ACM Template

Parallelc

October 19, 2017

```
syntax on
set cin
set nu
set ts=4
set sw=4
set et
set sta
set cino=g1
map<C-T> :call Run()<CR>
func! Run()
exec "w"
exec "!g++ -O2 -Wall -std=c++11 % -o %<"
exec "!./%<"
endfunc
```

Contents

1 图论

1.1 拓扑排序

```cpp
#include <bits/stdc++.h>
using namespace std;
vector<int> topsort(vector<vector<int>>& lj, vector<int>& rd) {  // !!! & rd
    int n = rd.size();
    stack<int, vector<int>> s;
    for (int i = 0; i < n; i++) {
        if (rd[i] == 0) s.push(i);
    }
    vector<int> ans;
    ans.reserve(n);
    while (!s.empty()) {
        // if (s.size() != 1)  // not only one answer
        int k = s.top();
        s.pop();
        ans.push_back(k);
        for (auto i : lj[k]) {
            rd[i]--;
            if (rd[i] == 0) s.push(i);
        }
    }
    if (ans.size() != n) return {};  // fail
    return ans;
}
```

1.2 最小生成树

1.2.1 Prim 堆优化

```cpp
// Copyright 2017 Parallelc
// o(ElogV) sparse map, from 1 to n
#include <bits/stdc++.h>
using namespace std;  // NOLINT
using LL = int64_t;
using T = int;
const T INF = 0x3f3f3f3f;
struct edge {
    int v;
    T w;
    bool operator<(const edge& e) const {return w > e.w;}
};
T prim(const vector<vector<edge>>& lj) {
    int num = 0, n = lj.size();
    T qz = 0;
    vector<T> a(n, INF);
    a[0] = 0;
    priority_queue<edge> q;
    q.push({0, 0});
    edge mini;
    while (!q.empty() && num < n) {
        do {
            mini = q.top();
            q.pop();
        }while(mini.w > a[mini.v] && !q.empty());
        if (mini.w == a[mini.v]) {
            qz += mini.w;
```

```
            num++;
            for (auto& i : lj[mini.v]) {
                if (a[i.v] > i.w) {
                    a[i.v] = i.w;
                    q.push({i.v, i.w});
                }
            }
        }
    }
    if (num == n) return qz;
    else return INF;
}
```

1.2.2 Kruskal

```
// Copyright 2017 Parallelc
// o(ElogE) sparse map, from 1 to n
#include <bits/stdc++.h>
using namespace std;  // NOLINT
using LL = int64_t;
using T = int;
const T INF = 0x3f3f3f3f;

vector<int> bcj;
int gr(int k) {
    return k == bcj[k]?k:bcj[k] = gr(bcj[k]);
}

struct edge {
    int u, v;
    T w;
    bool operator<(const edge& a) const {return w < a.w;}
};
T kruskal(vector<edge>& ed, int n) {
    bcj.resize(n);
    iota(bcj.begin(), bcj.end(), 0);
    sort(ed.begin(), ed.end());
    T qz = 0;
    int us_num = 0;
    for (auto& i : ed) {
        if (gr(i.u) != gr(i.v)) {
            bcj[gr(i.u)] = gr(i.v);
            qz += i.w;
            us_num++;
        }
        if (us_num == n - 1) break;
    }
    if (us_num == n - 1) return qz;
    else return INF;
}
```

1.3 最短路

1.3.1 Dijkstra 堆优化

```
// Copyright 2017 Parallelc
// o((E+V)logV)
#include <bits/stdc++.h>
using namespace std;  // NOLINT
```

```
using LL = int64_t;
using T = int;
const T INF = 0x3f3f3f3f;
struct edge {
    int v;
    T w;
    bool operator< (const edge& e) const {return w > e.w;}
};
T dij(const vector<vector<edge>>& lj, int S, int N) {
    int n = lj.size();
    vector<int> pre(n);
    vector<T> a(n, INF);
    a[S] = 0;
    priority_queue<edge> q;
    q.push({S, 0});
    edge mini;
    while (!q.empty()) {
        do {
            mini = q.top();
            q.pop();
        }while(mini.w > a[mini.v] && !q.empty());
        if (mini.v == N) break;
        if (mini.w == a[mini.v]) {
            for (auto& i : lj[mini.v]) {
                T k = mini.w + i.w;
                if (a[i.v] > k) {
                    a[i.v] = k;
                    q.push({i.v, k});
                    pre[i.v] = mini.v;
                }
            }
        }
    }
    if (a[N] != INF) {
        function<void(int)> pri = [&](int k)->void {
            if (k != S) pri(pre[k]);
            cout << k;
            if (k != N) cout << ' ';
        };
        pri(N);
    }
    return a[N];
}
```

1.3.2 SPFA

```
// Copyright 2016 Parallelc
#include <bits/stdc++.h>
using namespace std;  // NOLINT
using T = int;
const T INF = 0x3f3f3f3f;
struct node {
    int v;
    T w;
};
T spfa(vector<vector<node>>& lj, int S, int N) {
    int n = lj.size();
    static vector<T> dis;
    fill(dis.begin(), dis.end(), INF), dis.resize(n, INF);
```

```
    static vector<int> us, cnt, pre;
    fill(us.begin(), us.end(), 0), us.resize(n);
    fill(cnt.begin(), cnt.end(), 0), cnt.resize(n);
    pre.resize(n);
    us[S] = 1;
    dis[S] = 0;
    queue<int> q;
    q.push(S);
    cnt[S] = 1;
    pre[S] = -1;
    while (!q.empty()) {
        int now = q.front();
        q.pop();
        us[now] = 0;
        for (auto& i : lj[now]) {
            if (dis[i.v] > dis[now] + i.w) {
                dis[i.v] = dis[now] + i.w;
                pre[i.v] = now;
                if (!us[i.v]) {
                    us[i.v] = 1;
                    q.push(i.v);
                    if (++cnt[i.v] > n) return -1;
                }
            }
        }
    }
    return dis[N];
}
/*
// u -> v shortest == max(u - v) == -min(v - u)
lj[u].push_back({v, w});  // u - v <= w
lj[i].push_back({i - 1, 1});  // (i) - (i - 1) <= 1
lj[i - 1].push_back({i, 0});  // (i - 1) - (i) <= 0
if (dis[N] == INF) return -2;  // -1 no solution, -2 any solution
*/
```

1.3.3 k 短路

```
#include <bits/stdc++.h>
using namespace std;
using LL = int64_t;
using T = int;
const T INF = 0x3f3f3f3f;
struct node {
    int v;
    T w, g;
    bool operator< (const node& e) const {return w > e.w;}
};
vector<T> dij(const vector<vector<node>>& lj, int S, int N) {
    int n = lj.size();
    vector<T> a(n, INF);
    a[S] = 0;
    priority_queue<node> q;
    q.push({S, 0});
    node mini;
    while (!q.empty()) {
        do {
            mini = q.top();
            q.pop();
```

```
        }while(mini.w > a[mini.v] && !q.empty());
        if (mini.w == a[mini.v]) {
            for (auto& i : lj[mini.v]) {
                T k = mini.w + i.w;
                if (a[i.v] > k) {
                    a[i.v] = k;
                    q.push({i.v, k});
                }
            }
        }
    }
    return a;
}

T kshort(vector<vector<node>>& lj, int S, int N, int k) {
    int n = lj.size();
    auto h = dij(lj, N, S);
    if (h[S] == INF) return INF;
    vector<int> num(n);
    node now = {S, h[S], 0};
    priority_queue<node> q;
    q.push(now);
    while (!q.empty()) {
        now = q.top();
        q.pop();
        num[now.v]++;
        if (num[N] == k) return now.w;
        if (num[now.v] > k) continue;
        for (auto& i : lj[now.v]) {
            node tmp;  // new node
            tmp.v = i.v;
            if (h[tmp.v] == INF) continue;
            tmp.g = now.g + i.w;
            tmp.w = tmp.g + h[tmp.v];
            q.push(tmp);
        }
    }
    return INF;
}
```

1.4 网络流

1.4.1 最大流

```
#include <bits/stdc++.h>
using namespace std;
using T = int;
const T INF = 0x3f3f3f3f;
class dinic {
 private:
     struct node {
         int num;
         T cap;
         int rev;
     };
     vector<vector<node>> lj;
     vector<int> dis, cur;
     int s, d;
     int bfs() {
```

```
        fill(dis.begin(), dis.end(), 0);
        dis[s] = 1;
        queue<int> q;
        q.push(s);
        while (!q.empty()) {
            int p = q.front();
            q.pop();
            for (auto &i : lj[p]) {
                if (!dis[i.num] && i.cap > 0) {
                    dis[i.num] = dis[p] + 1;
                    if (i.num == d) return 1;
                    q.push(i.num);
                }
            }
        }
        return 0;
    }
    T dfs(int p, T low = INF) {
        if (p == d) return low;
        T flow = 0;
        for (int &i = cur[p]; i < lj[p].size(); i++) {
            auto &j = lj[p][i];
            if (dis[j.num] == dis[p] + 1 && j.cap > 0) {
                int k = dfs(j.num, min(low, j.cap));
                j.cap -= k;
                lj[j.num][j.rev].cap += k;
                flow += k;
                low -= k;
                if (low == 0) break;
            }
        }
        return flow;
    }

public:
    dinic(int n) {
        lj.resize(n);
        dis.resize(n);
        cur.resize(n);
    }
    void add(int u, int v, int w) {
        lj[u].push_back({v, w, lj[v].size()});
        lj[v].push_back({u, 0, lj[u].size() - 1});
    }
    T solve(int s, int d) {
        T ans = 0;
        this->s = s;
        this->d = d;
        while (bfs()) {
            fill(cur.begin(), cur.end(), 0);
            ans += dfs(s);
        }
        return ans;
    }
};
```

1.4.2 费用流

```cpp
// Copyright 2016 Parallelc
#include <bits/stdc++.h>
using namespace std;  // NOLINT
using T = int;
using L = int;
const int INF = 0x3f3f3f3f;
class minCost {
 private:
    struct node{
        int v;
        L cap;
        T cost;
        int rev;
    };
    vector<vector<node>> lj;
    vector<bool> vis;
    vector<int> cnt, pre;
    vector<T> dis;
    bool SPFA(int start, int end) {
        fill(dis.begin(), dis.end(), INF);
        fill(vis.begin(), vis.end(), 0);
        fill(cnt.begin(), cnt.end(), 0);
        vis[start] = true;
        dis[start] = 0;
        queue<int> que;
        que.push(start);
        cnt[start] = 1;
        pre[start] = -1;
        while (!que.empty()) {
            int u = que.front();
            que.pop();
            vis[u] = false;
            for (auto& i : lj[u]) {
                if (i.cap && dis[i.v] > dis[u] + i.cost) {
                    dis[i.v] = dis[u] + i.cost;
                    pre[i.v] = i.rev;
                    if (!vis[i.v]) {
                        vis[i.v] = true;
                        que.push(i.v);
                        if (++cnt[i.v] > n) return false;
                    }
                }
            }
        }
        // if (dis[end] >= 0) return false; // not maxFlow
        if (dis[end] == INF) return false;
        return true;
    }

 public:
    int n;
    minCost() {}
    minCost(int n) : n(n) {
        lj.resize(n);
        vis.resize(n);
        cnt.resize(n);
        dis.resize(n);
```

```
        pre.resize(n);
    }
    void add(int u, int v, L w, T c) {
        lj[u].push_back({v, w, c, lj[v].size()});
        lj[v].push_back({u, 0, -c, lj[u].size() - 1});
    }
    pair<L, T> solve(int s, int t) {
        L flow = 0;
        T cost = 0;
        while (SPFA(s, t)) {
            L Min = INF;
            for (int i = t; pre[i] != -1; i = lj[i][pre[i]].v) {
                auto& j = lj[i][pre[i]];
                Min = min(Min, lj[j.v][j.rev].cap);
            }
            for (int i = t; pre[i] != -1; i = lj[i][pre[i]].v) {
                auto& j = lj[i][pre[i]];
                auto& k = lj[j.v][j.rev];
                k.cap -= Min;
                j.cap += Min;
            }
            cost += dis[t] * Min;
            flow += Min;
        }
        return {flow, cost};
    }
};
```

1.5 二分图匹配

1.5.1 匈牙利算法

(1) DFS

```
// Copyright 2016 Parallelc
#include <bits/stdc++.h>
using namespace std;  // NOLINT
int hungary(const vector<vector<int>>& lj, int m) {
    int n = lj.size();
    static vector<int> rlink, us;  //, llink;
    // llink.resize(n), fill(llink.begin(), llink.end(), -1);
    rlink.resize(m), fill(rlink.begin(), rlink.end(), -1);
    us.resize(m), fill(us.begin(), us.end(), -1);
    static function<bool(int, int)> dfs = [&](int k, int d) {  // make k link to right
        for (auto i : lj[k]) {
            if (us[i] != d) {
                us[i] = d;
                if (rlink[i] == -1|| dfs(rlink[i], d)) {
                    rlink[i] = k;  // make or change link
                    // llink[k] = i;
                    return true;
                }
            }
        }
        return false;
    };
    int num = 0;  // max linked egde number
    for (int i = 0; i < n; i++) {
        if (dfs(i, i)) {  // link successfully
```

```
            num++;
        }
    }
    return num;
}
```

(2) BFS

```
// Copyright 2016 Parallelc
#include <bits/stdc++.h>
using namespace std;  // NOLINT
int hungary(const vector<vector<int>>& lj, int m) {
    int n = lj.size();
    static vector<int> llink, rlink, pre, us;
    llink.resize(n), fill(llink.begin(), llink.end(), -1);
    rlink.resize(m), fill(rlink.begin(), rlink.end(), -1);
    pre.resize(n), fill(pre.begin(), pre.end(), -1);
    us.resize(m), fill(us.begin(), us.end(), -1);
    int ans = 0;
    for (int i = 0; i < n; i++) {
        if (llink[i] == -1) {
            queue<int> q;
            q.push(i);
            bool flag = 0;
            while (!q.empty() && !flag) {
                int u = q.front();
                q.pop();
                for (auto v : lj[u]) {
                    if (us[v] != i) {
                        us[v] = i;
                        if (rlink[v] != -1) {
                            q.push(rlink[v]);
                            pre[rlink[v]] = u;
                        } else {
                            flag = 1;
                            while (u != -1) {
                                int temp = llink[u];
                                rlink[v] = u;
                                llink[u] = v;
                                u = pre[u];
                                v = temp;
                            }
                            break;
                        }
                    }
                }
            }
        }
        if (llink[i] != -1) ans++;
    }
    return ans;
}
```

1.5.2 最大权匹配

```
// Copyright 2016 Parallelc
#include <bits/stdc++.h>
using namespace std;  // NOLINT
const int INF = 0x3f3f3f3f;
```

```cpp
template<typename T>
T KM(vector<vector<T>>& lj) {
    int n = lj.size();  // left num
    int m = lj[0].size();  // right num
    static vector<int> rlink; //, llink;  // right link to left, left link to right
    //llink.resize(n), fill(llink.begin(), llink.end(), -1);
    rlink.resize(m), fill(rlink.begin(), rlink.end(), -1);
    vector<T> fl, fr;  // flag num
    fl.resize(n), fill(fl.begin(), fl.end(), -INF);
    fr.resize(m), fill(fr.begin(), fr.end(), 0);
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            fl[i] = max(fl[i], lj[i][j]);
        }
    }
    static vector<T> slack;
    slack.resize(m);
    static vector<int> lused, rused;  // record used point in a dfs
    lused.resize(n), rused.resize(m);
    static function<bool(int)> dfs = [&](int k) {  // find a method that can make k link to
    ↪ right
        lused[k] = 1;
        for (int i = 0; i < m; i++) {
            if (lj[k][i] == -INF || rused[i]) continue;
            T tmp = fl[k] + fr[i] - lj[k][i];
            if (!tmp) {
                rused[i] = 1;
                if (rlink[i] == -1 || dfs(rlink[i])) {
                    rlink[i] = k;  // make or change link
                    // llink[k] = i;
                    return true;
                }
            } else slack[i] = min(slack[i], tmp);
        }
        return false;
    };
    for (int d = 0; d < n; d++) {
        fill(slack.begin(), slack.end(), INF);
        while (1) {
            fill(lused.begin(), lused.end(), 0);
            fill(rused.begin(), rused.end(), 0);
            if (dfs(d)) break;
            T e = INF;
            for (int i = 0; i < m; i++) {  // min slack
                if (!rused[i]) e = min(e, slack[i]);
            }
            if (e == INF) return -1;
            for (int i = 0; i < n; i++) {
                if (lused[i]) fl[i] -= e;
            }
            for (int i = 0; i < m; i++) {
                if (rused[i]) fr[i] += e;
                else if (lj[d][i] != -INF) slack[i] -= e;
            }
        }
    }
    T res = 0;
    for (int i = 0; i < n; i++) {
```

```
        res += lj[rlink[i]][i];
    }
    return res;
}
```

1.6 最近公共祖先

1.6.1 ST

```
#include <bits/stdc++.h>
using namespace std;
template <typename T, class op = less<pair<T, int>>>
struct RMQ {
    vector<vector<pair<T, int>>> dp;
    RMQ() {}
    RMQ(vector<T>& a) {
        int n = a.size();
        int m = __lg(n) + 1;
        dp.resize(n);
        for (int i = 0; i < n; i++) {
            dp[i].resize(m);
            dp[i][0] = {a[i], i};
        }
        for (int j = 1; j < m; j++) {
            for (int i = 0; i + (1 << j) <= n; i++) {
                dp[i][j] = min(dp[i][j - 1], dp[i + (1 << (j - 1))][j - 1], op());
            }
        }
    }
    pair<T, int> que(int l, int r) {
        int k = 31 - __builtin_clz(r - l);
        return min(dp[l][k], dp[r - (1 << k)][k], op());
    }
};

using T = int;
struct node {
    int v;
    T w;
};
struct LCA {
    vector<int> dep, pos, olx;
    vector<T> dis;
    RMQ<int> st;
    LCA(vector<vector<node>>& lj, int r) {
        int n = lj.size();
        dep.reserve(2 * n);
        olx.reserve(2 * n);
        pos.resize(n);
        dis.resize(n);
        function<void(int, int)> dfs = [&](int k, int pre) {
            pos[k] = olx.size();
            olx.push_back(k);
            dep.push_back(dep.size() ? dep.back() + 1 : 1);
            for (auto& i : lj[k]) {
                if (i.v != pre) {
                    dis[i.v] = dis[k] + i.w;
                    dfs(i.v, k);
                    olx.push_back(k);
```

```
                    dep.push_back(dep.back() - 1);
                }
            }
        };
        dfs(r, r);
        st = RMQ<int>(dep);
    }
    int que(int u, int v) {
        if (pos[u] > pos[v]) swap(u, v);
        return olx[st.que(pos[u], pos[v] + 1).second];
    }
};
```

1.6.2 倍增

```
#include <bits/stdc++.h>
using namespace std;
using T = int;
struct node {
    int v;
    T w;
};
struct LCA {
    vector<int> dep;
    vector<vector<int>> fa;
    vector<T> dis;
    LCA(vector<vector<node>>& lj, int r) {
        int n = lj.size();
        int m = log2(n) + 1;
        fa.resize(n);
        for (auto& i : fa) i.resize(m);
        dep.resize(n);
        dis.resize(n);
        function<void(int, int)> dfs = [&](int k, int pre) {
            fa[k][0] = pre;
            dep[k] = dep[pre] + 1;
            for (auto& i : lj[k]) {
                if (i.v != pre) {
                    dis[i.v] = dis[k] + i.w;
                    dfs(i.v, k);
                }
            }
        };
        dfs(r, r);
        for (int j = 1; j < m; j++) {
            for (int i = 0; i < n; i++) {
                fa[i][j] = fa[fa[i][j - 1]][j - 1];
            }
        }
    }
    int que(int u, int v) {
        if (dep[u] < dep[v]) swap(u, v);
        for (int i = log2(dep[u]); i >= 0; i--) {
            if (dep[u] - (1 << i) >= dep[v]) u = fa[u][i];
        }
        if (u == v) return u;
        for (int i = log2(dep[u]); i >= 0; i--) {
            if (fa[u][i] != fa[v][i]) {
                u = fa[u][i];
```

```
                v = fa[v][i];
            }
        }
        return fa[u][0];
    }
};
```

1.6.3 tarjan

```
// Copyright 2017 Parallelc
#include <bits/stdc++.h>
using namespace std;  // NOLINT
using T = int;

vector<int> bcj;
int gr(int k) {
    return k == bcj[k]?k:bcj[k] = gr(bcj[k]);
}

struct node {
    int v;
    T w;
};
void LCA(vector<vector<node>>& lj, int r, vector<node>& p) {
    int n = lj.size(), q = p.size();
    vector<vector<node>> que(n);
    for (int i = 0; i < q; i++) {
        que[p[i].v].push_back({p[i].w, i});
        que[p[i].w].push_back({p[i].v, i});
    }
    bcj.resize(n);
    iota(bcj.begin(), bcj.end(), 0);
    vector<int> ans(q);
    vector<T> dis(n);
    function<void(int, int)> dfs = [&](int k, int pre) {
        for (auto& i : lj[k]) {
            if (i.v != pre) {
                dis[i.v] = dis[k] + i.w;
                dfs(i.v, k);
                bcj[i.v] = k;
            }
        }
        for (auto& i : que[k]) {
            if (bcj[i.v] != i.v || i.v == k) {
                ans[i.w] = gr(i.v);
            }
        }
    };
    dfs(r, r);
    // TODO(Parallelc)
}
```

1.7 dfs 序

```
#include <bits/stdc++.h>
using namespace std;
vector<int> in, out;
template <typename T>
vector<T> dfsx(vector<vector<int>>& lj, int r, vector<T>& dq) {
```

```
    int n = lj.size();
    in.resize(n), out.resize(n);
    vector<T> ans;
    ans.reserve(2 * n);
    function<void(int, int)> dfs = [&](int k, int pre) {
        in[k] = ans.size(); // [
        ans.push_back(dq[k]);
        for (auto i : lj[k]) {
            if (i == pre) continue;
            dfs(i, k);
        }
        out[k] = ans.size();  // )
        // ans.push_back(-dq[k]);  // ], sum of [0, in[k]] == sum from root to k
    };
    dfs(r, r);
    return ans;
}
```

1.8 树链剖分

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

struct node {
    int v;
    T w;
};
struct HDL {
    vector<int> fa, son, top, dep, num, p, fp;
    vector<T> bq, dis;
    HDL(vector<vector<node>>& lj, int r) {
        int n = lj.size();
        int pos = 0;
        fa.resize(n), son.resize(n, -1), top.resize(n), dep.resize(n);
        num.resize(n), p.resize(n), fp.resize(n), bq.resize(n);
        function<void(int, int)> dfs = [&](int k, int pre) {
            fa[k] = pre;
            dep[k] = dep[pre] + 1;
            num[k] = 1;
            for (auto& i : lj[k]) {
                if (i.v != pre) {
                    dfs(i.v, k);
                    num[k] += num[i.v];
                    if (son[k] == -1 || num[i.v] > num[son[k]]) son[k] = i.v;
                } else {
                    bq[i.v] = i.w;
                }
            }
        };
        function<void(int, int)> getpos = [&](int k, int t) {
            top[k] = t;
            p[k] = pos++;
            fp[p[k]] = k;
            if (son[k] == -1) return;
            getpos(son[k], t);
            for (auto& i : lj[k]) {
                if (i.v != son[k] && i.v != fa[k]) getpos(i.v, i.v);
```

```
            }
        };
        dfs(r, r);
        getpos(r, r);
    }
    int LCA(int u, int v) {
        for (;top[u] != top[v];dep[top[u]] > dep[top[v]]?u = fa[top[u]] : v = fa[top[v]]);
        return dep[u] < dep[v]? u : v;
    }
    vector<pair<int, int>> path(int u, int v) {  // [,]
        vector<pair<int, int>> a;
        a.reserve(p.size());
        int f1 = top[u], f2 = top[v];
        while (f1 != f2) {
            if (dep[f1] < dep[f2]) {
                swap(f1, f2);
                swap(u, v);
            }
            a.emplace_back(p[f1], p[u] + 1);
            u = fa[f1];
            f1 = top[u];
        }
        if (dep[u] > dep[v]) swap(u, v);
        a.emplace_back(p[u], p[v] + 1);  // point
        // if (u != v) a.emplace_back(p[u] + 1, p[v] + 1);  // edge
        return a;  // [,)
    }
};
```

1.9 连通性

1.9.1 割点-桥

```
// Copyright 2017 Parallelc
#include <bits/stdc++.h>
using namespace std;  // NOLINT
vector<int> tarjan(const vector<vector<int>>& lj) {
    int n = lj.size();
    vector<int> dfn(n), low(n), add_block(n);
    int ind = 1;
    function<void(int, int)> dfs = [&](int k, int pre) {
        dfn[k] = low[k] = ind++;
        int son = 0;
        for (auto i : lj[k]) {
            if (!dfn[i]) {
                son++;
                dfs(i, k);
                low[k] = min(low[k], low[i]);
                // bridge - (i, k), (k, i)
                if (low[i] > dfn[k]) {
                    // todo
                }
                // not root - cut point
                if (k != pre && low[i] >= dfn[k]) {
                    add_block[k]++;
                }
            } else if (i != pre) {
                low[k] = min(low[k], dfn[i]);
            }
```

```
        }
        // root - cut point
        if (k == pre)
            add_block[k] = son - 1;
    };
    for (int i = 0; i < n; i++) {
        if (!dfn[i]) {
            ind = 1;
            dfs(i, i);
        }
    }
    return add_block;
}
```

1.9.2 点双连通分量

```
// Copyright 2017 Parallelc
#include <bits/stdc++.h>
using namespace std;  // NOLINT
vector<vector<pair<int, int>>> tarjan(const vector<vector<int>>& lj, int m) {
    int n = lj.size();
    vector<int> dfn(n), low(n);
    int ind = 1, num = 0;
    stack<pair<int, int>> s;
    vector<vector<pair<int, int>>> edge(m);
    function<void(int, int)> dfs = [&](int k, int pre) {
        dfn[k] = low[k] = ind++;
        for (auto i : lj[k]) {
            if (!dfn[i]) {
                s.emplace(k, i);
                dfs(i, k);
                low[k] = min(low[k], low[i]);
                // find it is cut
                if (dfn[k] <= low[i]) {
                    pair<int, int> tmp;
                    do {
                        tmp = s.top();
                        edge[num].push_back(tmp);
                        s.pop();
                    } while (tmp.first != k || tmp.second != i);
                    num++;
                }
            } else if (i != pre) {
                low[k] = min(low[k], dfn[i]);
                // edge connect to ancestors
                if (dfn[k] > dfn[i])
                    s.emplace(k, i);
            }
        }
    };
    for (int i = 0; i < n; i++) {
        if (!dfn[i]) {
            ind = 1;
            dfs(i, i);
        }
    }
    return edge;
}
```

1.9.3 边双连通分量

```cpp
// Copyright 2017 Parallelc
#include <bits/stdc++.h>
using namespace std;  // NOLINT
vector<int> tarjan(const vector<vector<int>>& lj) {
    int n = lj.size();
    vector<int> dfn(n), low(n), bh(n);
    int ind = 1, num = 1;
    stack<int> s;
    function<void(int, int)> dfs = [&](int k, int pre) {
        dfn[k] = low[k] = ind++;
        for (auto i : lj[k]) {
            if (!dfn[i]) {
                s.push(i);
                dfs(i, k);
                low[k] = min(low[k], low[i]);
                // bridge
                if (low[i] > dfn[k]) {
                    int tmp;
                    do {
                        tmp = s.top();
                        bh[tmp] = num;
                        s.pop();
                    } while (tmp != i);
                    num++;
                }
            } else if (i != pre)
                low[k] = min(low[k], dfn[i]);
        }
    };
    for (int i = 0; i < n; i++) {
        if (!dfn[i]) {
            ind = 1;
            dfs(i, i);
        }
    }
    while (!s.empty()) {
        bh[s.top()] = num;
        s.pop();
    }
    return bh;
}
```

1.9.4 有向图强连通分量

```cpp
// Copyright 2017 Parallelc
#include <bits/stdc++.h>
using namespace std;  // NOLINT
vector<vector<int>> tarjan(const vector<vector<int>> &lj) {
    int n = lj.size();
    vector<int> dfn(n), low(n), bh(n, -1);
    int ind = 1;
    int num = 0;
    stack<int, vector<int>> s;
    function<void(int)> dfs = [&](int k) {
        dfn[k] = low[k] = ind++;
        s.push(k);
        for (auto i : lj[k]) {
```

```
            if (!dfn[i]) {
                dfs(i);
                low[k] = min(low[k], low[i]);
            } else if (bh[i] == -1)
                low[k] = min(dfn[i], low[k]);
        }
        if (low[k] == dfn[k]) {
            int tmp;
            do {
                tmp = s.top();
                bh[tmp] = num;
                s.pop();
            } while (tmp != k);
            num++;
        }
    };
    for (int i = 0; i < n; i++) {
        if (bh[i] == -1) {
            ind = 1;
            dfs(i);
        }
    }
    vector<vector<int>> tiny(num);
    for (int i = 0; i < n; i++) {
        for (auto j : lj[i]) {
            if (bh[i] != bh[j]) {
                tiny[bh[i]].push_back(bh[j]); // multiple edge
            }
        }
    }
    return tiny;
}
```

1.9.5 2sat

```
#include <bits/stdc++.h>
using namespace std;
vector<int> tsat(vector<vector<int>>& lj, vector<pair<int, int>>& p) {
    int n = lj.size();
    int num = 0;
    auto tarjan = [&](const vector<vector<int>> &lj) {
        int n = lj.size();
        vector<int> dfn(n), low(n), bh(n, -1);
        int ind = 1;
        stack<int> s;
        function<void(int)> dfs = [&](int k) {
            dfn[k] = low[k] = ind++;
            s.push(k);
            for (auto i : lj[k]) {
                if (!dfn[i]) {
                    dfs(i);
                    low[k] = min(low[k], low[i]);
                } else if (bh[i] == -1)
                    low[k] = min(dfn[i], low[k]);
            }
            if (low[k] == dfn[k]) {
                int tmp;
                do {
                    tmp = s.top();
```

```
                bh[tmp] = num;
                s.pop();
            } while (tmp != k);
            num++;
        }
    };
    for (int i = 0; i < n; i++) {
        if (bh[i] == -1) {
            ind = 1;
            dfs(i);
        }
    }
    return bh;
};
auto bh = tarjan(lj);
vector<int> fx(n);
for (auto& i : p) {
    if (bh[i.first] == bh[i.second]) return vector<int>();
    fx[bh[i.first]] = bh[i.second];
    fx[bh[i.second]] = bh[i.first];
}
vector<int> deg(num);
vector<vector<int>> dag(num);
for (int i = 0; i < n; i++) {
    for (auto j : lj[i]) {
        if (bh[i] != bh[j]) {
            dag[bh[j]].push_back(bh[i]);
            deg[bh[i]]++;
        }
    }
}
stack<int> s;
for (int i = 0; i < num; i++) {
    if (!deg[i]) s.push(i);
}
vector<int> us(num);
while (!s.empty()) {
    int k = s.top();
    s.pop();
    if (us[k] == 0) {
        us[k] = 1;
        us[fx[k]] = -1;
    }
    for (auto i : dag[k]) {
        if (!--deg[i]) s.push(i);
    }
}
vector<int> ans;
ans.reserve(n >> 1);
for (int i = 0; i < n; i++) {
    if (us[bh[i]] == 1) ans.push_back(i);
}
return ans;
}
```

2 数论

2.1 快速幂

```cpp
#include <bits/stdc++.h>
using namespace std;
using LL = int64_t;

LL PowMod(LL a, LL n, LL mod) {
    LL ans = 1;
    while (n) {
        if (n & 1)
            ans = (ans * a) % mod;
        a = (a * a) % mod;
        n >>= 1;
    }
    return ans;
}
```

2.2 矩阵快速幂

```cpp
// Copyright 2017 Parallelc
#include <bits/stdc++.h>
using namespace std;  // NOLINT
using LL = int64_t;
const LL mod = 1e9 + 7;

struct Mat {
    int n;
    static const int N = 3;
    LL val[N][N] = {{0}};
    Mat(int n, int op = 0) : n(n) {
        if (op) {
            for (int i = 0; i < n; i++) val[i][i] = 1;
        }
    }
    friend const Mat operator+ (const Mat& t, const Mat& s) {
        auto ans = t;
        for (int i = 0; i < ans.n; i++) {
            for (int j = 0; j < ans.n; j++) {
                ans.val[i][j] += s.val[i][j];
                if (ans.val[i][j] >= mod) ans.val[i][j] -= mod;
            }
        }
        return ans;
    }
    friend const Mat operator* (const Mat& t, const Mat& s) {
        Mat ans(t.n);
        for (int i = 0; i < ans.n; i++) {
            for (int k = 0; k < ans.n; k++) {
                if (ans.val[i][k]) {
                    for (int j = 0; j < ans.n; j++) {
                        ans.val[i][j] += t.val[i][k] * s.val[k][j] % mod;
                        if (ans.val[i][j] >= mod) ans.val[i][j] -= mod;
                    }
                }
            }
        }
        return ans;
```

```
    }
};

Mat PowMod(Mat a, LL n) {
    Mat ans(a.n, 1);
    while(n) {
        if(n & 1) ans = ans * a;
        a = a * a;
        n >>= 1;
    }
    return ans;
}

Mat PowSumMod(Mat& a, LL n) {// return (a+ a^2 + ... + a^n) Mod p;
    Mat ans(a.n, 1);
    if(n == 1) return a;
    if (n % 2 == 0)
        return (ans + PowMod(a, n / 2)) * PowSumMod(a, n / 2);
    else
        return (ans + PowMod(a, (n - 1) / 2)) * PowSumMod(a, (n - 1) / 2) +
            PowMod(a, n);
}
```

2.3 组合数取模

```
#include <bits/stdc++.h>
using namespace std;
using LL = int64_t;

namespace Cmod {
    const LL mod = 1e9 + 7;
    vector<LL> fac, finv;

    void init(int n) {
        fac.resize(n);
        finv.resize(n);
        fac[0] = 1;
        for (int i = 1; i < n; i++) {
            fac[i] = fac[i - 1] * i % mod;
        }
        finv[1] = 1;
        for (int i = 2; i < n; i++) {
            finv[i] = finv[mod % i] * (mod - mod / i) % mod;
        }
        finv[0] = 1;
        for (int i = 1; i < n; i++) {
            finv[i] = finv[i - 1] * finv[i] % mod;
        }
    }
    LL Comb(LL n, LL m) {
        return fac[n] * finv[m] % mod * finv[n - m] % mod;
    }

    LL Lucas(LL n, LL m) {
        LL ans = 1;
        while (n && m) {
            if (n < m) return 0;
            ans = (ans * Comb(n % mod, m % mod)) % mod;
```

```
            n /= mod;
            m /= mod;
        }
        return ans;
    }
}
```

2.4 斯特林数

```
#include <bits/stdc++.h>
using namespace std;
const int mod = 1e9 + 7;
const int N = 1005;
int S[N][N];
// divide i to j circles
void init1() {
    for (int i = 0; i < N; i++) {
        for (int j = 1; j < i; j++) {
            S[i][j] = (i - 1) * S[i - 1][j] % mod + S[i - 1][j - 1];
            if (S[i][j] >= mod) S[i][j] -= mod;
        }
        S[i][i] = 1;
    }
}

// divide i to j sets
void init2() {
    for (int i = 0; i < N; i++) {
        for (int j = 1; j < i; j++) {
            S[i][j] = j * S[i - 1][j] % mod + S[i - 1][j - 1];
            if (S[i][j] >= mod) S[i][j] -= mod;
        }
        S[i][i] = 1;
    }
}
```

2.5 素数打表

```
// Copyright 2016 Parallelc
#include <bits/stdc++.h>
using namespace std; // NOLINT
using LL = int64_t;
vector<int> prime, pd, u;
void db(int n) {
    pd.resize(n);
    u.resize(n);
    prime.reserve(n);
    u[1] = 1;
    for (int i = 2; i < n; i++) {
        if (pd[i] == 0) {
            prime.push_back(i);
            u[i] = -1;
        }
        for (auto j : prime) {
            if (i * j >= n) break;
            pd[i * j] = 1;
            if (i % j) u[i * j] = -u[i];
            else break;
        }
```

```
    }
}

vector<LL> bprime, bpd;
void getprime(LL l, LL r) {
    int n = r - l;
    bpd.clear();
    bprime.clear();
    bpd.resize(n);
    bprime.reserve(n);
    for (LL i : prime) {
        if (i * i >= r) break;
        LL s = l / i + !!(l % i);
        for (LL j = max(2LL, s); j * i < r; j++) bpd[j * i - l] = 1;
    }
    if (l < 2) for (int i = l; i < 2; i++) bpd[i] = 1;
    for (int i = 0; i < n; i++) if (!bpd[i]) bprime.push_back(l + i);
}
```

2.6 逆元

```
#include <bits/stdc++.h>
using namespace std;
using LL = int64_t;
using T = LL;
T exgcd(T a, T b, T& x, T& y) {  // a solution of ax + by = gcd(a, b)
    if (b == 0) return x = 1, y = 0, a;
    T d = exgcd(b, a % b, y, x);
    y -= a / b * x;
    return d;
}
T inv(T a, T mod) {  // ax = 1(mod p)
    T x, y;
    T d = exgcd(a, mod, x, y);
    if (d == 1) return (x % mod + mod) % mod;
    else return -1;
}
```

2.7 逆元打表

```
#include <bits/stdc++.h>
using namespace std;
using LL = int64_t;

vector<LL> inv;
void init(int n, LL mod) {
    inv.resize(n);
    inv[1] = 1;
    for(int i = 2; i < n; i++)
        inv[i] = (mod - mod / i) * inv[mod % i] % mod;
}
```

2.8 不定方程

```
#include <bits/stdc++.h>
using namespace std;
using LL = int64_t;
using T = LL;
T exgcd(T a, T b, T& x, T& y) {  // a solution of ax + by = gcd(a, b)
```

```
    if (b == 0) return x = 1, y = 0, a;
    T d = exgcd(b, a % b, y, x);
    y -= a / b * x;
    return d;
}
bool equ(T a, T b, T c, T& x, T& y) {  // solutions of ax + by = c || ax = c (mod b)
    T d = exgcd(a, b, x, y);
    if (d == 0) return !c;  // a == b == 0
    if (c % d) return false;
    x *= c / d, y *= c / d;  // a solution
    a /= d, b /= d;  // all solution: x = x + b * t, y = y - a * t; loop of x % b is d
    T tx = x;
    x = (x % b + b) % b, y += (tx - x) / b * a;  // solution of min{Z*}
    return true;
}
```

2.9 中国剩余定理

```
#include <bits/stdc++.h>
using namespace std;
using LL = int64_t;
using T = LL;
T exgcd(T a, T b, T& x, T& y) {  // a solution of ax + by = gcd(a, b)
    if (b == 0) return x = 1, y = 0, a;
    T d = exgcd(b, a % b, y, x);
    y -= a / b * x;
    return d;
}
pair<T, T> CRT(T mf, T af, T ms, T as) {
    T x, y;
    T g = exgcd(mf, ms, x, y);  // x is the inv of mf mod ms
    if ((as - af) % g) return {-1, -1};
    T m = ms / g;
    T a = af + ((as - af) / g * x % m + m) % m * mf;
    m *= mf;
    return {m, a};
}
```

2.10 高斯消元

```
// Copyright 2016 Parallelc
#include <bits/stdc++.h>
using namespace std;  // NOLINT
const double eps = 1e-9;
vector<double> Gauss(vector<vector<double>>& a, vector<double>& x) {
    int n = a.size();
    int m = a[0].size();
    vector<double> ans(m);
    vector<int> pos(n), free;
    int z = 0;
    for (int i = 0; i < m && z < n; i++, z++) {
        int r = z;
        for (int j = z + 1; j < n; j++) {
            if (abs(a[j][i]) - abs(a[r][i]) > eps) r = j;
        }
        if (abs(a[r][i]) <= eps) {
            free.push_back(i);
            z--;
            continue;
```

```
        }
        swap(a[z], a[r]);
        swap(x[z], x[r]);
        x[z] /= a[z][i];
        for (int j = i + 1; j < m; j++) a[z][j] /= a[z][i];
        a[z][i] = 1;
        for (int j = z + 1; j < n; j++) {
            if (abs(a[j][i]) > eps) {
                x[j] -= x[z] * a[j][i];
                for (int k = i + 1; k < m; k++) {
                    if (abs(a[z][k]) > eps) a[j][k] -= a[z][k] * a[j][i];
                }
                a[j][i] = 0;
            }
        }
        ans[i] = x[z];
        pos[z] = i;
    }
    for (int i = z; i < n; i++) if (abs(x[i]) > eps) return vector<double>();
    // TODO: enumerate free_x
    for (int i = z - 1; i >= 0; i--) {
        for (int j = pos[i] + 1; j < m; j++) {
            if (abs(a[i][j]) > eps) ans[pos[i]] -= a[i][j] * ans[j];
        }
    }
    return ans;
}

#include <tr2/dynamic_bitset>
using namespace tr2;
using db = dynamic_bitset<>;
template<size_t N, size_t M>
db Gauss(vector<db>& a) {
    int n = a.size();
    int m = a[0].size() - 1;
    db ans(m, 0);
    vector<int> pos(n), free;
    int z = 0;
    for (int i = 0; i < m && z < n; i++, z++) {
        int r = z;
        while (r < n - 1 && !a[r][i]) r++;
        if (!a[r][i]) {
            free.push_back(i);
            z--;
            continue;
        }
        swap(a[z], a[r]);
        for (int j = 0; j < n; j++) {
            if (j != z && a[j][i]) {
                a[j] ^= a[z];
            }
        }
        ans[i] = a[z][m];
        pos[z] = i;
    }
    for (int i = z; i < n; i++) if (a[i][m]) return db();
    // TODO: enumerate free_x
    /*
```

```
    for (int i = 0; i < n; i++) {
        for (auto& j : free) {
            if (a[i][j]) ans[pos[i]] = ans[pos[i]] ^ ans[j];
        }
    }
    */
    return ans;
}

using LL = int64_t;
using T = LL;
T exgcd(T a, T b, T& x, T& y) {  // a solution of ax + by = gcd(a, b)
    if (b == 0) return x = 1, y = 0, a;
    T d = exgcd(b, a % b, y, x);
    y -= a / b * x;
    return d;
}
T inv(T a, T mod) {  // ax = 1(mod p)
    T x, y;
    T d = exgcd(a, mod, x, y);
    if (d == 1) return (x % mod + mod) % mod;
    else return -1;
}
template <typename T>
vector<T> Gauss(vector<vector<T>>& a, vector<T>& x, T mod) {  // two mods for check
    int n = a.size();
    int m = a[0].size();
    vector<T> ans(m);
    vector<int> pos(n), free;
    int z = 0;
    for (int i = 0; i < m && z < n; i++, z++) {
        int r = z;
        while (r < n - 1 && !a[r][i]) r++;
        if (!a[r][i]) {
            free.push_back(i);
            z--;
            continue;
        }
        swap(a[z], a[r]);
        swap(x[z], x[r]);
        T inv = ::inv(a[z][i], mod);
        if (inv == -1) return vector<T>();
        x[z] = x[z] * inv % mod;
        for (int j = i + 1; j < m; j++) a[z][j] = a[z][j] * inv % mod;
        a[z][i] = 1;
        for (int j = z + 1; j < n; j++) {
            if (a[j][i]) {
                x[j] = (x[j] - x[z] * a[j][i] % mod + mod) % mod;
                for (int k = i + 1; k < m; k++) {
                    if (a[z][k]) a[j][k] = (a[j][k] - a[z][k] * a[j][i] % mod + mod) % mod;
                }
                a[j][i] = 0;
            }
        }
        ans[i] = x[z];
        pos[z] = i;
    }
    for (int i = z; i < n; i++) if (x[i]) return vector<T>();
```

```
    // TODO: enumerate free_x
    for (int i = z - 1; i >= 0; i--) {
        for (int j = pos[i] + 1; j < m; j++) {
            if (a[i][j]) ans[pos[i]] = (ans[pos[i]] - a[i][j] * ans[j] % mod + mod) % mod;
        }
    }
    return ans;
}
```

2.11 容斥原理

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

template <typename T>
vector<T> inc_exc(vector<T>& a) {
    vector<T> b;
    b.push_back(-1);
    for (auto& i : a) {
        int m = b.size();
        for (int j = 0; j < m; j++) b.push_back(- i * b[j]);
    }
    return b;
}
```

2.12 1-n 异或和

```
#include <bits/stdc++.h>
using namespace std;
using uint = uint64_t;
uint xor_n(uint n) {
    uint t = n & 3;
    if (t & 1) return t / 2u ^ 1;
    return t / 2u ^ n;
}
```

2.13 BM

```
#include <bits/stdc++.h>
using namespace std;
using LL = int64_t;
const LL mod = 1e9 + 7;
LL PowMod(LL a, LL n) {
    LL ans = 1;
    while (n) {
        if (n & 1)
            ans = (ans * a) % mod;
        a = (a * a) % mod;
        n >>= 1;
    }
    return ans;
}

vector<LL> BM(const vector<LL>& a) {
    vector<LL> B{1}, C{1};
    int l = 0, m = 1, b = 1;
    for (int n = 0; n < a.size(); n++) {
        LL d = 0;
        for (int i = 0; i <= l; i++) d = (d + C[i] * a[n - i]) % mod;
```

```
        if (d == 0) m++;
        else {
            LL c = mod - d * PowMod(b, mod - 2) % mod;
            C.resize(B.size() + m);
            if (2 * l <= n) {
                auto T = C;
                for (int i = 0; i < B.size(); i++) C[i + m] = (C[i + m] + c * B[i]) % mod;
                l = n + 1 - l; B = move(T); b = d; m = 1;
            } else {
                for (int i = 0; i < B.size(); i++) C[i + m] = (C[i + m] + c * B[i]) % mod;
                m++;
            }
        }
    }
    reverse(C.begin(), C.end());
    C.pop_back();
    for (auto& i : C) i = (mod - i) % mod;
    return C;
}

void test(const vector<LL>& a) {
    auto ans = BM(a);
    int n = ans.size();
    vector<LL> b;
    for (int i = 0; i < n; i++) {
        b.push_back(a[i]);
        cout << b[i] << endl;
    }
    for (int i = n; i < 20; i++) {
        LL tmp = 0;
        for (int j = 0; j < n; j++) {
            tmp += b[i - n + j] * ans[j] % mod;
            tmp %= mod;
        }
        b.push_back(tmp);
        cout << tmp << endl;
    }
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    test({2,24,96,416,1536,5504,18944,64000,212992,702464});
}
```

2.14 线性递推

```
#include <bits/stdc++.h>
using namespace std;
using LL = int64_t;
const LL mod = 1e9 + 7;

LL linear(const vector<LL>& a, const vector<LL>& b, LL n) {  // b[k + 1] = a[0] * b[0] + ...
↪  + a[k - 1] * b[k - 1];
    int k = a.size();
    vector<LL> d, res(k), base(k);
    for (int i = 0; i < k; i++) if (a[i]) d.push_back(i);
    res[0] = 1;
```

```
    int p = 0;
    while ((1LL << p) <= n) p++;
    for (; p >= 0; p--) {
        vector<LL> c(2 * k);
        for (int i = 0; i < k; i++) {
            if (res[i]) for (int j = 0; j < k; j++) {
                c[i + j] += res[i] * res[j] % mod;
                if (c[i + j] >= mod) c[i + j] -= mod;
            }
        }
        for (int i = 2 * k - 1; i >= k; i--) {
            if (c[i]) for (auto j : d) {
                c[i - k + j] += c[i] * a[j] % mod;
                if (c[i - k + j] >= mod) c[i - k + j] -= mod;
            }
        }
        c.resize(k);
        res = move(c);
        if ((n >> p) & 1) {
            LL tmp = res.back();
            for (int i = k - 1; i > 0; i--) res[i] = res[i - 1];
            res[0] = 0;
            for (auto i : d) {
                res[i] += tmp * a[i] % mod;
                if (res[i] >= mod) res[i] -= mod;
            }
        }
    }
    LL ans = 0;
    for (int i = 0; i < k; i++) {
        ans += res[i] * b[i] % mod;
        if (ans >= mod) ans -= mod;
    }
    return ans;
}
```

2.15 FFT

```
#include <bits/stdc++.h>
using namespace std;
using LD = long double;
const LD PI = acos(-1);
const double eps = 0.5;
using CLD = complex<LD>;
namespace FFT {
    void rader(vector<CLD>& y) {
        int len = y.size();
        for (int i = 1, j = len / 2; i < len - 1; i++) {
            if (i < j) swap(y[i], y[j]);
            int k = len / 2;
            while (j >= k) {
                j -= k;
                k /= 2;
            }
            if (j < k) j += k;
        }
    }
    void DFT(vector<CLD>& y, int on) {
```

```
    int len = y.size();
    rader(y);
    for (int h = 2; h <= len; h <<= 1) {
        CLD wn(cos(-on * 2 * PI / h), sin(-on * 2 * PI / h));
        for (int j = 0; j < len; j += h) {
            CLD w(1, 0);
            for (int k = j; k < j + h / 2; k++) {
                CLD u = y[k];
                CLD t = w * y[k + h / 2];
                y[k] = u + t;
                y[k + h / 2] = u - t;
                w = w * wn;
            }
        }
    }
    if (on == -1) for (auto& i : y) i.real(i.real() / len);
}
void conv(vector<CLD>& a, vector<CLD>& b) {
    int n = 1, m = a.size() + b.size() - 1;
    while (n < m) n <<= 1;
    a.resize(n), b.resize(n);
    DFT(a, 1);
    DFT(b, 1);
    for (int i = 0; i < n; i++) a[i] *= b[i];
    DFT(a, -1);
    a.resize(m);
}
void conv2(vector<CLD>& a, vector<CLD>& b) {
    int n = max(a.size(), b.size());
    a.resize(n), b.resize(n);
    reverse(b.begin(), b.end());
    conv(a, b);
    for (int i = 0; i < n - 1; i++) a[i] += a[i + n];
    a.resize(n);
    reverse(a.begin(), a.end());
}
vector<CLD> pinv(vector<CLD>& a, size_t n) {  // ax = 1 (mod x ^ n)
    if (n == 1) return vector<CLD>{(LD)1.0 / a[0]};
    else {
        auto b = pinv(a, (n + 1) >> 1);
        int len = 1;
        while (len < n << 1) len <<= 1;
        b.resize(len);
        vector<CLD> tmp(len);
        copy_n(a.begin(), min(n, a.size()), tmp.begin());
        DFT(b, 1);
        DFT(tmp, 1);
        for (int i = 0; i < len; i++) b[i] *= (LD)2.0 - tmp[i] * b[i];
        DFT(b, -1);
        b.resize(n);
        return b;
    }
}
void div(vector<CLD>& a, vector<CLD>& b) {
    if (a.size() < b.size()) {
        b = move(a);
        return;
    }
```

```
        int n = a.size() - b.size() + 1;
        int m = b.size();
        vector<CLD> c(a.rbegin(), a.rbegin() + n), d(b.rbegin(), b.rend());
        d = pinv(d, n);
        conv(c, d);
        c.resize(n);
        reverse(c.begin(), c.end());
        conv(b, c);
        b.resize(m);
        for (int i = 0; i < m; i++) b[i] = a[i] - b[i];
        DFT(c, -1);
        c.resize(n);
        a = move(c);
    }
}
```

2.16 NTT

```
#include <bits/stdc++.h>
using namespace std;
using LL = int64_t;
using LD = long double;

inline LL mul(LL x, LL y, LL mod) {
    return x * y % mod;
    // LL res = x * y - (LL)((LD)x / mod * y) * mod;
    // if (abs(res) >= mod) res %= mod;
    // if (res < 0) res += mod;
    // return res;
}
inline LL add(LL x, LL y, LL mod) {
    x += y;
    if (x >= mod) x -= mod;
    return x;
}
inline LL sub(LL x, LL y, LL mod) {
    x -= y;
    if (x < 0) x += mod;
    return x;
}
LL PowMod(LL a, LL n, LL mod) {
    LL ans = 1;
    while (n) {
        if (n & 1) ans = mul(ans, a, mod);
        a = mul(a, a, mod);
        n >>= 1;
    }
    return ans;
}
using T = LL;
T exgcd(T a, T b, T& x, T& y) {  // a solution of ax + by = gcd(a, b)
    if (b == 0) return x = 1, y = 0, a;
    T d = exgcd(b, a % b, y, x);
    y -= a / b * x;
    return d;
}
T inv(T a, T mod) {  // ax = 1(mod p)
    T x, y;
```

```
    T d = exgcd(a, mod, x, y);
    if (d == 1) return (x % mod + mod) % mod;
    else return -1;
}

class NTT {
 private:
     vector<LL> wn, inv2;

 public:
     LL mod;
     int g;
     NTT() {}
     NTT(LL mod, int g = 3, int k = 20) : mod(mod), g(g) {
         wn.resize(1 << k);
         wn[0] = 1;
         wn[1] = PowMod(g, (mod - 1) >> k, mod);
         for (int i = 2; i < (1 << k); i++) {
             wn[i] = mul(wn[i - 1], wn[1], mod);
         }
         inv2.resize(k + 1);
         inv2[0] = 1;
         inv2[1] = (mod + 1) >> 1;
         for (int i = 2; i <= k; i++) {
             inv2[i] = mul(inv2[i - 1], inv2[1], mod);
         }
     }
     void DFT(vector<LL>& y, int on) {
         int len = y.size();
         for (int i = 0, j = 0; i < len; i++) {
             if (i > j) swap(y[i], y[j]);
             for (int l = len >> 1; (j ^= l) < l; l >>= 1);
         }
         for (int i = 1, d = 1; d < len; i++, d <<= 1) {
             for (int j = 0; j < len; j += d << 1) {
                 for (int k = 0; k < d; k++) {
                     LL t = mul(wn[(wn.size() >> i) * k], y[j + k + d], mod);
                     y[j + d + k] = sub(y[j + k], t, mod);
                     y[j + k] = add(y[j + k], t, mod);
                 }
             }
         }
         if (on == -1) {
             reverse(y.begin() + 1, y.end());
             LL val = inv2[__lg(len)];
             for (auto &i : y) i = mul(i, val, mod);
         }
     }
     void conv(vector<LL>& a, vector<LL>& b) {
         int n = 1, m = a.size() + b.size() - 1;
         while (n < m) n <<= 1;
         a.resize(n), b.resize(n);
         DFT(a, 1);
         DFT(b, 1);
         for (int i = 0; i < n; i++) a[i] = mul(a[i], b[i], mod);
         DFT(a, -1);
         a.resize(m);
     }
```

```
void conv2(vector<LL>& a, vector<LL>& b) {
    int n = max(a.size(), b.size());
    a.resize(n), b.resize(n);
    reverse(b.begin(), b.end());
    conv(a, b);
    for (int i = 0; i < n - 1; i++) a[i] += a[i + n];
    a.resize(n);
    reverse(a.begin(), a.end());
}
vector<LL> pinv(vector<LL>& a, int n, LL P) {  // ax = 1 (mod x^n)(mod P)
    LL k = inv(a[0], P);
    if (k == -1) return vector<LL>();
    vector<LL> b, tmp;
    b.push_back(k);
    if (P == mod) {
        for (int i = 2; (i >> 1) < n; i <<= 1) {
            int len = min(i, n);
            b.resize(i << 1);
            tmp.resize(i << 1);
            copy_n(a.begin(), len, tmp.begin());
            DFT(b, 1);
            DFT(tmp, 1);
            for (int j = 0; j < (i << 1); j++) b[j] = mul(b[j], sub(2, mul(b[j],
            ↪ tmp[j], mod), mod), mod);
            DFT(b, -1);
            b.resize(len);
        }
    } else {
        for (int i = 2; (i >> 1) < n; i <<= 1) {
            int len = min(i, n);
            b.resize(i << 1);
            tmp.resize(i << 1);
            copy_n(a.begin(), len, tmp.begin());
            DFT(b, 1);
            DFT(tmp, 1);
            for (int j = 0; j < (i << 1); j++) tmp[j] = mul(b[j], tmp[j], mod);
            DFT(tmp, -1);
            for (int j = 0; j < (i << 1); j++) {
                tmp[j] %= P;
                if (tmp[j]) tmp[j] = P - tmp[j];
            }
            tmp[0] = add(tmp[0], 2, P);
            DFT(tmp, 1);
            for (int j = 0; j < (i << 1); j++) b[j] = mul(b[j], tmp[j], mod);
            DFT(b, -1);
            for (int j = 0; j < (i << 1); j++) tmp[j] %= P;
            b.resize(len);
        }
    }
    return b;
}
bool div(vector<LL>& a, vector<LL>& b, LL mod) {
    if (a.size() < b.size()) {
        b = move(a);
        return true;
    }
    int n = a.size() - b.size() + 1;
    int m = b.size();
```

```
        vector<LL> c(a.rbegin(), a.rbegin() + n), d(b.rbegin(), b.rend());
        d = pinv(d, n, mod);
        if (d.empty()) return false;
        conv(c, d);
        c.resize(n);
        reverse(c.begin(), c.end());
        conv(b, c);
        b.resize(m);
        for (int i = 0; i < m; i++) b[i] = sub(a[i], b[i] % mod, mod);
        DFT(c, -1);
        c.resize(n);
        for (int i = 0; i < n; i++) c[i] %= mod;
        a = move(c);
        return true;
    }
};

/*
 * 2281701377 = 17 * 2 ^ 27 + 1 平方刚好不会爆 long long
 * 1004535809 = 479 * 2 ^ 21 + 1 加起来刚好不会爆 int
 * 998244353 = 119 * 2 ^ 23 + 1
 * g 均为 3

 * mod                 r   k   g
 * 3                   1   1   2
 * 5                   1   2   2
 * 17                  1   4   3
 * 97                  3   5   5
 * 193                 3   6   5
 * 257                 1   8   3
 * 7681                15  9   17
 * 12289               3   12  11
 * 40961               5   13  3
 * 65537               1   16  3
 * 786433              3   18  10
 * 5767169             11  19  3
 * 7340033             7   20  3
 * 23068673            11  21  3
 * 104857601           25  22  3
 * 167772161           5   25  3
 * 469762049           7   26  3
 * 998244353           119 23  3
 * 1004535809          479 21  3
 * 2013265921          15  27  31
 * 2281701377          17  27  3
 * 3221225473          3   30  5
 * 75161927681         35  31  3
 * 77309411329         9   33  7
 * 206158430209        3   36  22
 * 2061584302081       15  37  7
 * 2748779069441       5   39  3
 * 6597069766657       3   41  5
 * 39582418599937      9   42  5
 * 79164837199873      9   43  5
 * 263882790666241     15  44  7
 * 1231453023109121    35  45  3
 * 1337006139375617    19  46  3
 * 3799912185593857    27  47  5
```

```
 * 4222124650659841    15  48  19
 * 7881299347898369    7   50  6
 * 31525197391593473   7   52  3
 * 18014398509481841   5   55  6
 * 194555039024054273  27  56  5
 * 4179340454199820289 29  57  3
 */
```

2.17 Java 大数

```
import java.io.*;
import java.math.*;
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner cin = new Scanner(new BufferedInputStream(System.in));
        while(cin.hasNext()) {
            BigInteger x, y;
            x = cin.nextBigInteger();
            y = cin.nextBigInteger();
            x = x.add(BigInteger.ONE); x = x.subtract(BigInteger.TEN);
            x = x.multiply(BigInteger.ZERO);
            x = x.divide(y); x = x.remainder(y); x = x.mod(y);
            x = x.pow(10); x = x.gcd(y); x = x.abs(); x = x.negate();
            x = x.max(y); x = x.min(y);
            if (x.compareTo(y) > 0) { }
            if (x.equals(y)) { }
            x = new BigInteger("123456");
            x = new BigInteger("1010101", 2);
            x.toString();
            System.out.println(x);
            BigDecimal a, b;
            a = cin.nextBigDecimal();
            b = cin.nextBigDecimal();
            a = a.divide(b, 100, RoundingMode.HALF_UP);
            System.out.println(String.format("%.6f", a));
        }
    }
}
```

2.18 多项式

```
#include <bits/stdc++.h>
using namespace std;
using LL = int64_t;
using LD = long double;

inline LL mul(LL x, LL y, LL mod) {
    return x * y % mod;
    // return (x * y - (LL)((LD)x / mod * y + 1e-3) * mod + mod) % mod;
}
LL PowMod(LL a, LL n, LL mod) {
    LL ans = 1;
    while (n) {
        if (n & 1) ans = mul(ans, a, mod);
        a = mul(a, a, mod);
        n >>= 1;
    }
```

```
    return ans;
}
LL inv(LL a, LL mod) {return PowMod(a, mod - 2, mod);}

class NTT {
 private:
     vector<LL> wn;
     void rader(vector<LL>& y) {
         int len = y.size();
         for (int i = 1, j = len / 2; i < len - 1; i++) {
             if (i < j) swap(y[i], y[j]);
             int k = len / 2;
             while (j >= k) {
                 j -= k;
                 k /= 2;
             }
             if (j < k) j += k;
         }
     }

 public:
     LL mod;
     int g;
     NTT() {}
     NTT(LL mod, int g = 3, int k = 20) : mod(mod), g(g) {
         wn.resize(k);
         for (int i = 1; i < k; i++) {
             int t = 1 << i;
             wn[i] = PowMod(g, (mod - 1) / t, mod);
         }
     }
     void DFT(vector<LL>& y, int on) {
         int len = y.size();
         rader(y);
         for (int h = 2, i = 1; h <= len; h <<= 1, i++) {
             for (int j = 0; j < len; j += h) {
                 LL w = 1;
                 for (int k = j; k < j + h / 2; k++) {
                     LL u = y[k];
                     LL t = mul(w, y[k + h / 2], mod);
                     y[k] = (u + t) % mod;
                     y[k + h / 2] = (u - t + mod) % mod;
                     w = mul(w, wn[i], mod);
                 }
             }
         }
         if (on == -1) {
             reverse(y.begin() + 1, y.end());
             LL inv = PowMod(len, mod - 2, mod);
             for (auto& i : y) i = mul(i, inv, mod);
         }
     }
     vector<LL> conv(vector<LL>& a, vector<LL>& b) {
         int n = 1;
         while (n < a.size() + b.size() - 1) n <<= 1;
         vector<LL> c, d;
         c.reserve(n), d.reserve(n);
         for (auto& i : a) c.push_back(i % mod);
```

```
        for (auto& i : b) d.push_back(i % mod);
        c.resize(n), d.resize(n);
        DFT(c, 1);
        DFT(d, 1);
        for (int i = 0; i < n; i++) c[i] = mul(c[i], d[i], mod);
        DFT(c, -1);
        return c;
    }
    vector<LL> pinv(vector<LL>& a, int n, LL P) {  // ax = 1 (mod x^n)(mod P)
        if (n == 1) return vector<LL>{inv(a[0], mod)};
        else {
            auto b = pinv(a, (n + 1) >> 1, P);
            int len = 1;
            while (len < n << 1) len <<= 1;
            b.resize(len);
            vector<LL> tmp(len);
            copy_n(a.begin(), n, tmp.begin());
            DFT(b, 1);
            DFT(tmp, 1);
            for (int i = 0; i < len; i++) tmp[i] = mul(b[i], tmp[i], mod);
            DFT(tmp, -1);
            for (int i = 0; i < len; i++) {
                tmp[i] = (tmp[i] % P + P) % P;
                if (tmp[i]) tmp[i] = P - tmp[i];
            }
            tmp[0] = (tmp[0] + 2) % P;
            DFT(tmp, 1);
            for (int i = 0; i < len; i++) b[i] = mul(b[i], tmp[i], mod);
            DFT(b, -1);
            for (int i = 0; i < len; i++) b[i] = (b[i] % P + P) % P;
            b.resize(n);
            return b;
        }
    }
};
```

2.19 素数个数 $n^{\frac{3}{4}}$

```
//1e11 1500ms 6000k O(n^(3/4))
#include <bits/stdc++.h>
using namespace std;
using LL = int64_t;
LL countp(LL k){
    static vector<LL> f, g;
    int num = sqrt(k) + 5;
    f.resize(num), g.resize(num);
    LL m;
    for (m = 1; m * m <= k; m++) f[m] = k / m - 1;
    iota(g.begin() + 1, g.begin() + m + 1, 0);
    for (LL i = 2; i <= m; i++){
        if (g[i] == g[i - 1]) continue;
        for (LL j = 1; j <= min(m - 1, k / i / i); j++) {
            if (i * j < m) f[j] -= f[i * j] - g[i - 1];
            else f[j] -= g[k / i / j] - g[i - 1];
        }
        for (LL j = m; j >= i * i; j--) g[j] -= g[j / i] - g[i - 1];
    }
    return f[1];
```

```
}
```

2.20 素数个数 $n^{\frac{2}{3}}$

```
// 2e12 200ms 40000k  O(n^(2/3))
#include <bits/stdc++.h>
using namespace std;
using LL = int64_t;
const int N = 5e6 + 2;
bool np[N];
int prime[N], pi[N];
int getprime() {
    int cnt = 0;
    np[0] = np[1] = true;
    pi[0] = pi[1] = 0;
    for (int i = 2; i < N; ++i) {
        if (!np[i]) prime[++cnt] = i;
        pi[i] = cnt;
        for (int j = 1; j <= cnt && i * prime[j] < N; ++j) {
            np[i * prime[j]] = true;
            if (i % prime[j] == 0) break;
        }
    }
    return cnt;
}
const int M = 7;
const int PM = 2 * 3 * 5 * 7 * 11 * 13 * 17;
int phi[PM + 1][M + 1], sz[M + 1];
void init() {
    getprime();
    sz[0] = 1;
    for (int i = 0; i <= PM; ++i) phi[i][0] = i;
    for (int i = 1; i <= M; ++i) {
        sz[i] = prime[i] * sz[i - 1];
        for (int j = 1; j <= PM; ++j) phi[j][i] = phi[j][i - 1] - phi[j / prime[i]][i - 1];
    }
}
int sqrt2(LL x) {
    LL r = (LL)sqrt(x - 0.1);
    while (r * r <= x) ++r;
    return int(r - 1);
}
int sqrt3(LL x) {
    LL r = (LL)cbrt(x - 0.1);
    while (r * r * r <= x) ++r;
    return int(r - 1);
}
LL getphi(LL x, int s) {
    if (s == 0) return x;
    if (s <= M) return phi[x % sz[s]][s] + (x / sz[s]) * phi[sz[s]][s];
    if (x <= prime[s] * prime[s]) return pi[x] - s + 1;
    if (x <= prime[s] * prime[s] * prime[s] && x < N) {
        int s2x = pi[sqrt2(x)];
        LL ans = pi[x] - (s2x + s - 2) * (s2x - s + 1) / 2;
        for (int i = s + 1; i <= s2x; ++i) ans += pi[x / prime[i]];
        return ans;
    }
    return getphi(x, s - 1) - getphi(x / prime[s], s - 1);
```

```
}
LL getpi(LL x) {
    if (x < N) return pi[x];
    LL ans = getphi(x, pi[sqrt3(x)]) + pi[sqrt3(x)] - 1;
    for (int i = pi[sqrt3(x)] + 1, ed = pi[sqrt2(x)]; i <= ed; ++i) ans -= getpi(x /
    ↪  prime[i]) - i + 1;
    return ans;
}
LL lehmer_pi(LL x) { // prime-num
    if (x < N) return pi[x];
    int a = (int)lehmer_pi(sqrt2(sqrt2(x)));
    int b = (int)lehmer_pi(sqrt2(x));
    int c = (int)lehmer_pi(sqrt3(x));
    LL sum = getphi(x, a) + (LL)(b + a - 2) * (b - a + 1) / 2;
    for (int i = a + 1; i <= b; i++) {
        LL w = x / prime[i];
        sum -= lehmer_pi(w);
        if (i > c) continue;
        LL lim = lehmer_pi(sqrt2(w));
        for (int j = i; j <= lim; j++) sum -= lehmer_pi(w / prime[j]) - (j - 1);
    }
    return sum;
}
```

2.21 等比数列求和

```
#include<iostream>
using namespace std;
int PowMod(int a, int b, int c)
{
        int ans = 1;
        a = a % c;
        while(b>0)
        {
                if(b & 1)
                ans = (ans * a) % c;
                b >>= 1;
                a = (a * a) % c;
        }
        return ans;
}
int PowSumMod(int a,int n,int p)
{// return (a+ a^2 + ... + a^n) Mod p;
        if( n == 1) return a%p;
        if( n % 2 == 0)return (1+PowMod(a,n/2,p))*PowSumMod(a,n/2,p) % p;
        else return ((1+PowMod(a,(n-1)/2,p)) * PowSumMod(a,(n-1)/2,p)+ PowMod(a,n,p)) % p;
}
int main()
{
        int a,n,p;
        cin>>a>>n>>p;
        cout<<PowSumMod(a,n,p)<<endl;
}
```

2.22 自然数幂和

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

```cpp
using LL = int64_t;
using LD = long double;
const LL mod = 1e9 + 7;
const LL mf = 1231453023109121;
const LL ms = 1337006139375617;
inline LL add(LL x, LL y, LL mod) {
    x += y;
    if (x >= mod) x -= mod;
    return x;
}
inline LL sub(LL x, LL y, LL mod) {
    x -= y;
    if (x < 0) x += mod;
    return x;
}
inline LL mul(LL x, LL y, LL mod) {
    // return x * y % mod;
    LL res = x * y - (LL)((LD)x / mod * y) * mod;
    if (abs(res) >= mod) res %= mod;
    if (res < 0) res += mod;
    return res;
}
LL PowMod(LL a, LL n, LL mod) {
    LL ans = 1;
    while (n) {
        if (n & 1)
            ans = mul(ans, a, mod);
        a = mul(a, a, mod);
        n >>= 1;
    }
    return ans;
}
LL CRT(LL af, LL as) {
    static LL x = PowMod(mf, ms - 2, ms);
    static LL m = mf % mod;
    return add(af % mod, mul(mul(sub(as, af, ms), x, ms) % mod, m, mod), mod);
}
class NTT {
 private:
     vector<LL> wn, inv2;

 public:
     LL mod;
     int g;
     NTT() {}
     NTT(LL mod, int g = 3, int k = 20) : mod(mod), g(g) {
         wn.resize(1 << k);
         wn[0] = 1;
         wn[1] = PowMod(g, (mod - 1) >> k, mod);
         for (int i = 2; i < (1 << k); i++) {
             wn[i] = mul(wn[i - 1], wn[1], mod);
         }
         inv2.resize(k + 1);
         inv2[0] = 1;
         inv2[1] = (mod + 1) >> 1;
         for (int i = 2; i <= k; i++) {
             inv2[i] = mul(inv2[i - 1], inv2[1], mod);
         }
```

```
    }
    void DFT(vector<LL> &y, int on) {
        int len = y.size();
        for (int i = 0, j = 0; i < len; i++) {
            if (i > j) swap(y[i], y[j]);
            for (int l = len >> 1; (j ^= l) < l; l >>= 1);
        }
        for (int i = 1, d = 1; d < len; i++, d <<= 1) {
            for (int j = 0; j < len; j += d << 1) {
                for (int k = 0; k < d; k++) {
                    LL t = mul(wn[(wn.size() >> i) * k], y[j + k + d], mod);
                    y[j + d + k] = sub(y[j + k], t, mod);
                    y[j + k] = add(y[j + k], t, mod);
                }
            }
        }
        if (on == -1) {
            reverse(y.begin() + 1, y.end());
            LL val = inv2[__lg(len)];
            for (auto &i : y) i = mul(i, val, mod);
        }
    }
    vector<LL> conv(vector<LL> &a, vector<LL> &b) {
        DFT(b, 1);
        vector<LL> c(a.size());
        for (int i = 0; i < a.size(); i++)
            c[i] = mul(a[i], b[i], mod);
        DFT(c, -1);
        return c;
    }
};
NTT ntt[2];
vector<LL> pinv(const vector<LL> &a, int n) {
    vector<LL> tmp[2], b[2], d[2];
    b[0].push_back(1);
    for (int i = 2; (i >> 1) < n; i <<= 1) {
        int len = min(i, n);
        b[0].resize(i << 1);
        b[1] = b[0];
        tmp[0].resize(i << 1);
        copy_n(a.begin(), len, tmp[0].begin());
        tmp[1] = tmp[0];
        ntt[0].DFT(b[0], 1);
        ntt[1].DFT(b[1], 1);
        d[0] = ntt[0].conv(b[0], tmp[0]);
        d[1] = ntt[1].conv(b[1], tmp[1]);
        for (int j = 0; j < (i << 1); j++) {
            tmp[0][j] = CRT(d[0][j], d[1][j]);
            tmp[0][j] = tmp[0][j] ? mod - tmp[0][j] : 0;
        }
        tmp[0][0] = add(tmp[0][0], 2, mod);
        tmp[1] = tmp[0];
        d[0] = ntt[0].conv(b[0], tmp[0]);
        d[1] = ntt[1].conv(b[1], tmp[1]);
        for (int j = 0; j < (i << 1); j++) b[0][j] = CRT(d[0][j], d[1][j]);
        b[0].resize(len);
    }
    return b[0];
```

```
}
vector<LL> jc, inv, B, npo;
void init(LL k) {
    jc.resize(k + 2);
    jc[0] = 1;
    for (int i = 1; i < k + 2; i++)
        jc[i] = jc[i - 1] * i % mod;
    inv.resize(k + 2);
    inv[k + 1] = PowMod(jc[k + 1], mod - 2, mod);
    for (int i = k; i >= 0; i--)
        inv[i] = inv[i + 1] * (i + 1) % mod;
    ntt[0] = NTT(1231453023109121);
    ntt[1] = NTT(1337006139375617);
    B = pinv(vector<LL>(inv.begin() + 1, inv.end()), k + 1);
    for (int i = 0; i < k + 1; i++) {
        B[i] = B[i] * jc[i] % mod;
    }
    npo.resize(k + 2);
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    init(50000);
    int t;
    cin >> t;
    while (t--) {
        LL n, k;
        cin >> n >> k;
        if (n >= mod) n %= mod;
        LL ans = 0;
        npo[0] = 1;
        for (int i = 1; i < k + 2; i++) {
            npo[i] = npo[i - 1] * (n + 1) % mod;
            ans += jc[k + 1] * inv[i] % mod * inv[k + 1 - i] % mod * B[k + 1 - i] % mod *
            ↪  npo[i] % mod;
            if (ans >= mod) ans -= mod;
        }
        ans *= PowMod(k + 1, mod - 2, mod);
        cout << ans % mod << '\n';
    }
}
```

3 数据结构

3.1 树状数组

```cpp
// Copyright 2017 Parallelc
#include <bits/stdc++.h>
#include <ext/numeric>
using namespace std;
using namespace __gnu_cxx;
template<typename T, class op = plus<T>, class sub = minus<T>>
class BIT {
 private:
    vector<T> tr;
    int lowbit(int x) {return x & (-x);}

 public:
    BIT() {}
    BIT(int n) {
        tr.resize(n + 1);
    }
    BIT(vector<T>& a) {
        int n = a.size();
        tr.resize(n + 1);
        for (int i = 0; i < n; i++) add(i, a[i]);
    }
    void add(int x, T k) {
        x++;
        for (int i = x; i < tr.size(); i += lowbit(i)) {
            tr[i] = op()(tr[i], k);
        }
    }
    T sum(int x) {
        T ans = identity_element(op());
        for (int i = x; i; i -= lowbit(i)) {
            ans = op()(ans, tr[i]);
        }
        return ans;
    }
    T que(int l, int r) {
        return sub()(sum(r), sum(l));
    }
};
```

3.2 二维树状数组

```cpp
// Copyright 2017 Parallelc
#include <bits/stdc++.h>
#include <ext/numeric>
using namespace std;
using namespace __gnu_cxx;
template<typename T, class op = plus<T>, class sub = minus<T>>
class BIT {
 private:
    vector<vector<T>> tr;
    T lowbit(T x) { return x & (-x); }

 public:
    BIT() {}
    BIT(int n, int m) {
```

```
        tr.resize(n + 1);
        for (auto& i : tr) i.resize(m + 1);
    }
    BIT(vector<vector<T>>& a) {
        int n = a.size();
        int m = a[0].size();
        tr.resize(n + 1);
        for (auto& i : tr) i.resize(m + 1);
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) {
                add(i, j, a[i][j]);
            }
        }
    }
    void add(int x, int y, T k) {
        x++, y++;
        for (int i = x; i < tr.size(); i += lowbit(i)) {
            for (int j = y; j < tr[x].size(); j += lowbit(j)) {
                tr[i][j] = op()(tr[i][j], k);
            }
        }
    }
    T sum(int x, int y) {
        T ans = identity_element(op());
        for (int i = x; i; i -= lowbit(i)) {
            for (int j = y; j; j -= lowbit(j)) {
                ans = op()(ans, tr[i][j]);
            }
        }
        return ans;
    }
    T que(int x1, int y1, int x2, int y2) {
        return op()(sub()(sub()(sum(x2, y2), sum(x2, y1)), sum(x1, y2)), sum(x1, y1));
    }
};
```

3.3 线段树

```
#include <bits/stdc++.h>
#include <ext/numeric>
using namespace std;
using namespace __gnu_cxx;
template <typename T, class op = plus<T>>
class SegTree {
 private:
    struct node {
        int l, r;
        T v, lazy = 0;
        node() {}
        node(int l, int r, T v) : l(l), r(r), v(v) {}
        void mod(T k) {  // mutable
            v += k * (r - l);
            lazy += k;
        }
    };
    vector<node> tr;
    void pd(int now) {
        if (!tr[now].lazy) return;
```

```
        tr[now * 2 + 1].mod(tr[now].lazy);
        tr[now * 2 + 2].mod(tr[now].lazy);
        tr[now].lazy = 0;
    }

public:
    SegTree() {}
    SegTree(int n) : SegTree(vector<T>(n, identity_element(op()))) {}
    SegTree(const vector<T>& a) {
        int n = a.size();
        tr.resize(4 * n);
        function<void(int, int, int)> cre = [&](int l, int r, int now) {
            if (l + 1 == r) tr[now] = node(l, r, a[l]);
            else {
                int mid = (l + r) / 2;
                cre(l, mid, now * 2 + 1);
                cre(mid, r, now * 2 + 2);
                tr[now] = node(l, r, op()(tr[now * 2 + 1].v, tr[now * 2 + 2].v));
            }
        };
        cre(0, n, 0);
    }
    void upd(int l, int r, T tag, int now = 0) {
        if (l >= r || tr[now].r <= l || tr[now].l >= r) return;
        else if (tr[now].r <= r && tr[now].l >= l) tr[now].mod(tag);
        else {
            pd(now);
            upd(l, r, tag, now * 2 + 1);
            upd(l, r, tag, now * 2 + 2);
            tr[now].v = op()(tr[now * 2 + 1].v, tr[now * 2 + 2].v);
        }
    }
    T que(int l, int r, int now = 0) {
        if (l >= r || tr[now].r <= l || tr[now].l >= r) return identity_element(op());
        else if (tr[now].r <= r && tr[now].l >= l) return tr[now].v;
        else {
            pd(now);
            return op()(que(l, r, now * 2 + 1), que(l, r, now * 2 + 2));
        }
    }
};

const int INF = 0x3f3f3f3f;
template<typename T>
struct Max {
    T operator()(const T& a, const T& b) const { return max(a, b); }
    friend T identity_element(const Max& s) { return -INF; }
};

template<typename T>
struct Min {
    T operator()(const T& a, const T& b) const { return min(a, b); }
    friend T identity_element(const Min& s) { return INF; }
};
```

3.4 可持久化线段树

```cpp
#include <bits/stdc++.h>
using namespace std;
using LL = int64_t;
using T = LL;
struct node {
    int ls, rs;
    T v, lazy;
    void mod(int l, int r, T k) {  // mutable
        v += k * (r - l);
        lazy += k;
    }
};
const int N = 1e5;
node tr[30 * N];  // 4n + mlogn
int cur = 0;

template <class op = plus<T>>
class PerTree {
 private:
     int n;
     int pd(int l, int r, int now) {
         if (!tr[now].lazy) return 0;
         tr[cur] = tr[tr[now].ls];
         tr[now].ls = cur++;
         tr[cur] = tr[tr[now].rs];
         tr[now].rs = cur++;
         int m = (l + r) >> 1;
         tr[tr[now].ls].mod(l, m, tr[now].lazy);
         tr[tr[now].rs].mod(m, r, tr[now].lazy);
         tr[now].lazy = 0;
         return 1;
     }
     int upd(int now, int L, int R, int l, int r, T tag, int f) {
         if (f) {
             tr[cur] = tr[now];
             now = cur++;
         } else f = 1;
         if (L == l && R == r) tr[now].mod(l, r, tag);
         else {
             if (pd(l, r, now)) f = 0;
             int m = (l + r) >> 1;
             if (R <= m) tr[now].ls = upd(tr[now].ls, L, R, l, m, tag, f);
             else if (L >= m) tr[now].rs = upd(tr[now].rs, L, R, m, r, tag, f);
             else {
                 tr[now].ls = upd(tr[now].ls, L, m, l, m, tag, f);
                 tr[now].rs = upd(tr[now].rs, m, R, m, r, tag, f);
             }
             tr[now].v = op()(tr[tr[now].ls].v, tr[tr[now].rs].v);
         }
         return now;
     }
     T que(int now, int L, int R, int l, int r) {
         if (L == l && R == r) return tr[now].v;
         else {
             int ls = tr[now].ls, rs = tr[now].rs;
             int m = (l + r) >> 1;
             if (tr[now].lazy) {
```

```
                ls = cur;
                tr[cur++] = tr[tr[now].ls];
                rs = cur;
                tr[cur++] = tr[tr[now].rs];
                tr[ls].mod(l, m, tr[now].lazy);
                tr[rs].mod(m, r, tr[now].lazy);
            }
            if (R <= m) return que(ls, L, R, l, m);
            else if (L >= m) return que(rs, L, R, m, r);
            else {
                return op()(que(ls, L, m, l, m), que(rs, m, R, m, r));
            }
        }
    }

public:
    vector<int> root;
    int tim = 0;
    PerTree() {}
    PerTree(int n, int m) : PerTree(vector<T>(n), m) {}
    PerTree(const vector<T>& a, int m) : n(a.size()) {
        root.resize(m + 233);
        function<int(int, int)> cre = [&](int l, int r) {
            int now = cur++;
            tr[now].lazy = 0;
            if (l + 1 == r) tr[now].v = a[l];
            else {
                int m = (l + r) >> 1;
                tr[now].ls = cre(l, m);
                tr[now].rs = cre(m, r);
                tr[now].v = op()(tr[tr[now].ls].v, tr[tr[now].rs].v);
            }
            return now;
        };
        root[0] = cre(0, n);
    }
    void back(int now) {
        if (now >= tim) return;
        tim = now;
        cur = root[now + 1];
    }
    void upd(int now, int l, int r, T tag) {
        if (l >= r) root[++tim] = root[now];
        else root[++tim] = upd(root[now], l, r, 0, n, tag, 1);
    }
    void upd(int l, int r, T tag) { upd(tim, l, r, tag); }
    T que(int now, int l, int r) {
        if (l >= r) return 0;
        int tmp = cur;
        T ans = que(root[now], l, r, 0, n);
        cur = tmp;
        return ans;
    }
    T que(int l, int r) { return que(tim, l, r); }
};
```

3.5 主席树

```cpp
#include <bits/stdc++.h>
using namespace std;
using LL = int64_t;
using T = int;
struct node {
    int ls, rs;
    T v;
    void mod(T k) {  // mutable
        v += k;
    }
};
const int N = 1e5;
node tr[30 * N];  // 4n + mlogn
int cur = 0;

template <class op = plus<T>>
class PerTree {
 protected:
     int n;
     void bui(const vector<T>& a, int m) {
         n = a.size();
         root.resize(m + 233);
         function<int(int, int)> cre = [&](int l, int r) {
             int now = cur++;
             if (l + 1 == r) tr[now].v = a[l];
             else {
                 int m = (l + r) >> 1;
                 tr[now].ls = cre(l, m);
                 tr[now].rs = cre(m, r);
                 tr[now].v = op()(tr[tr[now].ls].v, tr[tr[now].rs].v);
             }
             return now;
         };
         root[0] = cre(0, n);
     }
     int upd(int now, int x, int l, int r, T tag) {
         tr[cur] = tr[now];
         now = cur++;
         if (l + 1 == r) tr[now].mod(tag);
         else {
             int m = (l + r) >> 1;
             if (x < m) tr[now].ls = upd(tr[now].ls, x, l, m, tag);
             else tr[now].rs = upd(tr[now].rs, x, m, r, tag);
             tr[now].v = op()(tr[tr[now].ls].v, tr[tr[now].rs].v);
         }
         return now;
     }
     T que(int now, int L, int R, int l, int r) {
         if (L == l && R == r) return tr[now].v;
         else {
             int m = (l + r) >> 1;
             if (R <= m) return que(tr[now].ls, L, R, l, m);
             else if (L >= m) return que(tr[now].rs, L, R, m, r);
             else return op()(que(tr[now].ls, L, m, l, m), que(tr[now].rs, m, R, m, r));
         }
     }
```

```
 public:
     vector<int> root;
     int tim = 0;
     PerTree() {}
     PerTree(int n, int m) : PerTree(vector<T>(n), m) {}
     PerTree(const vector<T>& a, int m) { bui(a, m); }
     void back(int now) {
         if (now >= tim) return;
         tim = now;
         cur = root[now + 1];
     }
     void upd(int now, int x, T tag) { root[++tim] = upd(root[now], x, 0, n, tag); }
     void upd(int x, T tag) { upd(tim, x, tag); }
     T que(int now, int l, int r) {
         if (l >= r) return 0;
         return que(root[now], l, r, 0, n);
     }
     T que(int l, int r) { return que(tim, l, r); }
};

template<typename T>
class ChaTree : public PerTree<> {
    T find_by_order(int L, int R, int l, int r, int k) {
        if (l + 1 == r) return b[l];
        int m = (l + r) >> 1;
        int o = tr[tr[R].ls].v - tr[tr[L].ls].v;
        if (k < o) return find_by_order(tr[L].ls, tr[R].ls, l, m, k);
        else return find_by_order(tr[L].rs, tr[R].rs, m, r, k - o);
    }

 public:
    vector<T> b;
    ChaTree() {}
    ChaTree(const vector<T>& a) {
        b = a;
        sort(b.begin(), b.end());
        b.erase(unique(b.begin(), b.end()), b.end());
        bui(vector<int>(b.size()), a.size());
        for (auto& i : a) {
            int k = lower_bound(b.begin(), b.end(), i) - b.begin();
            upd(k, 1);
        }
    }
    T find_by_order(int l, int r, int k) {
        return find_by_order(root[l], root[r], 0, n, k);
    }
    int order_of_key(int l, int r, T v) {
        int k = lower_bound(b.begin(), b.end(), v) - b.begin();
        return que(r, 0, k) - que(l, 0, k);
    }
};
```

3.6 Treap

```
#include <bits/stdc++.h>
using namespace std;
using T = int;
const T INF = 0x3f3f3f3f;
```

```
struct node {
    int l, r, size;  // f
    uint32_t v; T k;  // m = INF;
    int rev;
    void mod() {
        swap(l, r);
        rev ^= 1;
    }
};
mt19937 rd(time(0));
const int N = 1e5;
node tr[N + 233];  // !!! 0
int cur = 1;

class null_tag {
 protected:
     int root = 0;
     inline int getr() { return root; }
     inline void setr(int x) { root = x; }
     inline int copy(int x) { return x; }
};

class pers_tag {
 protected:
     vector<int> root;
     int tim = 0;
     inline int getr() { return root[tim]; }
     inline void setr(int x) { tim = root.size(); root.push_back(x); }
     inline int copy(int x) { tr[cur] = tr[x]; return cur++; }

 public:
     pers_tag() { setr(0); }
     explicit pers_tag(int n) { root.reserve(n); setr(0); }
     inline void back(int x) { if (x >= root.size()) return; tim = x; }
};

class Treap : public null_tag {
    // using pers_tag::pers_tag;
 protected:
    int newnode(T k, uint32_t v = rd()) {
        tr[cur].k = k, tr[cur].size = 1, tr[cur].v = v, tr[cur].l = tr[cur].r = 0;
        tr[cur].rev = 0;  // tr[cur].f = 0; tr[cur].m = INF;
        return cur++;
    }
    virtual void pd(int now) {}
    virtual void pu(int now) { tr[now].size = 1 + tr[tr[now].l].size + tr[tr[now].r].size; }
    // if join after split, needn't copy
    int join(int A, int B) {
        if (!A && !B) return 0; if (!A) return copy(B); if (!B) return copy(A);
        if (tr[A].v < tr[B].v) {
            A = copy(A); pd(A); tr[A].r = join(tr[A].r, B); pu(A); return A;
        } else {
            B = copy(B); pd(B); tr[B].l = join(A, tr[B].l); pu(B); return B;
        }
    }
    pair<int, int> split_by_order(int x, int k) {
        if (!x) return {0, 0};
```

```
        pair<int, int> y;
        x = copy(x); pd(x);
        if (tr[tr[x].l].size >= k) {
            y = split_by_order(tr[x].l, k);
            tr[x].l = y.second; pu(x); y.second = x;
        } else {
            y = split_by_order(tr[x].r, k - tr[tr[x].l].size - 1);
            tr[x].r = y.first; pu(x); y.first = x;
        }
        return y;
    }

 public:
    Treap() {}
    size_t size() { return tr[getr()].size; }
    T find_by_order(int k) {
        int x = getr();
        while (tr[tr[x].l].size != k) {
            pd(x);
            if (tr[tr[x].l].size > k) x = tr[x].l;
            else { k -= tr[tr[x].l].size + 1; x = tr[x].r; }
        }
        return tr[x].k;
    }
    void show() {
        int f = 0;
        function<void(int)> dfs = [&](int now) {
            pd(now);
            if (tr[now].l) dfs(tr[now].l);
            if (f) cout << ' '; else f = 1; cout << tr[now].k;
            if (tr[now].r) dfs(tr[now].r);
        };
        dfs(getr());
    }
};

// change need pu
class BstTrp : public Treap {
    using Treap::Treap;
    using Treap::join;
    using Treap::split_by_order;
 private:
    pair<int, int> split_by_key(int x, T k) {
        if (!x) return {0, 0};
        pair<int, int> y;
        x = copy(x);
        if (k <= tr[x].k) {
            y = split_by_key(tr[x].l, k);
            tr[x].l = y.second; pu(x); y.second = x;
        } else {
            y = split_by_key(tr[x].r, k);
            tr[x].r = y.first; pu(x); y.first = x;
        }
        return y;
    }
    int insert(int &x, T k, uint32_t v) {
        int y;
        if (!x || tr[x].v > v) {
```

```
            y = newnode(k, v);
            tie(tr[y].l, tr[y].r) = split_by_key(x, tr[y].k); x = y;
        } else {
            x = copy(x);
            if (k < tr[x].k) y = insert(tr[x].l, k, v);
            else y = insert(tr[x].r, k, v);
        }
        pu(x); return y;
    }
    int erase(int &x, T k) {
        if (!x) return 0;
        int res = x;
        if (tr[x].k == k) { x = join(tr[x].l, tr[x].r); if (x) pu(x); }
        else {
            x = copy(x);
            if (k < tr[x].k) res = erase(tr[x].l, k);
            else res = erase(tr[x].r, k);
            if (res) pu(x);
        }
        return res;
    }

public:
    BstTrp() {}
    BstTrp& join(int x) { setr(join(getr(), x)); return *this; }
    BstTrp& join(BstTrp& x) { setr(join(getr(), x.getr())); x.setr(0); return *this; }
    BstTrp& join(BstTrp&& x) { setr(join(getr(), x.getr())); return *this; }
    BstTrp& split_by_order(int k) { setr(split_by_order(getr(), k).first); return *this; }
    BstTrp& split_by_order(int k, BstTrp& x) { auto r = split_by_order(getr(), k);
    ↪  setr(r.first); x.setr(r.second); return *this;}
    BstTrp& split_by_key(T k) { setr(split_by_key(getr(), k).first); return *this; }
    BstTrp& split_by_key(T k, BstTrp& x) { auto r = split_by_key(getr(), k); setr(r.first);
    ↪  x.setr(r.second); return *this; }
    int insert(T k) { int r = getr(); int res = insert(r, k, rd()); setr(r); return res; }
    int erase(T k) { int r = getr(); int res = erase(r, k); setr(r); return res; }
    int order_of_key(T k) {
        int x = getr();
        int ans = 0;
        while (x) {
            if (tr[x].k < k) { ans += tr[tr[x].l].size + 1; x = tr[x].r; }
            else x = tr[x].l;
        }
        return ans;
    }
    int find(T k) {
        int x = getr();
        while (x && tr[x].k != k) {
            if (k < tr[x].k) x = tr[x].l;
            else x = tr[x].r;
        }
        return x;
    }
    int lower_bound(T k) {
        int x = getr(), ans = 0;
        while (x) {
            if (tr[x].k >= k) ans = x, x = tr[x].l;
            else x = tr[x].r;
        }
```

```
        return ans;
    }
    int upper_bound(T k) {
        int x = getr(), ans = 0;
        while (x) {
            if (tr[x].k > k) ans = x, x = tr[x].l;
            else x = tr[x].r;
        }
        return ans;
    }
    T lower(T k) {
        T ans = -INF; int x = getr();
        while (x) {
            if (tr[x].k < k) { ans = max(ans, tr[x].k); x = tr[x].r; }
            else x = tr[x].l;
        }
        return ans;
    }
    T upper(T k) {
        T ans = INF; int x = getr();
        while (x) {
            if (tr[x].k > k) { ans = min(ans, tr[x].k); x = tr[x].l; }
            else x = tr[x].r;
        }
        return ans;
    }
};

// all need pd
// change need pu (pd !-> pu)
class RgeTrp : public Treap {
    using Treap::Treap;
    using Treap::join;
 private:
    void pd(int now) {
        if (!tr[now].rev) return;
        if (tr[now].l) { tr[now].l = copy(tr[now].l); tr[tr[now].l].mod(); }
        if (tr[now].r) { tr[now].r = copy(tr[now].r); tr[tr[now].r].mod(); }
        tr[now].rev ^= 1;
    }
    void pu(int now) {
        tr[now].size = 1 + tr[tr[now].l].size + tr[tr[now].r].size;
        // tr[tr[now].l].f = now, tr[tr[now].r].f = now, tr[now].f = 0;
        // tr[now].m = min({tr[now].k, tr[tr[now].l].m, tr[tr[now].r].m});
        // if (tr[now].l)
        // if (tr[now].r)
    }
    int insert(int &x, int pos, T k, uint32_t v) {
        int y;
        if (!x || tr[x].v > v) {
            y = newnode(k, v);
            tie(tr[y].l, tr[y].r) = split_by_order(x, pos); x = y;
        } else {
            x = copy(x); pd(x);
            if (tr[tr[x].l].size > pos) y = insert(tr[x].l, pos, k, v);
            else { pos -= tr[tr[x].l].size + 1; y = insert(tr[x].r, pos, k, v); }
        }
        pu(x); return y;
```

```
    }
    int erase_pos(int &x, int pos) {
        if (!x) return 0;
        int res = x;
        if (tr[tr[x].l].size == pos) { pd(x); x = join(tr[x].l, tr[x].r); if (x) pu(x); }
        else {
            x = copy(x); pd(x);
            if (tr[tr[x].l].size > pos) res = erase_pos(tr[x].l, pos);
            else { pos -= tr[tr[x].l].size + 1; res = erase_pos(tr[x].r, pos); }
            if (res) pu(x);
        }
        return res;
    }
    int cover(int &x, int pos, T k) {
        if (!x) return 0;
        int res = x; x = copy(x); pd(x);
        if (tr[tr[x].l].size == pos) { tr[x].k = k; pu(x); }
        else {
            if (tr[tr[x].l].size > pos) res = cover(tr[x].l, pos, k);
            else { pos -= tr[tr[x].l].size + 1; res = cover(tr[x].r, pos, k); }
            if (res) pu(x);
        }
        return res;
    }

public:
    RgeTrp() {}
    RgeTrp& join(int x) { setr(join(getr(), x)); return *this; }
    RgeTrp& join(RgeTrp& x) { setr(join(getr(), x.getr())); x.setr(0); return *this; }
    RgeTrp& join(RgeTrp&& x) { setr(join(getr(), x.getr())); return *this; }
    RgeTrp& split(int k) { setr(split_by_order(getr(), k).first); return *this; }
    RgeTrp& split(int k, RgeTrp& x) { auto r = split_by_order(getr(), k); setr(r.first);
    ↪  x.setr(r.second); return *this;}
    void build(vector<T>& a) {
        stack<int, vector<int>> s;
        for (auto i : a) {
            int x = newnode(i);
            while (!s.empty() && tr[s.top()].v > tr[x].v) pu(s.top()), s.pop();
            if (s.empty()) tr[x].l = getr(), setr(x);
            else tr[x].l = tr[s.top()].r, tr[s.top()].r = x;
            s.push(x);
        }
        while (!s.empty()) pu(s.top()), s.pop();
    }
    int insert(int pos, T k) { int r = getr(); int res = insert(r, pos, k, rd()); setr(r);
    ↪  return res; }
    void insert(int pos, RgeTrp& k) { RgeTrp x; this->split(pos, x).join(k).join(x); }
    void insert(int pos, RgeTrp&& k) { RgeTrp x; this->split(pos, x).join(k).join(x); }
    int erase(int pos) { int r = getr(); int res = erase_pos(r, pos); setr(r); return res; }
    RgeTrp erase(int l, int r) {
        RgeTrp x, y; this->split(l, x);
        x.split(r - l, y); this->join(y);
        return x;
    }
    int cover(int pos, T k) { int r = getr(); int res = cover(r, pos, k); setr(r); return
    ↪  res; }
    void reverse(int l, int r) {
        RgeTrp x, y; this->split(l, x);
```

```
        x.split(r - l, y); tr[x.getr()].mod();
        this->join(x).join(y);
    }
    /*
    int order_of_key(int k) {
        int ans = tr[tr[k].l].size;
        while (tr[k].f) {
            if (tr[tr[k].f].r == k) ans += tr[tr[tr[k].f].l].size + 1;
            k = tr[k].f;
        }
        return ans;
    }
    int find_min_order() {
        int x = getr(); pd(x); int ans = tr[tr[x].l].size;
        while (tr[x].m != tr[x].k) {  // || tr[x].m == tr[tr[x].l].m  // leftmost
            if (tr[x].m == tr[tr[x].l].m) { x = tr[x].l; ans -= tr[tr[x].r].size + 1; }
            else { x = tr[x].r; ans += tr[tr[x].l].size + 1; }
            pd(x);
        }
        return ans;
    }
    */
};

// cur = 1;
```

3.7 可持久化并查集

```
#include <bits/stdc++.h>
#include <ext/rope>
using namespace __gnu_cxx;
using namespace std;
vector<rope<int>> bcj;
inline int gr(int i, int x) {
    int z = bcj[i][x];
    if (z == x) return x;
    int f = gr(i, z);
    if (f != z) bcj[i].replace(x, f);
    return f;
}
inline void mg(int i, int x, int y) {
    x = gr(i, x), y = gr(i, y);
    if (x ^ y) bcj[i].replace(y, x);
}
```

3.8 分块

```
#include <bits/stdc++.h>
using namespace std;
using LL = int64_t;
template<typename T>
class Part_Blocks {
    int k;
    vector<T> a, tag, sum;

 public:
    Part_Blocks(vector<T>&& b) : a(b) {
        int n = a.size();
        k = sqrt(n);
```

```
        int m = n / k;
        if (m * k != n) m++;
        tag.resize(m); sum.resize(m);
        for (int i = 0; i < m; i++) {
            for (int j = anum(i); j < anum(i + 1); j++) {
                sum[i] += a[j];
            }
        }
    }
    inline int anum(int v) { return min<int>(v * k, a.size()); }
    inline int bnum(int v) { return v / k; }
    void aupd(int l, int r, T v) {
        int b = bnum(l);
        for (int i = l; i < r; i++) a[i] += v, sum[b] += v;
    }
    void bupd(int l, int r, T v) {
        for (int i = l; i < r; i++) tag[i] += v;
    }
    void upd(int l, int r, T v) {
        int L = bnum(l), R = bnum(r - 1);
        if (L == R) aupd(l, r, v);
        else { aupd(l, anum(L + 1), v), bupd(L + 1, R, v), aupd(anum(R), r, v); }
    }
    T aque(int l, int r) {
        int b = bnum(l); T ans = 0;
        for (int i = l; i < r; i++) ans += a[i] + tag[b];
        return ans;
    }
    T bque(int l, int r) {
        T ans = 0;
        for (int i = l; i < r; i++) ans += sum[i] + tag[i] * k;
        return ans;
    }
    T que(int l, int r) {
        int L = bnum(l), R = bnum(r - 1);
        if (L == R) return aque(l, r);
        else { return aque(l, anum(L + 1)) + bque(L + 1, R) + aque(anum(R), r); }
    }
};
```

3.9 莫队算法

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    struct Q {
        int l, r, id, block;  // [,)
    };
    int q;
    cin >> q;
    vector<Q> que(q);
    for (int i = 0; i < q; i++) {
        cin >> que[i].l >> que[i].r;
        que[i].id = i;
    }
    int v = sqrt(q);
    for (auto& i : que) i.block = i.l / v;
    sort(que.begin(), que.end(), [&](const Q& s, const Q& e) {
```

```
        if (s.block == e.block) return s.r < e.r;
        return s.l < e.l;
    });
    vector<T> ans(q);
    int l = 0, r = 0;
    for (auto& i : que) {
        for (; r < i.r; r++) upd(r, 1);
        for (; r > i.r; r--) upd(r - 1, -1);
        for (; l < i.l; l++) upd(l, -1);
        for (; l > i.l; l--) upd(l - 1, 1);
        ans[i.id] = ;
    }
}
```

4 字符串

4.1 KMP

```cpp
// Copyright 2017 Parallelc
// nex[i] 表示从 0 到 i - 1 的最长公共前后缀中前缀的下一位，如 abcdabc，nex[7] 为 3，即在 7 失
↪ 配时应该跳到 3 去匹配
#include <vector>
#include <string>
using namespace std;  // NOLINT
template <typename T>
class KMP {
 public:
    vector<T> sub_str;
    vector<int> nex;
    int sub_len;
    KMP() {}
    explicit KMP(const vector<T> &sub_str) {
        this->sub_str = sub_str;
        sub_len = sub_str.size();
        this->get_nex();
    }
    explicit KMP(const string &sub_str) {
        this->sub_str.assign(sub_str.begin(), sub_str.end());
        sub_len = sub_str.length();
        this->get_nex();
    }
    void get_nex() {
        nex.resize(sub_len + 1);
        int i = 0, j = -1;
        nex[0] = -1;
        while (i < sub_len) {
            if (-1 == j || sub_str[i] == sub_str[j]) {
                i++;
                j++;
             // if (sub_str[i] != sub_str[j]) nex[i] = j;
             // else nex[i] = nex[j];
                nex[i] = j;
            } else {
                j = nex[j];
            }
        }
    }
    int get_result(const vector<T> &ori_str, int ori_len, int pos = 0) {
        auto i = pos - 1, j = -1;
        while (i < ori_len) {
            if (-1 == j || ori_str[i] == sub_str[j]) {
                i++;
                j++;
                if (j == sub_len) {
                   return i - sub_len + 1;
                   j = nex[j];
                }
            } else {
                j = nex[j];
            }
        }
        return -1;
    }
```

```
    int get_cycle(int len) {
        if (len % (len - nex[len]) == 0)  return len / (len - nex[len]);
        else return 1;
    }
};
```

4.2 shift-or

```
#include <bits/stdc++.h>
using namespace std;
#include <tr2/dynamic_bitset>
using namespace tr2;
using db = dynamic_bitset<>;
int sh_or1(string& s, string& t) {
    int n = s.length();
    int m = t.length();
    static vector<db> b(26);
    for (auto& i : b) i.set(), i.resize(m, 1);
    for (int i = 0; i < n; i++) b[t[i] - 'a'].reset(i);
    static db d;
    d.resize(m);
    int ans = 0;
    for (int i = 0; i < n; i++) {
        (d <<= 1) |= b[s[i] - '0'];
        if (i >= m - 1 && !d[m - 1]) ans++;
    }
    return ans;
}
int sh_or2(string& s, string& t) {
    int n = s.length();
    int m = t.length();
    static vector<db> b(26);
    for (auto& i : b) i.set(), i.resize(n, 1);
    for (int i = 0; i < n; i++) b[s[i] - 'a'].reset(i);
    static db d;
    d.reset();
    d.resize(n);
    for (int i = 0; i < m; i++) (d <<= 1) |= b[t[i] - '0'];
    int ans = 0;
    for (int i = m - 1; i < n; i++) if (!d[i]) ans++;
    return ans;
}
```

4.3 哈希

```
#include <bits/stdc++.h>
using namespace std;
using ULL = uint64_t;
int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    int maxn = 50000;
    ULL base = 31;
    vector<ULL> x(maxn + 1);
    x[0] = 1;
    for (int i = 1; i <= maxn; i++) {
        x[i] = x[i - 1] * base;
    }
    auto get = [&] (vector<ULL>& h, int l, int r) {
```

```
        return h[r] - h[l] * x[r - l];
    };
    string s;
    cin >> s;
    vector<ULL> h(s.length() + 1);
    for (int i = 1; i < h.size(); i++) {
        h[i] = h[i - 1] * base + s[i - 1] - 'a' + 1;
    }
}
```

4.4 最小表示法

```
// o(n)
#include <bits/stdc++.h>
using namespace std;

int get_min(const string& s) {
    int n = s.length();
    int i = 0, j = 1, k = 0;
    while(i < n && j < n && k < n) {
        int t = s[(i + k) % n] - s[(j + k) % n];
        if (t) {
            if (t > 0) i += k + 1;
            else j += k + 1;
            if (i == j) j++;
            k = 0;
        } else k++;
    }
    return min(i, j);
}
int get_max(const string& s) {
    int n = s.length();
    int i = 0, j = 1, k = 0;
    while(i < n && j < n && k < n) {
        int t = s[(i + k) % n] - s[(j + k) % n];
        if (t) {
            if (t > 0) j += k + 1;
            else i += k + 1;
            if (i == j) j++;
            k = 0;
        } else k++;
    }
    return min(i, j);
}
```

4.5 子序列匹配

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

template <int N = 26>
struct SubSeq {
    vector<int> f[N];
    SubSeq(string& s) {
        int n = s.length();
        for (auto& i : f) i.resize(n + 1);
        vector<int> cur(N);
        for (int i = 0; i < n; i++) {
            int c = s[i] - 'a';
```

```
            for (int j = cur[c]; j < i + 1; j++) f[c][j] = i + 1;
            cur[c] = i + 1;
        }
    }
    int get_result(string& s) {
        int cur = 0, n = s.length();
        for (int i = 0; i < n; i++) {
            cur = f[s[i] - 'a'][cur];
            if (cur == 0) return i;
        }
        return n;
    }
};
```

5 动态规划

5.1 最长公共子序列

```cpp
#include <bits/stdc++.h>
using namespace std;
template <typename T>
int lcs(vector<T>& a, vector<T>& b) {
    int n = a.size();
    int m = b.size();
    static vector<vector<int>> dp;
    dp.resize(n + 1);
    for (auto& i : dp) i.resize(m + 1);
    for (int i = 1; i <= n; i++) {
        for(int j = 1; j <= m; j++) {
            if(a[i - 1] == b[j - 1]) dp[i][j] = dp[i - 1][j - 1] + 1;
            else dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
        }
    }
    return dp[n][m];
    static vector<T> ans;
    ans.resize(dp[n][m]);
    while (dp[n][m]) {
        if (a[n - 1] == b[m - 1]) {
            ans[dp[n - 1][m - 1]] = a[n - 1];
            n--, m--;
        } else {
            if (dp[n][m] == dp[n - 1][m]) n--;
            else if (dp[n][m] == dp[n][m - 1]) m--;
        }
    }
    return ans;
}
```

5.2 最长上升子序列

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

int LIS(vector<int>& a) {
    vector<int> b;
    b.reserve(a.size());
    for (auto i : a) {
        auto it = lower_bound(b.begin(), b.end(), i);  // upper: not decrease
        if (it != b.end()) *it = i;
        else b.push_back(i);
    }
    return b.size();
}
```

5.3 区间最值

```cpp
#include <bits/stdc++.h>
using namespace std;
template <typename T, class op = less<pair<T, int>>>
struct RMQ {
    vector<vector<pair<T, int>>> dp;
    RMQ() {}
    RMQ(vector<T>& a) {
```

```
        int n = a.size();
        int m = __lg(n) + 1;
        dp.resize(n);
        for (int i = 0; i < n; i++) {
            dp[i].resize(m);
            dp[i][0] = {a[i], i};
        }
        for (int j = 1; j < m; j++) {
            for (int i = 0; i + (1 << j) <= n; i++) {
                dp[i][j] = min(dp[i][j - 1], dp[i + (1 << (j - 1))][j - 1], op());
            }
        }
    }
    pair<T, int> que(int l, int r) {
        int k = 31 - __builtin_clz(r - l);
        return min(dp[l][k], dp[r - (1 << k)][k], op());
    }
};
```

5.4 背包

5.4.1 01 背包

```
#include <bits/stdc++.h>
using namespace std;
const int INF = 0x3f3f3f3f;
template <typename T>
T ZeroOne(vector<T>& val, vector<int>& wei, int w) {
    static vector<T> dp;
    fill(dp.begin(), dp.end(), 0);
    dp.resize(w + 1);
    // fill(dp.begin(), dp.end(), -INF);
    // dp.resize(w + 1, -INF);
    // dp[0] = 0;
    int n = val.size();
    int sum = accumulate(wei.begin(), wei.end(), 0);
    for (int i = 0; i < n; i++) {
        if (i) sum -= wei[i - 1];
        int bound = max(w - sum, wei[i]);
        for (int j = w; j >= bound; j--) dp[j] = max(dp[j], dp[j - wei[i]] + val[i]);
    }
    return dp[w];
    // if negative, can't fill fully
}
```

5.4.2 超大 01 背包

```
#include <bits/stdc++.h>
using namespace std;
using LL = int64_t;
const LL INF = 0x3f3f3f3f;
const LL mod = 1e9 + 7;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    int n;
    LL W;
    while (cin >> n >> W) {
```

```
        list<pair<LL, LL>> a;
        a.emplace_back(0, 0);
        for (int i = 0; i < n; i++) {
            LL v, w;
            cin >> w >> v;
            if (w > W || v <= 0) continue;
            auto b = a;
            auto s = a.begin();
            for (auto& i : b) {
                i.first += w;
                if (i.first > W) break;
                i.second += v;
                while (s != a.end() && s->first < i.first) s++;
                if (s != a.end()) {
                    if (s->first == i.first) {
                        if (s->second < i.second) s->second = i.second;
                        else continue;
                    } else {
                        if (prev(s)->second < i.second) s = a.insert(s, move(i));
                        else continue;
                    }
                } else {
                    s = a.insert(s, move(i));
                }
                s++;
                while (s != a.end() && s->second <= i.second) s = a.erase(s);
            }
        }
        LL ans = 0;
        for (auto& i : a) ans = max(ans, i.second);
        cout << ans << '\n';
    }
    return 0;
}
```

5.4.3 完全背包

```
#include <bits/stdc++.h>
using namespace std;
const int INF = 0x3f3f3f3f;
template <typename T>
T Complete(vector<T>& val, vector<int>& wei, int w) {
    static vector<T> dp;
    fill(dp.begin(), dp.end(), 0);
    dp.resize(w + 1);
    // fill(dp.begin(), dp.end(), -INF);
    // dp.resize(w + 1, -INF);
    // dp[0] = 0;
    int n = val.size();
    for (int i = 0; i < n; i++) {
        for (int j = wei[i]; j <= w; j++) dp[j] = max(dp[j], dp[j - wei[i]] + val[i]);
    }
    return dp[w];
    // if negative, can't fill fully
}
```

6 类

6.1 大整数类

```
// Copyright 2017 Parallelc
#include <bits/stdc++.h>
using namespace std;  // NOLINT
using LL = int64_t;
const double PI = acos(-1);
void rader(vector<complex<double> >& y) {
    int len = y.size();
    int i, j, k;
    for (i = 1, j = len / 2; i < len - 1; i++) {
        if (i < j) swap(y[i], y[j]);
        k = len / 2;
        while (j >= k) {
            j -= k;
            k /= 2;
        }
        if (j < k) j += k;
    }
}
void fft(vector<complex<double> >& y, int on) {
    int len = y.size();
    rader(y);
    for (int h = 2; h <= len; h <<= 1) {
        complex<double> wn(cos(-on * 2 * PI / h), sin(-on * 2 * PI / h));
        for (int j = 0; j < len; j += h) {
            complex<double> w(1, 0);
            for (int k = j; k < j + h / 2; k++) {
                complex<double> u = y[k];
                complex<double> t = w * y[k + h / 2];
                y[k] = u + t;
                y[k + h / 2] = u - t;
                w = w * wn;
            }
        }
    }
    if (on == -1) for (auto& i : y) i.real(i.real() / len);
}
class BigInt {
 private:
     string num;
     string sign;

 public:
     const string to_string() const {
         if (this->sign == "-") return this->sign + this->num;
         else return this->num;
     }
     const LL toll() { return stoll(this->to_string()); }
     const int toi() { return stoi(this->to_string()); }
     BigInt() : num("0"), sign("+") {}
     BigInt(const int t) {
         if (t < 0) {
             this->num = std::to_string(-t);
             this->sign = "-";
         } else {
             this->num = std::to_string(t);
```

```
        this->sign = "+";
    }
}
BigInt(const LL t) {
    if (t < 0) {
        this->num = std::to_string(-t);
        this->sign = "-";
    } else {
        this->num = std::to_string(t);
        this->sign = "+";
    }
}
BigInt(const string& t) {
    if (t[0] == '-') {
        this->num = t.substr(1);
        this->sign = "-";
    } else {
        this->num = t;
        this->sign = "+";
    }
    int flag = 0;
    while (flag < (int)this->num.length() - 1 && this->num[flag] == '0') flag++;
    this->num = this->num.substr(flag);
}
BigInt(char* const t) : BigInt(string(t)) {}
friend bool operator< (const BigInt& t, const BigInt& s) {
    if (t.sign != s.sign) {
        if (t.sign == "-") return true;
        else return false;
    } else {
        if (t.sign == "-") {
            if (t.num.length() == s.num.length()) {
                return t.num > s.num;
            } else {
                return t.num.length() > s.num.length();
            }
        } else {
            if (t.num.length() == s.num.length()) {
                return t.num < s.num;
            } else {
                return t.num.length() < s.num.length();
            }
        }
    }
}
friend bool operator> (const BigInt& t, const BigInt& s) {
    return s < t;
}
friend bool operator== (const BigInt& t, const BigInt& s) {
    return t.num == s.num && t.sign == s.sign;
}
friend bool operator!= (const BigInt& t, const BigInt& s) {
    return !(t == s);
}
friend bool operator<= (const BigInt& t, const BigInt& s) {
    return t == s || t < s;
}
friend bool operator>= (const BigInt& t, const BigInt& s) {
```

```
    return t == s || t > s;
}
friend const BigInt abs(const BigInt& t) {
    BigInt ans = t;
    if (ans.sign == "-") ans.sign = "+";
    return ans;
}
friend const BigInt operator- (const BigInt& t) {
    BigInt ans = t;
    if (ans.sign == "-") ans.sign = "+";
    else ans.sign = "-";
    return ans;
}
friend istream& operator>> (istream& in, BigInt& t) {
    string s;
    in >> s;
    t = s;
    return in;
}
friend ostream& operator<< (ostream& out, const BigInt& t) {
    out << t.to_string();
    return out;
}
friend const BigInt operator+ (const BigInt& t, const BigInt& s) {
    BigInt ans, sub;
    if (t.num.length() < s.num.length()) {
        ans = s;
        sub = t;
    } else if (t.num.length() == s.num.length()) {
        if (t.num < s.num) {
            ans = s;
            sub = t;
        } else {
            ans = t;
            sub = s;
        }
    } else {
        ans = t;
        sub = s;
    }
    int sub_l = sub.num.length();
    int ans_l = ans.num.length();
    if (t.sign == s.sign) {
        for (int i = 1; i <= sub_l; i++) {
            ans.num[ans_l - i] += sub.num[sub_l - i] - '0';
        }
        int flag = 0;
        for (int i = 1; i <= ans_l; i++) {
            if (ans.num[ans_l - i] > '9') {
                ans.num[ans_l - i] -= 10;
                if (i == ans_l) {
                    flag = 1;
                } else {
                    ans.num[ans_l - i - 1] += 1;
                }
            } else if (i >= sub_l) {
                break;
            }
```

```
        }
        if (flag) ans.num = "1" + ans.num;
    } else {
        for (int i = 1; i <= sub_l; i++) {
            ans.num[ans_l - i] -= sub.num[sub_l - i] - '0';
        }
        for (int i = 1; i <= ans_l; i++) {
            if (ans.num[ans_l - i] < '0') {
                ans.num[ans_l - i] += 10;
                ans.num[ans_l - i - 1] -= 1;
            } else if (i >= sub_l) {
                break;
            }
        }
        int flag = 0;
        while (flag < ans_l - 1 && ans.num[flag] == '0') flag++;
        ans.num = ans.num.substr(flag);
        if (ans.num == "0") ans.sign = "+";
    }
    return ans;
}
friend const BigInt operator- (const BigInt& t, const BigInt& s) {
    BigInt sub = s;
    if (sub.sign == "+") sub.sign = "-";
    else sub.sign = "+";
    return t + sub;
}
friend const BigInt operator* (const BigInt& t, const BigInt& s) {
    BigInt res;
    if (s.sign == t.sign) res.sign = "+";
    else res.sign = "-";
    vector<complex<double> > x1, x2;
    vector<int> sum;
    string str1 = t.num, str2 = s.num;
    int len1 = str1.length();
    int len2 = str2.length();
    int len = 1;
    while (len < len1 * 2 || len < len2 * 2) len <<= 1;
    for (int i = 0; i < len1; i++) {
        x1.push_back(complex<double>(str1[len1 - 1 - i] - '0', 0));
    }
    for (int i = len1; i < len; i++) {
        x1.push_back(complex<double>(0, 0));
    }
    for (int i = 0; i < len2; i++) {
        x2.push_back(complex<double>(str2[len2 - 1 -i] - '0', 0));
    }
    for (int i = len2; i < len; i++) {
        x2.push_back(complex<double>(0, 0));
    }
    fft(x1, 1);
    fft(x2, 1);
    for (int i = 0; i < len; i++) x1[i] = x1[i] * x2[i];
    fft(x1, -1);
    for (auto& i : x1) sum.push_back((int)(i.real() + 0.5));
    for (int i = 0; i < len; i++) {
        sum[i + 1] += sum[i] / 10;
        sum[i] %= 10;
```

```
    }
    len = len1 + len2 - 1;
    while (sum[len] <= 0 && len > 0) len--;
    res.num = "";
    for (int i = len; i >= 0; i--) res.num += sum[i] + '0';
    if (res.num == "0") res.sign = "+";
    return res;
}
friend const BigInt operator/ (const BigInt& t, const BigInt& s) {
    if (s == 0) throw;
    BigInt res;
    if (s.sign == t.sign) res.sign = "+";
    else res.sign = "-";
    BigInt sub = abs(t), ans = abs(s);
    int w = sub.num.length() - ans.num.length();
    for (int i = 0; i < w; i++) ans.num += "0";
    while (w >= 0) {
        int d = 0;
        while (ans <= sub) {
            sub -= ans;
            d++;
        }
        res.num += d + '0';
        ans.num = ans.num.substr(0, ans.num.length() - 1);
        w--;
    }
    int flag = 0;
    while (flag < (int)res.num.length() - 1 && res.num[flag] == '0') flag++;
    res.num = res.num.substr(flag);
    if (res.num == "0") res.sign = "+";
    return res;
}
friend const BigInt operator% (const BigInt& t, const BigInt& s) {
    if (s == 0) throw;
    BigInt sub = abs(t), ans = abs(s);
    int w = sub.num.length() - ans.num.length();
    for (int i = 0; i < w; i++) ans.num += "0";
    while (w >= 0) {
        int d = 0;
        while (ans <= sub) {
            sub -= ans;
            d++;
        }
        w--;
        ans.num = ans.num.substr(0, ans.num.length() - 1);
    }
    sub.sign = t.sign;
    if (sub.num == "0") sub.sign = "+";
    return sub;
}
friend BigInt& operator+= (BigInt& t, const BigInt& s) {
    return t = t + s;
}
friend BigInt& operator-= (BigInt& t, const BigInt& s) {
    return t = t - s;
}
friend BigInt& operator*= (BigInt& t, const BigInt& s) {
    return t = t * s;
```

```
    }
    friend BigInt& operator/= (BigInt& t, const BigInt& s) {
        return t = t / s;
    }
    friend BigInt& operator%= (BigInt& t, const BigInt& s) {
        return t = t % s;
    }
    const BigInt subnum(int r, int l) {
        BigInt ans = this->num.substr(this->num.length() - l, l - r);
        ans.sign = this->sign;
        return ans;
    }
    const BigInt subnum(int l) {
        return this->subnum(0, l);
    }
};
```

6.2 分数类

```
// Copyright 2017 Parallelc
#include <bits/stdc++.h>
using namespace std;  // NOLINT
template <typename T>
class Frac {
 private:
    void simply() {
        T g = __gcd(num, den);
        num /= g;
        den /= g;
        if (den < T(0)) {
            den = -den;
            num = -num;
        }
    }

 public:
    T num, den;
    Frac(const T& t = 0, const T& s = 1) : num(t), den(s) { simply(); }
    double to_double() const { return (double)num / den; }
    long double to_ldouble() const { return (long double)num / den; }
    friend const Frac operator- (const Frac& t) {
        auto ans = t;
        ans.num = -ans.num;
        return ans;
    }
    friend const Frac abs(const Frac& t) {
        auto ans = t;
        if (ans.num < T(0)) ans.num = -ans.num;
        return ans;
    }
    friend ostream& operator<< (ostream& out, const Frac& t) {
        out << t.num << "/" << t.den;
        return out;
    }
    friend const Frac operator+ (const Frac& t, const Frac& s) {
        auto ans = t;
        ans.num = ans.num * s.den + s.num * ans.den;
        ans.den *= s.den;
```

```
        ans.simply();
        return ans;
    }
    friend const Frac operator- (const Frac& t, const Frac& s) {
        return t + (-s);
    }
    friend const Frac operator* (const Frac& t, const Frac& s) {
        auto ans = t;
        ans.num *= s.num;
        ans.den *= s.den;
        ans.simply();
        return ans;
    }
    friend const Frac operator/ (const Frac& t, const Frac& s) {
        if (s == T(0)) throw;
        auto ans = s;
        swap(ans.num, ans.den);
        return t * ans;
    }
    friend Frac& operator+= (Frac& t, const Frac& s) {
        return t = t + s;
    }
    friend Frac& operator-= (Frac& t, const Frac& s) {
        return t = t - s;
    }
    friend Frac& operator*= (Frac& t, const Frac& s) {
        return t = t * s;
    }
    friend Frac& operator/= (Frac& t, const Frac& s) {
        return t = t / s;
    }
    friend bool operator< (const Frac& t, const Frac& s) {
        return t.num * s.den < s.num * t.den;
    }
    friend bool operator> (const Frac& t, const Frac& s) {
        return s < t;
    }
    friend bool operator== (const Frac& t, const Frac& s) {
        return t.num == s.num && t.den == s.den;
    }
    friend bool operator!= (const Frac& t, const Frac& s) {
        return !(t == s);
    }
    friend bool operator<= (const Frac& t, const Frac& s) {
        return t < s || t == s;
    }
    friend bool operator>= (const Frac& t, const Frac& s) {
        return t > s || t == s;
    }
};
```

6.3 矩阵类

```
// Copyright 2017 Parallelc
#include <bits/stdc++.h>
using namespace std;  // NOLINT
template <typename T>
class Mat {
```

```
public:
    int n, m;
    vector<vector<T>> val;
    Mat(int n, int m) : n(n), m(m) {
       val.resize(n);
       for (auto& i : val) {
           i.resize(m);
       }
    }
    explicit Mat(int n) : Mat(n, n) {}
    friend istream& operator>> (istream& in, Mat& t) {
        for (auto& i : t.val) {
            for (auto& j : i) in >> j;
        }
        return in;
    }
    friend ostream& operator<< (ostream& out, const Mat& t) {
        for (auto& i : t.val) {
            for (int j = 0; j < t.m; j++) {
                out << i[j];
                if (j == t.m - 1) out << endl;
                else out << " ";
            }
        }
        return out;
    }
    friend const Mat operator- (const Mat& t) {
        auto ans = t;
        for (auto& i : ans.val) {
            for (auto& j : i) {
                j = -j;
            }
        }
        return ans;
    }
    friend const Mat operator+ (const Mat& t, const Mat& s) {
        if (t.n != s.n || t.m != s.m) throw;
        auto ans = t;
        for (int i = 0; i < ans.n; i++) {
            for (int j = 0; j < ans.m; j++) {
               ans.val[i][j] += s.val[i][j];
            }
        }
        return ans;
    }
    friend const Mat operator- (const Mat& t, const Mat& s) {
        return t + (-s);
    }
    friend const Mat operator* (const Mat& t, const Mat& s) {
        if (t.m != s.n) throw;
        Mat ans(t.n, s.m);
        for (int i = 0; i < ans.n; i++) {
            for (int j = 0; j < ans.m; j++) {
                for (int k = 0; k < t.m; k++) {
                    ans.val[i][j] += t.val[i][k] * s.val[k][j];  // % mod;
                    // ans.val[i][j] %= mod;
                }
            }
```

```
        }
        return ans;
    }
    template <typename S>
    friend const Mat operator* (const Mat& t, const S& s) {
       auto ans = t;
       for (auto& i : ans.val) {
           for (auto& j : i) {
               j *= s;
           }
       }
       return ans;
    }
    template <typename S>
    friend const Mat operator/ (const Mat& t, const S& s) {
        auto ans = t;
        for (auto& i : ans.val) {
            for (auto& j : i) {
                j /= s;
            }
        }
        return ans;
    }
    template <typename S>
    friend const Mat operator% (const Mat& t, const S& s) {
        auto ans = t;
        for (auto& i : ans.val) {
            for (auto& j : i) {
                j %= s;
            }
        }
        return ans;
    }
    template <typename S>
    friend const Mat operator* (const S& s, const Mat& t) {
        return t * s;
    }
};
```

7 计算几何

```
/*
 * 二维 ACM 计算几何模板
 * 注意变量类型更改和 EPS
 */

#include <bits/stdc++.h>
using namespace std;
const double eps = 1e-8;
const double PI = acos(-1.0);
//点
class point
{
 public:
     double x, y;
     point(){};
     point(double x, double y):x(x),y(y){};

     static int xmult(const point &ps, const point &pe, const point &po)
     {
         return (ps.x - po.x) * (pe.y - po.y) - (pe.x - po.x) * (ps.y - po.y);
     }

     //相对原点的差乘结果，参数：点 [_Off]
     //即由原点和这两个点组成的平行四边形面积
     double operator *(const point &_Off) const
     {
         return x * _Off.y - y * _Off.x;
     }
     //相对偏移
     point operator - (const point &_Off) const
     {
         return point(x - _Off.x, y - _Off.y);
     }
     //点位置相同 (double 类型)
     bool operator == (const point &_Off) const
     {
         return fabs(_Off.x - x) < eps && fabs(_Off.y - y) < eps;
     }
     //点位置不同 (double 类型)
     bool operator != (const point &_Off) const
     {
         return ((*this) == _Off) == false;
     }
     //两点间距离的平方
     double dis2(const point &_Off) const
     {
         return (x - _Off.x) * (x - _Off.x) + (y - _Off.y) * (y - _Off.y);
     }
     //两点间距离
     double dis(const point &_Off) const
     {
         return sqrt((x - _Off.x) * (x - _Off.x) + (y - _Off.y) * (y - _Off.y));
     }
};

//两点表示的向量
```

```
class pVector
{
 public:
     point s, e;//两点表示，起点 [s]，终点 [e]
     double a, b, c;//一般式,ax+by+c=0

     pVector(){}
     pVector(const point &s, const point &e):s(s),e(e){}

     //向量与点的叉乘，参数：点 [_Off]
     //[点相对向量位置判断]
     double operator *(const point &_Off) const
     {
         return (_Off.y - s.y) * (e.x - s.x) - (_Off.x - s.x) * (e.y - s.y);
     }
     //向量与向量的叉乘，参数：向量 [_Off]
     double operator *(const pVector &_Off) const
     {
         return (e.x - s.x) * (_Off.e.y - _Off.s.y) - (e.y - s.y) * (_Off.e.x - _Off.s.x);
     }
     //从两点表示转换为一般表示
     bool pton()
     {
         a = s.y - e.y;
         b = e.x - s.x;
         c = s.x * e.y - s.y * e.x;
         return true;
     }

     //-----------点和直线（向量）-----------
     //点在向量左边（右边的小于号改成大于号即可，在对应直线上则加上 = 号）
     //参数：点 [_Off]，向量 [_Ori]
     friend bool operator<(const point &_Off, const pVector &_Ori)
     {
         return (_Ori.e.y - _Ori.s.y) * (_Off.x - _Ori.s.x)
             < (_Off.y - _Ori.s.y) * (_Ori.e.x - _Ori.s.x);
     }

     //点在直线上，参数：点 [_Off]
     bool lhas(const point &_Off) const
     {
         return fabs((*this) * _Off) < eps;
     }
     //点在线段上，参数：点 [_Off]
     bool shas(const point &_Off) const
     {
         return lhas(_Off)
             && _Off.x - min(s.x, e.x) > -eps && _Off.x - max(s.x, e.x) < eps
             && _Off.y - min(s.y, e.y) > -eps && _Off.y - max(s.y, e.y) < eps;
     }

     //点到直线/线段的距离
     //参数：  点 [_Off]，是否是线段 [isSegment](默认为直线)
     double dis(const point &_Off, bool isSegment = false)
     {
         //化为一般式
         pton();
```

```
    //到直线垂足的距离
    double td = (a * _Off.x + b * _Off.y + c) / sqrt(a * a + b * b);

    //如果是线段判断垂足
    if(isSegment)
    {
        double xp = (b * b * _Off.x - a * b * _Off.y - a * c) / ( a * a + b * b);
        double yp = (-a * b * _Off.x + a * a * _Off.y - b * c) / (a * a + b * b);
        double xb = max(s.x, e.x);
        double yb = max(s.y, e.y);
        double xs = s.x + e.x - xb;
        double ys = s.y + e.y - yb;
        if(xp > xb + eps || xp < xs - eps || yp > yb + eps || yp < ys - eps)
            td = min(_Off.dis(s), _Off.dis(e));
    }

    return fabs(td);
}

//关于直线对称的点
point mirror(const point &_Off) const
{
    //注意先转为一般式
    point ret;
    double d = a * a + b * b;
    ret.x = (b * b * _Off.x - a * a * _Off.x - 2 * a * b * _Off.y - 2 * a * c) / d;
    ret.y = (a * a * _Off.y - b * b * _Off.y - 2 * a * b * _Off.x - 2 * b * c) / d;
    return ret;
}
//计算两点的中垂线
static pVector ppline(const point &_a, const point &_b)
{
    pVector ret;
    ret.s.x = (_a.x + _b.x) / 2;
    ret.s.y = (_a.y + _b.y) / 2;
    //一般式
    ret.a = _b.x - _a.x;
    ret.b = _b.y - _a.y;
    ret.c = (_a.y - _b.y) * ret.s.y + (_a.x - _b.x) * ret.s.x;
    //两点式
    if(fabs(ret.a) > eps)
    {
        ret.e.y = 0.0;
        ret.e.x = - ret.c / ret.a;
        if(ret.e == ret. s)
        {
            ret.e.y = 1e10;
            ret.e.x = - (ret.c - ret.b * ret.e.y) / ret.a;
        }
    }
    else
    {
        ret.e.x = 0.0;
        ret.e.y = - ret.c / ret.b;
        if(ret.e == ret. s)
        {
            ret.e.x = 1e10;
            ret.e.y = - (ret.c - ret.a * ret.e.x) / ret.b;
```

```
            }
        }
        return ret;
    }

    //------------直线和直线（向量）-------------
    //直线重合，参数：直线向量 [_Off]
    bool equal(const pVector &_Off) const
    {
        return lhas(_Off.e) && lhas(_Off.s);
    }
    //直线平行，参数：直线向量 [_Off]
    bool parallel(const pVector &_Off) const
    {
        return fabs((*this) * _Off) < eps;
    }
    //两直线交点，参数：目标直线 [_Off]
    point crossLPt(pVector _Off)
    {
        //注意先判断平行和重合
        point ret = s;
        double t = ((s.x - _Off.s.x) * (_Off.s.y - _Off.e.y) - (s.y - _Off.s.y) * (_Off.s.x
        ↪  - _Off.e.x))
            / ((s.x - e.x) * (_Off.s.y - _Off.e.y) - (s.y - e.y) * (_Off.s.x - _Off.e.x));
        ret.x += (e.x - s.x) * t;
        ret.y += (e.y - s.y) * t;
        return ret;
    }

    //------------线段和直线（向量）----------
    //线段和直线交
    //参数：线段 [_Off]
    bool crossSL(const pVector &_Off) const
    {
        double rs = (*this) * _Off.s;
        double re = (*this) * _Off.e;
        return rs * re < eps;
    }

    //------------线段和线段（向量）----------
    //判断线段是否相交（注意添加 eps），参数：线段 [_Off]
    bool isCrossSS(const pVector &_Off) const
    {
        //1. 快速排斥试验判断以两条线段为对角线的两个矩形是否相交
        //2. 跨立试验（等于 0 时端点重合）
        return (
                (max(s.x, e.x) >= min(_Off.s.x, _Off.e.x)) &&
                (max(_Off.s.x, _Off.e.x) >= min(s.x, e.x)) &&
                (max(s.y, e.y) >= min(_Off.s.y, _Off.e.y)) &&
                (max(_Off.s.y, _Off.e.y) >= min(s.y, e.y)) &&
                ((pVector(_Off.s, s) * _Off) * (_Off * pVector(_Off.s, e)) >= 0.0) &&
                ((pVector(s, _Off.s) * (*this)) * ((*this) * pVector(s, _Off.e)) >= 0.0)
               );
    }
};

class polygon
{
```

```
public:
    const static long maxpn = 100;
    point pt[maxpn];//点（顺时针或逆时针）
    long n;//点的个数

    point& operator[](int _p)
    {
        return pt[_p];
    }

    //求多边形面积，多边形内点必须顺时针或逆时针
    double area() const
    {
        double ans = 0.0;
        int i;
        for(i = 0; i < n; i ++)
        {
            int nt = (i + 1) % n;
            ans += pt[i].x * pt[nt].y - pt[nt].x * pt[i].y;
        }
        return fabs(ans / 2.0);
    }
    //求多边形重心，多边形内点必须顺时针或逆时针
    point gravity() const
    {
        point ans;
        ans.x = ans.y = 0.0;
        int i;
        double area = 0.0;
        for(i = 0; i < n; i ++)
        {
            int nt = (i + 1) % n;
            double tp = pt[i].x * pt[nt].y - pt[nt].x * pt[i].y;
            area += tp;
            ans.x += tp * (pt[i].x + pt[nt].x);
            ans.y += tp * (pt[i].y + pt[nt].y);
        }
        ans.x /= 3 * area;
        ans.y /= 3 * area;
        return ans;
    }
    //判断点在凸多边形内，参数：点 [_Off]
    bool chas(const point &_Off) const
    {
        double tp = 0, np;
        int i;
        for(i = 0; i < n; i ++)
        {
            np = pVector(pt[i], pt[(i + 1) % n]) * _Off;
            if(tp * np < -eps)
                return false;
            tp = (fabs(np) > eps)?np: tp;
        }
        return true;
    }
    //判断点是否在任意多边形内 [射线法]，O(n)
    bool ahas(const point &_Off) const
    {
```

```
    int ret = 0;
    double infv = 1e-10;//坐标系最大范围
    pVector l = pVector(_Off, point( -infv ,_Off.y));
    for(int i = 0; i < n; i ++)
    {
        pVector ln = pVector(pt[i], pt[(i + 1) % n]);
        if(fabs(ln.s.y - ln.e.y) > eps)
        {
            point tp = (ln.s.y > ln.e.y)? ln.s: ln.e;
            if(fabs(tp.y - _Off.y) < eps && tp.x < _Off.x + eps)
                ret ++;
        }
        else if(ln.isCrossSS(l))
            ret ++;
    }
    return (ret % 2 == 1);
}
//凸多边形被直线分割，参数：直线 [_Off]
polygon split(pVector _Off)
{
    //注意确保多边形能被分割
    polygon ret;
    point spt[2];
    double tp = 0.0, np;
    bool flag = true;
    int i, pn = 0, spn = 0;
    for(i = 0; i < n; i ++)
    {
        if(flag)
            pt[pn ++] = pt[i];
        else
            ret.pt[ret.n ++] = pt[i];
        np = _Off * pt[(i + 1) % n];
        if(tp * np < -eps)
        {
            flag = !flag;
            spt[spn ++] = _Off.crossLPt(pVector(pt[i], pt[(i + 1) % n]));
        }
        tp = (fabs(np) > eps)?np: tp;
    }
    ret.pt[ret.n ++] = spt[0];
    ret.pt[ret.n ++] = spt[1];
    n = pn;
    return ret;
}

//-------------凸包-------------
//Graham 扫描法，复杂度 O(nlg(n))，结果为逆时针
static bool graham_cmp(const point &l, const point &r)//凸包排序函数
{
    return l.y < r.y || (l.y == r.y && l.x < r.x);
}
polygon& graham(point _p[], int _n)
{
    int i, len;
    sort(_p, _p + _n, polygon::graham_cmp);
    n = 1;
    pt[0] = _p[0], pt[1] = _p[1];
```

```
    for(i = 2; i < _n; i ++)
    {
        while(n && point::xmult(_p[i], pt[n], pt[n - 1]) >= 0)
            n --;
        pt[++ n] = _p[i];
    }
    len = n;
    pt[++ n] = _p[_n - 2];
    for(i = _n - 3; i >= 0; i --)
    {
        while(n != len && point::xmult(_p[i], pt[n], pt[n - 1]) >= 0)
            n --;
        pt[++ n] = _p[i];
    }
    return (*this);
}

//凸包旋转卡壳 (注意点必须顺时针或逆时针排列)
//返回值凸包直径的平方 (最远两点距离的平方)
double rotating_calipers()
{
    int i = 1;
    double ret = 0.0;
    pt[n] = pt[0];
    for(int j = 0; j < n; j ++)
    {
        while(fabs(point::xmult(pt[j], pt[j + 1], pt[i + 1])) >
        ↪  fabs(point::xmult(pt[j], pt[j + 1], pt[i])) + eps)
            i = (i + 1) % n;
        //pt[i] 和 pt[j],pt[i + 1] 和 pt[j + 1] 可能是对踵点
        ret = max(ret, max(pt[i].dis(pt[j]), pt[i + 1].dis(pt[j + 1])));
    }
    return ret;
}

//凸包旋转卡壳 (注意点必须逆时针排列)
//返回值两凸包的最短距离
double rotating_calipers(polygon &_Off)
{
    int i = 0;
    double ret = 1e10;//inf
    pt[n] = pt[0];
    _Off.pt[_Off.n] = _Off.pt[0];
    //注意凸包必须逆时针排列且 pt[0] 是左下角点的位置
    while(_Off.pt[i + 1].y > _Off.pt[i].y)
        i = (i + 1) % _Off.n;
    for(int j = 0; j < n; j ++)
    {
        double tp;
        //逆时针时为 >, 顺时针则相反
        while((tp = point::xmult(pt[j], pt[j + 1], _Off.pt[i + 1]) - point::xmult(
        ↪  pt[j], pt[j + 1], _Off.pt[i])) > eps)
            i = (i + 1) % _Off.n;
        //(pt[i],pt[i+1]) 和 (_Off.pt[j],_Off.pt[j + 1]) 可能是最近线段
        ret = min(ret, pVector(pt[j], pt[j + 1]).dis(_Off.pt[i], true));
        ret = min(ret, pVector(_Off.pt[i], _Off.pt[i + 1]).dis(pt[j + 1], true));
        if(tp > -eps)//如果不考虑 TLE 问题最好不要加这个判断
        {
```

```
            ret = min(ret, pVector(pt[j], pt[j + 1]).dis(_Off.pt[i + 1], true));
            ret = min(ret, pVector(_Off.pt[i], _Off.pt[i + 1]).dis(pt[j], true));
        }
    }
    return ret;
}

//-----------半平面交-------------
//复杂度:O(nlog2(n))
//半平面计算极角函数 [如果考虑效率可以用成员变量记录]
static double hpc_pa(const pVector &_Off)
{
    return atan2(_Off.e.y - _Off.s.y, _Off.e.x - _Off.s.x);
}
//半平面交排序函数 [优先顺序: 1. 极角 2. 前面的直线在后面的左边]
static bool hpc_cmp(const pVector &l, const pVector &r)
{
    double lp = hpc_pa(l), rp = hpc_pa(r);
    if(fabs(lp - rp) > eps)
        return lp < rp;
    return point::xmult(l.s, r.e, r.s) < 0.0;
}
//用于计算的双端队列
pVector dequeue[maxpn];
//获取半平面交的多边形 (多边形的核)
//参数: 向量集合 [l], 向量数量 [ln];(半平面方向在向量左边)
//函数运行后如果 n[即返回多边形的点数量] 为 0 则不存在半平面交的多边形 (不存在区域或区域面
↪  积无穷大)
polygon& halfPanelCross(pVector _Off[], int ln)
{
    int i, tn;
    n = 0;
    sort(_Off, _Off + ln, hpc_cmp);
    //平面在向量左边的筛选
    for(i = tn = 1; i < ln; i ++)
        if(fabs(hpc_pa(_Off[i]) - hpc_pa(_Off[i - 1])) > eps)
            _Off[tn ++] = _Off[i];
    ln = tn;
    int bot = 0, top = 1;
    dequeue[0] = _Off[0];
    dequeue[1] = _Off[1];
    for(i = 2; i < ln; i ++)
    {
        if(dequeue[top].parallel(dequeue[top - 1]) ||
                dequeue[bot].parallel(dequeue[bot + 1]))
            return (*this);
        while(bot < top &&
                point::xmult(dequeue[top].crossLPt(dequeue[top - 1]), _Off[i].e,
                ↪  _Off[i].s) > eps)
            top --;
        while(bot < top &&
                point::xmult(dequeue[bot].crossLPt(dequeue[bot + 1]), _Off[i].e,
                ↪  _Off[i].s) > eps)
            bot ++;
        dequeue[++ top] = _Off[i];
    }

    while(bot < top &&
```

```
                point::xmult(dequeue[top].crossLPt(dequeue[top - 1]), dequeue[bot].e,
                ↪  dequeue[bot].s) > eps)
            top --;
        while(bot < top &&
                point::xmult(dequeue[bot].crossLPt(dequeue[bot + 1]), dequeue[top].e,
                ↪  dequeue[top].s) > eps)
            bot ++;
        //计算交点 (注意不同直线形成的交点可能重合)
        if(top <= bot + 1)
            return (*this);
        for(i = bot; i < top; i ++)
            pt[n ++] = dequeue[i].crossLPt(dequeue[i + 1]);
        if(bot < top + 1)
            pt[n ++] = dequeue[bot].crossLPt(dequeue[top]);
        return (*this);
    }
};
class circle
{
 public:
    point c;//圆心
    double r;//半径
    double db, de;//圆弧度数起点， 圆弧度数终点 (逆时针 0-360)

    //-------圆---------

    //判断圆在多边形内
    bool inside(const polygon &_Off) const
    {
        if(_Off.ahas(c) == false)
            return false;
        for(int i = 0; i < _Off.n; i ++)
        {
            pVector l = pVector(_Off.pt[i], _Off.pt[(i + 1) % _Off.n]);
            if(l.dis(c, true) < r - eps)
                return false;
        }
        return true;
    }

    //判断多边形在圆内 (线段和折线类似)
    bool has(const polygon &_Off) const
    {
        for(int i = 0; i < _Off.n; i ++)
            if(_Off.pt[i].dis2(c) > r * r - eps)
                return false;
        return true;
    }

    //-------圆弧-------
    //圆被其他圆截得的圆弧, 参数: 圆 [_Off]
    circle operator-(circle &_Off) const
    {
        //注意圆必须相交, 圆心不能重合
        double d2 = c.dis2(_Off.c);
        double d = c.dis(_Off.c);
        double ans = acos((d2 + r * r - _Off.r * _Off.r) / (2 * d * r));
        point py = _Off.c - c;
```

```
    double oans = atan2(py.y, py.x);
    circle res;
    res.c = c;
    res.r = r;
    res.db = oans + ans;
    res.de = oans - ans + 2 * PI;
    return res;
}
//圆被其他圆截得的圆弧，参数：圆 [_Off]
circle operator+(circle &_Off) const
{
    //注意圆必须相交，圆心不能重合
    double d2 = c.dis2(_Off.c);
    double d = c.dis(_Off.c);
    double ans = acos((d2 + r * r - _Off.r * _Off.r) / (2 * d * r));
    point py = _Off.c - c;
    double oans = atan2(py.y, py.x);
    circle res;
    res.c = c;
    res.r = r;
    res.db = oans - ans;
    res.de = oans + ans;
    return res;
}

//过圆外一点的两条切线
//参数：点 [_Off](必须在圆外)，返回：两条切线 (切线的 s 点为 _Off,e 点为切点)
pair<pVector, pVector>  tangent(const point &_Off) const
{
    double d = c.dis(_Off);
    //计算角度偏移的方式
    double angp = acos(r / d), ango = atan2(_Off.y - c.y, _Off.x - c.x);
    point pl = point(c.x + r * cos(ango + angp), c.y + r * sin(ango + angp)),
          pr = point(c.x + r * cos(ango - angp), c.y + r * sin(ango - angp));
    return make_pair(pVector(_Off, pl), pVector(_Off, pr));
}

//计算直线和圆的两个交点
//参数：直线 [_Off](两点式)，返回两个交点，注意直线必须和圆有两个交点
pair<point, point> cross(pVector _Off) const
{
    _Off.pton();
    //到直线垂足的距离
    double td = fabs(_Off.a * c.x + _Off.b * c.y + _Off.c) / sqrt(_Off.a * _Off.a +
    ↪  _Off.b * _Off.b);
    if (fabs(td) < eps) {
        double ango = atan2(_Off.s.y - c.y, _Off.s.x - c.x);
        return make_pair(point(c.x + r * cos(ango), c.y + r * sin(ango)),
                point(c.x + r * cos(ango + PI), c.y + r * sin(ango + PI)));
    } else {
        //计算垂足坐标
        double xp = (_Off.b * _Off.b * c.x - _Off.a * _Off.b * c.y - _Off.a * _Off.c) /
        ↪  ( _Off.a * _Off.a + _Off.b * _Off.b);
        double yp = (- _Off.a * _Off.b * c.x + _Off.a * _Off.a * c.y - _Off.b * _Off.c)
        ↪  / (_Off.a * _Off.a + _Off.b * _Off.b);

        double ango = atan2(yp - c.y, xp - c.x);
        double angp = acos(td / r);
```

```
            return make_pair(point(c.x + r * cos(ango + angp), c.y + r * sin(ango + angp)),
                    point(c.x + r * cos(ango - angp), c.y + r * sin(ango - angp)));
        }
    }
};

class triangle
{
 public:
    point a, b, c;//顶点
    triangle(){}
    triangle(point a, point b, point c): a(a), b(b), c(c){}

    //计算三角形面积
    double area()
    {
        return fabs(point::xmult(a, b, c)) / 2.0;
    }

    //计算三角形外心
    //返回: 外接圆圆心
    point circumcenter()
    {
        pVector u,v;
        u.s.x = (a.x + b.x) / 2;
        u.s.y = (a.y + b.y) / 2;
        u.e.x = u.s.x - a.y + b.y;
        u.e.y = u.s.y + a.x - b.x;
        v.s.x = (a.x + c.x) / 2;
        v.s.y = (a.y + c.y) / 2;
        v.e.x = v.s.x - a.y + c.y;
        v.e.y = v.s.y + a.x - c.x;
        return u.crossLPt(v);
    }

    //计算三角形内心
    //返回: 内接圆圆心
    point incenter()
    {
        pVector u, v;
        double m, n;
        u.s = a;
        m = atan2(b.y - a.y, b.x - a.x);
        n = atan2(c.y - a.y, c.x - a.x);
        u.e.x = u.s.x + cos((m + n) / 2);
        u.e.y = u.s.y + sin((m + n) / 2);
        v.s = b;
        m = atan2(a.y - b.y, a.x - b.x);
        n = atan2(c.y - b.y, c.x - b.x);
        v.e.x = v.s.x + cos((m + n) / 2);
        v.e.y = v.s.y + sin((m + n) / 2);
        return u.crossLPt(v);
    }

    //计算三角形垂心
    //返回: 高的交点
    point perpencenter()
```

```
    {
        pVector u,v;
        u.s = c;
        u.e.x = u.s.x - a.y + b.y;
        u.e.y = u.s.y + a.x - b.x;
        v.s = b;
        v.e.x = v.s.x - a.y + c.y;
        v.e.y = v.s.y + a.x - c.x;
        return u.crossLPt(v);
    }

    //计算三角形重心
    //返回：重心
    //到三角形三顶点距离的平方和最小的点
    //三角形内到三边距离之积最大的点
    point barycenter()
    {
        pVector u,v;
        u.s.x = (a.x + b.x) / 2;
        u.s.y = (a.y + b.y) / 2;
        u.e = c;
        v.s.x = (a.x + c.x) / 2;
        v.s.y = (a.y + c.y) / 2;
        v.e = b;
        return u.crossLPt(v);
    }

    //计算三角形费马点
    //返回：到三角形三顶点距离之和最小的点
    point fermentpoint()
    {
        point u, v;
        double step = fabs(a.x) + fabs(a.y) + fabs(b.x) + fabs(b.y) + fabs(c.x) +
        ↪  fabs(c.y);
        int i, j, k;
        u.x = (a.x + b.x + c.x) / 3;
        u.y = (a.y + b.y + c.y) / 3;
        while (step > eps)
        {
            for (k = 0; k < 10; step /= 2, k ++)
            {
                for (i = -1; i <= 1; i ++)
                {
                    for (j =- 1; j <= 1; j ++)
                    {
                        v.x = u.x + step * i;
                        v.y = u.y + step * j;
                        if (u.dis(a) + u.dis(b) + u.dis(c) > v.dis(a) + v.dis(b) +
                        ↪  v.dis(c))
                            u = v;
                    }
                }
            }
        }
        return u;
    }
};
```

8 搜索

8.1 A*

```cpp
// h(x) <= h*(x)
#include <bits/stdc++.h>
using namespace std;
using Status = int;
using T = int;
const T INF = 0x3f3f3f3f;
unordered_map<Status, T> h;
struct node {
    Status x;
    // value
    T f, g;
    bool operator< (const node &a) const { return f > a.f; }
    void update() {  // update f
        if (h.find(x) == h.end()) h[x] = ;
        f = g + h[x];
    }
};
// ???
// Status encode() { return hash ??? }
// void decode(Status) { get ??? by Status }
T Astar(Status S, Status N) {  // n status, from S to N
    unordered_map<Status, T> a;  // status's f
    // start
    node now;  // current node
    now.x = S;
    now.g = 0;
    now.update();
    a[now.x] = now.f;
    priority_queue<node> q;
    q.push(now);
    while (!q.empty()) {
        do {
            now = q.top();
            q.pop();
        } while (now.f > a[now.x] && !q.empty());
        if (now.x == N) break;
        if (now.f == a[now.x]) {
            // decode(now.x);
            for () {
                // change ???
                node tmp;  // new node
                tmp.x = ;  // encode();
                tmp.g = now.g + ;
                tmp.update();
                if (a.find(tmp.x) == a.end() || a[tmp.x] > tmp.f) {
                    a[tmp.x] = tmp.f;
                    q.push(tmp);
                }
                // restore ???
            }
        }
    }
    if (a.find(N) != a.end()) return a[N];
    else return INF;
}
```

8.2 模拟退火

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

const double PI = acos(-1.0);
double eps = 1e-8;

int main() {
    mt19937 e(time(NULL));
    uniform_real_distribution<double> u(0.0, 1.0);
    double t = ;  // step length
    // some init val
    double E = check();
    while (t > eps) {
        double nE = 0.0;
        for () {  // go around
            // use random make some change
            double ans = check();  // new
            if (nE < ans) {  // best
                nE = ans;
            }
        }
        double dE = nE - E;  // max, min: E - nE
        if (dE >= eps || exp(dE / t) > u(e)) {
            E = nE;
        }
        t *= 0.99;  // rate
    }
}
```

9 其他

9.1 读入挂

```cpp
#include <bits/stdc++.h>
using namespace std;
template <typename T>
bool in(T& x){
    x = 0; T f = 1; char ch = cin.get();
    while (!isdigit(ch)) {
        if (ch == -1) return false;
        if (ch == '-') f = -1;
        ch = cin.get();
    }
    while (isdigit(ch)) {x = x * 10 + ch - '0'; ch = cin.get();}
    x *= f; return true;
}
template <typename T, typename... S>
bool in(T& t, S&... s) {if(!in(t)) return false; return in(s...);}

namespace fastIO {
    const int BUF_SIZE = 100000;
    char buf[BUF_SIZE], *s, *t;
    inline char read() {
        if (s == t) {
            s = buf;
            t = buf + cin.rdbuf()->sgetn(buf, BUF_SIZE);
            if(s == t) return -1;
        }
        return *s++;
    }
    template <typename T>
        bool in(T& x){
            x = 0; T f = 1; char ch = read();
            while (!isdigit(ch)) {
                if (ch == -1) return false;
                if (ch == '-') f = -1;
                ch = read();
            }
            while (isdigit(ch)) {x = x * 10 + ch - '0'; ch = read();}
            x *= f; return true;
        }
    template <typename T, typename... S>
        bool in(T& t, S&... s) {if(!in(t)) return false; return in(s...);}

    const int OUT_LEN = 10000000;
    char obuf[OUT_LEN], *oh = obuf;
    inline void print(char c) {
        if (oh == obuf + OUT_LEN) cout.rdbuf()->sputn(obuf, OUT_LEN), oh = obuf;
        *oh++ = c;
    }
    inline void print(string& s) {
        for (auto i : s) print(i);
    }
    inline void print(const char* s) {
        while (*s) print(*s++);
    }
    template <typename T>
        void print(T x) {
```

```
            static int buf[30], cnt;
            if (x < 0) x = -x, print('-');
            cnt = 0;
            do {
                buf[++cnt] = x % 10 | 48;
                x /= 10;
            } while (x);
            while (cnt) print((char)buf[cnt--]);
        }
    template <typename T, typename... S>
        void print(T t, S... s) {print(t); print(s...);}

    inline void flush() { cout.rdbuf()->sputn(obuf, oh - obuf); }
};
using namespace fastIO;

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);

    fastIO::flush();
}
```

9.2 离散化

```
#include <bits/stdc++.h>
using namespace std;
using T = int;
vector<T> LSH(vector<T>& a) {
    auto lsh = a;
    sort(lsh.begin(), lsh.end());
    lsh.erase(unique(lsh.begin(), lsh.end()), lsh.end());
    for (auto& i : a) {
        i = lower_bound(lsh.begin(), lsh.end(), i) - lsh.begin();
    }
    return lsh;
}
```

9.3 编译优化

```
#pragma GCC diagnostic error "-std=c++11"
#pragma GCC optimize ("Ofast")
#include <bits/stdc++.h>
using namespace std;
__attribute__((optimize("-Ofast"))) int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
}
```

9.4 扩展

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

#include <ext/numeric>
#include <ext/rope>
using namespace __gnu_cxx;

#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/hash_policy.hpp>
```

```cpp
#include <ext/pb_ds/tree_policy.hpp>
#include <ext/pb_ds/priority_queue.hpp>
using namespace __gnu_pbds;
template<typename T>
using pbq = priority_queue<T>;
template<typename T, typename S = null_type>
using rbt = tree<T, S, std::less<T>, rb_tree_tag, tree_order_statistics_node_update>;

#define Branch detail::branch_policy<Node_CItr, Node_Itr, _Alloc>
template <class Node_CItr, class Node_Itr, class Cmp_Fn, class _Alloc>
class multi_node_update : private Branch {
 private:
    typedef typename Node_Itr::value_type Itr;
    typedef typename Branch::key_type key_type;
    typedef typename Branch::value_type::second_type val_type;
    virtual Node_CItr node_begin() const = 0;
    virtual Node_CItr node_end() const = 0;
    virtual Node_Itr node_begin() = 0;
    virtual Node_Itr node_end() = 0;

 protected:
    inline void operator() (Node_Itr it , Node_CItr end_it) const {
        Node_Itr l = it.get_l_child(), r = it.get_r_child();
        size_t left = 0, right = 0;
        if (l != end_it) left = l.get_metadata();
        if (r != end_it) right = r.get_metadata();
        const_cast<metadata_type &>(it.get_metadata()) = left + right + (*it)->second;
    }
    virtual ~multi_node_update() {};

 public:
    typedef size_t metadata_type;
    inline Itr find_by_order(size_t order) {
        Node_Itr it = node_begin();
        Node_Itr end_it = node_end();
        while (it != end_it) {
            Node_Itr l = it.get_l_child();
            size_t o = 0;
            if (l != end_it) o = l.get_metadata();
            if (order >= o && order < o + (*it)->second) return *it;
            else if (order < o) it = l;
            else {
                order -= o + (*it)->second;
                it = it.get_r_child();
            }
        }
        return Branch::end_iterator();
    }
    inline size_t order_of_key(const key_type& r_key) const {
        Node_CItr it = node_begin();
        Node_CItr end_it = node_end();
        size_t ord = 0;
        Cmp_Fn op;
        while (it != end_it) {
            Node_CItr l = it.get_l_child();
            if (op(r_key, (*it)->first)) it = l;
            else if (op((*it)->first, r_key)) {
                ord += (l == end_it)? (*it)->second : (*it)->second + l.get_metadata();
```

```
                it = it.get_r_child();
            } else {
                ord += (l == end_it)? 0 : l.get_metadata();
                it = end_it;
            }
        }
        return ord;
    }
};
template <typename T>
using multirbt = tree<T, size_t, std::less<T>, rb_tree_tag, multi_node_update>;

using namespace std;

#include <tr2/dynamic_bitset>
using namespace tr2;
using db = dynamic_bitset<>;
```