Олимпиада «Ломоносов» по информатике
2025-2026 учебный год. Заключительный этап
Работа участника с id заявки 1631274, логином inf26f_331

Сводный итог по всем задачам в проверяющей системе

```
RunID Time Username Prob Lang Result Tests Score
207 2:59:52 inf26f_331 6 g++ Partial solution 0 0
102 1:48:06 inf26f_331 5 g++ OK 103 100
95 1:41:14 inf26f_331 4 g++ OK 103 100
90 1:35:57 inf26f_331 3 g++ OK 23 100
53 1:12:27 inf26f_331 2 g++ Partial solution 4 2
49 1:03:30 inf26f_331 1 g++ Partial solution 16 65
367 (триста шестьдесят семь) технических баллов
61 (шестьдесят один) итоговый балл
```

председатель Жюри кандидат физ.-мат. наук Малышко В. В.

зам. председателя Жюри кандидат физ.-мат. наук Корныхин Е. В.

Посылка по задаче 1

```
[1]  #include <bits/stdc++.h>
[2]  #pragma optimize("Ofast")
[3]  #pragma optimize("O3")
[4]
[5]  using namespace std;
[6]  using ll = long long;
[7]  using si = short int;
[8]  using ld = long double;
[9]  const int INF = 1e9 + 7;
[10] const ll inf = 1e18 + 7;
[11] const ld eps = 1e-7;
[12]
[13] #ifdef LOCAL
[14] const int MAXA = 100, TREESZ = 1e3, MAXN = 1e3;
[15] #else
[16] const int MAXA = 1e7 + 1, TREESZ = 1e6, MAXN = 2e5;
[17] #endif
[18]
[19] // vector<int> pr, is_prime(MAXA, 1), min_div(MAXA, 1);
[20]
[21] // void build() {
[22] //     is_prime[0] = is_prime[1] = 0;
[23] //     for (int i = 2; i < MAXA; i++) {
[24] //         if (is_prime[i]) {
[25] //             pr.push_back(i);
[26] //             min_div[i] = i;
[27] //         }
[28] //         for (int p : pr) {
[29] //             if ((ll)p * i >= MAXA || min_div[i] < p)
[30] //                 break;
[31] //             is_prime[p * i] = 0;
[32] //             min_div[p * i] = p;
[33] //         }
[34] //     }
[35] // }
[36]
[37] // struct Node {
[38] //     ll val;
[39] //     Node(ll val = 0) : val(val) {}
[40] //     Node operator+(const Node &n) const {
[41] //         return val + n.val;
[42] //     }
[43] // };
[44]
[45] // struct SegTree {
[46] //     vector<Node> tree, add;
[47] //     vector<ll> arr;
[48] //     SegTree(int treesz = 0, int arrsz = 0) {
[49] //         tree.resize(treesz);
[50] //         arr.resize(arrsz);
[51] //     }
[52] //     void push(int i, int l, int r) {
[53] //         if (add[i].val == 0)
[54] //             return;
[55] //         int m = (l + r) >> 1;
[56] //         tree[2 * i + 1].val += (ll)(m - l) * add[i].val;
[57] //         tree[2 * i + 2].val += (ll)(r - m) * add[i].val;
[58] //         add[2 * i + 1] = add[2 * i + 2] = add[i];
[59] //         add[i] = {0};
[60] //     }
[61] //     void build(int i, int l, int r) {
[62] //         if (l + 1 == r) {
[63] //             tree[i].val = arr[l];
[64] //             return;
[65] //         }
[66] //         int m = (l + r) >> 1;
[67] //         build(2 * i + 1, l, m);
[68] //         build(2 * i + 2, m, r);
[69] //         tree[i] = tree[2 * i + 1] + tree[2 * i + 2];
[70] //     }
[71] //     void upd(int i, int l, int r, int ql, int qr, int val) {
[72] //         if (qr <= l || r <= ql)
[73] //             return;
[74] //         if (ql <= l && r <= qr) {
[75] //             tree[i].val += (ll)(r - l) * val;
[76] //             add[i].val += val;
[77] //             return;
[78] //         }
```

```
[79] //         push(i, l, r);
[80] //         int m = (l + r) >> 1;
[81] //         upd(2 * i + 1, l, m, ql, qr, val);
[82] //         upd(2 * i + 2, m, r, ql, qr, val);
[83] //         tree[i] = tree[2 * i + 1] + tree[2 * i + 2];
[84] //     }
[85] //     Node get(int i, int l, int r, int ql, int qr) {
[86] //         if (qr <= l || r <= ql)
[87] //             return Node();
[88] //         if (ql <= l && r <= qr)
[89] //             return tree[i];
[90] //         push(i, l, r);
[91] //         int m = (l + r) >> 1;
[92] //         return get(2 * i + 1, l, m, ql, qr) + get(2 * i + 2, m, r, ql, qr);
[93] //     }
[94] // };
[95]
[96] ll getmn(string s) {
[97]     int cnti = 0;
[98]     ll ans = 0;
[99]     for (char ch : s) {
[100]         if (ch == 'i')
[101]             cnti++;
[102]         else if (ch == 'v') {
[103]             if (cnti <= 5)
[104]                 ans += 5 - cnti;
[105]             else
[106]                 ans += cnti - 5;
[107]             cnti = 0;
[108]         } else {
[109]             if (cnti <= 10)
[110]                 ans += 10 - cnti;
[111]             else
[112]                 ans += cnti - 10;
[113]             cnti = 0;
[114]         }
[115]     }
[116]     ans += cnti;
[117]     return ans;
[118] }
[119]
[120] ll getmx(string s) {
[121]     int cnti = 0;
[122]     ll ans = 0;
[123]     for (char ch : s) {
[124]         if (ch == 'i')
[125]             cnti++;
[126]         else if (ch == 'v') {
[127]             ans += 5 - cnti;
[128]             cnti = 0;
[129]         } else {
[130]             ans += 10 - cnti;
[131]             cnti = 0;
[132]         }
[133]     }
[134]     ans += cnti;
[135]     return ans;
[136] }
[137]
[138] string mn(string s) {
[139]     if(s == "n") return "a";
[140]     string now = "";
[141]     string a = "";
[142]     for (char ch : s) {
[143]         if (ch == '.') {
[144]             now += 'a' + getmn(a);
[145]             a = "";
[146]         } else
[147]             a += ch;
[148]     }
[149]     now += 'a' + getmn(a);
[150]     return now;
[151] }
[152]
```

```
[153] string mx(string s) {
[154]     if(s == "n") return "a";
[155]     string now = "";
[156]     string a = "";
[157]     for (char ch : s) {
[158]         if (ch == '.') {
[159]             now += 'a' + getmx(a);
[160]             a = "";
[161]         } else
[162]             a += ch;
[163]     }
[164]     now += 'a' + getmx(a);
[165]     return now;
[166] }
[167]
[168] string ans(ll a) {
[169]     if (a == 0)
[170]         return "a";
[171]     string now;
[172]     while (a) {
[173]         now += 'a' + (a % 26);
[174]         a /= 26;
[175]     }
[176]     reverse(now.begin(), now.end());
[177]     return now;
[178] }
[179]
[180] int main() {
[181] #ifdef LOCAL
[182]     freopen("input.txt", "r", stdin);
[183]     freopen("output.txt", "w", stdout);
[184] #endif
[185]     ios::sync_with_stdio(false);
[186]     cin.tie(0);
[187]     cout.tie(0);
[188]     int k;
[189]     cin >> k;
[190]     string mnans = "zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz", mxans = "a";
[191]     while (k--) {
[192]         string s;
[193]         cin >> s;
[194]         mnans = min(mnans, mn(s));
[195]         mxans = max(mxans, mx(s));
[196]     }
[197]     cout << mxans << '\n'
[198]          << mnans;
[199]     return 0;
[200] }
```

Посылка по задаче 2

```
[1]  #include <bits/stdc++.h>
[2]  #pragma optimize("Ofast")
[3]  #pragma optimize("O3")
[4]
[5]  using namespace std;
[6]  using ll = long long;
[7]  using si = short int;
[8]  using ld = long double;
[9]  const int INF = 1e9 + 7;
[10] const ll inf = 1e18 + 7;
[11] const ld eps = 1e-7;
[12]
[13] #ifdef LOCAL
[14] const int MAXA = 100, TREESZ = 1e3, MAXN = 1e3;
[15] #else
[16] const int MAXA = 1e7 + 1, TREESZ = 1e6, MAXN = 2e5;
[17] #endif
[18]
[19] vector<int> pr, is_prime(MAXA, 1), min_div(MAXA, 1);
[20]
[21] void build() {
[22]     is_prime[0] = is_prime[1] = 0;
[23]     for (int i = 2; i < MAXA; i++) {
[24]         if (is_prime[i]) {
[25]             pr.push_back(i);
[26]             min_div[i] = i;
[27]         }
[28]         for (int p : pr) {
[29]             if ((ll)p * i >= MAXA || min_div[i] < p)
[30]                 break;
[31]             is_prime[p * i] = 0;
[32]             min_div[p * i] = p;
[33]         }
[34]     }
[35] }
[36]
[37] // struct Node {
[38] //     ll val;
[39] //     Node(ll val = 0) : val(val) {}
[40] //     Node operator+(const Node &n) const {
[41] //         return val + n.val;
[42] //     }
[43] // };
[44]
[45] // struct SegTree {
[46] //     vector<Node> tree, add;
[47] //     vector<ll> arr;
[48] //     SegTree(int treesz = 0, int arrsz = 0) {
[49] //         tree.resize(treesz);
[50] //         arr.resize(arrsz);
[51] //     }
[52] //     void push(int i, int l, int r) {
[53] //         if (add[i].val == 0)
[54] //             return;
[55] //         int m = (l + r) >> 1;
[56] //         tree[2 * i + 1].val += (ll)(m - l) * add[i].val;
[57] //         tree[2 * i + 2].val += (ll)(r - m) * add[i].val;
[58] //         add[2 * i + 1] = add[2 * i + 2] = add[i];
[59] //         add[i] = {0};
[60] //     }
[61] //     void build(int i, int l, int r) {
[62] //         if (l + 1 == r) {
[63] //             tree[i].val = arr[l];
[64] //             return;
[65] //         }
[66] //         int m = (l + r) >> 1;
[67] //         build(2 * i + 1, l, m);
[68] //         build(2 * i + 2, m, r);
[69] //         tree[i] = tree[2 * i + 1] + tree[2 * i + 2];
[70] //     }
[71] //     void upd(int i, int l, int r, int ql, int qr, int val) {
[72] //         if (qr <= l || r <= ql)
[73] //             return;
[74] //         if (ql <= l && r <= qr) {
[75] //             tree[i].val += (ll)(r - l) * val;
[76] //             add[i].val += val;
[77] //             return;
[78] //         }
```

```
[79] //         push(i, l, r);
[80] //         int m = (l + r) >> 1;
[81] //         upd(2 * i + 1, l, m, ql, qr, val);
[82] //         upd(2 * i + 2, m, r, ql, qr, val);
[83] //         tree[i] = tree[2 * i + 1] + tree[2 * i + 2];
[84] //     }
[85] //     Node get(int i, int l, int r, int ql, int qr) {
[86] //         if (qr <= l || r <= ql)
[87] //             return Node();
[88] //         if (ql <= l && r <= qr)
[89] //             return tree[i];
[90] //         push(i, l, r);
[91] //         int m = (l + r) >> 1;
[92] //         return get(2 * i + 1, l, m, ql, qr) + get(2 * i + 2, m, r, ql, qr);
[93] //     }
[94] // };
[95]
[96] ll check(ll a) {
[97]     for (ll i = 2; i * i < a; i++) {
[98]         if (i * i + i + 1 == a && is_prime[i])
[99]             return 1;
[100]    }
[101]    return 0;
[102] }
[103]
[104] int main() {
[105] #ifdef LOCAL
[106]    freopen("input.txt", "r", stdin);
[107]    freopen("output.txt", "w", stdout);
[108] #endif
[109]    ios::sync_with_stdio(false);
[110]    cin.tie(0);
[111]    cout.tie(0);
[112]    build();
[113]    int n;
[114]    cin >> n;
[115]    int cnt = 0;
[116]    vector<int> answer(MAXA);
[117]    while (n--) {
[118]        ll a;
[119]        cin >> a;
[120]        answer[a] = check(a);
[121]    }
[122]    for (int i : answer)
[123]        cnt += i;
[124]    cout << cnt;
[125]    return 0;
[126] }
```

Посылка по задаче 3

```
[1]  #include <bits/stdc++.h>
[2]  #pragma optimize("Ofast")
[3]  #pragma optimize("O3")
[4]
[5]  using namespace std;
[6]  using ll = long long;
[7]  using si = short int;
[8]  using ld = long double;
[9]  const int INF = 1e9 + 7;
[10] const ll inf = 1e18 + 7;
[11] const ld eps = 1e-7;
[12]
[13] #ifdef LOCAL
[14] const int MAXA = 100, TREESZ = 1e3, MAXN = 1e3;
[15] #else
[16] const int MAXA = 1e7 + 1, TREESZ = 1e6, MAXN = 2e5;
[17] #endif
[18]
[19] // vector<int> pr, is_prime(MAXA, 1), min_div(MAXA, 1);
[20]
[21] // void build() {
[22] //     is_prime[0] = is_prime[1] = 0;
[23] //     for (int i = 2; i < MAXA; i++) {
[24] //         if (is_prime[i]) {
[25] //             pr.push_back(i);
[26] //             min_div[i] = i;
[27] //         }
[28] //         for (int p : pr) {
[29] //             if ((ll)p * i >= MAXA || min_div[i] < p)
[30] //                 break;
[31] //             is_prime[p * i] = 0;
[32] //             min_div[p * i] = p;
[33] //         }
[34] //     }
[35] // }
[36]
[37] // struct Node {
[38] //     ll val;
[39] //     Node(ll val = 0) : val(val) {}
[40] //     Node operator+(const Node &n) const {
[41] //         return val + n.val;
[42] //     }
[43] // };
[44]
[45] // struct SegTree {
[46] //     vector<Node> tree, add;
[47] //     vector<ll> arr;
[48] //     SegTree(int treesz = 0, int arrsz = 0) {
[49] //         tree.resize(treesz);
[50] //         arr.resize(arrsz);
[51] //     }
[52] //     void push(int i, int l, int r) {
[53] //         if (add[i].val == 0)
[54] //             return;
[55] //         int m = (l + r) >> 1;
[56] //         tree[2 * i + 1].val += (ll)(m - l) * add[i].val;
[57] //         tree[2 * i + 2].val += (ll)(r - m) * add[i].val;
[58] //         add[2 * i + 1] = add[2 * i + 2] = add[i];
[59] //         add[i] = {0};
[60] //     }
[61] //     void build(int i, int l, int r) {
[62] //         if (l + 1 == r) {
[63] //             tree[i].val = arr[l];
[64] //             return;
[65] //         }
[66] //         int m = (l + r) >> 1;
[67] //         build(2 * i + 1, l, m);
[68] //         build(2 * i + 2, m, r);
[69] //         tree[i] = tree[2 * i + 1] + tree[2 * i + 2];
[70] //     }
[71] //     void upd(int i, int l, int r, int ql, int qr, int val) {
[72] //         if (qr <= l || r <= ql)
[73] //             return;
[74] //         if (ql <= l && r <= qr) {
[75] //             tree[i].val += (ll)(r - l) * val;
[76] //             add[i].val += val;
[77] //             return;
[78] //         }
```

```
[79] //         push(i, l, r);
[80] //         int m = (l + r) >> 1;
[81] //         upd(2 * i + 1, l, m, ql, qr, val);
[82] //         upd(2 * i + 2, m, r, ql, qr, val);
[83] //         tree[i] = tree[2 * i + 1] + tree[2 * i + 2];
[84] //     }
[85] //     Node get(int i, int l, int r, int ql, int qr) {
[86] //         if (qr <= l || r <= ql)
[87] //             return Node();
[88] //         if (ql <= l && r <= qr)
[89] //             return tree[i];
[90] //         push(i, l, r);
[91] //         int m = (l + r) >> 1;
[92] //         return get(2 * i + 1, l, m, ql, qr) + get(2 * i + 2, m, r, ql, qr);
[93] //     }
[94] // };
[95]
[96] int main() {
[97] #ifdef LOCAL
[98]     freopen("input.txt", "r", stdin);
[99]     freopen("output.txt", "w", stdout);
[100]#endif
[101]    ios::sync_with_stdio(false);
[102]    cin.tie(0);
[103]    cout.tie(0);
[104]    int n, i1, j1;
[105]    cin >> n >> i1 >> j1;
[106]    vector<vector<int>> a(n, vector<int>(n, -1));
[107]    int i = 0, j = 0, type = 0, ind = 0, cnt = INF;
[108]    while (cnt--) {
[109]        a[i][j] = ind++;
[110]        if (type == 0) {
[111]            if (j + 1 < n && a[i][j + 1] == -1)
[112]                j++;
[113]            else {
[114]                type = 1;
[115]                i++;
[116]            }
[117]        } else if (type == 1) {
[118]            if (i + 1 < n && a[i + 1][j] == -1)
[119]                i++;
[120]            else {
[121]                type = 2;
[122]                j--;
[123]            }
[124]        } else if (type == 2) {
[125]            if (j - 1 >= 0 && a[i][j - 1] == -1)
[126]                j--;
[127]            else {
[128]                type = 3;
[129]                i--;
[130]            }
[131]        } else if (type == 3) {
[132]            if (i - 1 >= 0 && a[i - 1][j] == -1)
[133]                i--;
[134]            else {
[135]                type = 0;
[136]                j++;
[137]            }
[138]        }
[139]        if (ind == n * n)
[140]            break;
[141]    }
[142]    cout << a[j1 - 1][i1 - 1];
[143]    return 0;
[144]}
```

Посылка по задаче 4

```
[1]  #include <bits/stdc++.h>
[2]  #pragma optimize("Ofast")
[3]  #pragma optimize("O3")
[4]
[5]  using namespace std;
[6]  using ll = long long;
[7]  using si = short int;
[8]  using ld = long double;
[9]  const int INF = 1e9 + 7;
[10] const ll inf = 1e18 + 7;
[11] const ld eps = 1e-7;
[12]
[13] #ifdef LOCAL
[14] const int MAXA = 100, TREESZ = 1e3, MAXN = 1e3;
[15] #else
[16] const int MAXA = 1e7 + 1, TREESZ = 1e6, MAXN = 2e5;
[17] #endif
[18]
[19] // vector<int> pr, is_prime(MAXA, 1), min_div(MAXA, 1);
[20]
[21] // void build() {
[22] //     is_prime[0] = is_prime[1] = 0;
[23] //     for (int i = 2; i < MAXA; i++) {
[24] //         if (is_prime[i]) {
[25] //             pr.push_back(i);
[26] //             min_div[i] = i;
[27] //         }
[28] //         for (int p : pr) {
[29] //             if ((ll)p * i >= MAXA || min_div[i] < p)
[30] //                 break;
[31] //             is_prime[p * i] = 0;
[32] //             min_div[p * i] = p;
[33] //         }
[34] //     }
[35] // }
[36]
[37] // struct Node {
[38] //     ll val;
[39] //     Node(ll val = 0) : val(val) {}
[40] //     Node operator+(const Node &n) const {
[41] //         return val + n.val;
[42] //     }
[43] // };
[44]
[45] // struct SegTree {
[46] //     vector<Node> tree, add;
[47] //     vector<ll> arr;
[48] //     SegTree(int treesz = 0, int arrsz = 0) {
[49] //         tree.resize(treesz);
[50] //         arr.resize(arrsz);
[51] //     }
[52] //     void push(int i, int l, int r) {
[53] //         if (add[i].val == 0)
[54] //             return;
[55] //         int m = (l + r) >> 1;
[56] //         tree[2 * i + 1].val += (ll)(m - l) * add[i].val;
[57] //         tree[2 * i + 2].val += (ll)(r - m) * add[i].val;
[58] //         add[2 * i + 1] = add[2 * i + 2] = add[i];
[59] //         add[i] = {0};
[60] //     }
[61] //     void build(int i, int l, int r) {
[62] //         if (l + 1 == r) {
[63] //             tree[i].val = arr[l];
[64] //             return;
[65] //         }
[66] //         int m = (l + r) >> 1;
[67] //         build(2 * i + 1, l, m);
[68] //         build(2 * i + 2, m, r);
[69] //         tree[i] = tree[2 * i + 1] + tree[2 * i + 2];
[70] //     }
[71] //     void upd(int i, int l, int r, int ql, int qr, int val) {
[72] //         if (qr <= l || r <= ql)
[73] //             return;
[74] //         if (ql <= l && r <= qr) {
[75] //             tree[i].val += (ll)(r - l) * val;
[76] //             add[i].val += val;
[77] //             return;
[78] //         }
```

```
[79] //         push(i, l, r);
[80] //         int m = (l + r) >> 1;
[81] //         upd(2 * i + 1, l, m, ql, qr, val);
[82] //         upd(2 * i + 2, m, r, ql, qr, val);
[83] //         tree[i] = tree[2 * i + 1] + tree[2 * i + 2];
[84] //     }
[85] //     Node get(int i, int l, int r, int ql, int qr) {
[86] //         if (qr <= l || r <= ql)
[87] //             return Node();
[88] //         if (ql <= l && r <= qr)
[89] //             return tree[i];
[90] //         push(i, l, r);
[91] //         int m = (l + r) >> 1;
[92] //         return get(2 * i + 1, l, m, ql, qr) + get(2 * i + 2, m, r, ql, qr);
[93] //     }
[94] // };
[95]
[96] int main() {
[97] #ifdef LOCAL
[98]     freopen("input.txt", "r", stdin);
[99]     freopen("output.txt", "w", stdout);
[100] #endif
[101]     ios::sync_with_stdio(false);
[102]     cin.tie(0);
[103]     cout.tie(0);
[104]     int n, opt = 0;
[105]     cin >> n;
[106]     vector<int> a(n);
[107]     for(int & i : a) cin >> i;
[108]     vector<int> dp(n, INF);
[109]     for(int i = 0; i < n; i++){
[110]         auto it = upper_bound(dp.begin(), dp.end(), a[i]) - dp.begin();
[111]         dp[it] = a[i];
[112]     }
[113]     for(int i = 0; i < n; i++){
[114]         if(dp[i] != INF) opt = i + 1;
[115]         else break;
[116]     }
[117]     cout << n - opt;
[118]     return 0;
[119] }
```

Посылка по задаче 5

```
[1]  #include <bits/stdc++.h>
[2]  #pragma optimize("Ofast")
[3]  #pragma optimize("O3")
[4]
[5]  using namespace std;
[6]  using ll = long long;
[7]  using si = short int;
[8]  using ld = long double;
[9]  const int INF = 1e9 + 7;
[10] const ll inf = 1e18 + 7;
[11] const ld eps = 1e-7;
[12]
[13] #ifdef LOCAL
[14] const int MAXA = 100, TREESZ = 1e3, MAXN = 1e3;
[15] #else
[16] const int MAXA = 1e7 + 1, TREESZ = 1e6, MAXN = 2e5;
[17] #endif
[18]
[19] // vector<int> pr, is_prime(MAXA, 1), min_div(MAXA, 1);
[20]
[21] // void build() {
[22] //     is_prime[0] = is_prime[1] = 0;
[23] //     for (int i = 2; i < MAXA; i++) {
[24] //         if (is_prime[i]) {
[25] //             pr.push_back(i);
[26] //             min_div[i] = i;
[27] //         }
[28] //         for (int p : pr) {
[29] //             if ((ll)p * i >= MAXA || min_div[i] < p)
[30] //                 break;
[31] //             is_prime[p * i] = 0;
[32] //             min_div[p * i] = p;
[33] //         }
[34] //     }
[35] // }
[36]
[37] // struct Node {
[38] //     ll val;
[39] //     Node(ll val = 0) : val(val) {}
[40] //     Node operator+(const Node &n) const {
[41] //         return val + n.val;
[42] //     }
[43] // };
[44]
[45] // struct SegTree {
[46] //     vector<Node> tree, add;
[47] //     vector<ll> arr;
[48] //     SegTree(int treesz = 0, int arrsz = 0) {
[49] //         tree.resize(treesz);
[50] //         arr.resize(arrsz);
[51] //     }
[52] //     void push(int i, int l, int r) {
[53] //         if (add[i].val == 0)
[54] //             return;
[55] //         int m = (l + r) >> 1;
[56] //         tree[2 * i + 1].val += (ll)(m - l) * add[i].val;
[57] //         tree[2 * i + 2].val += (ll)(r - m) * add[i].val;
[58] //         add[2 * i + 1] = add[2 * i + 2] = add[i];
[59] //         add[i] = {0};
[60] //     }
[61] //     void build(int i, int l, int r) {
[62] //         if (l + 1 == r) {
[63] //             tree[i].val = arr[l];
[64] //             return;
[65] //         }
[66] //         int m = (l + r) >> 1;
[67] //         build(2 * i + 1, l, m);
[68] //         build(2 * i + 2, m, r);
[69] //         tree[i] = tree[2 * i + 1] + tree[2 * i + 2];
[70] //     }
[71] //     void upd(int i, int l, int r, int ql, int qr, int val) {
[72] //         if (qr <= l || r <= ql)
[73] //             return;
[74] //         if (ql <= l && r <= qr) {
[75] //             tree[i].val += (ll)(r - l) * val;
[76] //             add[i].val += val;
[77] //             return;
[78] //         }
```

```
[79] //         push(i, l, r);
[80] //         int m = (l + r) >> 1;
[81] //         upd(2 * i + 1, l, m, ql, qr, val);
[82] //         upd(2 * i + 2, m, r, ql, qr, val);
[83] //         tree[i] = tree[2 * i + 1] + tree[2 * i + 2];
[84] //     }
[85] //     Node get(int i, int l, int r, int ql, int qr) {
[86] //         if (qr <= l || r <= ql)
[87] //             return Node();
[88] //         if (ql <= l && r <= qr)
[89] //             return tree[i];
[90] //         push(i, l, r);
[91] //         int m = (l + r) >> 1;
[92] //         return get(2 * i + 1, l, m, ql, qr) + get(2 * i + 2, m, r, ql, qr);
[93] //     }
[94] // };
[95]
[96] int main() {
[97] #ifdef LOCAL
[98]     freopen("input.txt", "r", stdin);
[99]     freopen("output.txt", "w", stdout);
[100]#endif
[101]    ios::sync_with_stdio(false);
[102]    cin.tie(0);
[103]    cout.tie(0);
[104]    int n, opt = 0;
[105]    cin >> n;
[106]    vector<tuple<int, int, int>> a(n);
[107]    tuple<int, int, int> t1;
[108]    for (int i = 0; i < n; i++) {
[109]        int a1, a2, a3;
[110]        cin >> a1 >> a2 >> a3;
[111]        a[i] = {a1, a2, a3};
[112]    }
[113]    t1 = a[0];
[114]    sort(a.begin(), a.end());
[115]    vector<int> dp(n, 0);
[116]    for (int i = 0; i < n; i++)
[117]        if (a[i] == t1)
[118]            dp[i] = 1;
[119]    for (int i = 0; i < n; i++) {
[120]        auto [a1, a2, a3] = a[i];
[121]        if (!dp[i])
[122]            continue;
[123]        for (int j = i + 1; j < n; j++) {
[124]            auto [b1, b2, b3] = a[j];
[125]            if (b1 > a1 && b2 > a2 && b3 < a3)
[126]                dp[j] = max(dp[j], dp[i] + 1);
[127]        }
[128]    }
[129]    int ans = 1;
[130]    for(int i : dp) ans = max(ans, i);
[131]    cout << ans;
[132]    return 0;
[133]}
```

Посылка по задаче 6

```
[1]  #include <bits/stdc++.h>
[2]  #pragma optimize("Ofast")
[3]  #pragma optimize("O3")
[4]
[5]  using namespace std;
[6]  using ll = long long;
[7]  using si = short int;
[8]  using ld = long double;
[9]  const int INF = 1e9 + 7;
[10] const ll inf = 1e18 + 7;
[11] const ld eps = 1e-7;
[12]
[13] #ifdef LOCAL
[14] const int MAXA = 100, TREESZ = 1e3, MAXN = 1e3;
[15] #else
[16] const int MAXA = 1e7 + 1, TREESZ = 1e6, MAXN = 2e5;
[17] #endif
[18]
[19] // vector<int> pr, is_prime(MAXA, 1), min_div(MAXA, 1);
[20]
[21] // void build() {
[22] //     is_prime[0] = is_prime[1] = 0;
[23] //     for (int i = 2; i < MAXA; i++) {
[24] //         if (is_prime[i]) {
[25] //             pr.push_back(i);
[26] //             min_div[i] = i;
[27] //         }
[28] //         for (int p : pr) {
[29] //             if ((ll)p * i >= MAXA || min_div[i] < p)
[30] //                 break;
[31] //             is_prime[p * i] = 0;
[32] //             min_div[p * i] = p;
[33] //         }
[34] //     }
[35] // }
[36]
[37] // struct Node {
[38] //     ll val;
[39] //     Node(ll val = 0) : val(val) {}
[40] //     Node operator+(const Node &n) const {
[41] //         return val + n.val;
[42] //     }
[43] // };
[44]
[45] // struct SegTree {
[46] //     vector<Node> tree, add;
[47] //     vector<ll> arr;
[48] //     SegTree(int treesz = 0, int arrsz = 0) {
[49] //         tree.resize(treesz);
[50] //         arr.resize(arrsz);
[51] //     }
[52] //     void push(int i, int l, int r) {
[53] //         if (add[i].val == 0)
[54] //             return;
[55] //         int m = (l + r) >> 1;
[56] //         tree[2 * i + 1].val += (ll)(m - l) * add[i].val;
[57] //         tree[2 * i + 2].val += (ll)(r - m) * add[i].val;
[58] //         add[2 * i + 1] = add[2 * i + 2] = add[i];
[59] //         add[i] = {0};
[60] //     }
[61] //     void build(int i, int l, int r) {
[62] //         if (l + 1 == r) {
[63] //             tree[i].val = arr[l];
[64] //             return;
[65] //         }
[66] //         int m = (l + r) >> 1;
[67] //         build(2 * i + 1, l, m);
[68] //         build(2 * i + 2, m, r);
[69] //         tree[i] = tree[2 * i + 1] + tree[2 * i + 2];
[70] //     }
[71] //     void upd(int i, int l, int r, int ql, int qr, int val) {
[72] //         if (qr <= l || r <= ql)
[73] //             return;
[74] //         if (ql <= l && r <= qr) {
[75] //             tree[i].val += (ll)(r - l) * val;
[76] //             add[i].val += val;
[77] //             return;
[78] //         }
[79] //         push(i, l, r);
[80] //         int m = (l + r) >> 1;
```

```
[81] //         upd(2 * i + 1, l, m, ql, qr, val);
[82] //         upd(2 * i + 2, m, r, ql, qr, val);
[83] //         tree[i] = tree[2 * i + 1] + tree[2 * i + 2];
[84] //     }
[85] //     Node get(int i, int l, int r, int ql, int qr) {
[86] //         if (qr <= l || r <= ql)
[87] //             return Node();
[88] //         if (ql <= l && r <= qr)
[89] //             return tree[i];
[90] //         push(i, l, r);
[91] //         int m = (l + r) >> 1;
[92] //         return get(2 * i + 1, l, m, ql, qr) + get(2 * i + 2, m, r, ql, qr);
[93] //     }
[94] // };
[95]
[96] vector<pair<int, int>> p;
[97] ll ans = 0;
[98] int n1, n2, n3, n4, r, c;
[99] int _n1 = 0, _n2 = 0, _n3 = 0, _n4 = 0;
[100] // int i1 = 0, i2 = 0, i3 = 0, i4 = 0;
[101] vector<vector<int>> cur, now;
[102]
[103] bool check() {
[104]     for (int i = 0; i < r; i++) {
[105]         for (int j = 0; j < c; j++) {
[106]             if (min(1, now[i][j]) != cur[i][j])
[107]                 return 0;
[108]         }
[109]     }
[110]     return 1;
[111] }
[112]
[113] void dfs() {
[114]     if (_n1 == n1 && _n2 == n2 && _n3 == n3 && _n4 == n4) {
[115]         ans += check();
[116]         return;
[117]     }
[118]     if (_n1 < n1) {
[119]         for (auto [i, j] : p) {
[120]             now[i][j]++;
[121]             _n1++;
[122]             dfs();
[123]             _n1--;
[124]             now[i][j]--;
[125]         }
[126]     }
[127]     if (_n2 < n2) {
[128]         for (auto [i, j] : p) {
[129]             if (i + 1 < r) {
[130]                 now[i][j]++;
[131]                 now[i + 1][j]++;
[132]                 _n2++;
[133]                 dfs();
[134]                 now[i][j]--;
[135]                 now[i + 1][j]--;
[136]                 _n2--;
[137]             }
[138]             if (j + 1 < c) {
[139]                 now[i][j]++;
[140]                 now[i][j + 1]++;
[141]                 _n2++;
[142]                 dfs();
[143]                 now[i][j]--;
[144]                 now[i][j + 1]--;
[145]                 _n2--;
[146]             }
[147]         }
[148]     }
[149]     if (_n3 < n3) {
[150]         for (auto [i, j] : p) {
[151]             if (i + 2 < r) {
[152]                 now[i][j]++;
[153]                 now[i + 1][j]++;
[154]                 now[i + 2][j]++;
[155]                 _n3++;
[156]                 dfs();
[157]                 now[i][j]--;
[158]                 now[i + 1][j]--;
[159]                 now[i + 2][j]--;
[160]                 _n3--;
```

```
[161]            }
[162]            if (j + 2 < c) {
[163]                now[i][j]++;
[164]                now[i][j + 1]++;
[165]                now[i][j + 2]++;
[166]                _n3++;
[167]                dfs();
[168]                now[i][j]--;
[169]                now[i][j + 1]--;
[170]                now[i][j + 2]--;
[171]                _n3--;
[172]            }
[173]        }
[174]    }
[175]    if (_n4 < n4) {
[176]        for (auto [i, j] : p) {
[177]            if (i + 3 < r) {
[178]                now[i][j]++;
[179]                now[i + 1][j]++;
[180]                now[i + 2][j]++;
[181]                now[i + 3][j]++;
[182]                _n4++;
[183]                dfs();
[184]                now[i][j]--;
[185]                now[i + 1][j]--;
[186]                now[i + 2][j]--;
[187]                now[i + 3][j]--;
[188]                _n4--;
[189]            }
[190]            if (j + 2 < c) {
[191]                now[i][j]++;
[192]                now[i][j + 1]++;
[193]                now[i][j + 2]++;
[194]                now[i][j + 3]++;
[195]                _n4++;
[196]                dfs();
[197]                now[i][j]--;
[198]                now[i][j + 1]--;
[199]                now[i][j + 2]--;
[200]                now[i][j + 3]--;
[201]                _n4--;
[202]            }
[203]        }
[204]    }
[205]}
[206]
[207]int main() {
[208]#ifdef LOCAL
[209]    freopen("input.txt", "r", stdin);
[210]    freopen("output.txt", "w", stdout);
[211]#endif
[212]    ios::sync_with_stdio(false);
[213]    cin.tie(0);
[214]    cout.tie(0);
[215]    cin >> n1 >> n2 >> n3 >> n4 >> r >> c;
[216]    cur.resize(r, vector<int>(c));
[217]    now.resize(r, vector<int>(c));
[218]    for (int i = 0; i < r; i++) {
[219]        for (int j = 0; j < c; j++) {
[220]            char ch;
[221]            cin >> ch;
[222]            cur[i][j] = (ch == '#');
[223]            if (cur[i][j])
[224]                p.push_back({i, j});
[225]        }
[226]    }
[227]    dfs();
[228]    ll fact = 1;
[229]    vector<ll> p(20);
[230]    for(int i = 1; i <= n1 + n2 + n3 + n4; i++){
[231]        fact *= i;
[232]        p[i] = fact;
[233]        }
[234]    cout << ans / p[n1] / p[n2] / p[n3] / p[n4];
[235]    return 0;
[236]}
```