Олимпиада СПбГУ
по информатике
2023/24 учебного года

24fin0363 – Писарев Егор Дмитриевич

A	B	C	D	E	F	sum
100	100	100	100	100	90	590

Task A

```
#include <iostream>
#include <vector>

int main() {
    long long a, b, n;
    std::cin >> a >> b >> n;
    std::vector<long long> dp(n + 1, 0);
    if (a <= n) {
        dp[a] = 1;
    }
    for (int i = 0; i <= n; i++) {
        if (i + a <= n) {
            dp[i + a] += dp[i];
        }
        if (i + b <= n) {
            dp[i + b] += dp[i];
        }
    }
    long long resa = dp[n];
    dp.assign(n + 1, 0);
    if (b <= n) {
        dp[b] += 1;
    }
    for (int i = 0; i <= n; i++) {
        if (i + a <= n) {
            dp[i + a] += dp[i];
        }
        if (i + b <= n) {
            dp[i + b] += dp[i];
```

```
        }
    }
    long long resb = dp[n];
    std::cout << resa * a + resb * b << '\n';
    return 0;
}
```

Task B

```
#include <iostream>
#include <vector>

long long calcd(long long x1, long long y1, long long z1, long long x2,
    long long y2, long long z2) {
    return (std::abs(x1 - x2) + std::abs(y1 - y2) + std::abs(z1 - z2)) /
    2;
}

int main() {
    long long n;
    long long a1, b1, c1;
    long long a2, b2, c2;
    std::cin >> n >> a1 >> b1 >> c1 >> a2 >> b2 >> c2;
    long long res = calcd(a1, b1, c1, a2, b2, c2);
    int x = n;
    int y;
    int z;
    int xn, yn, zn;
    for (z = -n; z <= 0; z++) {
        y = -x - z;
        xn = -n;
        yn = n + y;
        zn = n + z;
        res = std::min(res, calcd(a1, b1, c1, x, y, z) + calcd(xn, yn, zn
    , a2, b2, c2));
        res = std::min(res, calcd(a1, b1, c1, xn, yn, zn) + calcd(x, y, z
    , a2, b2, c2));
    }
    y = n;
    for (x = -n; x <= 0; x++) {
        z = -x - y;
        yn = -n;
        xn = n + x;
        zn = n + z;
        res = std::min(res, calcd(a1, b1, c1, x, y, z) + calcd(xn, yn, zn
    , a2, b2, c2));
        res = std::min(res, calcd(a1, b1, c1, xn, yn, zn) + calcd(x, y, z
    , a2, b2, c2));
    }
    z = n;
    for (x = -n; x <= 0; x++) {
        y = -z - x;
        zn = -n;
```

```
        xn = n + x;
        yn = n + y;
        res = std::min(res, calcd(a1, b1, c1, x, y, z) + calcd(xn, yn, zn
    , a2, b2, c2));
        res = std::min(res, calcd(a1, b1, c1, xn, yn, zn) + calcd(x, y, z
    , a2, b2, c2));
    }
    std::cout << res << '\n';
    return 0;
}
```

Task C

```
#include <iostream>
#include <vector>
#include <string>

constexpr long long p = 3;
constexpr long long mod = 1000000007;

long long pow(long long a, long long b) {
    long long res = 1;
    while (b > 0) {
        if (b & 1) {
            res = res * 1ll * a % mod;
        }
        a = a * 1ll * a % mod;
        b >>= 1;
    }
    return res;
}

void solve1() {
    int n;
    std::cin >> n;
    std::string s;
    std::getline(std::cin, s);
    std::getline(std::cin, s);
    long long h = 0;
    for (int i = 0; i < n; i++) {
        h = (h * p + (s[i] - '0' + 1)) % mod;
    }
    std::cout << h << '\n';
}

void solve2() {
    int n;
    std::cin >> n;
    std::string s;
    std::cin >> s;
    long long h1;
    std::cin >> h1;
    long long cpw = pow(p, n);
```

```
    long long h = 0;
    for (int i = 0; i < n; i++) {
        h = (h * p + (s[i] - '0' + 1)) % mod;
    }
    for (int firstpos = 0; firstpos < n; firstpos++) {
        if (h == h1) {
            std::cout << "YES\n";
            return;
        }
        h = (h * p + (s[firstpos] - '0' + 1)) % mod;
        h = h - (s[firstpos] - '0' + 1) * cpw % mod;
        if (h < 0) {
            h += mod;
        }
        if (h >= mod) {
            h -= mod;
        }
    }
    std::cout << "NO\n";
}

int main() {
    std::string run;
    std::cin >> run;
    if (run[0] == 'f') {
        solve1();
    } else {
        solve2();
    }
}
```

Task D

```
#include <algorithm>
#include <iostream>
#include <numeric>
#include <vector>

struct Perm {
    int n;
    std::vector<int> a;
    int rotate;
    Perm() = default;
    Perm(int _n) {
        n = _n;
        a.resize(n);
        std::iota(a.begin(), a.end(), 0);
        rotate = 0;
    }

    int operator[](int i) {
        return a[(i + rotate) - n * (i + rotate >= n)];
    }
```

```
    void rot() {
        rotate = rotate + 1 - n * (rotate + 1 == n);
    }

    void swp(int x, int y) {
        int pos1 = (x + rotate - n * (x + rotate >= n));
        int pos2 = (y + rotate - n * (y + rotate >= n));
        std::swap(a[pos1], a[pos2]);
    }
};

Perm p;
// Perm p1;

char ask(int i, int j) {
    std::cout << i + 1 << ' ' << j + 1 << std::endl;
    p.swp(i, j);
    // p1.swp(i, j);
    p.rot();
    // int cpr = p1[0];
    // p1.rot();
    // int ccur = p1[0];
    char res;
    std::cin >> res;
    if (res == '!') {
        exit(0);
    }
    return res;
    // if (ccur > cpr) {
    //     return '>';
    // } else {
    //     return '<';
    // }
}

int main() {
    int n;
    std::cin >> n;
    p = Perm(n);
    // p1 = Perm(n);
    // p1.a[0] = 1;
    // p1.a[1] = 4;
    // p1.a[2] = 2;
    // p1.a[3] = 3;
    // p1.a[4] = 0;
    // for (int i = 0; i < n; i++) {
    //     std::cout << p1[i] << " \n"[i == n - 1];
    // }
    std::vector<int> cnt(n, 0);
    for (int first = 0; first < n; first++) {
        for (int second = first + 1; second < n; second++) {
            int firstpos = 0;
            int secondpos = 0;
```

```
            for (int j = 0; j < n; j++) {
                if (p[j] == first) {
                    firstpos = j;
                }
                if (p[j] == second) {
                    secondpos = j;
                }
            }
            if (secondpos == 1) {
                secondpos = firstpos;
            }
            ask(firstpos, 1);
            secondpos = (secondpos - 1 + n * (secondpos - 1 < 0));
            char cres = ask(secondpos, 1);
            if (cres == '<') {
                cnt[first]++;
            } else {
                cnt[second]++;
            }
        }
    }
    for (int i = n - 1; i >= 0; i--) {
        for (int j = 0; j < n; j++) {
            if (cnt[j] == i) {
                int jpos = 0;
                for (int k = 0; k < n; k++) {
                    if (p[k] == j) {
                        jpos = k;
                    }
                }
                ask(jpos, 0);
            }
        }
    }
    while (ask(0, 0) != '!');
    return 0;
}
```

Task E

```
#include <algorithm>
#include <iostream>
#include <random>
#include <vector>

std::vector<std::vector<int>> gr;
std::vector<int> pr;

void dfs0(int v) {
    for (int to : gr[v]) {
```

```
        if (to != pr[v]) {
            pr[to] = v;
            dfs0(to);
        }
    }
}

std::mt19937 rnd;

struct Node {
    int val;
    unsigned y;
    int sz;
    Node* pr;
    Node* l;
    Node* r;
    Node(int _val) {
        val = _val;
        y = rnd();
        sz = 1;
        pr = nullptr;
        l = r = nullptr;
    }
};

int getsz(Node* v) {
    if (!v) {
        return 0;
    }
    return v->sz;
}

void recalc(Node* v) {
    if (!v) {
        return;
    }
    v->sz = getsz(v->l) + 1 + getsz(v->r);
    v->pr = nullptr;
    if (v->l) {
        v->l->pr = v;
    }
    if (v->r) {
        v->r->pr = v;
    }
}

std::pair<Node*, Node*> split(Node* v, int szl) {
    if (!v) {
        return {nullptr, nullptr};
    }
    if (getsz(v->l) >= szl) {
        auto sl = split(v->l, szl);
        v->l = sl.second;
```

```
        recalc(v);
        return {sl.first, v};
    } else {
        auto sr = split(v->r, szl - getsz(v->l) - 1);
        v->r = sr.first;
        recalc(v);
        return {v, sr.second};
    }
}

Node* merge(Node* l, Node* r) {
    if (!l) {
        return r;
    }
    if (!r) {
        return l;
    }
    if (l->y > r->y) {
        l->r = merge(l->r, r);
        recalc(l);
        return l;
    } else {
        r->l = merge(l, r->l);
        recalc(r);
        return r;
    }
}

int getind(Node* v) {
    int res = getsz(v->l);
    while (v->pr != nullptr) {
        if (v == v->pr->r) {
            res += getsz(v->pr->l) + 1;
        }
        v = v->pr;
    }
    return res;
}

void dfs1(int v, Node*& et, Node*& nlroot, std::vector<Node*>& firstnodes
    ) {
    firstnodes[v] = new Node(v);
    et = merge(et, firstnodes[v]);
    for (int to : gr[v]) {
        if (to == pr[v]) {
            if (to != gr[v].back()) {
                nlroot = merge(nlroot, new Node(v));
            }
            break;
        } else {
            dfs1(to, et, nlroot, firstnodes);
            et = merge(et, new Node(v));
        }
```

```
    }
}

int getv(Node* v, int ind) {
    if (ind < getsz(v->l)) {
        return getv(v->l, ind);
    } else if (ind == getsz(v->l)) {
        return v->val;
    } else {
        return getv(v->r, ind - getsz(v->l) - 1);
    }
}

int main() {
    int n, q;
    std::cin >> n >> q;
    gr.resize(n);
    for (int i = 0; i < n; i++) {
        int cnt;
        std::cin >> cnt;
        gr[i].resize(cnt);
        for (int j = 0; j < cnt; j++) {
            std::cin >> gr[i][j];
            gr[i][j]--;
        }
    }
    pr.resize(n);
    pr[0] = -1;
    dfs0(0);

    Node* cet = nullptr;
    Node* cnlroot = nullptr;
    std::vector<Node*> firstnodes(n, nullptr);
    dfs1(0, cet, cnlroot, firstnodes);

    auto [cetl, r] = split(cet, getsz(cet) - 1);
    cet = cetl;

    long long ctime = 0;

    for (int i = 0; i < q; i++) {
        long long cur;
        std::cin >> cur;
        if (getsz(cnlroot) == 0) {
            long long caddtime = cur - ctime;
            caddtime %= getsz(cet);
            std::cout << getv(cet, caddtime) + 1 << " \n"[i == q - 1];
            continue;
        } else {
            while (ctime + getsz(cet) <= cur && getsz(cnlroot) > 0) {
                ctime += getsz(cet);
                Node* nwcnlroot = nullptr;
                while (getsz(cnlroot) != 0) {
```

```
                    auto [cf, cr] = split(cnlroot, 1);
                    cnlroot = cr;
                    int cv = cf->val;
                    int cind = getind(firstnodes[cv]);
                    auto [etl, etr] = split(cet, cind);
                    Node* net = nullptr;
                    firstnodes[cv] = new Node(cv);
                    net = merge(net, firstnodes[cv]);
                    int cprind = std::find(gr[cv].begin(), gr[cv].end(),
    pr[cv]) - gr[cv].begin();
                    for (int j = cprind + 1; j < (int)gr[cv].size(); j++)
     {
                        dfs1(gr[cv][j], net, nwcnlroot, firstnodes);
                        if (j != (int)gr[cv].size() - 1) {
                            net = merge(net, new Node(cv));
                        }
                    }
                    cet = merge(merge(etl, net), etr);
                }
                cnlroot = nwcnlroot;
            }
            if (getsz(cnlroot) == 0) {
                long long caddtime = cur - ctime;
                caddtime %= getsz(cet);
                std::cout << getv(cet, caddtime) + 1 << " \n"[i == q -
    1];
                continue;
            } else {
                int cind = cur - ctime;
                std::cout << getv(cet, cind) + 1 << " \n"[i == q - 1];
            }
        }
    }
    return 0;
}
```

Task F

```
#pragma GCC optimize("O3")
#pragma GCC optimize("unroll-loops")
#pragma GCC optimize("inline")

#include <algorithm>
#include <cmath>
#include <iostream>
#include <random>
#include <set>
#include <vector>

int cres = 0;
```

```
std::vector<std::vector<int>> a;
std::vector<std::pair<std::pair<int, int>, std::pair<int, int>>> pairs;
std::vector<std::vector<bool>> used;

std::mt19937 rnd;

long double getp() {
    return rnd() / (long double)(rnd.max());
}

void change(long double iter) {
    bool type = rnd() % 2;
    if (pairs.empty()) {
        type = true;
    }
    if (type == false) {
        int cind = rnd() % pairs.size();
        int ccost = a[pairs[cind].first.first][pairs[cind].first.second]
    * a[pairs[cind].second.first][pairs[cind].second.second];
        if (getp() < (exp((long double)ccost / 5000 - iter))) {
            auto cp = pairs[cind];
            used[cp.first.first][cp.first.second] = false;
            used[cp.second.first][cp.second.second] = false;
            cres -= ccost;
            pairs.erase(pairs.begin() + cind);
        }
    } else {
        std::vector<std::pair<std::pair<int, int>, std::pair<int, int>>>
    np;
        for (int i = 0; i < (int)a.size(); i++) {
            for (int j = 0; j < (int)a[i].size(); j++) {
                if (!used[i][j] && !used[(i + 1) % a.size()][j]) {
                    np.emplace_back(std::make_pair(i, j), std::make_pair
    ((i + 1) % a.size(), j));
                }
                if (!used[i][j] && !used[i][(j + 1) % a[i].size()]) {
                    np.emplace_back(std::make_pair(i, j), std::make_pair(
    i, (j + 1) % a[i].size()));
                }
            }
        }
        if (np.empty()) {
            return;
        }
        auto cp = np[rnd() % np.size()];
        int ccost = a[cp.first.first][cp.first.second] * a[cp.second.
    first][cp.second.second];
        cres += ccost;
        pairs.emplace_back(cp);
        used[cp.first.first][cp.first.second] = true;
        used[cp.second.first][cp.second.second] = true;
    }
}
```

```
int main() {
    int n, m;
    std::cin >> n >> m;
    a.assign(n, std::vector<int>(m));
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            char cur;
            std::cin >> cur;
            a[i][j] = cur - '0';
        }
    }
    used.assign(n, std::vector<bool>(m, false));
    // for (int i = 0; i < n; i += 2) {
    //     for (int j = 0; j < m; j += 2) {
    //         if (a[i][j] * a[i][j + 1] + a[i + 1][j] * a[i + 1][j + 1]
   > a[i][j] * a[i + 1][j] + a[i][j + 1] * a[i + 1][j + 1]) {
    //             pairs.emplace_back(std::make_pair(i, j), std::
   make_pair(i, j + 1));
    //             pairs.emplace_back(std::make_pair(i + 1, j), std::
   make_pair(i + 1, j + 1));
    //             cres += a[i][j] * a[i][j + 1] + a[i + 1][j] * a[i +
   1][j + 1];
    //         } else {
    //             pairs.emplace_back(std::make_pair(i, j), std::
   make_pair(i + 1, j));
    //             pairs.emplace_back(std::make_pair(i, j + 1), std::
   make_pair(i + 1, j + 1));
    //             cres += a[i][j] * a[i + 1][j] + a[i][j + 1] * a[i +
   1][j + 1];
    //         }
    //     }
    // }
    std::vector<std::pair<std::pair<int, int>, std::pair<int, int>>> cand
   ;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            cand.emplace_back(std::make_pair(i, j), std::make_pair(i, (j
   + 1) % m));
            cand.emplace_back(std::make_pair(i, j), std::make_pair((i +
   1) % n, j));
        }
    }
    std::sort(cand.begin(), cand.end(), [](const std::pair<std::pair<int,
     int>, std::pair<int, int>>& p1, const std::pair<std::pair<int, int>,
   std::pair<int, int>>& p2) {
        return a[p1.first.first][p1.first.second] * a[p1.second.first][p1
   .second.second] > a[p2.first.first][p2.first.second] * a[p2.second.
   first][p2.second.second];
    });
    int cnt = 0;
    for (auto [r1, r2] : cand) {
        if (!used[r1.first][r1.second] && !used[r2.first][r2.second]) {
```

```
            pairs.emplace_back(r1, r2);
            cres += a[r1.first][r1.second] * a[r2.first][r2.second];
            used[r1.first][r1.second] = used[r2.first][r2.second] = true;
        }
        cnt++;
    }
    for (int iter = 0; iter < 600000; iter++) {
        // if (iter % 512 == 0) {
        //     if (clock() > 1.9 * CLOCKS_PER_SEC) {
        //         break;
        //     }
        // }
        change(iter);
    }
    std::vector<std::vector<char>> res(n, std::vector<char>(m, '.'));
    for (auto [f, s] : pairs) {
        if (f.first == s.first) {
            res[f.first][f.second] = 'L';
            res[s.first][s.second] = 'R';
        } else {
            res[f.first][f.second] = 'T';
            res[s.first][s.second] = 'B';
        }
    }
    std::cout << cres << '\n';
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            std::cout << res[i][j];
        }
        std::cout << '\n';
    }
    return 0;
}
```