

Contents

1	Basic	1
1.1	vimrc	1
1.2	BigInt	1
1.3	Random	2
2	Mathmatics	2
2.1	Miller Rabin	2
2.2	ax+by=gcd(a,b)	2
2.3	FFT	3
2.4	Hash	3
2.5	Convex Hull	3
2.6	Eratosthenes	4
2.7	GaussElimination	4
2.8	Inverse	4
2.9	IterSet	4
2.10	LinearPrime	4
2.11	SG	4
3	Geometry	4
4	Flow	4
4.1	Dinic	4
5	Graph	5
5.1	Strongly Connected Component(SCC)	5
5.2	Euler Circuit	5
5.3	Hungarian	6
5.4	Maximum Clique	6
5.5	Tarjan	6
5.6	LCA	6
6	Data Structure	7
6.1	Disjoint Set	7
6.2	Sparse Table	7
6.3	Treap	7
7	String	8
7.1	KMP	8
7.2	AC	8
7.3	Z-value	8
8	Dark Code	8
8.1	輸入優化	8
9	Search	9
10	Others	9
11	Persistence	9

Basic

vimrc

```

1 set nu #行號
1 set tabstop=4
1 set softtabstop=4
2
2 set autoindent
2 set shiftwidth=4
2 set cindent
3
3 set smartindent
4
4 se ai ar sm nu rnu is
4 se mouse=a bs=2 so=6 ts=4 ttm=100
4
4 nmap <F4> :! cat -n % > %.print ; lpr %.print
4 nmap <F9> :! clear ; g++ -std=c++11 -static -Wall -O2
4 % -o %.out; ./%.out

```

BigInt

```

struct BigInt{
    static const int LEN = 60;
    static const int BIGMOD = 10000;
    int s;
    int vl, v[LEN];
    // vector<int> v;
    BigInt() : s(1) { vl = 0; }
    BigInt(long long a) {
        s = 1; vl = 0;
        if (a < 0) { s = -1; a = -a; }
        while (a) {
            push_back(a % BIGMOD);
            a /= BIGMOD;
        }
    }
    BigInt(string str) {
        s = 1; vl = 0;
        int stPos = 0, num = 0;
        if (!str.empty() && str[0] == '-') {
            stPos = 1;
            s = -1;
        }
        for (int i=SZ(str)-1, q=1; i>=stPos; i--) {
            num += (str[i] - '0') * q;
            if ((q *= 10) >= BIGMOD) {
                push_back(num);
                num = 0; q = 1;
            }
        }
        if (num) push_back(num);
    }
    int len() const { return vl; /* return SZ(v); */ }
    bool empty() const { return len() == 0; }
    void push_back(int x) { v[vl++] = x; /* v.PB(x); */ }
    void pop_back() { vl--; /* v.pop_back(); */ }
    int back() const { return v[vl-1]; /* return v.back()
    ; */ }
    void n() { while (!empty() && !back()) pop_back(); }
    void resize(int nl) {
        vl = nl; fill(v, v+vl, 0);
        // v.resize(nl); // fill(ALL(v), 0);
    }
    void print() const {
        if (empty()) { putchar('0'); return; }
        if (s == -1) putchar('-');
        printf("%d", back());
        for (int i=len()-2; i>=0; i--) printf("%.4d", v[i]);
    }
    friend std::ostream& operator << (std::ostream& out,
        const BigInt &a) {
        if (a.empty()) { out << "0"; return out; }
        if (a.s == -1) out << "-";
        out << a.back();
        for (int i=a.len()-2; i>=0; i--) {
            char str[10];

```

```

    snprintf(str, 5, "%.4d", a.v[i]);
    out << str;
}
return out;
}
int cp3(const Bigint &b) const {
    if (s != b.s) return s > b.s ? 1 : -1;
    if (s == -1) return -(*this).cp3(-b);
    if (len() != b.len()) return len() > b.len() ? 1 : -1;
    for (int i = len() - 1; i >= 0; i--)
        if (v[i] != b.v[i]) return v[i] > b.v[i] ? 1 : -1;
    return 0;
}
bool operator < (const Bigint &b) const { return cp3(b) == -1; }
bool operator <= (const Bigint &b) const { return cp3(b) <= 0; }
bool operator >= (const Bigint &b) const { return cp3(b) >= 0; }
bool operator == (const Bigint &b) const { return cp3(b) == 0; }
bool operator != (const Bigint &b) const { return cp3(b) != 0; }
bool operator > (const Bigint &b) const { return cp3(b) == 1; }
Bigint operator - () const {
    Bigint r = (*this);
    r.s = -r.s;
    return r;
}
Bigint operator + (const Bigint &b) const {
    if (s == -1) return -(-(*this) + (-b));
    if (b.s == -1) return (*this) - (-b);
    Bigint r;
    int nl = max(len(), b.len());
    r.resize(nl + 1);
    for (int i = 0; i < nl; i++) {
        if (i < len()) r.v[i] += v[i];
        if (i < b.len()) r.v[i] += b.v[i];
        if (r.v[i] >= BIGMOD) {
            r.v[i+1] += r.v[i] / BIGMOD;
            r.v[i] %= BIGMOD;
        }
    }
    r.n();
    return r;
}
Bigint operator - (const Bigint &b) const {
    if (s == -1) return -(-(*this) - (-b));
    if (b.s == -1) return (*this) + (-b);
    if ((*this) < b) return -(b - (*this));
    Bigint r;
    r.resize(len());
    for (int i = 0; i < len(); i++) {
        r.v[i] += v[i];
        if (i < b.len()) r.v[i] -= b.v[i];
        if (r.v[i] < 0) {
            r.v[i] += BIGMOD;
            r.v[i+1]--;
        }
    }
    r.n();
    return r;
}
Bigint operator * (const Bigint &b) {
    Bigint r;
    r.resize(len() + b.len() + 1);
    r.s = s * b.s;
    for (int i = 0; i < len(); i++) {
        for (int j = 0; j < b.len(); j++) {
            r.v[i+j] += v[i] * b.v[j];
            if (r.v[i+j] >= BIGMOD) {
                r.v[i+j+1] += r.v[i+j] / BIGMOD;
                r.v[i+j] %= BIGMOD;
            }
        }
    }
    r.n();
    return r;
}
Bigint operator / (const Bigint &b) {
    Bigint r;

```

```

    r.resize(max(1, len() - b.len() + 1));
    int oriS = s;
    Bigint b2 = b; // b2 = abs(b)
    s = b2.s = r.s = 1;
    for (int i = r.len() - 1; i >= 0; i--) {
        int d = 0, u = BIGMOD - 1;
        while (d < u) {
            int m = (d + u + 1) >> 1;
            r.v[i] = m;
            if ((r * b2) > (*this)) u = m - 1;
            else d = m;
        }
        r.v[i] = d;
    }
    s = oriS;
    r.s = s * b.s;
    r.n();
    return r;
}
Bigint operator % (const Bigint &b) {
    return (*this) - (*this) / b * b;
}
};

```

Random

```

inline int ran() {
    static int x = 20167122;
    return x = (x * 0xdefaced + 1) & INT_MAX;
}

```

Mathmatics

Miller Rabin

```

typedef long long LL;
LL bin_pow(LL a, LL n, LL MOD) {
    LL re = 1;
    while (n > 0) {
        if (n & 1) re = re * a % MOD;
        a = 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 < 7; 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;
}

```

ax+by=gcd(a,b)

```

typedef pair<int, int> pii;
pii extgcd(int a, int b) {
    if (b == 0) return make_pair(1, 0);
    else {

```

```

int p = a / b;
pii q = extgcd(b, a % b);
return make_pair(q.second, q.first - q.second * p);
}
}

```

FFT

```

const double pi = atan(1.0)*4;
struct Complex {
    double x,y;
    Complex(double _x=0,double _y=0)
        :x(_x),y(_y) {}
    Complex operator + (Complex &tt) { return Complex(x
        +tt.x,y+tt.y); }
    Complex operator - (Complex &tt) { return Complex(x
        -tt.x,y-tt.y); }
    Complex operator * (Complex &tt) { return Complex(x
        *tt.x-y*tt.y,x*tt.y+y*tt.x); }
};
void fft(Complex *a, int n, int rev) {
    // n是大于等于相乘的两个数组长度的2的幂次
    // 从0开始表示长度，对a进行操作
    // rev==1进行DFT，== -1进行IDFT
    for (int i = 1, j = 0; i < n; ++ i) {
        for (int k = n >> 1; k > (j^=k); k >>= 1);
        if (i < j) std::swap(a[i], a[j]);
    }
    for (int m = 2; m <= n; m <= 1) {
        Complex wm(cos(2*pi*rev/m), sin(2*pi*rev/m));
        for (int i = 0; i < n; i += m) {
            Complex w(1.0, 0.0);
            for (int j = i; j < i+m/2; ++ j) {
                Complex t = w*a[j+m/2];
                a[j+m/2] = a[j] - t;
                a[j] = a[j] + t;
                w = w * wm;
            }
        }
    }
    if (rev == -1) {
        for (int i = 0; i < n; ++ i) a[i].x /= n, a[i].y
            /= n;
    }
}

```

Hash

```

typedef long long LL;
LL X=7122;
LL P1=712271227;
LL P2=179433857;
LL P3=179434999;

struct HASH{
    LL a, b, c;
    HASH(LL a=0, LL b=0, LL c=0):a(a),b(b),c(c){ }
    HASH operator + (HASH B){
        return HASH((a+B.a)%P1,(b+B.b)%P2,(c+B.c)%P3);
    }
    HASH operator + (LL B){
        return (*this)+HASH(B,B,B);
    }
    HASH operator * (LL B){
        return HASH(a*B%P1,a*B%P2,a*B%P3);
    }
    bool operator < (const HASH &B)const{
        if (a!=B.a)return a<B.a;
        if (b!=B.b)return b<B.b;
        return c<B.c;
    }
    void up(){ (*this) = (*this)*X; }
};

int main(){
}

```

Convex Hull

```

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

typedef long long LL;
const int MAXN = 100005;
const LL INF = (1LL)<<62;

struct Point{
    LL x, y;
    Point (LL x=0, LL y=0):x(x),y(y){}
    bool operator < (const Point &B)const {
        if (x!=B.x)return x<B.x;
        return y<B.y;
    }
    Point operator - (Point B){
        return Point(x-B.x,y-B.y);
    }
};

LL cross(Point A, Point B){
    return A.x*B.y-A.y*B.x;
}

LL Abs(LL x){
    return x>0?x:-x;
}

LL AreaU[MAXN], AreaD[MAXN];
void find_CH(int N, Point P[], LL Area[]){
    static vector<Point> U, D;
    static vector<int> Ui, Di;
    U.clear(), Ui.clear();
    D.clear(), Di.clear();
    int uz=0, dz=0;

    for (int i=0; i<N; i++){
        while (uz>=2 && cross(P[i]-U[uz-2],U[uz-1]-U[uz-2])<=0)U.pop_back(), Ui.pop_back(), uz--;
        if (uz<=1)AreaU[i]=0;
        else AreaU[i] = AreaU[ Ui[uz-1] ] + Abs(cross(P[i]-U[0],U[uz-1]-U[0]));
        U.push_back(P[i]),Ui.push_back(i),uz++;

        while (dz>=2 && cross(P[i]-D[dz-2],D[dz-1]-D[dz-2])>=0)D.pop_back(), Di.pop_back(), dz--;
        if (dz<=1)AreaD[i]=0;
        else AreaD[i] = AreaD[ Di[dz-1] ] + Abs(cross(P[i]-D[0],D[dz-1]-D[0]));
        D.push_back(P[i]),Di.push_back(i),dz++;

        Area[i] = AreaU[i] + AreaD[i];
        //printf("Area[%d]=%lld\n",i,Area[i]);
    }
    //puts("");
}

int N;
Point P[MAXN];
LL AreaL[MAXN], AreaR[MAXN];

int main(){
    input();

    find_CH(N,P,AreaL);
    for (int i=0; i<N; i++)P[i].x*=-1;
    reverse(P,P+N);
    find_CH(N,P,AreaR);
    reverse(AreaR,AreaR+N);
    reverse(P,P+N);

    LL Ans = min(AreaL[N-1],AreaR[0]);
    for (int i=0; i<N-1; i++){
        if (P[i].x!=P[i+1].x){
            Ans = min (Ans,AreaL[i]+AreaR[i+1]);
        }
    }
    if (P[0].x==P[N-1].x)Ans=0;
    printf("%lld\n",(Ans+1)/2LL);
}

```

Eratosthenes

```
bool p[MAXP] = {0};

void eratosthenes() {
    p[0] = p[1] = 0;
    for (int i = 2; i < MAXP; i++) {
        p[i] = 1;
    }
    for (int i = 2; i < MAXP; i++) {
        if (!p[i]) {
            for (int j = i + i; j < MAXP; j++) {
                p[j] = 0;
            }
        }
    }
}
```

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;
            }
        }
    }
}
```

Inverse

```
int inverse[100000];
void invTable(int b, int p) {
    inverse[1] = 1;
    for( int i = 2; i <= b; i++ ) {
        inverse[i] = (long long)inverse[p%i] * (p-p/i) % p;
    }
}

int inv(int b, int p) {
    return b == 1 ? 1 : ((long long)inv(p % b, p) * (p-p/
    b) % p);
}
```

IterSet

```
// get all subset in set S

for (int i = S; i ; i = (i-1) & S ) {

}
```

LinearPrime

```
int p[MAXN], np = 0;
int a[MAXN];

for ( int i = 2 ; i < n ; i++ ) {
    if ( a[i] == 0 ) p[np++] = i;
    for ( int j = 0; j < np && i * p[j] < n ; j++ ) {
        a[i * p[j]] = 1;
        if ( i % p[j] == 0 ) break;
    }
}
```

SG

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]
}
```

Geometry

Flow

Dinic

```
struct Edge{
    int from, to, cap, flow;
};

const int INF = 1<<29;
const int MAXV = 5003;
struct Dinic{ //O(VVE)
    int n, m, s, t;
    vector<Edge> edges;
    vector<int> G[MAXV];
    bool vis[MAXV];
    int d[MAXV];
    int cur[MAXV];

    void AddEdge(int from, int to, int cap){
        edges.push_back( {from,to,cap,0} );
        edges.push_back( {to,from,0,0} );
        m = edges.size();
        G[from].push_back(m-2);
        G[to].push_back(m-1);
    }

    bool dinicBFS(){
        memset(vis,0,sizeof(vis));
```

```

queue<int> que;
que.push(s); vis[s]=1;
while (!que.empty()){
    int u = que.front(); que.pop();
    for (int ei:G[u]){
        Edge &e = edges[ei];
        if (!vis[e.to] && e.cap>e.flow ){
            vis[e.to]=1;
            d[e.to] = d[u]+1;
            que.push(e.to);
        }
    }
}
return vis[t];
}

int dinicDFS(int u, int a){
    if (u==t || a==0)return a;
    int 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.to])continue;
        f = dinicDFS(e.to, min(a, e.cap-e.flow) );
        if (f>0){
            e.flow += f;
            edges[ G[u][i]^1 ].flow -=f;
            flow += f;
            a -= f;
            if (a==0)break;
        }
    }
    return flow;
}

int maxflow(int s, int t){
    this->s = s, this->t = t;
    int flow=0, mf;
    while ( dinicBFS() ){
        memset(cur,0,sizeof(cur));
        while ( (mf=diniDFS(s,INF)) )flow+=mf;
    }
    return flow;
}
};

```

Graph

Strongly Connected Component(SCC)

```

#define MXN 100005
#define PB push_back
#define FZ(s) memset(s,0,sizeof(s))

struct Scc{
    int n, nScc, vst[MXN], bln[MXN];
    vector<int> E[MXN], rE[MXN], vec;
    void init(int _n){
        n = _n;
        for (int i=0; i<MXN; i++){
            E[i].clear();
            rE[i].clear();
        }
    }
    void add_edge(int u, int v){
        E[u].PB(v);
        rE[v].PB(u);
    }
    void DFS(int u){
        vst[u]=1;
        for (auto v : E[u])
            if (!vst[v]) DFS(v);
        vec.PB(u);
    }
    void rDFS(int u){
        vst[u] = 1;
        bln[u] = nScc;
        for (auto v : rE[u])
            if (!vst[v]) rDFS(v);
    }
};

```

```

}
void solve(){
    nScc = 0;
    vec.clear();
    FZ(vst);
    for (int i=0; i<n; i++)
        if (!vst[i]) DFS(i);
    reverse(vec.begin(),vec.end());
    FZ(vst);
    for (auto v : vec){
        if (!vst[v]){
            rDFS(v);
            nScc++;
        }
    }
}
};

```

Euler Circuit

```

//CF 723E
#include <bits/stdc++.h>
using namespace std;

const int MAXN = 300;

struct EDGE{
    int u ,v ;
    int type;
};

int n, m, deg[MAXN];
vector <EDGE> edges;
vector<int> G[MAXN];
bool vis[MAXN*MAXN];
bool alive[MAXN][MAXN];
bool visN[MAXN];
vector<int> ans;

void add_edge(int u, int v, int type=0){
    edges.push_back( EDGE{u,v,type} );
    edges.push_back( EDGE{v,u,type} );
    G[u].push_back( edges.size()-2 );
    G[v].push_back( edges.size()-1 );
    deg[u]++, deg[v]++;
    alive[u][v]=alive[v][u]=type^1;
}

void input(){
    memset(visN,0,sizeof(visN));
    memset(vis,0,sizeof(vis));
    memset(alive,0,sizeof(alive));
    memset(deg,0,sizeof(deg));
    edges.clear();
    ans.clear();
    for (int i=0; i<MAXN; i++)G[i].clear();

    scanf("%d%d",&n ,&m);
    for (int i=0, u, v; i<m; i++){
        scanf("%d%d", &u, &v);
        add_edge(u,v);
    }
}

void add_Graph(){
    vector<int> tmp;
    for (int i=1; i<=n; i++)if (deg[i]%2==1){
        tmp.push_back(i);
    }
    printf("%d\n",n-tmp.size());
    for (int i=0; i<tmp.size(); i+=2){
        add_edge(tmp[i],tmp[i+1],1);
    }
}

void dfs(int u){
    visN[u]=1;
    for (int i=0; i<G[u].size(); i++)if (!vis[ G[u][i]
        ]>1 ){
        EDGE &e = edges[ G[u][i] ];
    }
}

```

```

        int v = e.v;
        vis[ G[u][i]>>1 ]=1;
        dfs(v);
    }
    ans.push_back(u);
}

int main(){
    int T; scanf("%d",&T);
    while (T--){
        input();
        add_Graph();
        for (int i=1; i<=n; i++){
            if (!visN[i]){
                dfs(i);
                for (int j=0 ;j<ans.size()-1; j++){
                    int u = ans[j], v=ans[j+1];
                    if (alive[u][v]){
                        alive[u][v]=alive[v][u]=0;
                        printf("%d %d\n",u ,v);
                    }
                }
                ans.clear();
            }
        }
    }
}

```

Hungarian

```

vector<int> G[MAXN];
int n;
int match[MAXN]; // Matching Result
int visit[MAXN];

bool dfs(int u) {
    for ( auto v:G[u] ) {
        if (!visit[v]) {
            visit[v] = true;
            if (match[v] == -1 || dfs(match[v])) {
                match[v] = u;
                match[u] = v;
                return true;
            }
        }
    }
    return false;
}

int hungarian() {
    int res = 0;
    memset(match, -1, sizeof(match));
    for (int i = 0; i < n; i++) {
        if (match[i] == -1) {
            memset(visit, 0, sizeof(visit));
            if (dfs(i)) res += 1;
        }
    }
    return res;
}

```

Maximum Clique

```

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;
}

```

Tarjan

```

int n;
vector<int> G[MAXN];
stack<int> stk;
int dfn[MAXN], low[MAXN];
bool ins[MAXN];
int scc[MAXN], scn, count;

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]);
        }
    }

    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(){
    count = scn = 0;
    for(int i = 0 ; i < n ; i++){
        if(!dfn[i]) tarjan(i);
    }
}

```

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;
}

```

Data Structure

Disjoint Set

```

struct DisjointSet{
    int n, fa[MAXN];

    void init(int size) {
        for (int i = 0; i <= size; i++) {
            fa[i] = i;
        }
    }

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

    void unite(int x, int y) {
        p[find(x)] = find(y);
    }
} djs;

```

Sparse Table

```

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] );
    }
};

```

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=0;
        else if(size(o)==k)a=o,b=NULL;
        else{
            o->down();
            if(k<=size(o->l)){
                b=0;
                split(o->l,a,b->l,k);
                b->up();
            }else{
                a=0;
                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);
    }
};
treap<int ,20141223>bst;
int n,m,a,b;
int main(){
    //當成二分查找樹用
    while(~scanf("%d",&a))bst.insert(bst.root,a,bst.rank(
        a));
    while(~scanf("%d",&a))printf("%d\n",bst.rank(a));
    bst.clear(bst.root);
    return 0;
}

```

String

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(n,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;
}

```

AC

```

// by bcw_codebook
struct ACautomata{
    struct Node{
        int cnt,dp;
        Node *go[26], *fail;
        Node (){
            cnt = 0;
            dp = -1;
            memset(go,0,sizeof(go));
            fail = 0;
        }
    };

    Node *root, pool[1048576];
    int nMem;

    Node* new_Node(){
        pool[nMem] = Node();
        return &pool[nMem++];
    }

    void init(){
        nMem = 0;
        root = new_Node();
    }

    void add(const string &str){
        insert(root,str,0);
    }
}

```

```

void insert(Node *cur, const string &str, int pos){
    if (pos >= (int)str.size()){
        cur->cnt++;
        return;
    }
    int c = str[pos]-'a';
    if (cur->go[c] == 0){
        cur->go[c] = new_Node();
    }
    insert(cur->go[c],str,pos+1);
}

void make_fail(){
    queue<Node*> que;
    que.push(root);
    while (!que.empty()){
        Node* fr=que.front();
        que.pop();
        for (int i=0; i<26; i++){
            if (fr->go[i]){
                Node *ptr = fr->fail;
                while (ptr && !ptr->go[i]) ptr = ptr->fail;
                if (!ptr) fr->go[i]->fail = root;
                else fr->go[i]->fail = ptr->go[i];
                que.push(fr->go[i]);
            }
        }
    }
}

```

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;
}

```

Dark Code

輸入優化

```

#include <stdio.h>

char getc(){
    static const int bufsize = 1<<16;
    static char B[bufsize], *S=B, *T=B;
    return (S==T&&(T=(S=B)+fread(B,1,bufsize,stdin),S=T)
        ?0:*S++);
}

template <class T>
bool input(T& a){
    a=(T)0;
    register char p;
    while ((p = getc()) < '-')
        if (p==0 || p==EOF) return false;
    if (p == '-')
        while ((p = getc()) >= '0') a = a*10 - (p^'0');
    else {
        a = p ^ '0';
        while ((p = getc()) >= '0') a = a*10 + (p^'0');
    }
    return true;
}

template <class T, class... U>
bool input(T& a, U&... b){
    if (!input(a)) return false;
    return input(b...);
}

```


Search

Others

Persistence