Algorithm Template Library

ytz123

May 31, 2019

Contents

1 图论

1.1 线段树维护树直径

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

using namespace std;

const int N = 2e5 + 5;

int T, n, m;

int len, head[N], ST[20][N];

struct edge{int u, v, w;}ee[N];

int cnt, fa[N], log_2[N], st[N], en[N], dfn[N], dis[N],
↪  dep[N], pos[N];

struct edges{int to, next, cost;}e[N];

void add(int u, int v, int w) {
    e[++ len] = (edges){v, head[u], w}, head[u] = len;
    e[++ len] = (edges){u, head[v], w}, head[v] = len;
}

void dfs1(int u) {
    st[u] = ++ cnt, dfn[cnt] = u;
    for (int v, i = head[u]; i; i = e[i].next) {
        v = e[i].to;
        if (v == fa[u]) continue;
        fa[v] = u, dep[v] = dep[u] + 1;
        dis[v] = dis[u] + e[i].cost, dfs1(v);
    }
    en[u] = cnt;
}

void dfs2(int u) {
    dfn[++ cnt] = u, pos[u] = cnt;
    for (int v, i = head[u]; i; i = e[i].next) {
        v = e[i].to;
        if (v == fa[u]) continue;
        dfs2(v), dfn[++ cnt] = u;
    }
```

```
}

int mmin(int x, int y) {
    if (dep[x] < dep[y]) return x;
    return y;
}

int lca(int u, int v) {
    static int w;
    if (pos[u] > pos[v]) swap(u, v);
    w = log_2[pos[v] - pos[u] + 1];
    return mmin(ST[w][pos[u]], ST[w][pos[v] - (1 << w) + 1]);
}

int dist(int u, int v) {
    int Lca = lca(u, v);
    return dis[u] + dis[v] - dis[Lca] * 2;
}

void build() {
    for (int i = 1; i <= cnt; i ++)
        ST[0][i] = dfn[i];
    for (int i = 1; i < 20; i ++)
        for (int j = 1; j <= cnt; j ++)
            if (j + (1 << (i - 1)) > cnt) ST[i][j] = ST[i -
            ↪  1][j];
            else ST[i][j] = mmin(ST[i - 1][j], ST[i - 1][j +
            ↪  (1 << (i - 1))]);
}

int M;

struct node {
    int l, r, dis;
}tr[N << 1];

void update(int o, int o1, int o2) {
    static int d; static node tmp;
    if (tr[o1].dis == -1) {tr[o] = tr[o2]; return;}
    if (tr[o2].dis == -1) {tr[o] = tr[o1]; return;}
    if (tr[o1].dis > tr[o2].dis) tmp = tr[o1];
    else tmp = tr[o2];
    d = dist(tr[o1].l, tr[o2].l);
```

```
    if (d > tmp.dis) tmp.l = tr[o1].l, tmp.r = tr[o2].l,
    ↪  tmp.dis = d;
    d = dist(tr[o1].l, tr[o2].r);
    if (d > tmp.dis) tmp.l = tr[o1].l, tmp.r = tr[o2].r,
    ↪  tmp.dis = d;
    d = dist(tr[o1].r, tr[o2].l);
    if (d > tmp.dis) tmp.l = tr[o1].r, tmp.r = tr[o2].l,
    ↪  tmp.dis = d;
    d = dist(tr[o1].r, tr[o2].r);
    if (d > tmp.dis) tmp.l = tr[o1].r, tmp.r = tr[o2].r,
    ↪  tmp.dis = d;
    tr[o] = tmp;
}

void ask(int s, int t) {
    if (s > t) return;
    for (s += M - 1, t += M + 1; s ^ t ^ 1; s >>= 1, t >>= 1)
    ↪  {
        if (~s&1) update(0, 0, s ^ 1);
        if ( t&1) update(0, 0, t ^ 1);
    }
}

int main() {
    ios::sync_with_stdio(false);
    int u, v, w, ans; log_2[1] = 0;
    for (int i = 2; i <= 200000; i ++)
        if (i == 1 << (log_2[i - 1] + 1))
            log_2[i] = log_2[i - 1] + 1;
        else log_2[i] = log_2[i - 1];
    for (cin >> T; T --; ) {
        cin >> n >> m, cnt = len = 0;
        for (int i = 1; i <= n; i ++)
            head[i] = 0;
        for (int i = 1; i < n; i ++) {
            cin >> ee[i].u >> ee[i].v >> ee[i].w;
            add(ee[i].u, ee[i].v, ee[i].w);
        }
        dfs1(1);
        for (M = 1; M < n + 2; M <<= 1);
        for (int i = 1; i <= n; i ++)
            tr[i + M].l = tr[i + M].r = dfn[i], tr[i + M].dis
            ↪  = 0;
        for (int i = n + M + 1; i <= (M << 1) + 1; i ++)
```

```
            tr[i].dis = -1;
        cnt = 0, dfs2(1), build();
        for (int i = M; i; i --)
            update(i, i << 1, i << 1 | 1);
        for (int i = 1; i < n; i ++)
            if (dep[ee[i].u] > dep[ee[i].v])
                swap(ee[i].u, ee[i].v);
        for (int u, v, i = 1; i <= m; i ++) {
            cin >> u >> v, ans = 0;
            u = ee[u].v, v = ee[v].v, w = lca(u, v);
            if (w == u || w == v) {
                if (w != u) swap(u, v);
                tr[0].dis = -1, ask(1, st[u] - 1), ask(en[u] +
                ↪  1, n), ans = max(ans, tr[0].dis);
                tr[0].dis = -1, ask(st[u], st[v] - 1),
                ↪  ask(en[v] + 1, en[u]), ans = max(ans,
                ↪  tr[0].dis);
                tr[0].dis = -1, ask(st[v], en[v]), ans =
                ↪  max(ans, tr[0].dis);
            }
            else {
                if (st[u] > st[v]) swap(u, v);
                tr[0].dis = -1, ask(1, st[u] - 1), ask(en[u] +
                ↪  1, st[v] - 1), ask(en[v] + 1, n), ans =
                ↪  max(ans, tr[0].dis);
                tr[0].dis = -1, ask(st[u], en[u]), ans =
                ↪  max(ans, tr[0].dis);
                tr[0].dis = -1, ask(st[v], en[v]), ans =
                ↪  max(ans, tr[0].dis);
            }
            printf("%d\n", ans);
        }
    }
    return 0;
}
```

1.2 有向图判断两个点能否到达

```
/* 时间 O(n*m/32) 空间 O(n*n/blk) blk 随便取 */
#include <bits/stdc++.h>

using namespace std;

const int N = 1e5 + 2;
```

```
const int BLK = 5000;

int n, k, p;

int d[N];

vector <int> e[N], f[N], ck[N];

int top, sta[N], in[N];

int cnt, dfn[N], low[N], vis[N];

int sum, bel[N];

bitset <BLK> a[N];

queue <int> q;

int topo[N];

void tarjan(int u) {
        vis[u] = in[u] = 1;
        sta[++ top] = u, dfn[u] = low[u] = ++ cnt;
        for (int v : e[u])
                if (!vis[v]) {
                        tarjan(v);
                        low[u] = min(low[v], low[u]);
                }
                else if (in[v])
                        low[u] = min(low[v], low[u]);
        if (low[u] == dfn[u]) {
                sum ++; int i;
                while (1) {
                        i = sta[top --];
                        in[i] = 0, bel[i] = sum;
                        if (i == u) break;
                }
        }
}

int main() {
        freopen("input.txt", "r", stdin);
        freopen("output.txt", "w", stdout);
```

```
ios::sync_with_stdio(false);
cin >> n >> k;
for (int u, v, i = 1; i <= k; i ++) {
        cin >> u >> v;
        e[u].push_back(v);
}
cin >> p;
for (int u, v, i = 1; i <= p; i ++) {
        cin >> u >> v;
        f[u].push_back(v);
}

for (int i = 1; i <= n; i ++)
        if (!vis[i])
                tarjan(i);
for (int i = 1; i <= n; i ++) {
        for (int j : f[i]) {
                if (bel[i] == bel[j]) return
                ↪  puts("NO"), 0;
                ck[bel[i]].push_back(bel[j]);
        }
        f[i].clear();
}
for (int i = 1; i <= n; i ++)
        for (int j : e[i]) {
                if (bel[i] == bel[j]) continue;
                f[bel[i]].push_back(bel[j]);
                d[bel[j]] ++;
        }


cnt = 0;
for (int i = 1; i <= sum; i ++)
        if (!d[i])
                q.push(i);
while (!q.empty()) {
        int now = q.front(); q.pop();
        topo[++ cnt] = now;
        for (int j : f[now]) {
                d[j] --;
                if (d[j] == 0) q.push(j);
        }
}
```

```
        for (int i = 1, t = (sum + BLK - 1) / BLK; i <= t; i
        ↪  ++) {
                for (int j = sum; j; j --) {
                        int u = topo[j];
                        a[u].reset();
                        if (BLK * (i - 1) < u && u <= BLK * i)
                                a[u][u - BLK * (i - 1) - 1] =
                                ↪  1;
                        for (int v : f[u])
                                a[u] |= a[v];
                }
                for (int j = 1; j <= sum; j ++)
                        for (int v : ck[j])
                                if (BLK * (i - 1) < v && v <=
                                ↪  BLK * i && a[j][v -  BLK *
                                ↪  (i - 1) - 1] == 1) {
                                        puts("NO");
                                        return 0;
                                }
        }

        printf("YES\n%d\n", k);
        for (int i = 1; i <= n; i ++)
                for (int j : e[i])
                        printf("%d %d\n", i, j);
        return 0;
}
```

1.3 spfa 费用流

```
const int N = 600, M = 800000, inf = 0x3f3f3f3f;

int s, t, ans, len, maxflow;

int T, n, m, K, W;

int head[N], incf[N], path[N], pre[N], vis[N], d[N];

struct edge{
        int to, next, cap, cost;
}e[M];

struct video {z
        int s, t, w, op;
```

```
}a[N];

void add(int u, int v, int w, int c) {
        e[++ len] = (edge){v, head[u], w, c}, head[u] = len;
        e[++ len] = (edge){u, head[v], 0, -c}, head[v] = len;
}

bool spfa() {
        deque <int> q;
        q.push_back(s), incf[s] = inf;
        for (int i = 1; i <= t; i ++) d[i] = inf;
        d[s] = 0;
        while (!q.empty()) {
                int x = q.front();
                q.pop_front(), vis[x] = 0;
                for (int i = head[x]; i; i = e[i].next) {
                        if (e[i].cap && d[e[i].to] > d[x] +
                        ↪  e[i].cost) {
                                d[e[i].to] = d[x] + e[i].cost;
                                pre[e[i].to] = x,
                                ↪  path[e[i].to] = i;
                                incf[e[i].to] = min(incf[x],
                                ↪  e[i].cap);
                                if (!vis[e[i].to]) {
                                        vis[e[i].to] = 1;
                                        if (q.empty() ||
                                        ↪  d[e[i].to] <
                                        ↪  d[q.front()])
                                        ↪  q.push_front(e[i].to);
                                        else
                                        ↪  q.push_back(e[i].to);
                                }
                        }
                }
        }
        maxflow += incf[t];
        if (d[t] == inf) return 0;
        for (int i = t; i != s; i = pre[i]) {
                e[path[i]].cap -= incf[t];
                e[path[i] ^ 1].cap += incf[t];
        }
        return ans += incf[t] * d[t], 1;
}
```

```
int main() {
        /*build graph*/
        while(spfa());
}
```

1.4 dinic 最大流

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

using namespace std;

const int N = 20000;

const int M = 500000;

const int inf = 0x3f3f3f3f;

int n, m;

int s, t, len = 1;

int to[M], cap[M], nex[M];

int g[N], p[N], q[N], d[N];

void add(int x, int y, int v) {
        to[++ len] = y, cap[len] = v, nex[len] = g[x], g[x] =
        ↪  len;
        to[++ len] = x, cap[len] = 0, nex[len] = g[y], g[y] =
        ↪  len;
}

bool bfs() {
        int l = 1, r = 1, x, i;
        memset (d, 0, sizeof d);
        d[s] = 1, q[1] = s;
        while (l <= r) {
                x = q[l ++];
                for (i = g[x]; i; i = nex[i])
                        if (cap[i] && !d[to[i]])
                                d[to[i]] = d[x] + 1, q[++ r] =
                                ↪  to[i];
        }
        return d[t];
```

```
}

int dfs(int x, int y) {
        if (x == t || y == 0) return y;
        int flow = 0;
        for (int &i = p[x]; i; i = nex[i]) {
                if (!cap[i] || d[to[i]] != d[x] + 1) continue;
                int f = dfs(to[i], min(y, cap[i]));
                flow += f, y -= f;
                cap[i] -= f, cap[i ^ 1] += f;
                if (!y) break;
        }
        return flow;
}

int  dinic() {
        int maxflow = 0;
        while (bfs()) {
                memcpy(p, g, sizeof g);
                maxflow += dfs(s, inf);
        }
        return maxflow;
}

int main() {
        ios::sync_with_stdio(false);

        return 0;
}
```

1.5 DAG 删去无用边

```
/*
无用边定义：对于边 (u,v) 如果存在从 u 到 v 不经过该边的另一条路
↪  径，则称该边无用
时间复杂度:O(n^3)
*/
bool f[N][N];//i 是否可达 j

vector <int> e[N];

int main() {
        rep(i, 1, n)
                for (int j : e[i]) {
```

```
                        for (k, 1, n)
                                if (i != k && j != k &&
                                ↪  f[i][k] && f[k][j])
                                        no_use_edge;
                }
}
```

1.6 树分治

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

using namespace std;

typedef long long ll;

const int N = 1e5 + 5;

vector <int> e[N];

int n, a[N];

int root, _left, vis[N];

int siz[N], maxv[N];

void find_root(int u, int fr) {
        siz[u] = 1, maxv[u] = 0;
        for (int v : e[u]) {
                if (v == fr || vis[v]) continue;
                find_root(v, u);
                siz[u] += siz[v];
                maxv[u] = max(maxv[u], siz[v]);
        }
        maxv[u] = max(maxv[u], _left - siz[u]);
        if (!root || maxv[u] < maxv[root])
                root = u;
}

void dfs(int u, int fr) {
        siz[u] = 1;
        for (int v : e[u]) {
                if (v == fr || vis[v]) continue;
                find_root(v, u);
                siz[u] += siz[v];
```

```
        }
}

void solve(int u, int w) {
        dfs(u, u);//update siz[]
        a[u] = w, vis[u] = 1;
        for (int v : e[u]) {
                if (vis[v]) continue;
                _left = siz[v];
                root = 0;
                find_root(v, v);
                solve(root, w + 1);
        }
}

int main() {
    ios::sync_with_stdio(false);
    cin >> n;
    for (int u, v, i = 1; i < n; i ++) {
            cin >> u >> v;
            e[u].push_back(v);
            e[v].push_back(u);
    }
    _left = n, root = 0, find_root(1, 1);
    solve(root, 0);
    for (int i = 1; i <= n; i ++)
            printf("%c ", 'A' + a[i]);
    return 0;
}
```

2 数据结构

2.1 treap

```
/* 容易实现的预开内存池 treap,
↪ 每次 head 清空即可
如果初始要插入 n 个 1, 可改为类
↪ 似 splay 的 O(n)build 写法
*/
#include <bits/stdc++.h>

using namespace std;

const int poolSize = 5e5 + 10;

struct node {
        node *c[2];
        int v, r, siz;
        void update();
        void init(int x);
};

node *null = new node(), *root
↪  = null;

void node::update() {
        siz = c[0] -> siz +
        ↪  c[1] -> siz + 1;
}

void node::init(int x) {
        v = x, r = rand(), siz
        ↪  = 1;
        c[0] = c[1] = null;
}

node nodesPool[poolSize];

int head;//每次 head=0 清空

node *newnode(int x) {
        node *res =
        ↪  &nodesPool[head
        ↪  ++];
        res -> init(x);
        return res;
}

void rot(node *&o, int d) {
        node *tmp = o ->
        ↪  c[!d];
        o -> c[!d] = tmp ->
        ↪  c[d], tmp -> c[d]
        ↪  = o;
        o -> update(), tmp ->
        ↪  update(), o = tmp;
}

void insert(node *&o, int x) {
        if (o == null) {
                o =
                ↪  newnode(x);
                return;
        }
        int d = x > o -> v ? 0
        ↪  : 1;
        insert(o -> c[d], x);
        if (o -> c[d] -> r < o
        ↪  -> r) rot(o, !d);
        o -> update();
}

void del(node *&o, int x) {
        if (x == o -> v) {
                if (o -> c[0]
                ↪  == null)
                ↪  {o = o ->
                ↪  c[1];
                ↪  return;}
                if (o -> c[1]
                ↪  == null)
                ↪  {o = o ->
                ↪  c[0];
                ↪  return;}
```

```
                int d = o ->
                ↪  c[0] -> r
                ↪  < o ->
                ↪  c[1] -> r
                ↪  ? 1 : 0;
                rot(o, d),
                ↪  del(o ->
                ↪  c[d], x);
        }
        else del(o -> c[x <= o
        ↪  -> v], x);
        o -> update();
}

void build(node *&o, int l,
↪  int r) {
        o = newnode(1);
        if (l == r) return;
        int mid = l + r >> 1;
        if (l < mid) build(o
        ↪  -> c[0], l, mid -
        ↪  1);
        if (o -> c[0] != null
        ↪  && o -> c[0] -> r
        ↪  < o -> r) swap(o
        ↪  -> c[0] -> r, o ->
        ↪  r);
        if (mid < r) build(o
        ↪  -> c[1], mid + 1,
        ↪  r);
        if (o -> c[1] != null
        ↪  && o -> c[1] -> r
        ↪  < o -> r) swap(o
        ↪  -> c[1] -> r, o ->
        ↪  r);
        o -> update();
}
```

2.2 KDTree

2.2.1 3 维 KDtree

```
/*
 *O(n*n^(1-1/k)),k 为维度
 */
#include <bits/stdc++.h>

using namespace std;

const int N = 1e5 + 5;

const int Mod = 1e9 + 7;

int nowD, ans, x[3], y[3];

int n, m, a[N], b[N], c[N],
↪  d[N];

struct node {
    int Max[3], Min[3], d[3];
    int val, maxv;
    node *c[2];

    node() {
        c[0] = c[1] = NULL;
        val = maxv = 0;
    }

    void pushup();

    bool operator < (const
    ↪  node &a) const {
        return d[nowD] <
        ↪  a.d[nowD];
    }
}Null, nodes[N];

node *root = &Null;

inline void node::pushup() {
    if (c[0] != &Null) {
        if (c[0] -> Max[1] >
        ↪  Max[1]) Max[1] =
        ↪  c[0] -> Max[1];
        if (c[0] -> Max[2] >
        ↪  Max[2]) Max[2] =
        ↪  c[0] -> Max[2];
```

```
        if (c[0] -> Min[0] <
        ↪  Min[0]) Min[0] =
        ↪  c[0] -> Min[0];
        if (c[0] -> Min[2] <
        ↪  Min[2]) Min[2] =
        ↪  c[0] -> Min[2];
        if (c[0] -> maxv >
        ↪  maxv) maxv = c[0]
        ↪  -> maxv;
    }
    if (c[1] != &Null) {
        if (c[1] -> Max[1] >
        ↪  Max[1]) Max[1] =
        ↪  c[1] -> Max[1];
        if (c[1] -> Max[2] >
        ↪  Max[2]) Max[2] =
        ↪  c[1] -> Max[2];
        if (c[1] -> Min[0] <
        ↪  Min[0]) Min[0] =
        ↪  c[1] -> Min[0];
        if (c[1] -> Min[2] <
        ↪  Min[2]) Min[2] =
        ↪  c[1] -> Min[2];
        if (c[1] -> maxv >
        ↪  maxv) maxv = c[1]
        ↪  -> maxv;
    }
}

inline node *build(int l, int
↪  r) {
    int mid = l + r >> 1; nowD
    ↪  = rand() % 3;
    nth_element(nodes + l,
    ↪  nodes + mid, nodes + r
    ↪  + 1);
    node *res = &nodes[mid];
    if (l != mid) res -> c[0]
    ↪  = build(l, mid - 1);
    else res -> c[0] = &Null;
    if (r != mid) res -> c[1]
    ↪  = build(mid + 1, r);
    else res -> c[1] = &Null;
    res -> pushup();
    return res;
}

inline int calc(node *o) {
    if (y[0] < o -> Min[0] ||
    ↪  x[1] > o -> Max[1] ||
    ↪  x[2] > o -> Max[2] ||
    ↪  y[2] < o -> Min[2])
    ↪  return -1;
    return o -> maxv;
}

inline void query(node *o) {
    if (o -> val > ans && y[0]
    ↪  >= o -> d[0] && x[1]
    ↪  <= o -> d[1] && x[2]
    ↪  <= o -> d[2] && y[2]
    ↪  >= o -> d[2]) ans = o
    ↪  -> val;
    int dl, dr;
    if (o -> c[0] != &Null) dl
    ↪  = calc(o -> c[0]);
    else dl = -1;
    if (o -> c[1] != &Null) dr
    ↪  = calc(o -> c[1]);
    else dr = -1;
    if (dl > dr) {
        if (dl > ans) query(o
        ↪  -> c[0]);
        if (dr > ans) query(o
        ↪  -> c[1]);
    } else {
        if (dr > ans) query(o
        ↪  -> c[1]);
        if (dl > ans) query(o
        ↪  -> c[0]);
    }
}

int main() {
    ↪  ios::sync_with_stdio(false);
    cin >> n >> m;
```

```
    for (int i = 1; i <= n; i
    ↪  ++) {
        cin >> a[i];
        b[i] = d[a[i]];
        d[a[i]] = i;
    }
    for (int i = 1; i <= n; i
    ↪  ++) d[i] = n + 1;
    for (int i = n; i; i --) {
        c[i] = d[a[i]];
        d[a[i]] = i;
    }
    for (int i = 1; i <= n; i
    ↪  ++) {
        nodes[i].Min[0] =
        ↪  nodes[i].d[0] =
        ↪  b[i];
        nodes[i].Max[1] =
        ↪  nodes[i].d[1] =
        ↪  c[i];
        nodes[i].Max[2] =
        ↪  nodes[i].Min[2] =
        ↪  nodes[i].d[2] = i;
        nodes[i].val =
        ↪  nodes[i].maxv =
        ↪  a[i];
    }
    root = build(1, n);
    for (int l, r; m --; ) {
        cin >> l >> r;
        l = (l + ans) % n + 1;
        r = (r + ans) % n + 1;
        if (l > r) swap(l, r);
        y[0] = l - 1;
        x[1] = r + 1;
        x[2] = l, y[2] = r;
        ans = 0, query(root);
        cout << ans << endl;
    }
    cout << endl;
    return 0;
}
```

2.2.2 KDtree 二维空间区间覆盖单点查询

```
/* 类似线段树 */
#include <bits/stdc++.h>

using namespace std;

const int N = 1e5 + 5;

const int Mod = 1e9 + 7;

int nowD, x[2], y[2], z;

struct node {
    int Max[2], Min[2], d[2];
    int val, lazy;
    node *c[2];

    node() {
        c[0] = c[1] = NULL;
    }

    void pushup();

    void pushdown();

    bool operator < (const
    ↪  node &a) const {
        return d[nowD] <
        ↪  a.d[nowD];
    }
}Null, nodes[N];

node *root = &Null;

inline void node::pushup() {
    if (c[0] != &Null) {
        if (c[0] -> Max[0] >
        ↪  Max[0]) Max[0] =
        ↪  c[0] -> Max[0];
        if (c[0] -> Max[1] >
        ↪  Max[1]) Max[1] =
        ↪  c[0] -> Max[1];
```

```
        if (c[0] -> Min[0] <
        ↪  Min[0]) Min[0] =
        ↪  c[0] -> Min[0];
        if (c[0] -> Min[1] <
        ↪  Min[1]) Min[1] =
        ↪  c[0] -> Min[1];
    }
    if (c[1] != &Null) {
        if (c[1] -> Max[0] >
        ↪  Max[0]) Max[0] =
        ↪  c[1] -> Max[0];
        if (c[1] -> Max[1] >
        ↪  Max[1]) Max[1] =
        ↪  c[1] -> Max[1];
        if (c[1] -> Min[0] <
        ↪  Min[0]) Min[0] =
        ↪  c[1] -> Min[0];
        if (c[1] -> Min[1] <
        ↪  Min[1]) Min[1] =
        ↪  c[1] -> Min[1];
    }
}

inline void node::pushdown() {
    if (c[0] != &Null) c[0] ->
    ↪  val = c[0] -> lazy =
    ↪  lazy;
    if (c[1] != &Null) c[1] ->
    ↪  val = c[1] -> lazy =
    ↪  lazy;
    lazy = -1;
}

inline node *build(int l, int
↪  r, int D) {
    int mid = l + r >> 1; nowD
    ↪  = D;
    nth_element(nodes + l,
    ↪  nodes + mid, nodes + r
    ↪  + 1);
    node *res = &nodes[mid];
    if (l != mid) res -> c[0]
    ↪  = build(l, mid - 1,
    ↪  !D);
    else res -> c[0] = &Null;
    if (r != mid) res -> c[1]
    ↪  = build(mid + 1, r,
    ↪  !D);
    else res -> c[1] = &Null;
    res -> pushup();
    return res;
}

inline int query(node *o) {
    if (o == &Null) return -1;
    if (o -> lazy != -1) o ->
    ↪  pushdown();
    if (x[0] > o -> Max[0] ||
    ↪  y[0] > o -> Max[1] ||
    ↪  x[0] < o -> Min[0] ||
    ↪  y[0] < o -> Min[1])
    ↪  return -1;
    if (x[0] == o -> d[0])
    ↪  return o -> val;
    return max(query(o ->
    ↪  c[0]), query(o ->
    ↪  c[1]));
}

inline void modify(node *o) {
    if (o == &Null) return;
    if (o -> lazy != -1) o ->
    ↪  pushdown();
    if (x[0] > o -> Max[0] ||
    ↪  y[0] > o -> Max[1] ||
    ↪  x[1] < o -> Min[0] ||
    ↪  y[1] < o -> Min[1])
    ↪  return;
    if (x[0] <= o -> Min[0] &&
    ↪  y[0] <= o -> Min[1] &&
    ↪  x[1] >= o -> Max[0] &&
    ↪  y[1] >= o -> Max[1]) {
        o -> val = o -> lazy =
        ↪  z;
        return;
    }
```

```
    if (x[0] <= o -> d[0] &&
    ↪  y[0] <= o -> d[1] &&
    ↪  x[1] >= o -> d[0] &&
    ↪  y[1] >= o -> d[1]) o
    ↪  -> val = z;
    modify(o -> c[0]),
    ↪  modify(o -> c[1]);
}

int n, m, k, a[N], c[N], d[N];

int cnt, st[N], en[N], dfn[N],
↪  dep[N];

vector <int> e[N];

void dfs(int u) {
    st[u] = ++ cnt, dfn[cnt] =
    ↪  u;
    for (int v : e[u])
        dep[v] = dep[u] + 1,
        ↪  dfs(v);
    en[u] = cnt;
}

int main() {

    ↪  ios::sync_with_stdio(false);
    int T, ans;
    for (cin >> T; T --; ) {
        cin >> n >> m >> k,
        ↪  ans = cnt = 0;
        for (int i = 1; i <=
        ↪  n; i ++)
            e[i].clear();
        for (int u, i = 2; i
        ↪  <= n; i ++) {
            cin >> u;
            e[u].push_back(i);
        }
        dfs(1);
        for (int i = 1; i <=
        ↪  n; i ++) {
            nodes[i].Min[0] =
            ↪  nodes[i].Max[0]
            ↪  =
            ↪  nodes[i].d[0]
            ↪  = i;
            nodes[i].Min[1] =
            ↪  nodes[i].Max[1]
            ↪  =
            ↪  nodes[i].d[1]
            ↪  = dep[dfn[i]];
            nodes[i].val = 1,
            ↪  nodes[i].lazy
            ↪  = -1;
        }
        root = build(1, n, 0);
        for (int u, v, w, i =
        ↪  1; i <= k; i ++) {
            cin >> u >> v >>
            ↪  w;
            if (w == 0) {
                x[0] = st[u],
                ↪  y[0] =
                ↪  dep[u];
                ans = (ans +
                ↪  1ll * i *
                ↪  query(root)
                ↪  % Mod) %
                ↪  Mod;
            } else {
                x[0] = st[u],
                ↪  x[1] =
                ↪  en[u];
                y[0] = dep[u],
                ↪  y[1] =
                ↪  dep[u] +
                ↪  v;
                z = w,
                ↪  modify(root);
            }
        }
        cout << ans << endl;
    }
    return 0;
}
```

2.2.3 KDtree 二维空间单点修改 区间查询

```
/*
 * 调整重构系数可以影响常数
 * 询问多就让系数接近
↪  0.70-0.75, 询问少就让系数在
↪  0.8-0.90
 */
#include <bits/stdc++.h>

using namespace std;

const int inf = 1e9;

int n, m, tot, nowD;

struct node {
    int Max[2], Min[2], d[2];
    int sum, siz, val;
    node *c[2];

    node() {
        Max[0] = Max[1] =
        ↪  -inf;
        Min[0] = Min[1] = inf;
        sum = val = siz = 0;
        c[0] = c[1] = NULL;
        d[0] = d[1] = 0;
    }

    void update();
}Null, nodes[200010],
↪  *temp[200010];

node *root = &Null;

inline void node::update() {
    siz = c[0] -> siz + c[1]
    ↪  -> siz + 1;
    sum = c[0] -> sum + c[1]
    ↪  -> sum + val;
    if (c[0] != &Null) {
        if (c[0] -> Max[0] >
        ↪  Max[0]) Max[0] =
        ↪  c[0] -> Max[0];
        if (c[0] -> Max[1] >
        ↪  Max[1]) Max[1] =
        ↪  c[0] -> Max[1];
        if (c[0] -> Min[0] <
        ↪  Min[0]) Min[0] =
        ↪  c[0] -> Min[0];
        if (c[0] -> Min[1] <
        ↪  Min[1]) Min[1] =
        ↪  c[0] -> Min[1];
    }
    if (c[1] != &Null) {
        if (c[1] -> Max[0] >
        ↪  Max[0]) Max[0] =
        ↪  c[1] -> Max[0];
        if (c[1] -> Max[1] >
        ↪  Max[1]) Max[1] =
        ↪  c[1] -> Max[1];
        if (c[1] -> Min[0] <
        ↪  Min[0]) Min[0] =
        ↪  c[1] -> Min[0];
        if (c[1] -> Min[1] <
        ↪  Min[1]) Min[1] =
        ↪  c[1] -> Min[1];
    }
}

inline bool cmp(const node *a,
↪  const node *b) {
    return a -> d[nowD] < b ->
    ↪  d[nowD];
}

inline void traverse(node *o)
↪  {
    if (o == &Null) return;
    temp[++ tot] = o;
    traverse(o -> c[0]);
    traverse(o -> c[1]);
}
```

```
inline node *build(int l, int
↪  r, int D) {
    int mid = l + r >> 1; nowD
    ↪  = D;
    nth_element(temp + l, temp
    ↪  + mid, temp + r + 1,
    ↪  cmp);
    node *res = temp[mid];
    res -> Max[0] = res ->
    ↪  Min[0] = res -> d[0];
    res -> Max[1] = res ->
    ↪  Min[1] = res -> d[1];
    if (l != mid) res -> c[0]
    ↪  = build(l, mid - 1,
    ↪  !D);
    else res -> c[0] = &Null;
    if (r != mid) res -> c[1]
    ↪  = build(mid + 1, r,
    ↪  !D);
    else res -> c[1] = &Null;
    res -> update();
    return res;
}

int x, y, a, b, tmpD;

node **tmp;

inline void rebuild(node *&o,
↪  int D) {
    tot = 0;
    traverse(o);
    o = build(1, tot, D);
}

inline void insert(node *&o,
↪  node *p, int D) {
    if (o == &Null) {o = p;
    ↪  return;}
    if (p -> Max[0] > o ->
    ↪  Max[0]) o -> Max[0] =
    ↪  p -> Max[0];
    if (p -> Max[1] > o ->
    ↪  Max[1]) o -> Max[1] =
    ↪  p -> Max[1];
    if (p -> Min[0] < o ->
    ↪  Min[0]) o -> Min[0] =
    ↪  p -> Min[0];
    if (p -> Min[1] < o ->
    ↪  Min[1]) o -> Min[1] =
    ↪  p -> Min[1];
    o -> siz ++, o -> sum += p
    ↪  -> sum;
    insert(o -> c[p -> c[D] >=
    ↪  o -> c[D]], p, !D);
    if (max(o -> c[0] -> siz,
    ↪  o -> c[1] -> siz) >
    ↪  int(o -> siz * 0.75 +
    ↪  0.5)) tmpD = D, tmp =
    ↪  &o;
}

inline int query(node *o) {
    if (o == &Null) return 0;
    if (x > o -> Max[0] || y >
    ↪  o -> Max[1] || a < o
    ↪  -> Min[0] || b < o ->
    ↪  Min[1]) return 0;
    if (x <= o -> Min[0] && y
    ↪  <= o -> Min[1] && a >=
    ↪  o -> Max[0] && b >= o
    ↪  -> Max[1]) return o ->
    ↪  sum;
    return (x <= o -> d[0] &&
    ↪  y <= o -> d[1] && a >=
    ↪  o -> d[0] && b >= o ->
    ↪  d[1] ? o -> val : 0)
        + query(o -> c[1]) +
        ↪  query(o -> c[0]);
}

int main() {

    ↪  ios::sync_with_stdio(false);
    cin >> m;
    node *ttt = &Null;
```

```
for (int t, ans = 0; ; ) {
    cin >> t;
    if (t == 3) break;
    if (t == 1) {
        cin >> x >> y >>
        ↪ a;
        x ^= ans, y ^=
        ↪ ans, n ++;
        nodes[n].sum =
        ↪ nodes[n].val =
        ↪ a ^ ans,
        ↪ nodes[n].siz =
        ↪ 1;
        nodes[n].Max[0] =
        ↪ nodes[n].Min[0]
        ↪ =
        ↪ nodes[n].d[0]
        ↪ = x;
        nodes[n].Max[1] =
        ↪ nodes[n].Min[1]
        ↪ =
        ↪ nodes[n].d[1]
        ↪ = y;
        nodes[n].c[0] =
        ↪ nodes[n].c[1]
        ↪ = &Null;
        tmp = &(ttt),
        ↪ insert(root,
        ↪ &nodes[n], 0);
        if (*tmp != &Null)
        ↪ rebuild(*tmp,
        ↪ tmpD);
    } else {
        cin >> x >> y >> a
        ↪ >> b;
        x ^= ans, y ^=
        ↪ ans, a ^= ans,
        ↪ b ^= ans;
        if (x > a) swap(x,
        ↪ a);
        if (y > b) swap(y,
        ↪ b);
        ans = query(root);
        printf("%d\n",
        ↪ ans);
    }
}
return 0;
}
```

2.2.4 KDtree 找最近点

```
/*
 * 为了维持树的平衡，可以一开始
↪ 把所有点都读进来 build
 * 然后打 flag 标记该点是否被激
↪ 活
 */
#include <bits/stdc++.h>

using namespace std;

const int N = 5e5 + 5;

const int inf = 1 << 30;

int n, m;

int ql, qr, ans, tot, nowD;
//nowD = rand() & 1 ?
struct Node {
    int d[2];

    bool operator < (const
    ↪ Node &a) const {
        if (d[nowD] ==
        ↪ a.d[nowD]) return
        ↪ d[!nowD] <
        ↪ a.d[!nowD];
        return d[nowD] <
        ↪ a.d[nowD];
    }
}pot[N];

struct node {
    int min[2], max[2], d[2];
    node *c[2];
```

```
    node() {
        min[0] = min[1] =
        ↪ max[0] = max[1] =
        ↪ d[0] = d[1] = 0;
        c[0] = c[1] = NULL;
    }

    node(int x, int y);

    void update();


}t[N], Null, *root;

node::node(int x, int y) {
    min[0] = max[0] = d[0] =
    ↪ x;
    min[1] = max[1] = d[1] =
    ↪ y;
    c[0] = c[1] = &Null;
}

inline void node::update() {
    if (c[0] != &Null) {
        if (c[0] -> max[0] >
        ↪ max[0]) max[0] =
        ↪ c[0] -> max[0];
        if (c[0] -> max[1] >
        ↪ max[1]) max[1] =
        ↪ c[0] -> max[1];
        if (c[0] -> min[0] <
        ↪ min[0]) min[0] =
        ↪ c[0] -> min[0];
        if (c[0] -> min[1] <
        ↪ min[1]) min[1] =
        ↪ c[0] -> min[1];
    }
    if (c[1] != &Null) {
        if (c[1] -> max[0] >
        ↪ max[0]) max[0] =
        ↪ c[1] -> max[0];
        if (c[1] -> max[1] >
        ↪ max[1]) max[1] =
        ↪ c[1] -> max[1];
        if (c[1] -> min[0] <
        ↪ min[0]) min[0] =
        ↪ c[1] -> min[0];
        if (c[1] -> min[1] <
        ↪ min[1]) min[1] =
        ↪ c[1] -> min[1];
    }
}

inline void build(node *&o,
↪ int l, int r, int D) {
    int mid = l + r >> 1;
    nowD = D;
    nth_element(pot + l, pot +
    ↪ mid, pot + r + 1);
    o = new
    ↪ node(pot[mid].d[0],
    ↪ pot[mid].d[1]);

    if (l != mid) build(o ->
    ↪ c[0], l, mid - 1, !D);
    if (r != mid) build(o ->
    ↪ c[1], mid + 1, r, !D);
    o -> update();
}

inline void insert(node *o) {
    node *p = root;
    int D = 0;
    while (1) {
        if (o -> max[0] > p ->
        ↪ max[0]) p ->
        ↪ max[0] = o ->
        ↪ max[0];
        if (o -> max[1] > p ->
        ↪ max[1]) p ->
        ↪ max[1] = o ->
        ↪ max[1];
```

```
        if (o -> min[0] < p ->
        ↪  min[0]) p ->
        ↪  min[0] = o ->
        ↪  min[0];
        if (o -> min[1] < p ->
        ↪  min[1]) p ->
        ↪  min[1] = o ->
        ↪  min[1];

        if (o -> d[D] >= p ->
        ↪  d[D]) {
            if (p -> c[1] ==
            ↪  &Null) {
                p -> c[1] = o;
                return;
            } else p = p ->
            ↪  c[1];
        } else {
            if (p -> c[0] ==
            ↪  &Null) {
                p -> c[0] = o;
                return;
            } else p = p ->
            ↪  c[0];
        }
        D ^= 1;
    }
}

inline int dist(node *o) {
    int dis = 0;
    if (ql < o -> min[0]) dis
    ↪  += o -> min[0] - ql;
    if (ql > o -> max[0]) dis
    ↪  += ql - o -> max[0];
    if (qr < o -> min[1]) dis
    ↪  += o -> min[1] - qr;
    if (qr > o -> max[1]) dis
    ↪  += qr - o -> max[1];
    return dis;
}

inline void query(node *o) {
    int dl, dr, d0;
    d0 = abs(o -> d[0] - ql) +
    ↪  abs(o -> d[1] - qr);
    if (d0 < ans) ans = d0;
    if (o -> c[0] != &Null) dl
    ↪  = dist(o -> c[0]);
    else dl = inf;
    if (o -> c[1] != &Null) dr
    ↪  = dist(o -> c[1]);
    else dr = inf;

    if (dl < dr) {
        if (dl < ans) query(o
        ↪  -> c[0]);
        if (dr < ans) query(o
        ↪  -> c[1]);
    } else {
        if (dr < ans) query(o
        ↪  -> c[1]);
        if (dl < ans) query(o
        ↪  -> c[0]);
    }
}

int main() {

    ↪  ios::sync_with_stdio(false);
    cin >> n >> m;
    for (int i = 1; i <= n; i
    ↪  ++)
        cin >> pot[i].d[0] >>
        ↪  pot[i].d[1];
    build(root, 1, n, 0);

    for (int x, y, z; m --; )
    ↪  {
        cin >> x >> y >> z;
        if (x == 1) {
            t[tot].max[0] =
            ↪  t[tot].min[0]
            ↪  = t[tot].d[0]
            ↪  = y;
```

```
            t[tot].max[1] =
             ↪  t[tot].min[1]
             ↪  = t[tot].d[1]
             ↪  = z;
            t[tot].c[0] =
             ↪  t[tot].c[1] =
             ↪  &Null;
            insert(&t[tot
             ↪  ++]);
        } else {
            ans = inf, ql = y,
             ↪  qr = z;
            query(root),
             ↪  printf("%d\n",
             ↪  ans);
        }
    }
    return 0;
}
```

3 构造

3.1 若干排列使所有数对都出现一次

```cpp
/*n/2 个排列使得所有数对 (i,j) 且 i<j 都出现一次
 *n 为奇数则首尾相连，偶数不连
 */
#include <bits/stdc++.h>

using namespace std;

typedef vector<int> vi;

void get_even(int n, vi ans[]) {
        vi a(n);
        for (int i = 0; i < n; i ++)
                a[i] = i + 1;
        for (int i = 1; i <= n / 2; i ++) {
                ans[i].resize(n + 1);
                for (int j = 0; j < n / 2; j ++)
                        ans[i][j * 2] = a[j], ans[i][j * 2 + 1] = a[n - 1 - j];
                int t = a[n - 1];
                for (int j = n - 1; j > 0; j --)
                        a[j] = a[j - 1];
                a[0] = t;
        }
}

void get_odd(int n, vi ans[]) {
        get_even(n - 1, ans);
        for (int i = 1; i <= n / 2; i ++) {
                for (int j = n - 1; j > 0; j --)
                        ans[i][j] = ans[i][j - 1] + 1;
                ans[i][0] = 1;
        }
}

int main() {
        vi ans[2019];
        int n; cin >> n;
        if (n & 1) get_odd(n, ans);
        else get_even(n, ans);
        return 0;
}
```

3.2 rec-free

```
/* 不存在四个 1 构成一个矩形,
↪  并使得 1 尽量多, 输出 01 矩
↪  阵 */
#include <bits/stdc++.h>

using namespace std;

const int N = 200, n = 150, M
↪  = 13;
//M 为质数, N>M*M>n
int a[N][N], b[N][N], c[N][N];

void make() {
        for (int i = 1; i <=
        ↪  M; i ++)
        for (int j = 1; j <=
        ↪  M; j ++)
            a[i][j + 1] = M *
            ↪  (j - 1) + i;
    for (int i = 1; i <= M; i
    ↪  ++)
        for (int j = 1; j <=
        ↪  M; j ++)
            c[i][a[i][j]] = 1;
    for (int k = 1; k < M; k
    ↪  ++) {
        memcpy(b, a, sizeof
        ↪  b);
        for (int i = 1; i <=
        ↪  M; i ++)
            for (int j = 1; j
            ↪  <= M; j ++)
                a[i][j] = (b[i
                ↪  + j - 1 -
                ↪  ((i + j -
                ↪  1) > M ? M
                ↪  : 0)][j]);
        for (int i = 1; i <=
        ↪  M; i ++)
            for (int j = 1; j
            ↪  <= M; j ++)
                c[k * M +
                ↪  i][a[i][j]]
                ↪  = 1;
    }
}

int main() {
        make();
        return 0;
}
```

4 计算几何

4.1 最小矩形覆盖含凸包和旋转卡壳

```
/*
 * 最小矩形覆盖，保留六位小数，
↪  逆时针输出四个顶点坐标
 */
#include <bits/stdc++.h>

using namespace std;

namespace minRectCover {

        const int N = 1e5 + 5;
        const double eps =
        ↪  1e-8;

        struct point{
                double x, y;
                point(){}
                point(double
                ↪  x, double
                ↪  y):x(x),
                ↪  y(y){}

                bool operator
                ↪  < (const
                ↪  point &a)
                ↪  const {
                        return
                        ↪  fabs(y
                        ↪  -
                        ↪  a.y)
                        ↪  <
                        ↪  eps
                        ↪  ?
                        ↪  x
                        ↪  <
                        ↪  a.x
                        ↪  :
                        ↪  y
                        ↪  <
                        ↪  a.y;
        }

        point operator
        ↪  - (const
        ↪  point &a)
        ↪  const {
                return
                ↪  point(x
                ↪  -
                ↪  a.x,
                ↪  y
                ↪  -
                ↪  a.y);
        }

        point operator
        ↪  + (const
        ↪  point &a)
        ↪  const {
                return
                ↪  point(x
                ↪  +
                ↪  a.x,
                ↪  y
                ↪  +
                ↪  a.y);
        }
```

```
point operator
↪ / (const
↪ double &a)
↪ const {
        return
        ↪ point(x
        ↪ /
        ↪ a,
        ↪ y
        ↪ /
        ↪ a);
}

point operator
↪ * (const
↪ double &a)
↪ const {
        return
        ↪ point(x
        ↪ *
        ↪ a,
        ↪ y
        ↪ *
        ↪ a);
}

double
↪ operator /
↪ (const
↪ point &a)
↪ const { //
↪ .
        return
        ↪ x
        ↪ *
        ↪ a.x
        ↪ +
        ↪ y
        ↪ *
        ↪ a.y;
}

double
↪ operator *
↪ (const
↪ point &a)
↪ const { //
↪ X
        return
        ↪ x
        ↪ *
        ↪ a.y
        ↪ -
        ↪ y
        ↪ *
        ↪ a.x;
}

}p[N], q[N], rc[4];

double sqr(double x)
↪ {return x * x;}
double abs(point a)
↪ {return sqrt(a /
↪ a);}
int sgn(double x)
↪ {return fabs(x) <
↪ eps ? 0 : (x < 0 ?
↪ -1 : 1);}
point vertical(point
↪ a, point b)
↪ {return point(a.x
↪ + a.y - b.y, a.y -
↪ a.x + b.x) -
↪ a;}//与 ab 向量垂
↪ 直的向量
point vec(point
↪ a){return a /
↪ abs(a);}

void convexhull(int n,
↪ point *hull, int
↪ &top) {//如果要计
↪ 算周长需要特判 n==2
```

```
for (int i =
↪  1; i < n;
↪  i ++)
        if
        ↪  (p[i]
        ↪  <
        ↪  p[0])
                swap(p[i],
                ↪  p[0]);
sort (p + 1, p
↪  + n,
↪  [&](point
↪  a, point
↪  b){
        double
        ↪  t
        ↪  =
        ↪  (a
        ↪  -
        ↪  p[0])
        ↪  *
        ↪  (b
        ↪  -
        ↪  p[0]);
        if
        ↪  (fabs(t)
        ↪  <
        ↪  eps)
        ↪  return
        ↪  sgn(abs(p[0]
        ↪  -
        ↪  a)
        ↪  -
        ↪  abs(p[0]
        ↪  -
        ↪  b))
        ↪  <
        ↪  0;
        return
        ↪  t
        ↪  >
        ↪  0;
});
```

```
int cnt =
↪  0;//去重
for (int i =
↪  1; i < n;
↪  i ++)
        if
        ↪  (sgn(p[i].x
        ↪  -
        ↪  p[cnt].x)
        ↪  !=
        ↪  0
        ↪  ||
        ↪  sgn(p[i].y
        ↪  -
        ↪  p[cnt].y)
        ↪  !=
        ↪  0)
                p[++
                ↪  cnt]
                ↪  =
                ↪  p[i];
n = cnt + 1;

hull[top = 1]
↪  = p[0];
for (int i =
↪  1; i < n;
↪  i ++) {
```

```
                while
                ↪ (top
                ↪ >
                ↪ 1
                ↪ &&
                ↪ (hull[top]
                ↪ -
                ↪ hull[top
                ↪ -
                ↪ 1])
                ↪ *
                ↪ (p[i]
                ↪ -
                ↪ hull[top])
                ↪ <
                ↪ eps)
                ↪ top
                ↪ --;
                hull[++
                ↪ top]
                ↪ =
                ↪ p[i];
        }
        hull[0] =
        ↪ hull[top];
}

void main() {
        int n;
        scanf("%d",
        ↪ &n);
        for (int i =
        ↪ 0; i < n;
        ↪ i ++)
                scanf("%lf
                ↪ %lf",
                ↪ &p[i].x,
                ↪ &p[i].y);

        convexhull(n,
        ↪ q, n);

        double ans =
        ↪ 1e20;
        int l = 1, r =
        ↪ 1, t = 1;
        double L, R,
        ↪ D, H;
        for (int i =
        ↪ 0; i < n;
        ↪ i ++) {
                D =
                ↪ abs(q[i]
                ↪ -
                ↪ q[i
                ↪ +
                ↪ 1]);
                while
                ↪ (sgn((q[i
                ↪ +
                ↪ 1]
                ↪ -
                ↪ q[i])
                ↪ *
                ↪ (q[t
                ↪ +
                ↪ 1]
                ↪ -
                ↪ q[i])
                ↪ -
                ↪ (q[i
                ↪ +
                ↪ 1]
                ↪ -
                ↪ q[i])
                ↪ *
                ↪ (q[t]
                ↪ -
                ↪ q[i]))
                ↪ >
                ↪ -1)
                ↪ t
                ↪ =
                ↪ (t
                ↪ +
                ↪ 1)
                ↪ %
                ↪ n;
```

```
while
↪ (sgn((q[i
↪ +
↪ 1]
↪ -
↪ q[i])
↪ /
↪ (q[r
↪ +
↪ 1]
↪ -
↪ q[i])
↪ -
↪ (q[i
↪ +
↪ 1]
↪ -
↪ q[i])
↪ /
↪ (q[r]
↪ -
↪ q[i]))
↪ >
↪ -1)
↪ r
↪ =
↪ (r
↪ +
↪ 1)
↪ %
↪ n;
if (i
↪ ==
↪ 0)
↪ l
↪ =
↪ r;
```

```
while
↪ (sgn((q[i
↪ +
↪ 1]
↪ -
↪ q[i])
↪ /
↪ (q[l
↪ +
↪ 1]
↪ -
↪ q[i])
↪ -
↪ (q[i
↪ +
↪ 1]
↪ -
↪ q[i])
↪ /
↪ (q[l]
↪ -
↪ q[i]))
↪ <
↪ 1)
↪ l
↪ =
↪ (l
↪ +
↪ 1)
↪ %
↪ n;
L =
↪ fabs((q[i
↪ +
↪ 1]
↪ -
↪ q[i])
↪ /
↪ (q[l]
↪ -
↪ q[i])
↪ /
↪ D);
```

```
R =
↪ fabs((q[i
↪ +
↪ 1]
↪ -
↪ q[i])
↪ /
↪ (q[r]
↪ -
↪ q[i])
↪ /
↪ D);
H =
↪ fabs((q[i
↪ +
↪ 1]
↪ -
↪ q[i])
↪ *
↪ (q[t]
↪ -
↪ q[i])
↪ /
↪ D);
double
↪ tmp
↪ =
↪ (R
↪ +
↪ L)
↪ *
↪ H;
if
↪ (tmp
↪ <
↪ ans)
↪ {
        ans
        ↪ =
        ↪ tmp;
rc[0]
↪ =
↪ q[i]
↪ +
↪ (q[i
↪ +
↪ 1]
↪ -
↪ q[i])
↪ *
↪ (R
↪ /
↪ D);//右
↪ 下
rc[1]
↪ =
↪ rc[0]
↪ +
↪ vec(vertical(q[
↪ q[i
↪ +
↪ 1]))
↪ *
↪ H;//右
↪ 上
rc[2]
↪ =
↪ rc[1]
↪ -
↪ (rc[0]
↪ -
↪ q[i])
↪ *
↪ ((R
↪ +
↪ L)
↪ /
↪ abs(q[i]
↪ -
↪ rc[0]));//左
↪ 上
```

```
                                rc[3]
                                =
                                rc[2]
                                -
                                (rc[1]
                                -
                                rc[0]);
                        }
                }

                printf("%.6f\n",
                ans);
                int fir = 0;
                for (int i =
                1; i < 4;
                i ++)
                        if
                        (rc[i]
                        <
                        rc[fir])
                                fir
                                =
                                i;
                for (int i =
                0; i < 4;
                i ++)
                        printf("%.6f
                        %.6f\n",
                        rc[(fir
                        +
                        i)
                        %
                        4].x,
                        rc[(fir
                        +
                        i)
                        %
                        4].y);
        }
}

int main() {
        minRectCover::main();
        return 0;
```

5 其他

5.1 数字哈希

```
namespace my_hash {
        const int N = (1 <<
        ↪  19) - 1;//散列大小,
        ↪  一定要取 2^k-1, 不
        ↪  超内存的情况下, N
        ↪  越大碰撞越少

        struct E {
                int v;
                E *nxt;
        }*g[N + 1], pool[N],
        ↪  *cur = pool, *p;

        int vis[N + 1], T;

        void ins(int v) {
                int u = v & N;
                if (vis[u] <
                ↪  T) vis[u]
                ↪  = T, g[u]
                ↪  = NULL;
                for (p = g[u];
                ↪  p; p = p
                ↪  -> nxt) if
                ↪  (p -> v ==
                ↪  v) return;
                p = cur ++; p
                ↪  -> v = v;
                ↪  p -> nxt =
                ↪  g[u]; g[u]
                ↪  = p;
        }

        int ask(int v) {
                int u = v & N;
                if (vis[u] <
                ↪  T) return
                ↪  0;
                for (p = g[u];
                ↪  p; p = p
                ↪  -> nxt) if
                ↪  (p -> v ==
                ↪  v) return
                ↪  1;
                return 0;
        }

        void init() {T ++, cur
        ↪  = pool;}//应对多组
        ↪  数据使用
}
```

5.2 海岛分金币

5.2.1 海岛分金币 1

```
/*
非朴素模型, 有额外条件:
每个人做决定时如果有多种方案可
↪  以使自己获得最大收益
那么他会让决策顺序靠前的人获得
↪  的收益尽可能的大!
solution:
贪心模拟
*/
#include <bits/stdc++.h>

#define v first
#define id second

using namespace std;

typedef pair<int, int> pr;

const int N = 1010;

int a[N][N];

pr b[N];

int n, m;
```

```
int main() {
    cin >> n >> m;
    a[1][1] = m;
    for (int i = 2; i <= n; i
    ↪  ++) {
        for (int j = 1; j < i;
        ↪  j ++)
            b[j] = pr(a[i -
            ↪  1][j], j);
        sort (b + 1, b + i,
        ↪  [&](pr x, pr
        ↪  y){return x.v !=
        ↪  y.v ? (x.v < y.v)
        ↪  : (x.id >
        ↪  y.id);});
        //按照是否容易满足来排
        ↪  序，因为容易满足的
        ↪  人消耗掉的金币比较
        ↪  少，也就使得当前的
        ↪  人获利最大
        int s = m, nd = (i -
        ↪  1) / 2;
        for (int j = 1; j < i
        ↪  && nd; j ++) {
            nd --;
            s -=
            ↪  (a[i][b[j].id]
            ↪  = a[i -
            ↪  1][b[j].id] +
            ↪  1);
        }
        if (s < 0) {
            for (int j = 1; j
            ↪  < i; j ++)
                a[i][j] = a[i
                ↪  - 1][j];
            a[i][i] = -1;
        }
        else {
            a[i][i] = s;
        }
    }
    for (int i = n; i; i --)
        printf("%d ",
        ↪  a[n][i]);
    return 0;
}
```

5.2.2 海岛分金币 2

```
/*
海盗分金币朴素模型：
n 个海盗分 m 个金币，依次做决策，
↪  如果不少于半数的人同意则方
↪  案通过，否则当前做决策的人会
↪  被淘汰（收益视为-1），由下一
↪  人做出决策
如果一个海盗有多种方案均为最大
↪  收益，那么他会希望淘汰的人越
↪  多越好
求出第 x 个做决策的海盗的最大可
↪  能受益和最小可能收益
*/
#include <bits/stdc++.h>

using namespace std;

struct node {
    int min_v, max_v;
    node():min_v(0), max_v(0)
    ↪  {}
    node(int min_v, int
    ↪  max_v):min_v(min_v),
    ↪  max_v(max_v) {}
};

node ask(int n, int m, int x)
↪  {//n 个人分 m 个金币，第 x
↪  个做决策的人最少/最多分到多
↪  少个金币
    int y = n + 1 - x;
    if (n >= (m + 2) * 2) {
        int a = (m + 1) * 2, b
        ↪  = 2, c = 4;
        //前 a 个为 [0,1]，后 b
        ↪  个为 [0,0]，将持续
        ↪  c 个
```

```
    while (a + b + c <= n)
    ↪  {
        a += b;
        b *= 2;
        c *= 2;
    }
    if (y <= a) return
    ↪  node(0, 1);
    else if (y <= a + b)
    ↪  return node(0, 0);
    else return node(-1,
    ↪  -1);
}
else if (n == m * 2 + 3) {
    if (x == 1) return
    ↪  node(-1, -1);
    else if (y <= m * 2 &&
    ↪  y % 2 == 1 || x ==
    ↪  2) return node(0,
    ↪  0);
    else return node(0,
    ↪  1);
}
else if (n == m * 2 + 2) {
    if (y <= m * 2 && y %
    ↪  2 == 1 || x == 1)
    ↪  return node(0, 0);
    else return node(0,
    ↪  1);
}
else if (n == m * 2 + 1) {
    if (y <= m * 2 && y %
    ↪  2 == 1) return
    ↪  node(1, 1);
    else return node(0,
    ↪  0);
}
else {
    if (x & 1) {
        if (x != 1) return
        ↪  node(1, 1);
        else return node(m
        ↪  - (n - 1) / 2,
        ↪  m - (n - 1) /
        ↪  2);
    }
    else return node(0,
    ↪  0);
}
}

int main() {

    ↪  ios::sync_with_stdio(false);
    int x, n, m, k; node y;
    cin >> n >> m >> k;
    while (k --) {
        cin >> x;
        y = ask(n, m, x);
        printf("%d %d\n",
        ↪  y.min_v, y.max_v);
    }
    return 0;
}

/*
m = 5

1  5
2  0 5
3  1 0 4
4  0 1 0 4
5  1 0 1 0 3
6  0 1 0 1 0 3
7  1 0 1 0 1 0 2
8  0 1 0 1 0 1 0 2
9  1 0 1 0 1 0 1 0 1
10 0 1 0 1 0 1 0 1 0 1
11 1 0 1 0 1 0 1 0 1 0 0
12 0 _ 0 _ 0 _ 0 _ 0 _ _ 0
13 0 _ 0 _ 0 _ 0 _ 0 _ _ 0 -1
14 _ _ _ _ _ _ _ _ _ _ _ _ 0 0
15 _ _ _ _ _ _ _ _ _ _ _ _ 0 0
↪  -1
```

```
16 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ 0 0
↪  -1 -1
17 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ 0 0
↪  -1 -1 -1
18 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
↪  0 0 0 0
19 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
↪  0 0 0 0 -1
20 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
↪  0 0 0 0 -1 -1
21 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
↪  0 0 0 0 -1 -1 -1
22 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
↪  0 0 0 0 -1 -1 -1 -1
23 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
↪  0 0 0 0 -1 -1 -1 -1 -1
24 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
↪  0 0 0 0 -1 -1 -1 -1 -1 -1
25 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
↪  0 0 0 0 -1 -1 -1 -1 -1 -1
↪  -1
26 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
↪  _ _ _ _ 0 0 0 0 0 0 0 0
*/
```

5.3 根号枚举

```
for (int i = 1, last; i <= n;
↪  i = last + 1) {
        last = n / (n / i);
        //当前枚举区间为 [i,
        ↪  last]
}
```

5.4 读入输出外挂

```
namespace IO {//only for
↪  int!!!
    static const int SIZE = 1
    ↪  << 20;

    inline int get_char() {
        static char *S, *T =
        ↪  S, buf[SIZE];
        if (S == T) {
            T = fread(buf, 1,
            ↪  SIZE, stdin) +
            ↪  (S = buf);
            if (S == T) return
            ↪  -1;
        }
        return *S ++;
    }

    inline void in(int &x)
    ↪  {//for int
        static int ch;
        while (ch =
        ↪  get_char(), ch <
        ↪  48);x = ch ^ 48;
        while (ch =
        ↪  get_char(), ch >
        ↪  47) x = x * 10 +
        ↪  (ch ^ 48);
    }

    char buffer[SIZE];
    char *s = buffer;

    void flush() {//最后需要
    ↪  flush!!
        fwrite(buffer, 1, s -
        ↪  buffer, stdout);
        s = buffer;
        fflush(stdout);
    }

    inline void print(const
    ↪  char ch) {
        if(s - buffer > SIZE -
        ↪  2) flush();
        *s++ = ch;
    }

    inline void print(char
    ↪  *str) {//for string
        while(*str != 0)
```

```
            print(char(*str
            ↪  ++));
    }

    inline void print(int x) {
        static char buf[25];
        static char *p = buf;
        if (x <  0)
        ↪  print('-'), x =
        ↪  -x;
        if (x == 0)
        ↪  print('0');
        while(x) *(++ p) = x %
        ↪  10, x /= 10;
        while(p != buf)
        ↪  print(char(*(p --)
        ↪  ^ 48));
    }
};
```

5.5 给定小数化成分数

```
# 本题答案的分母不超过 1e9, 给
↪  定小数的小数点位为 18 位
inf, inff = 10 ** 9, 10 ** 18
for i in range(int(input())):
 n = int(input()[2:])
 if n == 0: print('0 1')
 else:
  lp, lq, rp, rq = 0, 1, 1, 1
  while max(lq, rq) <= inf:
   mp, mq = lp + rp, lq + rq
   if mp * inff <= mq * n:
    l, r, mid, cnt = 1, (inf -
    ↪  lq) // rq + 1, -1, -1
    while l <= r:
     mid = l + r >> 1
     if (lp + rp * mid) * inff
     ↪  <= (lq + rq * mid) *
     ↪  n:
      cnt, l = mid, mid + 1
     else:
      r = mid - 1
    lp, lq = lp + rp * cnt, lq
    ↪  + rq * cnt
   else:
    l, r, mid, cnt = 1, (inf -
    ↪  rq) // lq + 1, -1, -1
    while l <= r:
     mid = l + r >> 1
     if (rp + lp * mid) * inff
     ↪  > (rq + lq * mid) *
     ↪  n:
      cnt, l = mid, mid + 1
     else:
      r = mid - 1
    rp, rq = rp + lp * cnt, rq
    ↪  + lq * cnt
  if lq <= inf: print(lp, lq)
  else: print(rp, rq)
```