

NCTU_Oimo

Contents

1 Basic	1	8 Other	16
1.1 Vimrc	1	8.1 CYK	16
2 DataStructure	1	8.2 DP-optimization	17
2.1 KDTree	1	8.3 DigitCounting	17
2.2 SparseTable	2	8.4 Dp1D1D	17
2.3 Treap	2	8.5 ManhattanMST	17
2.4 Link Cut Tree	3	8.6 Count Spanning Tree	18
2.5 Pb Ds Heap	4	9 String	19
2.6 Pb Ds Rbtree	4	9.1 AC	19
3 Flow	4	9.2 BWT	19
3.1 Dinic	4	9.3 KMP	20
3.2 Gomory Hu	5	9.4 PalindromicTree	20
3.3 Min Cost Flow	5	9.5 SAM	20
3.4 SW-mincut	6	9.6 Z-value	21
4 Geometry	6	9.7 Smallest Rotation	21
4.1 2Dpoint	6	9.8 Suffix Array	21
4.2 ConvexHull	6	10 無權邊的生成樹個數 Kirchhoff's Theorem	21
4.3 Intersection Of Two Circle	6	11 monge	22
4.4 Intersection Of Two Lines	6	12 四心	22
4.5 Smallest Circle	7	13 Runge-Kutta	22
4.6 Circumcentre	7	14 Householder Matrix	22
4.7 Half Plane Intersection	7	1 Basic	
5 Graph	7	1.1 Vimrc	
5.1 BCC Edge	7	<code>set mouse=a</code>	
5.2 Dijkstra	8	<code>set t_Co=256</code>	
5.3 Directed MST	8	<code>set cursorline</code>	
5.4 LCA	8	<code>syntax on</code>	
5.5 MaximumClique	8	<code>syntax enable</code>	
5.6 Min Mean Cycle	9	<code>set number</code>	
5.7 MinimumSteinerTree	9	<code>map <F9> : !g++ % -o %<.out <CR></code>	
5.8 Tarjan	10	<code>map <F5> : !./%<.out < %<.in <CR></code>	
5.9 TwoSAT	10	<code>set bg=light</code>	
6 Matching	10	<code>set shiftwidth=2</code>	
6.1 KM	10	<code>set tabstop=2</code>	
6.2 Maximum General Matching	11	<code>set ai</code>	
6.3 Minimum General Weighted Matching	11	<code>set nu</code>	
6.4 Stable Marriage	12	<code>set ruler</code>	
7 Math	12	<code>set incsearch</code>	
7.1 FFT	12	<code>filetype indent on</code>	
7.2 GaussElimination	13	<code>hi Comment ctermfg=darkcyan</code>	
7.3 Karatsuba	13	2 DataStructure	
7.4 LinearPrime	13	2.1 2DSegmentTree	
7.5 Miller-Rabin	13	<code>//Not tested yet</code>	
7.6 Mobius	14	<code>const int MAXN = 1005;</code>	
7.7 Simplex	14	<code>int n, m;</code>	
7.8 Sprague-Grundy	15	<code>struct node {</code>	
7.9 Ax+by=gcd	15	<code>int num;</code>	
7.10 PollardRho	15	<code>} a[MAXN][MAXN];</code>	
7.11 Theorem	15	<code>struct segTree2D {</code>	
		<code>node tree[4 * MAXN][4 * MAXN];</code>	
		<code>void build() { build_x(1, 1, n); }</code>	
		<code>void update_node_x(int index_x, int index_y) {</code>	
		<code>tree[index_x][index_y].num =</code>	

```

        tree[index_x * 2][index_y].num + tree[
            index_x * 2 + 1][index_y].num;
    }
    void update_node_y(int index_x, int index_y) {
        tree[index_x][index_y].num =
            tree[index_x][index_y * 2].num + tree[
                index_x][index_y * 2 + 1].num;
    }
    void build_x(int vx, int lx, int rx) {
        if (lx != rx) {
            int mx = (lx + rx) / 2;
            build_x(vx * 2, lx, mx);
            build_x(vx * 2 + 1, mx + 1, rx);
        }
        build_y(vx, lx, rx, 1, 1, m);
    }
    void build_y(int vx, int lx, int rx, int vy, int ly
        , int ry) {
        if (ly == ry) {
            if (lx == rx)
                tree[vx][vy] = a[lx][ly];
            else
                update_node_x(vx, vy);
        } else {
            int my = (ly + ry) / 2;
            build_y(vx, lx, rx, vy * 2, ly, my);
            build_y(vx, lx, rx, vy * 2 + 1, my + 1, ry);
            ;
            update_node_y(vx, vy);
        }
    }
    void update(int x, int y){
        //Add modify the corresponding a first.
        update_x(1,1,n,x,y);
    }
    void update_x(int vx, int lx, int rx, int x, int y)
    {
        if (lx != rx) {
            int mx = (lx + rx) / 2;
            if (x <= mx)
                update_x(vx * 2, lx, mx, x, y);
            else
                update_x(vx * 2 + 1, mx + 1, rx, x, y);
        }
        update_y(vx, lx, rx, 1, 1, m, x, y);
    }
    void update_y(int vx, int lx, int rx, int vy, int
        ly, int ry, int x, int y) {
        if (ly == ry) {
            if (lx == rx)
                tree[vx][vy] = a[lx][ly];
            else
                update_node_x(vx, vy);
        } else {
            int my = (ly + ry) / 2;
            if (y <= my)
                update_y(vx, lx, rx, vy * 2, ly, my, x,
                    y);
            else
                update_y(vx, lx, rx, vy * 2 + 1, my +
                    1, ry, x, y);
            update_node_y(vx, vy);
        }
    }
    int ans_node(int vx, int vy){
        return tree[vx][vy].num;
    }
    int ans(int lx, int rx, int ly, int ry){
        return ans_x(1, 1,n,lx,rx,ly,ry);
    }
    int ans_x(int vx, int tlx, int trx, int lx, int rx,
        int ly, int ry){
        if (lx > rx){
            return 0;
        }
        if (lx == tlx && rx == trx){
            return ans_y(vx, 1, 1,m,ly,ry);

```

```

        }
        int tmx = (tlx+trx)/2;
        return ans_x(vx*2, tlx, tmx, lx, min(rx,tmx),
            ly, ry)+
            ans_x(vx*2+1, tmx+1, trx, max(lx,tmx+1), rx
                , ly, ry);
    }
    int ans_y(int vx, int vy, int tly, int try_, int ly
        , int ry){
        if (ly>ry)
            return 0;
        if (ly == tly&&ry == try_){
            return ans_node(vx, vy);
        }
        int tmy = (tly+try_)/2;
        return ans_y(vx,vy*2,tly, tmy,ly, min(ry,tmy))+
            ans_y(vx,vy*2+1,tmy+1, try_,max(ly,tmy+1),ry);
    }
} T;

```

2.2 KDTree

```

// from BCW
const int MXN = 100005;

struct KDTree {
    struct Node {
        int x,y,x1,y1,x2,y2;
        int id,f;
        Node *L, *R;
    }tree[MXN];
    int n;
    Node *root;

    long long dis2(int x1, int y1, int x2, int y2) {
        long long dx = x1-x2;
        long long dy = y1-y2;
        return dx*dx+dy*dy;
    }
    static bool cmpx(Node& a, Node& b){ return a.x<b.x; }
    static bool cmpy(Node& a, Node& b){ return a.y<b.y; }
    void init(vector<pair<int,int>> ip) {
        n = ip.size();
        for (int i=0; i<n; i++) {
            tree[i].id = i;
            tree[i].x = ip[i].first;
            tree[i].y = ip[i].second;
        }
        root = build_tree(0, n-1, 0);
    }
    Node* build_tree(int L, int R, int dep) {
        if (L>R) return nullptr;
        int M = (L+R)/2;
        tree[M].f = dep%2;
        nth_element(tree+L, tree+M, tree+R+1, tree[M].f ?
            cmpy : cmpx);
        tree[M].x1 = tree[M].x2 = tree[M].x;
        tree[M].y1 = tree[M].y2 = tree[M].y;

        tree[M].L = build_tree(L, M-1, dep+1);
        if (tree[M].L) {
            tree[M].x1 = min(tree[M].x1, tree[M].L->x1);
            tree[M].x2 = max(tree[M].x2, tree[M].L->x2);
            tree[M].y1 = min(tree[M].y1, tree[M].L->y1);
            tree[M].y2 = max(tree[M].y2, tree[M].L->y2);
        }

        tree[M].R = build_tree(M+1, R, dep+1);
        if (tree[M].R) {
            tree[M].x1 = min(tree[M].x1, tree[M].R->x1);
            tree[M].x2 = max(tree[M].x2, tree[M].R->x2);
            tree[M].y1 = min(tree[M].y1, tree[M].R->y1);
            tree[M].y2 = max(tree[M].y2, tree[M].R->y2);
        }
    }

```

```

    return tree+M;
}
int touch(Node* r, int x, int y, long long d2){
    long long dis = sqrt(d2)+1;
    if (x<r->x1-dis || x>r->x2+dis || y<r->y1-dis || y>
        r->y2+dis)
        return 0;
    return 1;
}
void nearest(Node* r, int x, int y, int &mID, long
    long &md2) {
    if (!r || !touch(r, x, y, md2)) return;
    long long d2 = dis2(r->x, r->y, x, y);
    if (d2 < md2 || (d2 == md2 && mID < r->id)) {
        mID = r->id;
        md2 = d2;
    }
    // search order depends on split dim
    if ((r->f == 0 && x < r->x) ||
        (r->f == 1 && y < r->y)) {
        nearest(r->L, x, y, mID, md2);
        nearest(r->R, x, y, mID, md2);
    } else {
        nearest(r->R, x, y, mID, md2);
        nearest(r->L, x, y, mID, md2);
    }
}
int query(int x, int y) {
    int id = 1029384756;
    long long d2 = 102938475612345678LL;
    nearest(root, x, y, id, d2);
    return id;
}
}tree;

```

2.3 SparseTable

```

const int MAXN = 200005;
const int lgN = 20;

struct SP{ //sparse table
    int Sp[MAXN][lgN];
    function<int(int,int)> opt;
    void build(int n, int *a){ // 0 base
        for (int i=0 ;i<n; i++) Sp[i][0]=a[i];

        for (int h=1; h<lgN; h++){
            int len = 1<<(h-1), i=0;
            for (; i+len<n; i++)
                Sp[i][h] = opt( Sp[i][h-1] , Sp[i+len][h-1] );
            for (; i<n; i++)
                Sp[i][h] = Sp[i][h-1];
        }
    }
    int query(int l, int r){
        int h = __lg(r-l+1);
        int len = 1<<h;
        return opt( Sp[l][h] , Sp[r-len+1][h] );
    }
};

```

2.4 Treap

```

#include<bits/stdc++.h>
using namespace std;
template<class T,unsigned seed>class treap{
public:
    struct node{
        T data;
        int size;
        node *l,*r;
        node(T d){
            size=1;

```

```

        data=d;
        l=r=NULL;
    }
    inline void up(){
        size=1;
        if(l)size+=l->size;
        if(r)size+=r->size;
    }
    inline void down(){
    }
}*root;
inline int size(node *p){return p?p->size:0;}
inline bool ran(node *a,node *b){
    static unsigned x=seed;
    x=0xdefaced*x+1;
    unsigned all=size(a)+size(b);
    return (x%all+all)%all<size(a);
}
void clear(node *p){
    if(p)clear(p->l),clear(p->r),delete p,p=NULL;
}
~treap(){clear(root);}
void split(node *o,node *a,node *b,int k){
    if(!k)a=NULL,b=o;
    else if(size(o)==k)a=o,b=NULL;
    else{
        o->down();
        if(k<=size(o->l)){
            b=o;
            split(o->l,a,b->l,k);
            b->up();
        }else{
            a=o;
            split(o->r,a->r,b,k-size(o->l)-1);
            a->up();
        }
    }
}
void merge(node *o,node *a,node *b){
    if(!a||!b)o=a?a:b;
    else{
        if(ran(a,b)){
            a->down();
            o=a;
            merge(o->r,a->r,b);
        }else{
            b->down();
            o=b;
            merge(o->l,a,b->l);
        }
        o->up();
    }
}
void build(node *p,int l,int r,T *s){
    if(l>r)return;
    int mid=(l+r)>>1;
    p=new node(s[mid]);
    build(p->l,l,mid-1,s);
    build(p->r,mid+1,r,s);
    p->up();
}
inline int rank(T data){
    node *p=root;
    int cnt=0;
    while(p){
        if(data<=p->data)p=p->l;
        else cnt+=size(p->l)+1,p=p->r;
    }
    return cnt;
}
inline void insert(node *p,T data,int k){
    node *a,*b,*now;
    split(p,a,b,k);
    now=new node(data);
    merge(a,a,now);
    merge(p,a,b);
}

```

```

inline void remove(node *p, int k) {
    node *a, *b, *res, *die;
    split(p, a, res, k);
    if (res == NULL) return;
    split(res, die, b, 1);
    merge(a, a, b);
    if (size(a) > size(b)) p = a;
    else p = b;
    clear(die);
}
};
treap<T, 20141223>bst;
int main(){
    bst.remove(bst.root, bst.rank(E));
    bst.insert(bst.root, E, bst.rank(E));
}

```

2.5 Link Cut Tree

// from bcw codebook

```

const int MXN = 100005;
const int MEM = 100005;

struct Splay {
    static Splay nil, mem[MEM], *pmem;
    Splay *ch[2], *f;
    int val, rev, size;
    Splay () : val(-1), rev(0), size(0) {
        f = ch[0] = ch[1] = &nil;
    }
    Splay (int _val) : val(_val), rev(0), size(1) {
        f = ch[0] = ch[1] = &nil;
    }
    bool isr() {
        return f->ch[0] != this && f->ch[1] != this;
    }
    int dir() {
        return f->ch[0] == this ? 0 : 1;
    }
    void setCh(Splay *c, int d) {
        ch[d] = c;
        if (c != &nil) c->f = this;
        pull();
    }
    void push() {
        if (rev) {
            swap(ch[0], ch[1]);
            if (ch[0] != &nil) ch[0]->rev ^= 1;
            if (ch[1] != &nil) ch[1]->rev ^= 1;
            rev=0;
        }
    }
    void pull() {
        size = ch[0]->size + ch[1]->size + 1;
        if (ch[0] != &nil) ch[0]->f = this;
        if (ch[1] != &nil) ch[1]->f = this;
    }
} Splay::nil, Splay::mem[MEM], *Splay::pmem = Splay::mem;
Splay *nil = &Splay::nil;

void rotate(Splay *x) {
    Splay *p = x->f;
    int d = x->dir();
    if (!p->isr()) p->f->setCh(x, p->dir());
    else x->f = p->f;
    p->setCh(x->ch[!d], d);
    x->setCh(p, !d);
    p->pull(); x->pull();
}

vector<Splay*> splayVec;
void splay(Splay *x) {
    splayVec.clear();
    for (Splay *q=x;; q=q->f) {

```

```

        splayVec.push_back(q);
        if (q->isr()) break;
    }
    reverse(begin(splayVec), end(splayVec));
    for (auto it : splayVec) it->push();
    while (!x->isr()) {
        if (x->f->isr()) rotate(x);
        else if (x->dir()==x->f->dir()) rotate(x->f), rotate(x);
        else rotate(x), rotate(x);
    }
}

Splay* access(Splay *x) {
    Splay *q = nil;
    for (;x!=nil;x=x->f) {
        splay(x);
        x->setCh(q, 1);
        q = x;
    }
    return q;
}

void evert(Splay *x) {
    access(x);
    splay(x);
    x->rev ^= 1;
    x->push(); x->pull();
}

void link(Splay *x, Splay *y) {
    // evert(x);
    access(x);
    splay(x);
    evert(y);
    x->setCh(y, 1);
}

void cut(Splay *x, Splay *y) {
    // evert(x);
    access(y);
    splay(y);
    y->push();
    y->ch[0] = y->ch[0]->f = nil;
}

int N, Q;
Splay *vt[MXN];

int ask(Splay *x, Splay *y) {
    access(x);
    access(y);
    splay(x);
    int res = x->f->val;
    if (res == -1) res=x->val;
    return res;
}

int main(int argc, char** argv) {
    scanf("%d%d", &N, &Q);
    for (int i=1; i<=N; i++)
        vt[i] = new (Splay::pmem++) Splay(i);
    while (Q--) {
        char cmd[105];
        int u, v;
        scanf("%s", cmd);
        if (cmd[1] == 'i') {
            scanf("%d%d", &u, &v);
            link(vt[u], vt[v]);
        } else if (cmd[0] == 'c') {
            scanf("%d", &v);
            cut(vt[1], vt[v]);
        } else {
            scanf("%d%d", &u, &v);
            int res=ask(vt[u], vt[v]);
            printf("%d\n", res);
        }
    }

    return 0;
}

```

2.6 Pb Ds Heap

```
#include <bits/extc++.h>
typedef __gnu_pbds::priority_queue<int> heap_t;
heap_t a,b;
int main() {
    a.clear();b.clear();
    a.push(1);a.push(3);b.push(2);b.push(4);
    // merge two heap
    a.join(b);
    assert(a.top() == 4);
    assert(b.empty());
    return 0;
}
```

2.7 Pb Ds Rbtree

```
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
#define ordered_set tree<int, null_type, less<int>,
    rb_tree_tag, tree_order_statistics_node_update>
int main() {
    ordered_set o_set;
    o_set.insert(5);
    o_set.insert(1);
    o_set.insert(2);
    cout << *(o_set.find_by_order(1)) << endl; // 2
    cout << o_set.order_of_key(4) << endl; // 2
    cout << o_set.order_of_key(5) << endl; // 2
    if (o_set.find(2) != o_set.end())
        o_set.erase(o_set.find(2));
    cout << *(o_set.find_by_order(1)) << endl; // 5
    cout << o_set.order_of_key(4) << endl; // 1
}
```

3 Flow

3.1 Dinic

- (a) Bounded Maxflow Construction:
1. add two node ss, tt
 2. add_edge(ss, tt, INF)
 3. **for** each edge u -> v with capacity [l, r]:
 add_edge(u, tt, l)
 add_edge(ss, v, l)
 add_edge(u, v, r-l)
 4. see (b), check **if** it is possible.
 5. answer is maxflow(ss, tt) + maxflow(s, t)
-
- (b) Bounded Possible Flow:
1. same construction method as (a)
 2. run maxflow(ss, tt)
 3. **for** every edge connected with ss **or** tt:
 rule: check **if** their rest flow is exactly 0
 4. answer is possible **if** every edge **do** satisfy the rule
 5. otherwise, it is NOT possible.
-
- (c) Bounded Minimum Flow:
1. same construction method as (a)
 2. answer is maxflow(ss, tt)
-
- (d) Bounded Minimum Cost Flow:
- * the concept is somewhat like bounded possible flow.
1. same construction method as (a)
 2. answer is maxflow(ss, tt) + (\sum l * cost **for** every edge)
-
- (e) Minimum Cut:
1. run maxflow(s, t)
 2. run cut(s)

3. ss[i] = 1: node i is at the same side with s.

```
-----
const long long INF = 1LL<<60;
struct Dinic { //O(VVE), with minimum cut
    static const int MAXN = 5003;
    struct Edge{
        int u, v;
        long long cap, rest;
    };

    int n, m, s, t, d[MAXN], cur[MAXN];
    vector<Edge> edges;
    vector<int> G[MAXN];

    void init(){
        edges.clear();
        for ( int i = 0 ; i < n ; i++ ) G[i].clear();
        n = 0;

        // min cut start
        bool side[MAXN];
        void cut(int u) {
            side[u] = 1;
            for ( int i : G[u] ) {
                if ( !side[ edges[i].v ] && edges[i].rest )
                    cut(edges[i].v);
            }
        }
        // min cut end

        int add_node(){
            return n++;
        }

        void add_edge(int u, int v, long long cap){
            edges.push_back( {u, v, cap, cap} );
            edges.push_back( {v, u, 0, 0LL} );
            m = edges.size();
            G[u].push_back(m-2);
            G[v].push_back(m-1);
        }

        bool bfs(){
            fill(d,d+n,-1);
            queue<int> que;
            que.push(s); d[s]=0;
            while (!que.empty()){
                int u = que.front(); que.pop();
                for (int ei : G[u]){
                    Edge &e = edges[ei];
                    if (d[e.v] < 0 && e.rest > 0){
                        d[e.v] = d[u] + 1;
                        que.push(e.v);
                    }
                }
            }
            return d[t] >= 0;
        }

        long long dfs(int u, long long a){
            if ( u == t || a == 0 ) return a;
            long long flow = 0, f;
            for ( int &i=cur[u]; i < (int)G[u].size() ; i++ ) {
                Edge &e = edges[ G[u][i] ];
                if ( d[u] + 1 != d[e.v] ) continue;
                f = dfs(e.v, min(a, e.rest) );
                if ( f > 0 ) {
                    e.rest -= f;
                    edges[ G[u][i]^1 ].rest += f;
                    flow += f;
                    a -= f;
                    if ( a == 0 ) break;
                }
            }
            return flow;
        }
    };
}
```

```

        return flow;
    }

    long long maxflow(int _s, int _t){
        s = _s, t = _t;
        long long flow = 0, mf;
        while ( bfs() ){
            fill(cur, cur+n, 0);
            while ( (mf = dfs(s, INF)) ) flow += mf;
        }
        return flow;
    }
} dinic;

```

3.2 Gomory Hu

Construct of Gomory Hu Tree

1. make sure the whole graph is clear
2. set node 0 as root, also be the parent of other nodes.
3. for every node $i > 0$, we run maxflow from i to $parent[i]$
4. hence we know the weight between i and $parent[i]$
5. for each node $j > i$, if j is at the same side with i , make the parent of j as i

```

int e[MAXN][MAXN];
int p[MAXN];

Dinic D; // original graph

void gomory_hu() {
    fill(p, p+n, 0);
    fill(e[0], e[n], INF);
    for ( int s = 1 ; s < n ; s++ ) {
        int t = p[s];
        Dinic F = D;
        int tmp = F.max_flow(s, t);

        for ( int i = 1 ; i < s ; i++ )
            e[s][i] = e[i][s] = min(tmp, e[t][i]);

        for ( int i = s+1 ; i <= n ; i++ )
            if ( p[i] == t && F.side[i] ) p[i] = s;
    }
}

```

3.3 Min Cost Flow

```

// long long version
typedef pair<long long, long long> pll;
struct CostFlow {
    static const int MAXN = 350;
    static const long long INF = 1LL<<60;
    struct Edge {
        int to, r;
        long long rest, c;
    };
    int n, pre[MAXN], preL[MAXN]; bool inq[MAXN];
    long long dis[MAXN], fl, cost;
    vector<Edge> G[MAXN];
    void init() {
        for ( int i = 0 ; i < MAXN ; i++ ) G[i].clear();
    }
    void add_edge(int u, int v, long long rest, long long c) {
        G[u].push_back({v, (int)G[v].size(), rest, c});
        G[v].push_back({u, (int)G[u].size()-1, 0, -c});
    }
}

```

```

pll flow(int s, int t) {
    fl = cost = 0;
    while (true) {
        fill(dis, dis+MAXN, INF);
        fill(inq, inq+MAXN, 0);
        dis[s] = 0;
        queue<int> que;
        que.push(s);
        while ( !que.empty() ) {
            int u = que.front(); que.pop();
            inq[u] = 0;
            for ( int i = 0 ; i < (int)G[u].size() ; i++ ) {
                int v = G[u][i].to;
                long long w = G[u][i].c;
                if ( G[u][i].rest > 0 && dis[v] >
                    dis[u] + w ) {
                    pre[v] = u; preL[v] = i;
                    dis[v] = dis[u] + w;
                    if (!inq[v]) {
                        inq[v] = 1;
                        que.push(v);
                    }
                }
            }
        }

        if (dis[t] == INF) break;
        long long tf = INF;
        for (int v = t, u, l ; v != s ; v = u ) {
            u = pre[v]; l = preL[v];
            tf = min(tf, G[u][l].rest);
        }
        for (int v = t, u, l ; v != s ; v = u ) {
            u = pre[v]; l = preL[v];
            G[u][l].rest -= tf;
            G[v][G[u][l].r].rest += tf;
        }
        cost += tf * dis[t];
        fl += tf;
    }
    return {fl, cost};
} flow;

```

3.4 SW-mincut

```

// all pair min cut
// global min cut
struct SW{ // O(V^3)
    static const int MXN = 514;
    int n, vst[MXN], del[MXN];
    int edge[MXN][MXN], wei[MXN];
    void init(int _n){
        n = _n; FZ(edge); FZ(del);
    }
    void addEdge(int u, int v, int w){
        edge[u][v] += w; edge[v][u] += w;
    }
    void search(int &s, int &t){
        FZ(vst); FZ(wei);
        s = t = -1;
        while (true){
            int mx=-1, cur=0;
            for (int i=0; i<n; i++){
                if (!del[i] && !vst[i] && mx<wei[i])
                    cur = i, mx = wei[i];
            }
            if (mx == -1) break;
            vst[cur] = 1;
            s = t; t = cur;
            for (int i=0; i<n; i++){
                if (!vst[i] && !del[i]) wei[i] += edge[cur][i];
            }
        }
    }
    int solve(){
        int res = 2147483647;
    }
}

```

```

    for (int i=0,x,y; i<n-1; i++){
        search(x,y);
        res = min(res,wei[y]);
        del[y] = 1;
        for (int j=0; j<n; j++){
            edge[x][j] = (edge[j][x] += edge[y][j]);
        }
        return res;
    }
}graph;

```

4 Geometry

4.1 2Dpoint

```

typedef double Double;
struct Point {
    Double x,y;

    bool operator < (const Point &b)const{
        //return tie(x,y) < tie(b.x,b.y);
        //return atan2(y,x) < atan2(b.y,b.x);
        assert(0 && "choose compare");
    }
    Point operator + (const Point &b)const{
        return (Point){x+b.x,y+b.y};
    }
    Point operator - (const Point &b)const{
        return (Point){x-b.x,y-b.y};
    }
    Point operator * (const Double &d)const{
        return Point(d*x,d*y);
    }
    Double operator * (const Point &b)const{
        return x*b.x + y*b.y;
    }
    Double operator % (const Point &b)const{
        return x*b.y - y*b.x;
    }
    friend Double abs2(const Point &p){
        return p.x*p.x + p.y*p.y;
    }
    friend Double abs(const Point &p){
        return sqrt( abs2(p) );
    }
};
typedef Point Vector;

struct Line{
    Point P; Vector v;
    bool operator < (const Line &b)const{
        return atan2(v.y,v.x) < atan2(b.v.y,b.v.x);
    }
};

```

4.2 ConvexHull

```

#include "2Dpoint.cpp"

// return H, 第一個點會在 H 出現兩次
void ConvexHull(vector<Point> &P, vector<Point> &H){
    int n = P.size(), m=0;
    sort(P.begin(),P.end());
    H.clear();

    for (int i=0; i<n; i++){
        while (m>=2 && (P[i]-H[m-2]) % (H[m-1]-H[m-2])
            <0)H.pop_back(), m--;
        H.push_back(P[i]), m++;
    }

    for (int i=n-2; i>=0; i--){

```

```

        while (m>=2 && (P[i]-H[m-2]) % (H[m-1]-H[m-2])
            <0)H.pop_back(), m--;
        H.push_back(P[i]), m++;
    }
}

```

4.3 Intersection Of Two Circle

```

vector<Double> interCircle(Double o1, Double r1, Double
    o2, Double r2) {
    Double d2 = abs2(o1 - o2);
    Double d = sqrt(d2);
    if (d < fabs(r1-r2) || r1+r2 < d) return {};
    Double u = 0.5*(o1+o2) + ((r2*r2-r1*r1)/(2.0*d2))*(o1
        -o2);
    Double A = sqrt((r1+r2+d) * (r1-r2+d) * (r1+r2-d) *
        (-r1+r2+d));
    Double v = A / (2.0*d2) * Double(o1.S-o2.S, -o1.F+o2.
        F);
    return {u+v, u-v};
}

```

4.4 Intersection Of Two Lines

```

Point interPnt(Point p1, Point p2, Point q1, Point q2,
    bool &res){
    Double f1 = cross(p2, q1, p1);
    Double f2 = -cross(p2, q2, p1);
    Double f = (f1 + f2);

    if(fabs(f) < EPS) {
        res = false;
        return {};
    }

    res = true;
    return (f2 / f) * q1 + (f1 / f) * q2;
}

```

4.5 Smallest Circle

```

#include "circumcentre.cpp"
pair<Point,Double> SmallestCircle(int n, Point _p[]){
    Point *p = new Point[n];
    memcpy(p,_p,sizeof(Point)*n);
    random_shuffle(p,p+n);

    Double r2=0;
    Point cen;
    for (int i=0; i<n; i++){
        if ( abs2(cen-p[i]) <= r2)continue;
        cen = p[i], r2=0;
        for (int j=0; j<i; j++){
            if ( abs2(cen-p[j]) <= r2)continue;
            cen = (p[i]+p[j])*0.5;
            r2 = abs2(cen-p[i]);
            for (int k=0; k<j; k++){
                if ( abs2(cen-p[k]) <= r2)continue;
                cen = circumcentre(p[i],p[j],p[k]);
                r2 = abs2(cen-p[k]);
            }
        }
    }

    delete[] p;
    return {cen,r2};
}

// auto res = SmallestCircle(,);

```

4.6 Circumcentre

```
#include "2Dpoint.cpp"

Point circumcentre(Point &p0, Point &p1, Point &p2){
    Point a = p1-p0;
    Point b = p2-p0;
    Double c1 = abs2(a)*0.5;
    Double c2 = abs2(b)*0.5;
    Double d = a % b;
    Double x = p0.x + ( c1*b.y - c2*a.y ) / d;
    Double y = p0.y + ( c2*a.x - c1*b.x ) / d;
    return {x,y};
}
```

4.7 Half Plane Intersection

```
bool OnLeft(const Line& L,const Point& p){
    return Cross(L.v,p-L.P)>0;
}

Point GetIntersection(Line a,Line b){
    Vector u = a.P-b.P;
    Double t = Cross(b.v,u)/Cross(a.v,b.v);
    return a.P + a.v*t;
}

int HalfplaneIntersection(Line* L,int n,Point* poly){
    sort(L,L+n);

    int first,last;
    Point *p = new Point[n];
    Line *q = new Line[n];
    q[first=last=0] = L[0];
    for(int i=1;i<n;i++){
        while(first < last && !OnLeft(L[i],p[last-1])) last--;
        while(first < last && !OnLeft(L[i],p[first])) first++;
        q[++last]=L[i];
        if(fabs(Cross(q[last].v,q[last-1].v))<EPS){
            last--;
            if(OnLeft(q[last],L[i].P)) q[last]=L[i];
        }
        if(first < last) p[last-1]=GetIntersection(q[last-1],q[last]);
    }
    while(first<last && !OnLeft(q[first],p[last-1])) last--;
    if(last-first<=1) return 0;
    p[last]=GetIntersection(q[last],q[first]);

    int m=0;
    for(int i=first;i<=last;i++) poly[m++]=p[i];
    return m;
}
```

5 Graph

5.1 BCC Edge

邊雙連通

任意兩點間至少有兩條不重疊的路徑連接，找法：

1. 標記出所有的橋
2. 對全圖進行 DFS，不走橋，每一次 DFS 就是一個新的邊雙連通

// from BCW

```
struct BccEdge {
    static const int MXN = 100005;
    struct Edge { int v,eid; };
```

```
int n,m,step,par[MXN],dfn[MXN],low[MXN];
vector<Edge> E[MXN];
DisjointSet djs;
void init(int _n) {
    n = _n; m = 0;
    for (int i=0; i<n; i++) E[i].clear();
    djs.init(n);
}
void add_edge(int u, int v) {
    E[u].PB({v, m});
    E[v].PB({u, m});
    m++;
}
void DFS(int u, int f, int f_eid) {
    par[u] = f;
    dfn[u] = low[u] = step++;
    for (auto it:E[u]) {
        if (it.eid == f_eid) continue;
        int v = it.v;
        if (dfn[v] == -1) {
            DFS(v, u, it.eid);
            low[u] = min(low[u], low[v]);
        } else {
            low[u] = min(low[u], dfn[v]);
        }
    }
}
void solve() {
    step = 0;
    memset(dfn, -1, sizeof(int)*n);
    for (int i=0; i<n; i++) {
        if (dfn[i] == -1) DFS(i, i, -1);
    }
    djs.init(n);
    for (int i=0; i<n; i++) {
        if (low[i] < dfn[i]) djs.uni(i, par[i]);
    }
}
}graph;
```

5.2 Dijkstra

```
from heapq import *
INF = 2*10**10000
t = input()
for pp in range(t):
    n, m = map(int, raw_input().split())
    g, d, q = [[[] for _ in range(n+1)], [0] + [INF] * n, [(0, 0)]]
    #for i in range(1, m):
    #    a[i], b[i], c[i], l[i], o[i] = map(int, input().split())
    for _ in range(m):
        u, v, c, l, o = map(int, raw_input().split())
        g[u] += [(o, v, c, l)]
    while q:
        u = heappop(q)[1]
        for e in g[u]:
            k = d[u] / e[2]
            if k < 0:
                k = 0
            else:
                k = k * e[3]
            t, v = d[u] + e[0] + k, e[1]
            if t < d[v]:
                d[v] = t
                heappush(q, (d[v], v))
    print(d[n])
```

5.3 Directed MST

```
template<typename T>
struct zhu_liu{
```



```

static const int MAXN=110,MAXM=10005;
struct node{
    int u,v;
    T w,tag;
    node *l,*r;
    node(int u=0,int v=0,T w=0):u(u),v(v),w(w),tag(0),l(0),r(0){}
    void down(){
        w+=tag;
        if(l)l->tag+=tag;
        if(r)r->tag+=tag;
        tag=0;
    }
}mem[MAXN];//靜態記憶體
node *pq[MAXN*2],*E[MAXN*2];
int st[MAXN*2],id[MAXN*2],m;
void init(int n){
    for(int i=1;i<=n;++i){
        pq[i]=E[i]=0;
        st[i]=id[i]=i;
    }m=0;
}
node *merge(node *a,node *b){//skew heap
    if(!a||!b)return a?a:b;
    a->down(),b->down();
    if(b->w<a->w)return merge(b,a);
    swap(a->l,a->r);
    a->l=merge(b,a->l);
    return a;
}
void add_edge(int u,int v,T w){
    if(u!=v)pq[v]=merge(pq[v],&(mem[m++]=node(u,v,w)));
}
int find(int x,int *st){
    return st[x]==x?x:st[x]=find(st[x],st);
}
T build(int root,int n){
    T ans=0;int N=n,all=n;
    for(int i=1;i<=N;++i){
        if(i==root||!pq[i])continue;
        while(pq[i]){
            pq[i]->down(),E[i]=pq[i];
            pq[i]=merge(pq[i]->l,pq[i]->r);
            if(find(E[i]->u,id)!=find(i,id))break;
        }
        if(find(E[i]->u,id)==find(i,id))continue;
        ans+=E[i]->w;
        if(find(E[i]->u,st)==find(i,st)){
            if(pq[i])pq[i]->tag-=E[i]->w;
            pq[++N]=pq[i],id[N]=N;
            for(int u=find(E[i]->u,id);u!=i;u=find(E[u]->u,id)){
                if(pq[u])pq[u]->tag-=E[u]->w;
                id[find(u,id)]=N;
                pq[N]=merge(pq[N],pq[u]);
            }
            st[N]=find(i,st);
            id[find(i,id)]=N;
        }else st[find(i,st)]=find(E[i]->u,st),--all;
    }
    return all==1?ans:-INT_MAX;//圖不連通就無解
};

```

5.4 LCA

```

//lv紀錄深度
//father[多少層次][誰]
//已經建好每個人的父親是誰 (father[0][i]已經建好)
//已經建好深度 (lv[i]已經建好)
void makePP(){
    for(int i = 1; i < 20; i++){
        for(int j = 2; j <= n; j++){
            father[i][j]=father[i-1][ father[i-1][j] ];
        }
    }
}

```

```

}
}
int find(int a, int b){
    if(lv[a] < lv[b]) swap(a,b);
    int need = lv[a] - lv[b];
    for(int i = 0; need!=0; i++){
        if(need&1) a=father[i][a];
        need >>= 1;
    }
    for(int i = 19 ; i >= 0 ; i--){
        if(father[i][a] != father[i][b]){
            a=father[i][a];
            b=father[i][b];
        }
    }
    return a!=b?father[0][a] : a;
}

```

5.5 MaximumClique

```

const int MAXN = 105;
int best;
int m ,n;
int num[MAXN];
// int x[MAXN];
int path[MAXN];
int g[MAXN][MAXN];

bool dfs( int *adj, int total, int cnt ){
    int i, j, k;
    int t[MAXN];
    if( total == 0 ){
        if( best < cnt ){
            // for( i = 0; i < cnt; i++) path[i] = x[i];
            best = cnt; return true;
        }
        return false;
    }
    for( i = 0; i < total; i++){
        if( cnt+(total-i) <= best ) return false;
        if( cnt+num[adj[i]] <= best ) return false;
        // x[cnt] = adj[i];
        for( k = 0, j = i+1; j < total; j++ )
            if( g[ adj[i] ][ adj[j] ] )
                t[ k++ ] = adj[j];
        if( dfs( t, k, cnt+1 ) ) return true;
    }
    return false;
}

int MaximumClique(){
    int i, j, k;
    int adj[MAXN];
    if( n <= 0 ) return 0;
    best = 0;
    for( i = n-1; i >= 0; i-- ){
        // x[0] = i;
        for( k = 0, j = i+1; j < n; j++ )
            if( g[i][j] ) adj[k++] = j;
        dfs( adj, k, 1 );
        num[i] = best;
    }
    return best;
}

```

5.6 Min Mean Cycle

```

// from BCW
/* minimum mean cycle */
const int MAXE = 1805;
const int MAXN = 35;
const double inf = 1029384756;
const double eps = 1e-6;

```

```

struct Edge {
    int v,u;
    double c;
};
int n,m,prv[MAXN][MAXN], prve[MAXN][MAXN], vst[MAXN];
Edge e[MAXE];
vector<int> edgeID, cycle, rho;
double d[MAXN][MAXN];
inline void bellman_ford() {
    for(int i=0; i<n; i++) d[0][i]=0;
    for(int i=0; i<n; i++) {
        fill(d[i+1], d[i+1]+n, inf);
        for(int j=0; j<m; j++) {
            int v = e[j].v, u = e[j].u;
            if(d[i][v]<inf && d[i+1][u]>d[i][v]+e[j].c) {
                d[i+1][u] = d[i][v]+e[j].c;
                prv[i+1][u] = v;
                prve[i+1][u] = j;
            }
        }
    }
}
double karp_mmc() {
    // returns inf if no cycle, mmc otherwise
    double mmc=inf;
    int st = -1;
    bellman_ford();
    for(int i=0; i<n; i++) {
        double avg=-inf;
        for(int k=0; k<n; k++) {
            if(d[n][i]<inf-eps) avg=max(avg, (d[n][i]-d[k][i])/(n-k));
            else avg=max(avg, inf);
        }
        if (avg < mmc) tie(mmc, st) = tie(avg, i);
    }
    for(int i=0; i<n; i++) vst[i] = 0;
    edgeID.clear(); cycle.clear(); rho.clear();
    for (int i=n; !vst[st]; st=prv[i-1][st]) {
        vst[st]++;
        edgeID.PB(prve[i][st]);
        rho.PB(st);
    }
    while (vst[st] != 2) {
        int v = rho.back(); rho.pop_back();
        cycle.PB(v);
        vst[v]++;
    }
    reverse(ALL(edgeID));
    edgeID.resize(SZ(cycle));
    return mmc;
}

```

5.7 MinimumSteinerTree

```

// Minimum Steiner Tree
// O(V^3T + V^2 2^T)
struct SteinerTree{
#define V 33
#define T 8
#define INF 1023456789
    int n , dst[V][V] , dp[1<<T][V] , tdst[V];
    void init( int _n ){
        n = _n;
        for( int i = 0 ; i < n ; i ++ ){
            for( int j = 0 ; j < n ; j ++ )
                dst[ i ][ j ] = INF;
            dst[ i ][ i ] = 0;
        }
    }
    void add_edge( int ui , int vi , int wi ){
        dst[ ui ][ vi ] = min( dst[ ui ][ vi ] , wi );
        dst[ vi ][ ui ] = min( dst[ vi ][ ui ] , wi );
    }
    void shortest_path(){
        for( int k = 0 ; k < n ; k ++ )

```

```

        for( int i = 0 ; i < n ; i ++ )
            for( int j = 0 ; j < n ; j ++ )
                dst[ i ][ j ] = min( dst[ i ][ j ] ,
                    dst[ i ][ k ] + dst[ k ][ j ] );
    }
    int solve( const vector<int>& ter ){
        int t = (int)ter.size();
        for( int i = 0 ; i < ( 1<<t ) ; i ++ )
            for( int j = 0 ; j < n ; j ++ )
                dp[ i ][ j ] = INF;
        for( int i = 0 ; i < n ; i ++ )
            dp[ 0 ][ i ] = 0;
        for( int msk = 1 ; msk < ( 1<<t ) ; msk ++ ){
            if( msk == ( msk & (-msk) ) ){
                int who = __lg( msk );
                for( int i = 0 ; i < n ; i ++ )
                    dp[ msk ][ i ] = dst[ ter[ who ] ][ i ];
                continue;
            }
            for( int i = 0 ; i < n ; i ++ )
                for( int submsk = ( msk - 1 ) & msk ; submsk ;
                    submsk = ( submsk - 1 ) & msk )
                    dp[ msk ][ i ] = min( dp[ msk ][ i ] ,
                        dp[ submsk ][ i ] +
                        dp[ msk ^ submsk ][ i ] );
            for( int i = 0 ; i < n ; i ++ ){
                tdst[ i ] = INF;
                for( int j = 0 ; j < n ; j ++ )
                    tdst[ i ] = min( tdst[ i ] ,
                        dp[ msk ][ j ] + dst[ j ][ i ] );
            }
            for( int i = 0 ; i < n ; i ++ )
                dp[ msk ][ i ] = tdst[ i ];
        }
        int ans = INF;
        for( int i = 0 ; i < n ; i ++ )
            ans = min( ans , dp[ ( 1<<t ) - 1 ][ i ] );
        return ans;
    }
} solver;

```

5.8 Tarjan

割點

點 u 為割點 **if and only if** 滿足 1. **or** 2.

1. u 為樹根，且 u 有多於一個子樹。
2. u 不為樹根，且滿足存在 (u,v) 為樹枝邊（或稱父子邊，即 u 為 v 在搜索樹中的父親），使得 $DFN(u) \leq Low(v)$ 。

橋

一條無向邊 (u,v) 是橋 **if and only if** (u,v) 為樹枝邊，且滿足 $DFN(u) < Low(v)$ 。

```

// 0 base
struct TarjanSCC{
    static const int MAXN = 1000006;
    int n, dfn[MAXN], low[MAXN], scc[MAXN], scn, count;
    vector<int> G[MAXN];
    stack<int> stk;
    bool ins[MAXN];

    void tarjan(int u){
        dfn[u] = low[u] = ++count;
        stk.push(u);
        ins[u] = true;

        for(auto v:G[u]){
            if(!dfn[v]){
                tarjan(v);
                low[u] = min(low[u], low[v]);
            } else if(ins[v]){
                low[u] = min(low[u], dfn[v]);
            }
        }
    }
}

```

6 Matching

6.1 KM

```
#define MAXN 100
#define INF INT_MAX
int g[MAXN][MAXN], lx[MAXN], ly[MAXN], slack_y[MAXN];
int px[MAXN], py[MAXN], match_y[MAXN], par[MAXN];
int n;

void adjust(int y) { //把增廣路上所有邊反轉
    match_y[y] = py[y];
    if (px[match_y[y]] != -2)
        adjust(px[match_y[y]]);
}

bool dfs(int x) { //DFS找增廣路
    for (int y = 0; y < n; ++y) {
        if (py[y] != -1) continue;
        int t = lx[x] + ly[y] - g[x][y];
        if (t == 0) {
            py[y] = x;
            if (match_y[y] == -1) {
                adjust(y);
                return 1;
            }
            if (px[match_y[y]] != -1) continue;
            px[match_y[y]] = y;
            if (dfs(match_y[y])) return 1;
        } else if (slack_y[y] > t) {
            slack_y[y] = t;
            par[y] = x;
        }
    }
    return 0;
}

inline int km() {
    memset(ly, 0, sizeof(int) * n);
    memset(match_y, -1, sizeof(int) * n);
    for (int x = 0; x < n; ++x) {
        lx[x] = -INF;
        for (int y = 0; y < n; ++y) {
            lx[x] = max(lx[x], g[x][y]);
        }
    }
    for (int x = 0; x < n; ++x) {
        for (int y = 0; y < n; ++y) slack_y[y] = INF;
        memset(px, -1, sizeof(int) * n);
        memset(py, -1, sizeof(int) * n);
        px[x] = -2;
        if (dfs(x)) continue;
        bool flag = 1;
        while (flag) {
            int cut = INF;
            for (int y = 0; y < n; ++y)
                if (py[y] == -1 && cut > slack_y[y]) cut = slack_y[y];
            for (int j = 0; j < n; ++j) {
                if (px[j] != -1) lx[j] -= cut;
                if (py[j] != -1) ly[j] += cut;
                else slack_y[j] -= cut;
            }
            for (int y = 0; y < n; ++y) {
                if (py[y] == -1 && slack_y[y] == 0) {
                    py[y] = par[y];
                    if (match_y[y] == -1) {
                        adjust(y);
                        flag = 0;
                        break;
                    }
                }
                px[match_y[y]] = y;
                if (dfs(match_y[y])) {
                    flag = 0;
                    break;
                }
            }
        }
    }
}
```

}

```
}

if (dfn[u] == low[u]) {
    int v;
    do {
        v = stk.top();
        stk.pop();
        scc[v] = scn;
        ins[v] = false;
    } while (v != u);
    scn++;
}

}

void getSCC() {
    memset(dfn, 0, sizeof(dfn));
    memset(low, 0, sizeof(low));
    memset(ins, 0, sizeof(ins));
    memset(scc, 0, sizeof(scc));
    count = scn = 0;
    for (int i = 0; i < n; i++) {
        if (!dfn[i]) tarjan(i);
    }
}

}SCC;
```

5.9 TwoSAT

```
const int MAXN = 2020;

struct TwoSAT {
    static const int MAXv = 2 * MAXN;
    vector<int> GO[MAXN], BK[MAXN], stk;
    bool vis[MAXN];
    int SC[MAXN];

    void imply(int u, int v) { // u imply v
        GO[u].push_back(v);
        BK[v].push_back(u);
    }

    int dfs(int u, vector<int> *G, int sc) {
        vis[u] = 1, SC[u] = sc;
        for (int v : G[u]) if (!vis[v])
            dfs(v, G, sc);
        if (G == GO) stk.push_back(u);
    }

    int scc(int n = MAXv) {
        memset(vis, 0, sizeof(vis));
        for (int i = 0; i < n; i++) if (!vis[i])
            dfs(i, GO, -1);
        memset(vis, 0, sizeof(vis));
        int sc = 0;
        while (!stk.empty()) {
            if (!vis[stk.back()])
                dfs(stk.back(), BK, sc++);
            stk.pop_back();
        }
    }
}SAT;

int main() {
    SAT.scc(2 * n);
    bool ok = 1;
    for (int i = 0; i < n; i++) {
        if (SAT.SC[2 * i] == SAT.SC[2 * i + 1]) ok = 0;
    }
    if (ok) {
        for (int i = 0; i < n; i++) {
            if (SAT.SC[2 * i] > SAT.SC[2 * i + 1]) {
                cout << i << endl;
            }
        }
    }
    else puts("NO");
}
```

```

int ans=0;
for(int y=0;y<n;++y)if(g[match_y[y]][y]!=-INF)ans+=g[
    match_y[y]][y];
return ans;
}

```

6.2 Maximum General Matching

// Maximum Cardinality Matching

```

struct Graph {
    vector<int> G[MAXN];
    int pa[MAXN], match[MAXN], st[MAXN], S[MAXN], vis[
        MAXN];
    int t, n;

    void init(int _n) {
        n = _n;
        for ( int i = 1 ; i <= n ; i++ ) G[i].clear();
    }

    void add_edge(int u, int v) {
        G[u].push_back(v);
        G[v].push_back(u);
    }

    int lca(int u, int v){
        for ( ++t ; ; swap(u, v) ) {
            if ( u == 0 ) continue;
            if ( vis[u] == t ) return u;
            vis[u] = t;
            u = st[ pa[ match[u] ] ];
        }
    }

    void flower(int u, int v, int l, queue<int> &q) {
        while ( st[u] != 1 ) {
            pa[u] = v;
            if ( S[ v = match[u] ] == 1 ) {
                q.push(v);
                S[v] = 0;
            }
            st[u] = st[v] = 1;
            u = pa[v];
        }
    }

    bool bfs(int u){
        for ( int i = 1 ; i <= n ; i++ ) st[i] = i;
        memset(S, -1, sizeof(S));
        queue<int>q;
        q.push(u);
        S[u] = 0;
        while ( !q.empty() ) {
            u = q.front(); q.pop();
            for ( int i = 0 ; i < (int)G[u].size(); i++ ) {
                int v = G[u][i];
                if ( S[v] == -1 ) {
                    pa[v] = u;
                    S[v] = 1;
                    if ( !match[v] ) {
                        for ( int lst ; u ; v = lst, u = pa[v] ) {
                            lst = match[u];
                            match[u] = v;
                            match[v] = u;
                        }
                        return 1;
                    }
                    q.push(match[v]);
                    S[ match[v] ] = 0;
                } else if ( !S[v] && st[v] != st[u] ) {
                    int l = lca(st[v], st[u]);
                    flower(v, u, l, q);
                    flower(u, v, l, q);
                }
            }
        }
        return 0;
    }

    int solve(){

```

```

memset(pa, 0, sizeof(pa));
memset(match, 0, sizeof(match));
int ans = 0;
for ( int i = 1 ; i <= n ; i++ )
    if ( !match[i] && bfs(i) ) ans++;
return ans;
}
} graph;

```

6.3 Minimum General Weighted Matching

// Minimum Weight Perfect Matching (Perfect Match)

```

struct Graph {
    static const int MAXN = 105;
    int n, e[MAXN][MAXN];
    int match[MAXN], d[MAXN], onstk[MAXN];
    vector<int> stk;
    void init(int _n) {
        n = _n;
        for( int i = 0 ; i < n ; i++ )
            for( int j = 0 ; j < n ; j++ )
                e[i][j] = 0;
    }

    void add_edge(int u, int v, int w) {
        e[u][v] = e[v][u] = w;
    }

    bool SPFA(int u){
        if (onstk[u]) return true;
        stk.push_back(u);
        onstk[u] = 1;
        for ( int v = 0 ; v < n ; v++ ) {
            if (u != v && match[u] != v && !onstk[v] )
            {
                int m = match[v];
                if ( d[m] > d[u] - e[v][m] + e[u][v] )
                {
                    d[m] = d[u] - e[v][m] + e[u][v];
                    onstk[v] = 1;
                    stk.push_back(v);
                    if (SPFA(m)) return true;
                    stk.pop_back();
                    onstk[v] = 0;
                }
            }
        }
        onstk[u] = 0;
        stk.pop_back();
        return false;
    }

    int solve() {
        for ( int i = 0 ; i < n ; i += 2 ) {
            match[i] = i+1;
            match[i+1] = i;
        }
        while (true){
            int found = 0;
            for ( int i = 0 ; i < n ; i++ )
                onstk[ i ] = d[ i ] = 0;
            for ( int i = 0 ; i < n ; i++ ) {
                stk.clear();
                if ( !onstk[i] && SPFA(i) ) {
                    found = 1;
                    while ( stk.size() >= 2 ) {
                        int u = stk.back(); stk.
                            pop_back();
                        int v = stk.back(); stk.
                            pop_back();
                        match[u] = v;
                        match[v] = u;
                    }
                }
            }
            if (!found) break;
        }
        int ret = 0;

```

```

    for ( int i = 0 ; i < n ; i++ )
        ret += e[i][match[i]];
    ret /= 2;
    return ret;
}
} graph;

```

6.4 Stable Marriage

```

#define F(n) Fi(i, n)
#define Fi(i, n) Fl(i, 0, n)
#define Fl(i, l, n) for(int i = l ; i < n ; ++i)
#include <bits/stdc++.h>
using namespace std;
int D, quota[205], weight[205][5];
int S, scoretoDep[12005][205], score[5];
int P, prefer[12005][85], iter[12005];
int ans[12005];
typedef pair<int, int> PII;
map<int, int> samescore[205];
typedef priority_queue<PII, vector<PII>, greater<PII>>
    QQQ;
QQQ pri[205];
void check(int d) {
    PII t = pri[d].top();
    int v;
    if (pri[d].size() - samescore[d][t.first] + 1 <=
        quota[d]) return;
    while (pri[d].top().first == t.first) {
        v = pri[d].top().second;
        ans[v] = -1;
        --samescore[d][t.first];
        pri[d].pop();
    }
}
void push(int s, int d) {
    if (pri[d].size() < quota[d]) {
        pri[d].push(PII(scoretoDep[s][d], s));
        ans[s] = d;
        ++samescore[s][scoretoDep[s][d]];
    } else if (scoretoDep[s][d] >= pri[d].top().first) {
        pri[d].push(PII(scoretoDep[s][d], s));
        ans[s] = d;
        ++samescore[s][scoretoDep[s][d]];
        check(d);
    }
}
void f() {
    int over;
    while (true) {
        over = 1;
        Fi (q, S) {
            if (ans[q] != -1 || iter[q] >= P) continue;
            push(q, prefer[q][iter[q]++]);
            over = 0;
        }
        if (over) break;
    }
}
main() {
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    int sadmit, stof, dexceed, dfew;
    while (cin >> D, D) { // Beware of the input format
        // or judge may troll us.
        sadmit = stof = dexceed = dfew = 0;
        memset(iter, 0, sizeof(iter));
        memset(ans, 0, sizeof(ans));
        Fi (q, 205) {
            pri[q] = QQQ();
            samescore[q].clear();
        }
        cin >> S >> P;
        Fi (q, D) {
            cin >> quota[q];
            Fi (w, 5) cin >> weight[q][w];

```

```

        }
        Fi (q, S) {
            Fi (w, 5) cin >> score[w];
            Fi (w, D) {
                scoretoDep[q][w] = 0;
                F (5) scoretoDep[q][w] += weight[w][i] * score[
                    i];
            }
        }
        Fi (q, S) Fi (w, P) {
            cin >> prefer[q][w];
            --prefer[q][w];
        }
    }
    f();
    Fi (q, D) sadmit += pri[q].size();
    Fi (q, S) if (ans[q] == prefer[q][0]) ++stof;
    Fi (q, D) if (pri[q].size() > quota[q]) ++dexceed;
    Fi (q, D) if (pri[q].size() < quota[q]) ++dfew;
    cout << sadmit << ' ' << stof << ' ' << dexceed <<
        ' ' << dfew << '\n';
}
}

```

7 Math

7.1 FFT

```

// use llround() to avoid EPS
typedef double Double;
const Double PI = acos(-1);

// STL complex may TLE
typedef complex<Double> Complex;
#define x real()
#define y imag()

template<typename Iter> // Complex*
void BitReverse(Iter a, int n) {
    for (int i=1, j=0; i<n; i++) {
        for (int k = n>>1; k>(j^=k); k>>=1);
        if (i<j) swap(a[i], a[j]);
    }
}

template<typename Iter> // Complex*
void FFT(Iter a, int n, int rev=1) { // rev = 1 or -1
    assert( (n&(-n)) == n ); // n is power of 2
    BitReverse(a, n);
    Iter A = a;

    for (int s=1; (1<<s)<=n; s++) {
        int m = (1<<s);

        Complex wm( cos(2*PI*rev/m), sin(2*PI*rev/m) );
        for (int k=0; k<n; k+=m) {
            Complex w(1, 0);
            for (int j=0; j<(m>>1); j++) {
                Complex t = w * A[k+j+(m>>1)];
                Complex u = A[k+j];
                A[k+j] = u+t;
                A[k+j+(m>>1)] = u-t;
                w = w*wm;
            }
        }
    }

    if (rev== -1) {
        for (int i=0; i<n; i++) {
            A[i] /= n;
        }
    }
}

```

7.2 GaussElimination

```
// by bcw_codebook

const int MAXN = 300;
const double EPS = 1e-8;

int n;
double A[MAXN][MAXN];

void Gauss() {
    for(int i = 0; i < n; i++) {
        bool ok = 0;
        for(int j = i; j < n; j++) {
            if(fabs(A[j][i]) > EPS) {
                swap(A[j], A[i]);
                ok = 1;
                break;
            }
        }
        if(!ok) continue;

        double fs = A[i][i];
        for(int j = i+1; j < n; j++) {
            double r = A[j][i] / fs;
            for(int k = i; k < n; k++) {
                A[j][k] -= A[i][k] * r;
            }
        }
    }
}
```

7.3 Karatsuba

```
// N is power of 2
template<typename Iter>
void DC(int N, Iter tmp, Iter A, Iter B, Iter res){
    fill(res, res+2*N, 0);
    if (N<=32){
        for (int i=0; i<N; i++){
            for (int j=0; j<N; j++){
                res[i+j] += A[i]*B[j];
            }
        }
        return;
    }
    int n = N/2;
    auto a = A+n, b = A;
    auto c = B+n, d = B;
    DC(n, tmp+n, a, c, res+2*N);
    for (int i=0; i<N; i++){
        res[i+n] += res[2*N+i];
        res[i+n] -= res[2*N+i];
    }
    DC(n, tmp+n, b, d, res+2*N);
    for (int i=0; i<N; i++){
        res[i] += res[2*N+i];
        res[i+n] -= res[2*N+i];
    }

    auto x = tmp;
    auto y = tmp+n;
    for (int i=0; i<n; i++) x[i] = a[i]+b[i];
    for (int i=0; i<n; i++) y[i] = c[i]+d[i];
    DC(n, tmp+n, x, y, res+2*N);
    for (int i=0; i<N; i++){
        res[i+n] += res[2*N+i];
    }
}

// DC(1<<16, tmp.begin(), A.begin(), B.begin(), res.begin())
);
```

7.4 LinearPrime

```
const int MAXP = 100; //max prime
vector<int> P; // primes
void build_prime(){
    static bitset<MAXP> ok;
    int np=0;
    for (int i=2; i<MAXP; i++){
        if (ok[i]==0)P.push_back(i), np++;
        for (int j=0; j<np && i*P[j]<MAXP; j++){
            ok[i*P[j]] = 1;
            if (i%P[j]==0) break;
        }
    }
}
```

7.5 Miller-Rabin

```
typedef long long LL;

inline LL bin_mul(LL a, LL n, const LL& MOD){
    LL re=0;
    while (n>0){
        if (n&1) re += a;
        a += a; if (a>=MOD) a-=MOD;
        n>>=1;
    }
    return re%MOD;
}

inline LL bin_pow(LL a, LL n, const LL& MOD){
    LL re=1;
    while (n>0){
        if (n&1) re = bin_mul(re, a, MOD);
        a = bin_mul(a, a, MOD);
        n>>=1;
    }
    return re;
}

bool is_prime(LL n){
    //static LL sprp[3] = { 2LL, 7LL, 61LL};
    static LL sprp[7] = { 2LL, 325LL, 9375LL,
        28178LL, 450775LL, 9780504LL,
        1795265022LL };
    if (n==1 || (n&1)==0) return n==2;
    int u=n-1, t=0;
    while ( (u&1)==0 ) u>>=1, t++;
    for (int i=0; i<3; i++){
        LL x = bin_pow( sprp[i]%n, u, n);
        if (x==0 || x==1 || x==n-1) continue;

        for (int j=1; j<t; j++){
            x=x*x%n;
            if (x==1 || x==n-1) break;
        }
        if (x==n-1) continue;
        return 0;
    }
    return 1;
}
```

7.6 Mobius

```
void mobius() {
    fill(isPrime, isPrime + MAXN, 1);
    mu[1] = 1, num = 0;
    for (int i = 2; i < MAXN; ++i) {
        if (isPrime[i]) primes[num++] = i, mu[i] = -1;
        static int d;
        for (int j = 0; j < num && (d = i * primes[j])
            < MAXN; ++j) {
            isPrime[d] = false;
            if (i % primes[j] == 0) {
                mu[d] = 0; break;
            }
        }
    }
}
```

```

    } else mu[d] = -mu[i];
}
}
}

```

7.7 Simplex

```

// Two-phase simplex algorithm for solving linear
// programs of the form
//
//      maximize      c^T x
//      subject to    Ax <= b
//                   x >= 0
//
// INPUT: A -- an m x n matrix
//        b -- an m-dimensional vector
//        c -- an n-dimensional vector
//        x -- a vector where the optimal solution will
//              be stored
//
// OUTPUT: value of the optimal solution (infinity if
//         unbounded
//         above, nan if infeasible)
//
// To use this code, create an LPSolver object with A,
// b, and c as
// arguments. Then, call Solve(x).

#include <iostream>
#include <iomanip>
#include <vector>
#include <cmath>
#include <limits>

using namespace std;

typedef long double DOUBLE;
typedef vector<DOUBLE> VD;
typedef vector<VD> VVD;
typedef vector<int> VI;

const DOUBLE EPS = 1e-9;

struct LPSolver {
    int m, n;
    VI B, N;
    VVD D;

    LPSolver(const VVD &A, const VD &b, const VD &c) :
        m(b.size()), n(c.size()), N(m + 1), B(m), D(m + 2,
            VD(n + 2)) {
        for (int i = 0; i < m; i++) for (int j = 0; j < n;
            j++) D[i][j] = A[i][j];
        for (int i = 0; i < m; i++) { B[i] = n + i; D[i][n]
            = -1; D[i][n + 1] = b[i]; }
        for (int j = 0; j < n; j++) { N[j] = j; D[m][j] = -
            c[j]; }
        N[n] = -1; D[m + 1][n] = 1;
    }

    void Pivot(int r, int s) {
        double inv = 1.0 / D[r][s];
        for (int i = 0; i < m + 2; i++) if (i != r)
            for (int j = 0; j < n + 2; j++) if (j != s)
                D[i][j] -= D[r][j] * D[i][s] * inv;
        for (int j = 0; j < n + 2; j++) if (j != s) D[r][j]
            *= inv;
        for (int i = 0; i < m + 2; i++) if (i != r) D[i][s]
            *= -inv;
        D[r][s] = inv;
        swap(B[r], N[s]);
    }

    bool Simplex(int phase) {
        int x = phase == 1 ? m + 1 : m;
        while (true) {

```

```

            int s = -1;
            for (int j = 0; j <= n; j++) {
                if (phase == 2 && N[j] == -1) continue;
                if (s == -1 || D[x][j] < D[x][s] || D[x][j] ==
                    D[x][s] && N[j] < N[s]) s = j;
            }
            if (D[x][s] > -EPS) return true;
            int r = -1;
            for (int i = 0; i < m; i++) {
                if (D[i][s] < EPS) continue;
                if (r == -1 || D[i][n + 1] / D[i][s] < D[r][n +
                    1] / D[r][s] ||
                    (D[i][n + 1] / D[i][s]) == (D[r][n + 1] / D[r
                        ][s]) && B[i] < B[r]) r = i;
            }
            if (r == -1) return false;
            Pivot(r, s);
        }
    }

    DOUBLE Solve(VD &x) {
        int r = 0;
        for (int i = 1; i < m; i++) if (D[i][n + 1] < D[r][
            n + 1]) r = i;
        if (D[r][n + 1] < -EPS) {
            Pivot(r, n);
            if (!Simplex(1) || D[m + 1][n + 1] < -EPS) return
                -numeric_limits<DOUBLE>::infinity();
            for (int i = 0; i < m; i++) if (B[i] == -1) {
                int s = -1;
                for (int j = 0; j <= n; j++)
                    if (s == -1 || D[i][j] < D[i][s] || D[i][j]
                        == D[i][s] && N[j] < N[s]) s = j;
                Pivot(i, s);
            }
        }
        if (!Simplex(2)) return numeric_limits<DOUBLE>::
            infinity();
        x = VD(n);
        for (int i = 0; i < m; i++) if (B[i] < n) x[B[i]] =
            D[i][n + 1];
        return D[m][n + 1];
    }
};

int main() {

    const int m = 4;
    const int n = 3;
    DOUBLE _A[m][n] = {
        { 6, -1, 0 },
        { -1, -5, 0 },
        { 1, 5, 1 },
        { -1, -5, -1 }
    };
    DOUBLE _b[m] = { 10, -4, 5, -5 };
    DOUBLE _c[n] = { 1, -1, 0 };

    VVD A(m);
    VD b(_b, _b + m);
    VD c(_c, _c + n);
    for (int i = 0; i < m; i++) A[i] = VD(_A[i], _A[i] +
        n);

    LPSolver solver(A, b, c);
    VD x;
    DOUBLE value = solver.Solve(x);

    cerr << "VALUE: " << value << endl; // VALUE: 1.29032
    cerr << "SOLUTION: "; // SOLUTION: 1.74194 0.451613 1
    for (size_t i = 0; i < x.size(); i++) cerr << " " <<
        x[i];
    cerr << endl;
    return 0;
}

```

7.8 Sprague-Grundy

Anti Nim (取走最後一個石子者敗)

先手必勝 **if and only if**

1. 「所有」堆的石子數都為 1 且遊戲的 SG 值為 0。
2. 「有些」堆的石子數大於 1 且遊戲的 SG 值不為 0。

Anti-SG (決策集合為空的遊戲者贏)

定義 SG 值為 0 時，遊戲結束，

則先手必勝 **if and only if**

1. 遊戲中沒有單一遊戲的 SG 函數大於 1 且遊戲的 SG 函數為 0。
2. 遊戲中某個單一遊戲的 SG 函數大於 1 且遊戲的 SG 函數不為 0。

Sprague-Grundy

1. 雙人、回合制
2. 資訊完全公開
3. 無隨機因素
4. 可在有限步內結束
5. 沒有和局
6. 雙方可採取的行動相同

SG(S) 的值為 0：後手(P)必勝

不為 0：先手(N)必勝

```
int mex(set S) {
    // find the min number >= 0 that not in the S
    // e.g. S = {0, 1, 3, 4} mex(S) = 2
}

state = []
int SG(A) {
    if (A not in state) {
        S = sub_states(A)
        if( len(S) > 1 ) state[A] = reduce(operator.xor, [
            SG(B) for B in S])
        else state[A] = mex(set(SG(B) for B in next_states(
            A)))
    }
    return state[A]
}
```

7.9 Ax+by=gcd

```
pair<int,int> extgcd(int a, int b){
    if (b==0) return {1,0};
    int k = a/b;
    pair<int,int> p = extgcd(b,a-k*b);
    return { p.second, p.first - k*p.second };
}

int inv[maxn];
LL invtable(int n,LL P){
    inv[1]=1;
    for(int i=2;i<n;++i)
        inv[i]=(P-(P/i))*inv[P%i]%P;
}
```

7.10 PollardRho

```
// does not work when n is prime
inline LL f(LL x, LL mod) {
    return (x * x % mod + 1) % mod;
}

inline LL pollard_rho(LL n) {
    if(!(n&1)) return 2;

```

```
while(true) {
    LL y = 2, x = rand() % (n - 1) + 1, res = 1;
    for(int sz = 2; res == 1; sz *= 2) {
        for(int i = 0; i < sz && res <= 1; i++) {
            x = f(x, n);
            res = __gcd(abs(x - y), n);
        }
        y = x;
    }
    if (res != 0 && res != n) return res;
}
}
```

7.11 Theorem

```
/*
Lucas's Theorem
For non-negative integer n,m and prime P,
C(m,n) mod P = C(m/M,n/M) * C(m%M,n%M) mod P
= mult_i ( C(m_i,n_i) )
where m_i is the i-th digit of m in base P.
-----
Kirchhoff's theorem
A_{ii} = deg(i), A_{ij} = (i,j) \in E ? -1 : 0
Deleting any one row, one column, and cal the det(A)
-----
Nth Catalan recursive function:
C_0 = 1, C_{n+1} = C_n * 2(2n + 1)/(n+2)
-----
Mobius Formula
u(n) = 1, if n = 1
      (-1)^m, 若 n 無平方數因數, 且 n = p1*p2*p3
      *...*pk
      0, 若 n 有大於 1 的平方數因數
- Property
1. (積性函數) u(a)u(b) = u(ab)
2. \sum_{d|n} u(d) = [n == 1]
-----
Mobius Inversion Formula
if f(n) = \sum_{d|n} g(d)
then g(n) = \sum_{d|n} u(n/d)f(d)
= \sum_{d|n} u(d)f(n/d)
- Application
the number/power of gcd(i, j) = k
- Trick
分塊, O(sqrt(n))
-----
Chinese Remainder Theorem (m_i 兩兩互質)

x = a_1 (mod m_1)
x = a_2 (mod m_2)
....
x = a_i (mod m_i)

construct a solution:

Let M = m_1 * m_2 * m_3 * ... * m_n
Let M_i = M / m_i

t_i = 1 / M_i
t_i * M_i = 1 (mod m_i)

solution x = a_1 * t_1 * M_1 + a_2 * t_2 * M_2 + ...
+ a_n * t_n * M_n + k * M
= k*M + \sum a_i * t_i * M_i, k is positive integer.

under mod M, there is one solution x = \sum a_i * t_i *
M_i
-----
Burnside's lemma
|G| * |X/G| = sum( |X^g| ) where g in G
總方法數：每一種旋轉下不動點的個數總和 除以 旋轉的方法數
-----
Lagrange multiplier

```


$f(x, y)$ 求極值。必須滿足 $g(x, y) = 0$ 。

湊得 $f(x, y) = f(x, y) + \lambda g(x, y)$

定義 $s(x, y, \lambda) = f(x, y) + \lambda g(x, y)$

$f(x, y)$ 的極值，等同 $s(x, y, \lambda) = f(x, y) + \lambda g(x, y)$ 的極值。

欲求極值：

對 x 偏微分，讓斜率是 0。

對 y 偏微分，讓斜率是 0。

不管 λ 如何變化， $\lambda g(x, y)$ 都是零， $s(x, y, \lambda)$ 永遠不變。

欲求永遠不變的地方：

對 λ 偏微分，讓斜率是 0。

三道偏微分方程式聯立之後，其解涵蓋了（不全是）所有符合約束條件的極值。

$\{ \partial/\partial x s(x, y, \lambda) = 0$

$\{ \partial/\partial y s(x, y, \lambda) = 0$

$\{ \partial/\partial \lambda s(x, y, \lambda) = 0$

Baby Step Giant Step

Get $a^x = b \pmod n$

$x = im - j, m = \text{ceil}(\text{sqrt}(n))$

$a^{im} = b * a^j \pmod n$

預處理 $a^{im}, 1 \leq i \leq m$ ，存 hash/map

接著去試從 $0 \leq j \leq m$ 有沒有 $a^{im} = b * a^j$ ，有的話答案是 $im - j$ 。

可能需要特判 $\text{if } b = 1 \text{ then } x = 0$

*/

8 Other

8.1 CYK

// 2016 NCP from sunmoon

// 轉換

#define MAXN 55

struct CNF{

int s, x, y; // s->xy | s->x, if y==-1

int cost;

CNF() {}

CNF(int s, int x, int y, int c): s(s), x(x), y(y), cost(c) {}
};

int state; // 規則數量

map<char, int> rule; // 每個字元對應到的規則，小寫字母為終端字符

vector<CNF> cnf;

inline void init() {

state=0;

rule.clear();

cnf.clear();

}

inline void add_to_cnf(char s, const string &p, int cost) {

if (rule.find(s) == rule.end()) rule[s] = state++;

for (auto c: p) if (rule.find(c) == rule.end()) rule[c] = state++;

if (p.size() == 1) {
cnf.push_back(CNF(rule[s], rule[p[0]], -1, cost));

} else {
int left = rule[s];

int sz = p.size();

for (int i = 0; i < sz - 2; ++i) {
cnf.push_back(CNF(left, rule[p[i]], state, 0));

left = state++;

}
cnf.push_back(CNF(left, rule[p[sz-2]], rule[p[sz-1]], cost));

}

}

// 計算

vector<long long> dp[MAXN][MAXN];

vector<bool> neg_INF[MAXN][MAXN]; // 如果花費是負的可能會有無限小的情形

inline void relax(int l, int r, const CNF &c, long long cost, bool neg_c=0) {

if (!neg_INF[l][r][c.s] && (neg_INF[l][r][c.x] || cost < dp[l][r][c.s])) {

if (neg_c || neg_INF[l][r][c.x]) {

dp[l][r][c.s] = 0;

neg_INF[l][r][c.s] = true;

} else dp[l][r][c.s] = cost;

}

}

inline void bellman(int l, int r, int n) {

for (int k = l; k <= state; ++k)

for (auto c: cnf)

if (c.y == -1) relax(l, r, c, dp[l][r][c.x] + c.cost, k == n);

;

}

inline void cyk(const vector<int> &tok) {

for (int i = 0; i < (int) tok.size(); ++i) {

for (int j = 0; j < (int) tok.size(); ++j) {

dp[i][j] = vector<long long>(state + 1, INT_MAX);

neg_INF[i][j] = vector<bool>(state + 1, false);

}

dp[i][i][tok[i]] = 0;

bellman(i, i, tok.size());

}

for (int r = 1; r < (int) tok.size(); ++r) {

for (int l = r - 1; l >= 0; --l) {

for (int k = l; k < r; ++k)

for (auto c: cnf)

if (~c.y) relax(l, r, c, dp[l][k][c.x] + dp[k+1][r][c.y] + c.cost);

bellman(l, r, tok.size());

}

}

}

8.2 DP-optimization

Monotonicity & 1D/1D DP & 2D/1D DP

Definition xD/yD

1D/1D DP[j] = min(0 ≤ i < j) { DP[i] + w(i, j) }; DP[0] = k

2D/1D DP[i][j] = min(i < k ≤ j) { DP[i][k - 1] + DP[k][j] } + w(i, j); DP[i][i] = 0

Monotonicity

c

d

a | w(a, c) w(a, d)

b | w(b, c) w(b, d)

Monge Condition

Concave (凹四邊形不等式): w(a, c) + w(b, d) ≥ w(a, d) + w(b, c)

Convex (凸四邊形不等式): w(a, c) + w(b, d) ≤ w(a, d) + w(b, c)

Totally Monotone

Concave (凹單調): w(a, c) ≤ w(b, d) -----> w(a, d) ≤ w(b, c)

Convex (凸單調): w(a, c) ≥ w(b, d) -----> w(a, d) ≥ w(b, c)

1D/1D DP O(n^2) -> O(n lg n)

CONSIDER THE TRANSITION POINT

Solve 1D/1D Concave by Stack

Solve 1D/1D Convex by Deque

2D/1D Convex DP (Totally Monotone) $O(n^3) \rightarrow O(n^2)$
 $h(i, j - 1) \leq h(i, j) \leq h(i + 1, j)$

8.3 DigitCounting

```
int dfs(int pos, int state1, int state2 ....., bool
    limit, bool zero) {
    if ( pos == -1 ) return 是否符合條件;
    int &ret = dp[pos][state1][state2][....];
    if ( ret != -1 && !limit ) return ret;
    int ans = 0;
    int upper = limit ? digit[pos] : 9;
    for ( int i = 0 ; i <= upper ; i++ ) {
        ans += dfs(pos - 1, new_state1, new_state2,
            limit & ( i == upper ), ( i == 0 ) && zero);
    }
    if ( !limit ) ret = ans;
    return ans;
}

int solve(int n) {
    int it = 0;
    for ( ; n ; n /= 10 ) digit[it++] = n % 10;
    return dfs(it - 1, 0, 0, 1, 1);
}
```

8.4 Dp1D1D

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

int t, n, L;
int p;
char s[MAXN][35];
ll sum[MAXN] = {0};
long double dp[MAXN] = {0};
int prevd[MAXN] = {0};

long double pw(long double a, int n) {
    if ( n == 1 ) return a;
    long double b = pw(a, n/2);
    if ( n & 1 ) return b*b*a;
    else return b*b;
}

long double f(int i, int j) {
    // cout << (sum[i] - sum[j]+i-j-1-L) << endl;
    return pw(abs(sum[i] - sum[j]+i-j-1-L), p) + dp[j];
}

struct INV {
    int L, R, pos;
};
INV stk[MAXN*10];
int top = 1, bot = 1;
void update(int i) {
    while ( top > bot && i < stk[top].L && f(stk[top].L
        , i) < f(stk[top].L, stk[top].pos) ) {
        stk[top - 1].R = stk[top].R;
        top--;
    }
    int lo = stk[top].L, hi = stk[top].R, mid, pos =
        stk[top].pos;
    //if ( i >= lo ) lo = i + 1;
    while ( lo != hi ) {
        mid = lo + (hi - lo) / 2;
        if ( f(mid, i) < f(mid, pos) ) hi = mid;
        else lo = mid + 1;
    }
    if ( hi < stk[top].R ) {
        stk[top + 1] = (INV) { hi, stk[top].R, i };
        stk[top++].R = hi;
    }
}

int main() {
```

```
cin >> t;
while ( t-- ) {
    cin >> n >> L >> p;
    dp[0] = sum[0] = 0;
    for ( int i = 1 ; i <= n ; i++ ) {
        cin >> s[i];
        sum[i] = sum[i-1] + strlen(s[i]);
        dp[i] = numeric_limits<long double>::max();
    }
    stk[top] = (INV) {1, n + 1, 0};
    for ( int i = 1 ; i <= n ; i++ ) {
        if ( i >= stk[bot].R ) bot++;
        dp[i] = f(i, stk[bot].pos);
        update(i);
        // cout << (ll) f(i, stk[bot].pos) << endl;
    }
    if ( dp[n] > 1e18 ) {
        cout << "Too hard to arrange" << endl;
    } else {
        vector<PI> as;
        cout << (ll)dp[n] << endl;
    }
}
return 0;
}
```

8.5 ManhattanMST

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

const int MAXN = 100005;
const int OFFSET = 2000; // y-x may < 0, offset it, if
// y-x too large, please write a unique function
const int INF = 0xFFFFFFFF;
int n;
int x[MAXN], y[MAXN], p[MAXN];

typedef pair<int, int> pii;
pii bit[MAXN]; // [ val, pos ]

struct P {
    int x, y, id;
    bool operator<(const P&b) const {
        if ( x == b.x ) return y > b.y;
        else return x > b.x;
    }
};
vector<P> op;

struct E {
    int x, y, cost;
    bool operator<(const E&b) const {
        return cost < b.cost;
    }
};
vector<E> edges;

int find(int x) {
    return p[x] == x ? x : p[x] = find(p[x]);
}

void update(int i, int v, int p) {
    while ( i ) {
        if ( bit[i].first > v ) bit[i] = {v, p};
        i -= i & (-i);
    }
}

pii query(int i) {
    pii res = {INF, INF};
    while ( i < MAXN ) {
        if ( bit[i].first < res.first ) res = {bit[i].
            first, bit[i].second};
        i += i & (-i);
    }
}
```

8.6 Count Spanning Tree

```
return res;
}

void input() {
    cin >> n;
    for (int i = 0; i < n; i++) cin >> x[i] >> y[i], op.push_back((P) {x[i], y[i], i});
}

void mst() {
    for (int i = 0; i < MAXN; i++) p[i] = i;
    int res = 0;
    sort(edges.begin(), edges.end());
    for (auto e : edges) {
        int x = find(e.x), y = find(e.y);
        if (x != y) {
            p[x] = y;
            res += e.cost;
        }
    }
    cout << res << endl;
}

void construct() {
    sort(op.begin(), op.end());
    for (int i = 0; i < n; i++) {
        pii q = query(op[i].y - op[i].x + OFFSET);
        update(op[i].y - op[i].x + OFFSET, op[i].x + op[i].y, op[i].id);
        if (q.first == INF) continue;
        edges.push_back((E) {op[i].id, q.second, abs(x[op[i].id] - x[q.second]) + abs(y[op[i].id] - y[q.second])});
    }
}

void solve() {
    // [45 ~ 90 deg]
    for (int i = 0; i < MAXN; i++) bit[i] = {INF, INF};
    construct();

    // [0 ~ 45 deg]
    for (int i = 0; i < MAXN; i++) bit[i] = {INF, INF};
    for (int i = 0; i < n; i++) swap(op[i].x, op[i].y);
    construct();
    for (int i = 0; i < n; i++) swap(op[i].x, op[i].y);

    // [-90 ~ -45 deg]
    for (int i = 0; i < MAXN; i++) bit[i] = {INF, INF};
    for (int i = 0; i < n; i++) op[i].y *= -1;
    construct();

    // [-45 ~ 0 deg]
    for (int i = 0; i < MAXN; i++) bit[i] = {INF, INF};
    for (int i = 0; i < n; i++) swap(op[i].x, op[i].y);
    construct();

    // mst
    mst();
}

int main() {
    input();
    solve();
    return 0;
}
```

新的方法介绍

下面我们介绍一种新的方法——Matrix-Tree定理(Kirchhoff矩阵-树定理)。

Matrix-Tree定理是解决生成树计数问题最有力的武器之一。它首先于1847年被Kirchhoff证明。在介绍定理之前，我们首先明确几个概念：

- 1、G的度数矩阵 $D[G]$ 是一个 $n \times n$ 的矩阵，并且满足：当 $i \neq j$ 时， $d_{ij} = 0$ ；当 $i = j$ 时， d_{ij} 等于 v_i 的度数。
- 2、G的邻接矩阵 $A[G]$ 也是一个 $n \times n$ 的矩阵，并且满足：如果 v_i 、 v_j 之间有边直接相连，则 $a_{ij} = 1$ ，否则为0。

我们定义G的Kirchhoff矩阵(也称为拉普拉斯算子) $C[G]$ 为 $C[G] = D[G] - A[G]$ ，

则Matrix-Tree定理可以描述为：G的所有不同的生成树的个数等于其Kirchhoff矩阵 $C[G]$ 任何一个 $n-1$ 阶主子式的行列式的绝对值。

所谓 $n-1$ 阶主子式，就是对于 $r (1 \leq r \leq n)$ ，将 $C[G]$ 的第 r 行、第 r 列同时去掉后得到的新矩阵，用 $Cr[G]$ 表示。

生成树计数

算法步骤：

- 1、构建拉普拉斯矩阵
 $Matrix[i][j] =$
 $\text{degree}(i), i=j$
 $-1, i-j \text{ 有边}$
 $0, \text{其他情况}$
- 2、去掉第 r 行，第 r 列 (r 任意)
- 3、计算矩阵的行列式

```
/* *****
MYID      : Chen Fan
LANG      : G++
PROG      : Count_Spaning_Tree_From_Kuangbin
***** */
#include <stdio.h>
#include <string.h>
#include <algorithm>
#include <iostream>
#include <math.h>
using namespace std;
const double eps = 1e-8;
const int MAXN = 110;
int sgn(double x)
{
    if(fabs(x) < eps) return 0;
    if(x < 0) return -1;
    else return 1;
}
double b[MAXN][MAXN];
double det(double a[][MAXN], int n)
{
    int i, j, k, sign = 0;
    double ret = 1;
    for(i = 0; i < n; i++)
        for(j = 0; j < n; j++) b[i][j] = a[i][j];
    for(i = 0; i < n; i++)
    {
        if(sgn(b[i][i]) == 0)
        {
            for(j = i + 1; j < n; j++)
                if(sgn(b[j][i]) != 0) break;
            if(j == n) return 0;
            for(k = i; k < n; k++) swap(b[i][k], b[j][k]);
            sign++;
        }
        ret *= b[i][i];
        for(k = i + 1; k < n; k++) b[i][k] /= b[i][i];
        for(j = i + 1; j < n; j++)
            for(k = i + 1; k < n; k++) b[j][k] -= b[j][i] * b[i][k];
    }
}
```

```

    if(sign & 1)ret = -ret;
    return ret;
}
double a[MAXN][MAXN];
int g[MAXN][MAXN];
int main()
{
    int T;
    int n,m;
    int u,v;
    scanf("%d",&T);
    while(T--){
        scanf("%d%d",&n,&m);
        memset(g,0,sizeof(g));
        while(m--){
            scanf("%d%d",&u,&v);
            u--;v--;
            g[u][v] = g[v][u] = 1;
        }
        memset(a,0,sizeof(a));
        for(int i = 0;i < n;i++)
            for(int j = 0;j < n;j++)
                if(i != j && g[i][j]){
                    a[i][i]++;
                    a[i][j] = -1;
                }
        double ans = det(a,n-1);
        printf("%.01f\n",ans);
    }
    return 0;
}

```

9 String

9.1 AC

```

// remember make_fail() !!!
// notice MLE

const int sigma = 62;
const int MAXC = 200005;

inline int idx(char c){
    if ('A' <= c && c <= 'Z')return c - 'A';
    if ('a' <= c && c <= 'z')return c - 'a' + 26;
    if ('0' <= c && c <= '9')return c - '0' + 52;
}

struct ACautomaton{
    struct Node{
        Node *next[sigma], *fail;
        int cnt; // dp
        Node(){
            memset(next,0,sizeof(next));
            fail=0;
            cnt=0;
        }
    } buf[MAXC], *bufp, *ori, *root;

    void init(){
        bufp = buf;
        ori = new (bufp++) Node();
        root = new (bufp++) Node();
    }

    void insert(int n, char *s){
        Node *ptr = root;
        for (int i=0; s[i]; i++){
            int c = idx(s[i]);
            if (ptr->next[c]==NULL)
                ptr->next[c] = new (bufp++) Node();

```

```

        ptr = ptr->next[c];
    }
    ptr->cnt++;
}

Node* trans(Node *o, int c){
    while (o->next[c]==NULL) o = o->fail;
    return o->next[c];
}

void make_fail(){
    static queue<Node*> que;

    for (int i=0; i<sigma; i++){
        ori->next[i] = root;
        root->fail = ori;

        que.push(root);
        while ( que.size() ){
            Node *u = que.front(); que.pop();
            for (int i=0; i<sigma; i++){
                if (u->next[i]==NULL)continue;
                u->next[i]->fail = trans(u->fail,i);
                que.push(u->next[i]);
            }
            u->cnt += u->fail->cnt;
        }
    }
} ac;

```

9.2 BWT

```

// BWT
const int N = 8; // 字串長度
int s[N+N+1] = "suffixes"; // 字串，後面預留一倍空間。
int sa[N]; // 後綴陣列
int pivot;

int cmp(const void* i, const void* j)
{
    return strcmp(s+(int*)i, s+(int*)j, N);
}

// 此處便宜行事，採用  $O(N^2 \log N)$  的後綴陣列演算法。
void BWT()
{
    strcpy(s + N, s, N);
    for (int i=0; i<N; ++i) sa[i] = i;
    qsort(sa, N, sizeof(int), cmp);
    // 當輸入字串的所有字元都相同，必須當作特例處理。
    // 或者改用 stable sort。

    for (int i=0; i<N; ++i)
        cout << s[(sa[i] + N-1) % N];

    for (int i=0; i<N; ++i)
        if (sa[i] == 0)
        {
            pivot = i;
            break;
        }
}

// Inverse BWT
const int N = 8; // 字串長度
char t[N+1] = "xuffessi"; // 字串
int pivot;
int next[N];

void IBWT()
{
    vector<int> index[256];
    for (int i=0; i<N; ++i)
        index[t[i]].push_back(i);

```

```

    for (int i=0, n=0; i<256; ++i)
        for (int j=0; j<index[i].size(); ++j)
            next[n++] = index[i][j];

    int p = pivot;
    for (int i=0; i<N; ++i)
        cout << t[p = next[p]];
}

```

9.3 KMP

```

template<typename T>
void build_KMP(int n, T *s, int *f){ // 1 base
    f[0]=-1, f[1]=0;
    for (int i=2; i<=n; i++){
        int w = f[i-1];
        while (w>=0 && s[w+1]!=s[i])w = f[w];
        f[i]=w+1;
    }
}

template<typename T>
int KMP(int n, T *a, int m, T *b){
    build_KMP(m,b,f);
    int ans=0;

    for (int i=1, w=0; i<=n; i++){
        while ( w>=0 && b[w+1]!=a[i] )w = f[w];
        w++;
        if (w==m){
            ans++;
            w=f[w];
        }
    }
    return ans;
}

```

9.4 PalindromicTree

```

// remember init()      !!!
// remember make_fail() !!!
// insert s need 1 base !!!
// notice MLE
const int sigma = 62;
const int MAXC = 1000006;
inline int idx(char c){
    if ('a'<= c && c <= 'z')return c-'a';
    if ('A'<= c && c <= 'Z')return c-'A'+26;
    if ('0'<= c && c <= '9')return c-'0'+52;
}

struct PalindromicTree{
    struct Node{
        Node *next[sigma], *fail;
        int len, cnt; // for dp
        Node(){
            memset(next,0,sizeof(next));
            fail=0;
            len = cnt = 0;
        }
    } buf[MAXC], *bufp, *even, *odd;

    void init(){
        bufp = buf;
        even = new (bufp++) Node();
        odd = new (bufp++) Node();
        even->fail = odd;
        odd->len = -1;
    }

    void insert(char *s){
        Node* ptr = even;
        for (int i=1; s[i]; i++){
            ptr = extend(ptr,s+i);

```

```

        }
    }

    Node* extend(Node *o, char *ptr){
        int c = idx(*ptr);
        while ( *ptr != *(ptr-1-o->len) )o=o->fail;
        Node *&np = o->next[c];
        if (!np){
            np = new (bufp++) Node();
            np->len = o->len+2;
            Node *f = o->fail;
            if (f){
                while ( *ptr != *(ptr-1-f->len) )f=f->fail;
                np->fail = f->next[c];
            }
            else {
                np->fail = even;
            }
            np->cnt = np->fail->cnt;
        }
        np->cnt++;
        return np;
    }
} PAM;

```

9.5 SAM

```

// par : fail link
// val : a topological order ( useful for DP )
// go[x] : automata edge ( x is integer in [0,26) )

struct SAM{
    struct State{
        int par, go[26], val;
        State () : par(0), val(0){ FZ(go); }
        State (int _val) : par(0), val(_val){ FZ(go); }
    };
    vector<State> vec;
    int root, tail;

    void init(int arr[], int len){
        vec.resize(2);
        vec[0] = vec[1] = State(0);
        root = tail = 1;
        for (int i=0; i<len; i++)
            extend(arr[i]);
    }

    void extend(int w){
        int p = tail, np = vec.size();
        vec.PB(State(vec[p].val+1));
        for ( ; p && vec[p].go[w]==0; p=vec[p].par)
            vec[p].go[w] = np;
        if (p == 0){
            vec[np].par = root;
        } else {
            if (vec[vec[p].go[w]].val == vec[p].val+1){
                vec[np].par = vec[p].go[w];
            } else {
                int q = vec[p].go[w], r = vec.size();
                vec.PB(vec[q]);
                vec[r].val = vec[p].val+1;
                vec[q].par = vec[np].par = r;
                for ( ; p && vec[p].go[w] == q; p=vec[p].par)
                    vec[p].go[w] = r;
            }
        }
        tail = np;
    }
};

```

9.6 Z-value

```

z[0] = 0;
for ( int bst = 0, i = 1; i < len ; i++ ) {
    if ( z[bst] + bst <= i ) z[i] = 0;
    else z[i] = min(z[i - bst], z[bst] + bst - i);
    while ( str[i + z[i]] == str[z[i]] ) z[i]++;
    if ( i + z[i] > bst + z[bst] ) bst = i;
}

// 回文版

void Zpal(const char *s, int len, int *z) {
    // Only odd palindrome len is considered
    // z[i] means that the longest odd palindrom
    // centered at
    // i is [i-z[i] .. i+z[i]]
    z[0] = 0;
    for (int b=0, i=1; i<len; i++) {
        if (z[b] + b >= i) z[i] = min(z[2*b-i], b+z[b]-i);
        else z[i] = 0;
        while (i+z[i]+1 < len and i-z[i]-1 >= 0 and
            s[i+z[i]+1] == s[i-z[i]-1]) z[i] ++;
        if (z[i] + i > z[b] + b) b = i;
    }
}

```

9.7 Smallest Rotation

```

string mcp(string s){
    int n = s.length();
    s += s;
    int i=0, j=1;
    while (i<n && j<n){
        int k = 0;
        while (k < n && s[i+k] == s[j+k]) k++;
        if (s[i+k] <= s[j+k]) j += k+1;
        else i += k+1;
        if (i == j) j++;
    }
    int ans = i < n ? i : j;
    return s.substr(ans, n);
}

```

9.8 Suffix Array

```

/*he[i]保存了在后缀数组中相邻两个后缀的最长公共前缀长度
*sa[i]表示的是字典序排名为i的后缀是谁（字典序越小的排名越靠前）
*rk[i]表示的是后缀我所对应的排名是多少 */

const int MAX = 1020304;
int ct[MAX], he[MAX], rk[MAX];
int sa[MAX], tsa[MAX], tp[MAX][2];
void suffix_array(char *ip){
    int len = strlen(ip);
    int alp = 256;
    memset(ct, 0, sizeof(ct));
    for(int i=0;i<len;i++) ct[ip[i]+1]++;
    for(int i=1;i<alp;i++) ct[i]+=ct[i-1];
    for(int i=0;i<len;i++) rk[i]=ct[ip[i]];
    for(int i=1;i<len;i*=2){
        for(int j=0;j<len;j++){
            if(j+i>len) tp[j][1]=0;
            else tp[j][1]=rk[j+i]+1;
            tp[j][0]=rk[j];
        }
        memset(ct, 0, sizeof(ct));
        for(int j=0;j<len;j++) ct[tp[j][1]+1]++;
        for(int j=1;j<len+2;j++) ct[j]+=ct[j-1];
        for(int j=0;j<len;j++) tsa[ct[tp[j][1]]+1]=j;
        memset(ct, 0, sizeof(ct));
        for(int j=0;j<len;j++) ct[tp[j][0]+1]++;
    }
}

```

```

for(int j=1;j<len+1;j++) ct[j]+=ct[j-1];
for(int j=0;j<len;j++){
    sa[ct[tp[tsa[j]][0]]+1]=tsa[j];
    rk[sa[0]]=0;
    for(int j=1;j<len;j++){
        if( tp[sa[j]][0] == tp[sa[j-1]][0] &&
            tp[sa[j]][1] == tp[sa[j-1]][1] )
            rk[sa[j]] = rk[sa[j-1]];
        else
            rk[sa[j]] = j;
    }
}
for(int i=0,h=0;i<len;i++){
    if(rk[i]==0) h=0;
    else{
        int j=sa[rk[i]-1];
        h=max(0,h-1);
        for(;ip[i+h]==ip[j+h];h++);
    }
    he[rk[i]]=h;
}
}

```

10 無權邊的生成樹個數 Kirchhoff's Theorem

1. 定義 $n \times m$ 矩陣 $E = (a_{i,j})$, n 為點數, m 為邊數, 若 i 點在 j 邊上, i 為小點 $a_{i,j} = 1$, i 為大點 $a_{i,j} = -1$, 否則 $a_{i,j} = 0$ 。
(證明省略)
4. 令 $E(E^T) = Q$, 他是一種有負號的 kirchhoff 的矩陣, 取 Q 的子矩陣即為 $F(F^T)$
結論: 做 Q 取子矩陣算 \det 即為所求。(除去第一行第一列 by mz)

11 monge

$$i \leq i' < j \leq j'$$

$$m(i, j) + m(i', j') \leq m(i', j) + m(i, j')$$

$$k(i, j - 1) \leq k(i, j) \leq k(i + 1, j)$$

12 四心

$$\frac{sa \cdot A + sb \cdot B + sc \cdot C}{sa + sb + sc}$$

外心 $\sin 2A : \sin 2B : \sin 2C$

內心 $\sin A : \sin B : \sin C$

垂心 $\tan A : \tan B : \tan C$

重心 $1 : 1 : 1$

13 Runge-Kutta

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$k_1 = f(t_n, y_n)$$

$$k_2 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1)$$

$$k_3 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2)$$

$$k_4 = f(t_n + h, y_n + hk_3)$$

14 Householder Matrix

$$I - 2 \frac{vv^T}{v^T v}$$