTT(a, true);
161
162     resize(a);
163     return a;
164 }
165
166 template<typename T>
167 void inv(vector<T>& ia, int N) {
168     vector<T> _a(move(ia));
169     ia.resize(1, pw[_a[0], MOD-2]);
170     vector<T> a(1, _a[0] + (-_a[0] < 0 ? MOD : 0));
171
172     for (int n = 1; n < N; n<=1) {
173         // n -> 2*n
174         // ia' = ia(2-a*ia);
175
176         for (int i = n; i < min(siz(_a), (n<1)); i++)
177             a.emplace_back(-_a[i] + (-_a[i] < 0 ? MOD : 0));
178
179         vector<T> tmp = ia;
180         ia *= a;
181         ia.resize(n<1);
182         ia[0] = ia[0] + 2 >= MOD ? ia[0] + 2 - MOD : ia[0] + 2;
183         ia *= tmp;
184         ia.resize(n<1);
185     }
186     ia.resize(N);
187 }
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] + ... + c[N-1]a[j-1])
212 Let B(x) = x^N - c[N-1]x^(N-1) - ... - c[1]x^1 - c[0]
213 Let R(x) = x^K mod 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 => 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    }

```

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 // Author: Benson
2 // Function: Max Flow, O(V^2 E)
3 struct Dinic {
4     struct Edge {
5         int t, c, r;
6         Edge() {}
7         Edge(int _t, int _c, int _r):
8             t(_t), c(_c), r(_r) {}
9     };
10    vector<vector<Edge>> G;
11    vector<int> dis, iter;
12    int s, t;
13    void init(int n) {
14        G.resize(n), dis.resize(n), iter.resize(n);
15        for (int i = 0; i < n; ++i)
16            G[i].clear();
17    }
18    void add(int a, int b, int c) {
19        G[a].eb(b, c, G[b].size());
20        G[b].eb(a, 0, G[a].size() - 1);
21    }
22    bool bfs() {
23        fill(ALL(dis), -1);
24        dis[s] = 0;
25        queue<int> que;
26        que.push(s);
27        while (!que.empty()) {
28            int u = que.front(); que.pop();
29            for (auto& e : G[u]) {

```

```

30         if(e.c > 0 && dis[e.t] == -1) {
31             dis[e.t] = dis[u] + 1;
32             que.push(e.t);
33         }
34     }
35 }
36 return dis[t] != -1;
37 }
38 int dfs(int u, int cur) {
39     if(u == t) return cur;
40     for(int &i = iter[u]; i < (int)G[u].size(); ++i) {
41         auto& e = G[u][i];
42         if(e.c > 0 && dis[u] + 1 == dis[e.t]) {
43             int ans = dfs(e.t, min(cur, e.c));
44             if(ans > 0) {
45                 G[e.t][e.r].c += ans;
46                 e.c -= ans;
47                 return ans;
48             }
49         }
50     }
51     return 0;
52 }
53
54 int flow(int a, int b) {
55     s = a, t = b;
56     int ans = 0;
57     while(bfs()) {
58         fill(ALL(iter), 0);
59         int tmp;
60         while((tmp = dfs(s, INF)) > 0)
61             ans += tmp;
62     }
63     return ans;
64 }
65 };

```

12.2 ISAP

```

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

```

12.3 MCMF

```

1 // Author: Unknown
2 // Function: First Maximize flow, then minimize flow cost
3 struct MCMF {
4     struct Edge {
5         int to, cap, rev;
6         ll cost;
7         Edge() {}
8         Edge(int _to, int _cap, int _rev, ll _cost) :
9             to(_to), cap(_cap), rev(_rev), cost(_cost) {}
10    };
11    static const int N = 2000;
12    vector<Edge> G[N];
13    int n, s, t;
14    void init(int _n, int _s, int _t) {
15        n = _n, s = _s, t = _t;
16        for(int i = 0; i <= n; ++i)
17            G[i].clear();
18    }
19    void add_edge(int from, int to, int cap, ll cost) {
20        G[from].eb(to, cap, (int)G[to].size(), cost);
21        G[to].eb(from, 0, (int)G[from].size() - 1, -cost);
22    }
23
24    bool vis[N];
25    int iter[N];
26    ll dis[N];
27    bool SPFA() {
28        for(int i = 0; i <= n; ++i)
29            vis[i] = 0, dis[i] = LINF;
30
31        dis[s] = 0; vis[s] = 1;
32        queue<int> que; que.push(s);
33        while(!que.empty()) {
34            int u = que.front(); que.pop();
35            vis[u] = 0;
36            for(auto& e : G[u]) if(e.cap > 0 && dis[e.to] > dis[u] + e.cost) {
37                dis[e.to] = dis[u] + e.cost;
38                if(!vis[e.to]) {
39                    que.push(e.to);
40                    vis[e.to] = 1;
41                }
42            }
43        }
44        return dis[t] != LINF;
45    }
46
47    int dfs(int u, int cur) {
48        if(u == t) return cur;
49        int ret = 0; vis[u] = 1;
50        for(int &i = iter[u]; i < (int)G[u].size(); ++i) {
51            auto &e = G[u][i];
52            if(e.cap > 0 && dis[e.to] == dis[u] + e.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        }
64        vis[u] = 0;
65        return ret;
66    }
67
68    pair<int, ll> flow() {
69        int flow = 0; ll cost = 0;
70        while(SPFA()) {
71            memset(iter, 0, sizeof(iter));
72            int tmp = dfs(s, INF);
73            flow += tmp, cost += tmp * dis[t];
74        }
75        return {flow, cost};
76    }

```



```
75 }
76 };
```

12.4 Hopcroft-Karp

```
1 // Author: Gino
2 // Function: Max Bipartite Matching in  $O(V \sqrt{E})$ 
3 // init() -> get() -> Ans = hk.MXCNT
4 struct HopcroftKarp {
5     // id:  $X = [1, nx], Y = [nx+1, nx+ny]$ 
6     int n, nx, ny, m, MXCNT;
7     vector<vector<int>> g;
8     vector<int> mx, my, dis, vis;
9     void init(int nnx, int nny, int mm) {
10         nx = nnx, ny = nny, m = mm;
11         n = nx + ny + 1;
12         g.clear(); g.resize(n);
13     }
14     void add(int x, int y) {
15         g[x].emplace_back(y);
16         g[y].emplace_back(x);
17     }
18     bool dfs(int x) {
19         vis[x] = true;
20         for (auto& y : g[x]) {
21             int px = my[y];
22             if (px == -1 ||
23                 (dis[px] == dis[x]+1 &&
24                  !vis[px] && dfs(px))) {
25                 mx[x] = y;
26                 my[y] = x;
27                 return true;
28             }
29         }
30         return false;
31     }
32     void get() {
33         mx.clear(); mx.resize(n, -1);
34         my.clear(); my.resize(n, -1);
35
36         while (true) {
37             queue<int> q;
38             dis.clear(); dis.resize(n, -1);
39             for (int x = 1; x <= nx; x++){
40                 if (mx[x] == -1) {
41                     dis[x] = 0;
42                     q.push(x);
43                 }
44             }
45             while (!q.empty()) {
46                 int x = q.front(); q.pop();
47                 for (auto& y : g[x]) {
48                     if (my[y] != -1 && dis[my[y]] ==
49                         -1) {
50                         dis[my[y]] = dis[x] + 1;
51                         q.push(my[y]);
52                     }
53                 }
54             }
55             bool brk = true;
56             vis.clear(); vis.resize(n, 0);
57             for (int x = 1; x <= nx; x++)
58                 if (mx[x] == -1 && dfs(x))
59                     brk = false;
60
61             if (brk) break;
62         }
63         MXCNT = 0;
64         for (int x = 1; x <= nx; x++) if (mx[x] != -1)
65             MXCNT++;
66     } 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 // Author: Unknown
2 // Function: Weighted Max Bipartite Matching in  $O(V^3)$ 
3 #include <bits/stdc++.h>
4
5 using namespace std;
6 const int inf = 1e9;
7
8 struct KuhnMunkres {
9     int n;
10    vector<vector<int>> g;
11    vector<int> lx, ly, slack;
12    vector<int> match, visx, visy;
13    KuhnMunkres(int n) : n(n), g(n, vector<int>(n)),
14        lx(n), ly(n), slack(n), match(n), visx(n), visy
15        (n)) {}
16    vector<int> & operator[](int i) { return g[i]; }
17    bool dfs(int i, bool aug) { // aug = true 表示要更新 match
18        if (visx[i]) return false;
19        visx[i] = true;
20        for (int j = 0; j < n; j++) {
21            if (visy[j]) continue;
22            // 一邊擴增交錯樹、尋找增廣路徑
23            // 一邊更新 slack: 樹上的點跟樹外的點所造成的最小權重
24            int d = lx[i] + ly[j] - g[i][j];
25            if (d == 0) {
26                visy[j] = true;
27                if (match[j] == -1 || dfs(match[j], aug))
28                    match[j] = i;
29                return true;
30            }
31            else {
32                slack[j] = min(slack[j], d);
33            }
34        }
35        return false;
36    }
37    bool augment() { // 回傳是否有增廣路
38        for (int j = 0; j < n; j++) if (!visy[j] && slack[j] == 0) {
39            visy[j] = true;
40            if (match[j] == -1 || dfs(match[j], false))
41                return true;
42        }
43        return false;
44    }
45    void relabel() {
46        int delta = inf;
47        for (int j = 0; j < n; j++) if (!visy[j]) delta =
48            min(delta, slack[j]);
49        for (int i = 0; i < n; i++) if (visx[i]) lx[i] -=
50            delta;
51        for (int j = 0; j < n; j++) {
52            if (visy[j]) ly[j] += delta;
53            else slack[j] -= delta;
54        }
55    }
56    int solve() {
57        for (int i = 0; i < n; i++) {
```

```

57     lx[i] = 0;
58     for(int j = 0; j < n; j++) lx[i] = max(lx[i]
59         ], g[i][j]);
60 }
61 fill(ly.begin(), ly.end(), 0);
62 fill(match.begin(), match.end(), -1);
63 for(int i = 0; i < n; i++) {
64     // slack 在每一輪都要初始化
65     fill(slack.begin(), slack.end(), inf);
66     fill(visx.begin(), visx.end(), false);
67     fill(visy.begin(), visy.end(), false);
68     if(dfs(i, true)) continue;
69     // 重複調整頂標直到找到增廣路徑
70     while(!augment()) relabel();
71     fill(visx.begin(), visx.end(), false);
72     fill(visy.begin(), visy.end(), false);
73     dfs(i, true);
74 }
75 int ans = 0;
76 for(int j = 0; j < n; j++) if(match[j] != -1)
77     ans += g[match[j]][j];
78 return ans;
79 }
80 };
81 signed main() {
82     ios_base::sync_with_stdio(0), cin.tie(0);
83     int n;
84     while(cin >> n && n) {
85         KuhnMunkres KM(n);
86         for(int i = 0; i < n; i++) {
87             for(int j = 0; j < n; j++) {
88                 int c;
89                 cin >> c;
90                 if(c > 0)
91                     KM[i][j] = c;
92             }
93         }
94         cout << KM.solve() << '\n';
95     }
96 }

```

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 Bertrand's Ballot Theorem

- A always $> B$: $C(p+q, p) - 2C(p+q-1, p)$
- A always $\geq B$: $C(p+q, p) \times \frac{p+1-q}{p+1}$

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