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

const int MAX_COORD = 1e9;
typedef long long li;

const string SMILE_BASE_STRING = R"eof(
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0
0 0 0 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 0 0 0
0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0
0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0
0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0
0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0
0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0
0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0
0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0
0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0
0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0
0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0
0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0
1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0
0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
)eof";

li query(int x0, int x1, int y0, int y1)
{
    cout << x0 << " " << y0 << " " << x1 << " " << y1 << endl;
    li answer;
    cin >> answer;
    return answer;
}

void answer(string verdict) {
    cout << -1 << endl << verdict << endl;
}

li total;
int base_x1, base_x2, base_y1, base_y2;
int size;

li inter(int a, int b, int c, int d) {
```

```
    return max(0, min(b, d) - max(a, c));
}

li query_base(int x0_q, int x1_q, int y0_q, int y1_q) {
    istringstream base_stream(SMILE_BASE_STRING);
    li answer = 0;
    for (int y = 40; y >= 0; y--) {
        for (int x = 0; x < 41; x++) {
            char c;
            assert(base_stream >> c);
            if (c == '1') {
                int x1_r = base_x1 + x * size;
                int x2_r = x1_r + size;

                int y1_r = base_y1 + y * size;
                int y2_r = y1_r + size;

                answer += inter(x1_r, x2_r, x0_q, x1_q) * inter(y1_r, y2_r, y0_q, y1_q);
            }
        }
    }
    return answer;
}

li query_smile(int x0_q, int x1_q, int y0_q, int y1_q) {
    li count = query_base(x0_q, x1_q + 1, y0_q, y1_q + 1) - query(x0_q, x1_q, y0_q, y1_q);
    assert(count >= 0);
    return count;
}

int main() {
    total = query(1, MAX_COORD, 1, MAX_COORD);
    int low, high;

    // locate base left: low outside, high inside
    low = 0, high = MAX_COORD;
    while (low + 1 != high) {
        int m = (low + high) / 2;
        if (query(1, m, 1, MAX_COORD) != 0)
            high = m;
        else
            low = m;
    }
    base_x1 = high;

    // locate base right: low inside, high outside
    low = 1, high = MAX_COORD + 1;
    while (low + 1 != high) {
        int m = (low + high) / 2;
        if (query(1, m, 1, MAX_COORD) != total)
            low = m;
        else
            high = m;
    }
    base_x2 = low + 1;

    // locate base bottom: low outside, high inside
    low = 0, high = MAX_COORD;
    while (low + 1 != high) {
        int m = (low + high) / 2;
        if (query(1, MAX_COORD, 1, m) != 0)
            high = m;
        else
            low = m;
    }
    base_y1 = high;

    base_y2 = base_y1 + (base_x2 - base_x1);
```

```
    assert((base_x2 - base_x1 + 1) % 41 == 0);
    size = (base_x2 - base_x1 + 1) / 41;

    // locate smile bottom:
    li total_smile = query_smile(1, MAX_COORD, 1, MAX_COORD);
    low = 0, high = MAX_COORD;
    while (low + 1 != high) {
        int m = (low + high) / 2;
        if (query_smile(1, MAX_COORD, 1, m) != 0)
            high = m;
        else
            low = m;
    }
    int smile_bottom = high;

    // locate smile top:
    low = 1, high = MAX_COORD + 1;
    while (low + 1 != high) {
        int m = (low + high) / 2;
        if (query_smile(1, MAX_COORD, 1, m) != total_smile)
            low = m;
        else
            high = m;
    }
    int smile_top = low + 1;

    int smile_size = (smile_top - smile_bottom + 1) / 5;

    li first_row = query_smile(1, MAX_COORD, 1, smile_bottom) / smile_size;
    assert(first_row == 4 || first_row == 18);

    answer(first_row == 4 ? "HAPPY" : "SAD");

    return 0;
}

=====================================================
======================== B ==========================
#include <bits/stdc++.h>
using namespace std;
#define fs  first
#define sc  second
#define mp  make_pair
#define pb  push_back
#define ll  long long
#define vi  vector<ll >
#define vvi vector<vi >

ll  mod = (ll)1e9 + 33;
int n, k, m;
vvi t;
vector<vvi > dp;
int x[7][2][2] = {
    { { 0, 0 }, { 0, 0 } },
    { { 0, 1 }, { 1, 1 } },
    { { 1, 1 }, { 1, 0 } },
    { { 1, 0 }, { 1, 1 } },
    { { 1, 1 }, { 0, 1 } },
    { { 1, 0 }, { 0, 0 } },
    { { 1, 1 }, { 1, 1 } }
};

bool z(int mask, int pos) { return ((mask & (1 << pos)) == 0); }

bool ok(int mask, int mas, int pos, int i, int& new_mask, int& new_mas) {
    new_mask = mask;
    new_mas  = mas;
    for (int l = 0; l < 2; ++l) {
        if (x[i][0][l]) {
```

```
            if (!z(mask, pos + l)) return false;
            new_mask ^= (1 << (pos + l));
        }
        if (x[i][1][l]) {
            if (!z(mas, pos + l)) return false;
            new_mas ^= (1 << (pos + l));
        }
    }
    if (i < 2 && z(mask, pos)) return false;
    return true;
}

void rec(int mask, int mas, int pos, vector<pair<int, int> >& v, int ii) {
    v.clear();
    if (pos == 6) {
        if (z(mask, 6) && t[pos][ii]) v.pb(mp(mas, 1));
        if (!z(mask, 6)) v.pb(mp(mas, 0));
        return;
    }
    for (int i = 0; i < 7; ++i) {
        int new_mask, new_mas;
        vector<pair<int, int> > w;
        if (i > 4 && t[pos][ii] == 0)
            continue;
        if (!ok(mask, mas, pos, i, new_mask, new_mas))
            continue;
        rec(new_mask, new_mas, pos + 1, w, ii);
        for (int j = 0; j < w.size(); ++j) {
            if (i > 4) ++w[j].sc;
            v.pb(w[j]);
        }
    }
}

int main() {
    ios_base::sync_with_stdio(false);
    cin >> n >> k >> m;
    k = 7 * n - 3 * k;
    t.assign(7, vi(n, 1));
    for (int i = 0; i < m; ++i) {
        int w, d; cin >> w >> d; --w; --d;
        t[d][w] = 0;
    }
    dp.assign(n + 1, vvi(k + 1, vi(128, 0)));
    dp[0][0][0] = 1;
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < k + 1; ++j) {
            for (int mask = 0; mask < 128; ++mask) {
                if (dp[i][j][mask] == 0)
                    continue;
                vector<pair<int, int> > v;
                rec(mask, 0, 0, v, i);
                for (int l = 0; l < v.size(); ++l) {
                    int new_mask = v[l].fs;
                    int jj       = v[l].sc;
                    if (j + jj > k)
                        continue;
                    dp[i + 1][j + jj][new_mask] += dp[i][j][mask];
                    if (dp[i + 1][j + jj][new_mask] >= mod)
                        dp[i + 1][j + jj][new_mask] -= mod;
                }
            }
        }
    }
    cout << dp[n][k][0] << endl;

    return 0;
}
```

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

static const int setSize = 1000000 + 5;
static bitset<setSize> reachablePositions, allowedPositions;

int main() {
    int n, m, k, l, h;
    scanf("%d%d%d%d%d", &n, &m, &k, &l, &h);
    vector<int> initPos(k);

    for (int i = 0; i < k; i++)
        scanf("%d", &initPos[i]);

    vector<pair<int, int> > snow(l); // first - initial height, second - cell number

    for (int i = 0; i < l; i++) {
        int cell, height;
        scanf("%d%d", &cell, &height);
        snow[i].first = height;
        snow[i].second = cell;
    }

    sort(snow.begin(), snow.end());
    vector<int> busyAmount(n + 1, 0);

    for (int i = 1; i <= n; i++) {
        reachablePositions.set(i);
        allowedPositions.set(i);
    }

    int snowBegin, snowEnd;
    snowEnd = snow.size();

    while (snowEnd > 0 && snow[snowEnd - 1].first - (m + 1) > h)
        snowEnd--;

    snowBegin = snowEnd;
    for (int turn = m; turn >= 0; turn--) {
        // Transfer situation from <after turn TURN+1> to <after turn TURN>
        while (snowEnd > 0 && snow[snowEnd - 1].first - turn > h) {
            int cell = snow[snowEnd - 1].second;
            if (busyAmount[cell] > 0) {
                busyAmount[cell]--;
                if (busyAmount[cell] == 0)
                    allowedPositions.set(cell);
            }
            snowEnd--;
        }
        snowBegin = std::min(snowBegin, snowEnd);
        while (snowBegin > 0 && snow[snowBegin - 1].first - turn >= 0) {
            int cell = snow[snowBegin - 1].second;
            busyAmount[cell]++;
            if (busyAmount[cell] == 1)
                allowedPositions.reset(cell);
            snowBegin--;
        }
        reachablePositions |= (reachablePositions >> 1);
        reachablePositions |= (reachablePositions << 1);
        reachablePositions &= allowedPositions;
    }

    int ans = 0;
    for (int i = 0; i < k; i++) {
        if (reachablePositions.test(initPos[i]))
            ans++;
    }
```

```
    printf("%d\n", ans);

    return 0;
}

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

static const int mod = 1000000000 + 7;

static int pow2(int degree) {
    long long accum = 2;
    long long res = 1;
    for (int i = 0; i < 31; i++) {
        if (degree & (1 << i))
            res = res * accum % mod;
        accum = accum * accum % mod;
    }
    return res % mod;
}

int main() {
    int k;
    scanf("%d", &k);
    int degree = k - (k & (k - 1));
    int res = (pow2(degree) + mod - 1) % mod;
    printf("%d", res);

    return 0;
}

==================================================
======================== E =======================
#include<bits/stdc++.h>
using namespace std;
#define int int64_t

const double eps = 1e-7;

signed main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    int n, m;
    double S;
    cin >> n >> m >> S;
    int s = round(2 * S);
    if (s > 2 * n * m || abs(s - 2 * S) > eps) {
        cout << -1 << endl;
        return 0;
    }
    bool flag = s % 2;
    s = s / 2 + flag;
    while (n * (m - 1) >= s)
        m--;
    if (m == 1) {
        cout << 4 - (s == flag) << endl;
        cout << "0 0" << endl;
        cout << "0 1" << endl;
        cout << s << " 1" << endl;
        if (s != flag)
            cout << s - flag << " 0" << endl;
        return 0;
    }
    if (m * n == s) {
        cout << 4 - (n == flag) + flag << endl;
        cout << "0 0" << endl;
        cout << "0 " << m << endl;
```

```
        cout << n << ' ' << m << endl;
        if (flag)
            cout << n << ' ' << 1 << endl;
        if (n != flag)
            cout << n - flag << " 0" << endl;
        return 0;
    }
    cout << (m == 2 ? 4 : 5) - (n == flag) + flag << endl;
    cout << "0 0" << endl;
    if (m > 2)
        cout << "0 " << m - 2 << endl;
    cout << n * m - s << ' ' << m << endl;
    cout << n << ' ' << m << endl;
    if (flag)
        cout << n << ' ' << 1 << endl;
    if (n != flag)
        cout << n - flag << " 0" << endl;

    return 0;
}

==================================================
======================== F =======================
#include <bits/stdc++.h>
using namespace std;
#define pb push_back

int main() {
    int n; cin >> n;
    for (int i = 0; i < n * (n - 1) / 2; ++i) {
        int u, v;
        string s;
        cin >> u >> v >> s;
        if (s.size() == 1)
            cout << u << " " << v << " " << s << "\n";
    }

    return 0;
}

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

int const maxn = 10004;
pair<int,int> a[maxn];
pair<long long, long long> p[maxn];

int getIntersect(pair<int,int> x, pair<int,int> y, pair<long long, long long>& res) {
    if (x.first == y.first) return 0;

    long long num = y.second - x.second + y.first * 1LL * y.first - x.first * 1LL * x.first;
    long long denum = 2LL * (y.first - x.first);

    int sgn = num >= 0 ? 1 : -1;
    if (denum < 0) sgn *= -1;
    num = abs(num);
    denum = abs(denum);
    long long d = __gcd(num, denum);
    num /= d;
    denum /= d;
    res = {num * sgn, denum};
    return 1;
}

int main() {
    ios_base::sync_with_stdio(false);
    int n; cin >> n;
```

```
    for(int i = 0; i < n; ++i)
        cin >> a[i].first >> a[i].second;

    sort(a, a + n);
    n = unique(a, a + n) - a;

    int ans = 1;
    for(int i = 0; i < n; ++i) {
        ++ans;
        int cntPoints = 0;
        for(int j = 0; j < i; ++j)
            cntPoints += getIntersect(a[i], a[j], p[cntPoints]);
        sort(p, p + cntPoints);
        ans += unique(p, p + cntPoints) - p;
    }
    cout << ans << endl;

    return 0;
}

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

vector<vector<int> > ST;

void build_ST(vector<int>& a) {
    int m = a.size();
    int n = 1;
    int k = 1;
    while (n < m)
        n *= 2, ++k;
    ST.assign(k, vector<int>(n, 0));
    for (int j = 0; j < m; ++j)
        ST[0][j] = a[j];
    for (int i = 1, l = 1; i < k; ++i, l *= 2)
        for (int j = 0; j < n - 2 * l + 1; ++j)
            ST[i][j] = max(ST[i - 1][j], ST[i - 1][j + l]);
}

int get_max(int l, int r) {
    int h = 1;
    int i = 0;
    while (h + h < r - l + 1)
        h *= 2, ++i;
    return max(ST[i][l], ST[i][r - h + 1]);
}

int n, m, l, r, h;
vector<int> a;
set<int> water;
set<int>::iterator it;

bool l_ok() {
    it = water.lower_bound(l);
    if (it == water.begin()) return true;
    return (get_max(*(--it) + 1, l) >= h);
}
bool r_ok() {
    it = water.lower_bound(l);
    if (it == water.end()) return true;
    if (*it <= r)          return false;
    return (get_max(r, *it - 1) >= h);
}

int main() {
    scanf("%d %d", &n, &m);
    a.resize(n);
```

```
    for (int i = 0; i < n; ++i) {
        scanf("%d", &a[i]);
        if (a[i] == 0)
            water.insert(i);
    }
    build_ST(a);

    for (int i = 0; i < m; ++i) {
        scanf("%d %d %d", &l, &r, &h);
        --l; --r;
       if (l_ok() && r_ok())
        printf("No\n");
       else
        printf("Yes\n");
    }

    return 0;
}

================================================
======================== I =======================
#include <bits/stdc++.h>
using namespace std;
#define sc  second
#define fs  first
#define ll  long long
#define mp  make_pair

int n;
vector<pair<ll, ll> > v;

bool solve() {
    if (n & 1)
        return false;
    n /= 2;
    pair<ll, ll> p = mp(v[0].fs + v[n].fs, v[0].sc + v[n].sc);
    for (int i = 1; i < n; ++i) {
        if (v[i].fs + v[n + i].fs != p.fs || v[i].sc + v[i + n].sc != p.sc)
            return false;
    }
    return true;
}

int main() {
    ios_base::sync_with_stdio(false);
    cin >> n;
    v.resize(n);
    for (int i = 0; i < n; ++i)
        cin >> v[i].fs >> v[i].sc;
    cout << (solve() ? "yes" : "no") << endl;

    return 0;
}

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

struct Tree {
    int n, sumRoot, sum;
    string s;
};

bool flag[21][200][2000];
vector<Tree> trees[21];

void tryAdd(int n, int sumRoot, int sum, string s) {
    if (flag[n][sumRoot][sum])
```

```
        return;
    flag[n][sumRoot][sum] = 1;
    trees[n].push_back(Tree{n, sumRoot, sum, s});
}

int main() {
    int n, m;
    scanf("%d%d", &n, &m);

    trees[1] = vector<Tree>(1, Tree{1, 0, 0, ""});
    flag[1][0][0] = 1;
    for (int i = 2; i <= n; ++i) {
        for (const Tree& tree : trees[i-1])
            tryAdd(i,
                   tree.sumRoot + i - 1,
                   tree.sum + tree.sumRoot + i - 1,
                   "(" + tree.s + ")"
                  );
        for (int j = i-1; j > 0; --j)
            for (const Tree& treeBase: trees[j])
                for (const Tree& treeAdd: trees[i-j])
                    tryAdd(i,
                           treeBase.sumRoot + treeAdd.sumRoot + i-j,
                           treeBase.sum + treeAdd.sum +
                                treeBase.sumRoot * (i - j) + treeAdd.sumRoot * j +
                                j * (i-j),
                           treeBase.s + "(" + treeAdd.s + ")"
                          );
    }

    Tree tree;
    tree.n = -1;
    for (const Tree& itree: trees[n])
        if (itree.sum == m) {
            tree = itree;
            break;
        }
    if (tree.n < 0) {
        cout << "NO\n";
        return 0;
    }

    cout << "YES\n";
    stack<int> ST;
    ST.push(1);
    int curv = 1;
    for (char ch : tree.s)
        if (ch == '(') {
            cout << ST.top() << ' ' << ++curv << "\n";;
            ST.push(curv);
        }
        else
            ST.pop();

    return 0;
}
```