Državno tekmovanje iz programiranja 2020

Rešitve

Preprosto zaporedje

```
#include <iostream>
#include <cstdio>
#include <cmath>

using namespace std;

long long solve(long long n) {
    long long res = 0;

    long long x = 1;
    for (; x * x <= n; ++x) {
        res += (x * x - (x - 1) * (x - 1)) * x;
    }
    res += (n - (x - 1) * (x - 1)) * x;

    long long y = 1;
    for (; y * y * y <= n; ++y) {
        res += (y * y * y - (y - 1) * (y - 1) * (y - 1)) * (y - 1);
        ++res;
    }
    res += (n - (y - 1) * (y - 1) * (y - 1)) * (y - 1);

    return res;
}

int main() {

    long long N;
    cin >> N;
    cout << solve(N) << endl;

    return 0;
}
```

Kvadrati

```
#include <math.h>
#include <stdio.h>
#define MAX(x, y) ( ((x) > (y)) ? (x) : (y) )

long stKvadratov(long a, long b);
long stKvadratov_neucinkovito(long a, long b);

int main() {
    long stPloscic;
    scanf("%ld", &stPloscic);
    long koren = (long) ceil(sqrt((double) stPloscic));
    long a;

    long najStKvadratov = 0L;
    for (a = 1;  a <= koren;  a++) {
        long b = stPloscic / a;
        najStKvadratov = MAX(najStKvadratov, stKvadratov(a, b));
    }
    printf("%ld\n", najStKvadratov);
    return 0;
}

long stKvadratov(long a, long b) {
    if (a > b) {
        long t = a;
        a = b;
        b = t;
    }
    return a * (a + 1) * (3 * b - a + 1) / 6;
}

long stKvadratov_neucinkovito(long a, long b) {
    if (a > b) {
        long t = a;
        a = b;
        b = t;
    }
    long i;
    long stKvadratov = 0L;
    for (i = 1L;  i <= a;  i++) {
        stKvadratov += (a - i + 1) * (b - i + 1);
    }
    return stKvadratov;
}
```

H-index

```
#include <stdio.h>
#include <stdlib.h>

int* g_arr;

int prim(const void* pa, const void* pb) {
    int a = *((int *) pa);
    int b = *((int *) pb);
    return (b - a);
}

int main() {
    int n;
    scanf("%d", &n);
    int* citiranost = malloc(n * sizeof(int));
    for (int i = 0;  i < n;  i++) {
        scanf("%d", &citiranost[i]);
    }
    qsort(citiranost, n, sizeof(int), prim);

    int zadnji = 0;
    for (int i = 0;  i < n;  i++) {
        if (i + 1 > citiranost[i]) {
            break;
        }
        zadnji = i + 1;
    }
    printf("%d\n", zadnji);
    return 0;
}
```

Šifrirano sporočilo

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define M 1000010

int main() {
    char* sporocilo = malloc(M * sizeof(char));
    scanf("%s", sporocilo);
    int dolzinaPodniza;
    scanf("%d", &dolzinaPodniza);

    int dolzinaSporocila = strlen(sporocilo);
    int dolzinaTabele = 1 << dolzinaPodniza;
    int* frekvenca = calloc(dolzinaTabele, sizeof(int));

    int vrednost = 0;
    for (int i = 0;  i < dolzinaSporocila;  i++) {
        if (i >= dolzinaPodniza) {
            vrednost -= (sporocilo[i - dolzinaPodniza] - 'A') << (dolzinaPodniza - 1);
        }
        vrednost <<= 1;
        vrednost += sporocilo[i] - 'A';
        if (i >= dolzinaPodniza - 1) {
            frekvenca[vrednost]++;
        }
    }

    int vMax = 0;
    for (int v = 0;  v < dolzinaTabele;  v++) {
        if (frekvenca[v] > frekvenca[vMax]) {
            vMax = v;
        }
    }

    char* cilj = malloc((dolzinaPodniza + 1) * sizeof(char));
    cilj[dolzinaPodniza] = '\0';
    int i = dolzinaPodniza - 1;
    while (i >= 0) {
        cilj[i--] = (vMax % 2) + 'A';
        vMax /= 2;
    }

    printf("%s\n", cilj);
    free(cilj);

    free(frekvenca);
    free(sporocilo);
    return 0;
}
```

Rektanglija

```
#include <stdio.h>

#define D 1000
#define MAX(x, y) (((x) > (y)) ? (x) : (y))
#define MIN(x, y) (((x) < (y)) ? (x) : (y))

int F[D + 2][D + 2];

int main() {
    int stObcin;
    scanf("%d", &stObcin);
    int xMax = 0;
    int yMax = 0;

    for (int i = 0;  i < stObcin;  i++) {
        int x1, y1, x2, y2;
        scanf("%d%d%d%d", &x1, &y1, &x2, &y2);
        F[x1][y1]++;
        F[x1][y2 + 1]--;
        F[x2 + 1][y1]--;
        F[x2 + 1][y2 + 1]++;
        xMax = MAX(x2 + 1, xMax);
        yMax = MAX(y2 + 1, yMax);
    }

    for (int x = 0;  x <= xMax;  x++) {
        for (int y = 1;  y <= yMax;  y++) {
            F[x][y] += F[x][y - 1];
        }
    }

    for (int y = 0;  y <= yMax;  y++) {
        for (int x = 1;  x <= xMax;  x++) {
            F[x][y] += F[x - 1][y];
        }
    }

    int stKrajev;
    scanf("%d", &stKrajev);

    for (int i = 0;  i < stKrajev;  i++) {
        int x, y;
        scanf("%d%d", &x, &y);
        printf("%d\n", F[x][y]);
    }
    return 0;
}
```

Popotnik

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

using namespace std;

typedef vector<int> vi;
typedef vector<vi> t_graph;

vi st, in_st, in_rn, visited;
vector<vi> rings;

void go(int v, int p, const t_graph& g) {
    in_st[v] = true;
    st.push_back(v);
    visited[v] = true;
    for (size_t i = 0; i < g[v].size(); ++i) {
        int u = g[v][i];
        if (u == p) continue;
        if (in_st[u]) {
            vi ring(1, u);
            while (true) {
                ring.push_back(st.back());
                in_st[st.back()] = false;
                in_rn[st.back()] = rings.size();
                if (st.back() == u)
                    break;
                st.pop_back();
            }
            rings.push_back(ring);
            st.pop_back();
            continue;
        }
        if (visited[u]) continue;
        go(u, v, g);
    }
    if (in_st[v]) {
        rings.push_back(vi(1, v));
        in_st[v] = false;
        in_rn[v] = rings.size() - 1;
        st.pop_back();
    }
}

int solve(int s, const t_graph& g) {
    if (g.size() == 1)
        return 1;
    visited.resize(g.size());
    in_st.resize(g.size());
    in_rn.resize(g.size());
    go(s, -1, g);
    int res = 0;
    for (size_t i = 0; i < rings.size(); ++i) {
        int v = rings[i][0];
        if (rings[i].size() == 1) {
            if (g[v].size() == 1 && v != s) {
                ++res;
            }
        } else {
            int p1 = rings[i][1];
            int p2 = rings[i][rings[i].size() - 2];
            if (g[p1].size() == 2)
                ++res;
            if (g[p2].size() == 2)
                ++res;
        }
    }
    return res;
}

int main() {
    int n, m, s;
    t_graph g;
    cin >> n >> m >> s;
    g.resize(n);
    for (int i = 0; i < m; ++i) {
        int x, y; cin >> x >> y;
        --x; --y;
        g[x].push_back(y);
        g[y].push_back(x);
    }
    cout << solve(s - 1, g) << endl;
    return 0;
}
```

Stolpi

```
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>

#define MAX(x, y) (((x) > (y)) ? (x) : (y))
#define MIN(x, y) (((x) < (y)) ? (x) : (y))
#define ABS(x) (((x) > 0) ? (x) : (-(x)))

#define MEJA 3000

int MEMO[MEJA + 1][MEJA + 1];

int poisci(int* visine, int n) {
    int zg = INT_MIN;
    int sp = INT_MAX;
    for (int i = 0;  i < n;  i++) {
        sp = MIN(sp, visine[i]);
        zg = MAX(zg, visine[i]);
    }

    for (int j = sp;  j <= zg;  j++) {
        MEMO[0][j] = ABS(visine[0] - j);
    }

    for (int i = 1;  i < n;  i++) {
        int minimum = INT_MAX;
        for (int j = sp;  j <= zg;  j++) {
            minimum = MIN(minimum, MEMO[i - 1][j]);
            MEMO[i][j] = minimum + ABS(visine[i] - j);
        }
    }

    int rezultat = INT_MAX;
    for (int j = sp;  j <= zg;  j++) {
        rezultat = MIN(rezultat, MEMO[n - 1][j]);
    }

    return rezultat;
}

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

    int* visine = malloc(n * sizeof(int));
    for (int i = 0;  i < n;  i++) {
        scanf("%d", &visine[i]);
    }

    printf("%d\n", poisci(visine, n));
    free(visine);

    return 0;
}
```

Kriminalist (1. del)

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>

#define MV (26 * 26)

char BUF[10000];
char BUF1[10];
char BUF2[10];
char BUF3[10];

typedef struct _Drevo {
    int oznaka;
    struct _Drevo* levo;
    struct _Drevo* desno;
} Drevo;

Drevo* ustvari(int oznaka, Drevo* levo, Drevo* desno) {
    Drevo* drevo = malloc(sizeof(Drevo));
    drevo->oznaka = oznaka;
    drevo->levo = levo;
    drevo->desno = desno;
    return drevo;
}

void pocisti(Drevo* drevo) {
    if (drevo != NULL) {
        pocisti(drevo->levo);
        pocisti(drevo->desno);
        free(drevo);
    }
}

char* i2o(int oznaka, char* buf) {
    buf[0] = oznaka / 26 + 'A';
    buf[1] = oznaka % 26 + 'A';
    buf[2] = '\0';
    return buf;
}

int o2i(char* oznaka) {
    return 26 * (oznaka[0] - 'A') + (oznaka[1] - 'A');
}

int steviloOtrok(Drevo* drevo) {
    int n = 0;
    if (drevo->levo != NULL) {
        n++;
    }
    if (drevo->desno != NULL) {
        n++;
    }
    return n;
}

void izpisi0(Drevo* drevo) {
    if (drevo != NULL) {
        printf("%s", i2o(drevo->oznaka, BUF1));
        int otr = steviloOtrok(drevo);
        if (otr > 0) {
            printf("[");
            if (otr == 1) {
                izpisi0(drevo->levo);
            } else {
                izpisi0(drevo->levo);
                printf(", ");
                izpisi0(drevo->desno);
            }
            printf("]");
        }
    }
}

void izpisi(Drevo* drevo) {
    izpisi0(drevo);
    printf("\n");
}
```

Kriminalist (2. del)

```
bool izomorfni(Drevo* prvo, Drevo* drugo) {
    if (drugo == NULL) {
        return false;
    }

    int otr = steviloOtrok(prvo);
    if (otr != steviloOtrok(drugo)) {
        return false;
    }

    if (otr == 0) {
        return true;
    }

    if (otr == 1) {
        return izomorfni(prvo->levo, drugo->levo);
    }

    return (izomorfni(prvo->levo, drugo->levo) && izomorfni(prvo->desno, drugo->desno)) ||
           (izomorfni(prvo->levo, drugo->desno) && izomorfni(prvo->desno, drugo->levo));
}

Drevo* zgradiDrevo(int nasledniki[][2], int koren) {
    if (koren < 0) {
        return NULL;
    }
    if (nasledniki[koren][0] < 0) {
        return ustvari(koren, NULL, NULL);
    }
    if (nasledniki[koren][1] < 0) {
        return ustvari(koren, zgradiDrevo(nasledniki, nasledniki[koren][0]), NULL);
    }
    return ustvari(koren, zgradiDrevo(nasledniki, nasledniki[koren][0]), zgradiDrevo(nasledniki, nasledniki[koren][1]));
}

Drevo* izdelajDrevo(char* opis) {
    int nasledniki[MV][2];
    bool vozlisca[MV];

    for (int i = 0;  i < MV;  i++) {
        nasledniki[i][0] = -1;
        nasledniki[i][1] = -1;
        vozlisca[i] = false;
    }

    int stPovezav = strlen(opis) / 4;
    for (int i = 0;  i < stPovezav;  i++) {
        int izvor = o2i(opis + 4 * i);
        int cilj = o2i(opis + 4 * i + 2);
        if (nasledniki[izvor][0] < 0) {
            nasledniki[izvor][0] = cilj;
        } else {
            nasledniki[izvor][1] = cilj;
        }
        vozlisca[izvor] = true;
        vozlisca[cilj] = true;
    }

    for (int i = 0;  i < MV;  i++) {
        if (nasledniki[i][0] >= 0) {
            vozlisca[nasledniki[i][0]] = false;
        }
        if (nasledniki[i][1] >= 0) {
            vozlisca[nasledniki[i][1]] = false;
        }
    }

    int koren = -1;
    for (int i = 0;  i < MV;  i++) {
        if (vozlisca[i]) {
            koren = i;
            break;
        }
    }

    return zgradiDrevo(nasledniki, koren);
}

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

    for (int i = 0;  i < stPrimerov;  i++) {
        scanf("%s", BUF);
        Drevo* prvo = izdelajDrevo(BUF);
        scanf("%s", BUF);
        Drevo* drugo = izdelajDrevo(BUF);

        //izpisi(prvo);
        //izpisi(drugo);
        printf("%d\n", izomorfni(prvo, drugo));
        pocisti(prvo);
        pocisti(drugo);
    }

    return 0;
}
```

Omara

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

using namespace std;

typedef vector<int> vi;

struct info {
    info(int pos = 0, int d = 0, long long best = 0) : pos(pos), d(d), best(best) {}
    int pos, d;
    long long best;
};

int gcd(int a, int b) {
    if (!b) return a; return gcd(b, a % b);
}

vector<long long> S;
void set_val(int pos, long long val) {
    while (pos > 0) {
        S[pos] = max(S[pos], val);
        pos &= pos - 1;
    }
}

long long get_max(int l, int r) {
    if (l > r)
        return -(1LL << 60);
    if (r == 0) return 0;
    if (l == 0) l = 1;
    long long res = S[l];
    while (l < S.size()) {
        res = max(res, S[l]);
        l = l * 2 - (l & (l - 1));
    }
    return res;
}

long long solve(vi a, int d) {
    a.insert(a.begin(), 0);
    if (d == 1) {
        long long res = 0;
        for (int i = 0; i < a.size(); ++i)
            res += a[i];
        return res;
    }
    vector<info> borders(1, info(0, 0, 0));
    vector<long long> ans(a.size(), -(1LL << 60)); ans[0] = 0;
    S.assign(a.size(), -(1LL << 60));
    for (int i = 1; i < a.size(); ++i) {
        for (int j = 1; j < borders.size(); ++j)
            borders[j].d = gcd(borders[j].d, a[i]);
        int k = 0;
        for (int j = 1; j < borders.size(); ++j) {
            if (borders[k].d == borders[j].d) {
                borders[k].best = max(borders[k].best, borders[j].best);
            } else {
                borders[++k] = borders[j];
            }
        }
        borders.resize(k + 1);
        if (i >= d) {
            set_val(i - d, ans[i - d]);
            for (int j = 0; j < borders.size(); ++j) {
                if (j + 1 < borders.size() && i - d + 1 >= borders[j + 1].pos) {
                    if (borders[j + 1].pos - borders[j].pos > 1)
                        ans[i] = max(ans[i], borders[j].best + borders[j].d);
                    ans[i] = max(ans[i], ans[borders[j + 1].pos - 1] + borders[j + 1].d);
                } else {
                    long long mx = get_max(borders[j].pos, i - d);
                    ans[i] = max(ans[i], mx + borders[j].d);
                    break;
                }
            }
        }
        borders.push_back(info(i, a[i], ans[i]));
    }
    return ans.back();
}

int main()
{
    int n, m, d;
    cin >> n >> m >> d;
    vi a;
    for (int i = 0; i < n; ++i) {
        int k;
        cin >> k;
        for (int j = 0; j < k; ++j) {
            int x; cin >> x; a.push_back(x);
        }
    }
    cout << solve(a, d) << endl;

    return 0;
}
```