Олимпиада СПбГУ
по информатике
2023/24 учебного года

24fin0043 – Безбородов Руслан Владиславович

A	B	C	D	E	F	sum
100	100	100	35	.	53	388

Task A

```
#include <stdint.h>

#include <array>
#include <iostream>

#define N_MAX 80

std::array<uint64_t, N_MAX> dp;

uint64_t getWaysCount(uint32_t a, uint32_t b, uint32_t n) {
    std::fill(dp.begin(), dp.end(), 0);

    dp[a]++;
    dp[b]++;

    for (uint32_t i = a; i <= n; i++) {
        dp[i] += dp[i - a] + (i >= b ? dp[i - b] : 0);
    }

    return dp[n];
}

int main() {
    uint32_t a, b, n;
    std::cin >> a >> b >> n;

    if (a > n && b > n) {
        std::cout << "0\n";
        return 0;
```

```
    }

    uint64_t firstCount = a == n;
    uint64_t secondCount = b == n;

    if (a < n) {
        firstCount += getWaysCount(a, b, n - a);
    }

    if (b < n) {
        secondCount += getWaysCount(a, b, n - b);
    }

    std::cout << firstCount * a + secondCount * b << "\n";

    return 0;
}
```

Task B

```
#include <stdint.h>

#include <iostream>
#include <queue>

struct vec3_t {
    int64_t x, y, z;

    friend std::ostream& operator<<(std::ostream& os, const vec3_t& v) {
        return (os << "(" << v.x << "; " << v.y << "; " << v.z << ")");
    }
};

vec3_t operator-(const vec3_t& a, const vec3_t& b) {
    return vec3_t{a.x - b.x, a.y - b.y, a.z - b.z};
}

vec3_t operator+(const vec3_t& a, const vec3_t& b) {
    return vec3_t{a.x + b.x, a.y + b.y, a.z + b.z};
}

vec3_t invertX(const vec3_t& a) { return vec3_t{-a.x, -a.z, -a.y}; }
vec3_t invertY(const vec3_t& a) { return vec3_t{-a.z, -a.y, -a.x}; }
vec3_t invertZ(const vec3_t& a) { return vec3_t{-a.y, -a.x, -a.z}; }

int64_t getRadius(const vec3_t& a) {
    return std::max(std::max(std::abs(a.x), std::abs(a.y)), std::abs(a.z)
    );
}

int64_t shortestPath(const vec3_t& a, const vec3_t& b) {
    return getRadius(b - a);
}
```

```
int main() {
    int64_t n;
    vec3_t a, b;
    std::cin >> n >> a.x >> a.y >> a.z >> b.x >> b.y >> b.z;

    int64_t path = shortestPath(a, b);

    path = std::min(path, shortestPath(a, b + vec3_t{-n - n, 0 + n, n}));
    path =
        std::min(path, shortestPath(a, b + vec3_t{-n + 2 * n, 0 - 2 * n,
   n}));
    path = std::min(path, shortestPath(a, b + vec3_t{n + n, 0 - n, 0 - n
   }));
    path = std::min(path,
                    shortestPath(a, b + vec3_t{n - 2 * n, 0 + 2 * n, 0 -
   n}));
    path =
        std::min(path, shortestPath(a, b + vec3_t{2 * n - n, 0 + n, -2 *
   n}));
    std::min(path,
             shortestPath(a, b + vec3_t{2 * n + n - 1, 0 - n + 1, -2 * n
   }));
    path =
        std::min(path, shortestPath(a, b - vec3_t{2 * n - n, 0 + n, -2 *
   n}));
    std::min(path,
             shortestPath(a, b - vec3_t{2 * n + n - 1, 0 - n + 1, -2 * n
   }));

    std::cout << path << "\n";

    return 0;
}
```

Task C

```
#include <stdint.h>

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

class HashFabric {
   public:
    int64_t kPowers[200002];
    int64_t p;

   public:
    HashFabric(int64_t k, int64_t p) {
        this->p = p;
        this->kPowers[0] = 1;
        this->kPowers[1] = k % p;
```

```
        for (uint32_t i = 2; i < 200002; i++) {
            this->kPowers[i] = (this->kPowers[i - 1] * k) % p;
        }
    }

    int64_t hashString(const std::string &str) {
        int64_t out = 0;
        uint32_t idx = 0;

        for (char c : str) {
            out = (out + ((c - '0' + 1) * this->kPowers[idx]) % p) % p;
            idx++;
        }

        return out;
    }
};

std::string encodeBase64(int64_t num) {
    std::string out;

    for (uint32_t i = 0; i < 64; i += 6) {
        out += char((num & 63) + 48);
        num >>= 6;
    }

    return out;
}

int64_t decodeBase64(const std::string &str) {
    int64_t out = 0;

    for (uint32_t i = str.size(); i > 0; i--) {
        char c = str[i - 1];

        out <<= 6;
        out |= c - 48;
    }

    return out;
}

int main() {
    const int64_t k1 = 7;
    const int64_t k2 = 13;
    const int64_t k3 = 11;
    const int64_t k4 = 19;
    const int64_t k5 = 3;
    const int64_t p1 = 999835393;
    const int64_t p2 = 765838421;
    const int64_t p3 = 745583863;
    const int64_t p4 = 83081249;
```

```
const int64_t p5 = 197205689;

HashFabric hash1(k1, p1);
HashFabric hash2(k2, p2);
HashFabric hash3(k3, p3);
HashFabric hash4(k4, p4);
HashFabric hash5(k5, p5);

int64_t n;
std::string test, input;
std::cin >> test >> n >> input;

int64_t inputHash1 = hash1.hashString(input);
int64_t inputHash2 = hash2.hashString(input);
int64_t inputHash3 = hash3.hashString(input);
int64_t inputHash4 = hash4.hashString(input);
int64_t inputHash5 = hash5.hashString(input);

const int64_t inputSize = input.size();

if (test == "first") {
    std::cout << encodeBase64(inputHash1) << encodeBase64(inputHash2)
              << encodeBase64(inputHash3) << encodeBase64(inputHash4)
              << encodeBase64(inputHash5) << "\n";
} else {
    std::string help;
    std::cin >> help;

    std::string helpHash1Base64;
    std::string helpHash2Base64;
    std::string helpHash3Base64;
    std::string helpHash4Base64;
    std::string helpHash5Base64;

    helpHash1Base64.resize(11);
    helpHash2Base64.resize(11);
    helpHash3Base64.resize(11);
    helpHash4Base64.resize(11);
    helpHash5Base64.resize(11);

    for (uint32_t i = 0; i < 11; i++) {
        helpHash1Base64[i] = help[i];
    }

    for (uint32_t i = 11; i < 22; i++) {
        helpHash2Base64[i - 11] = help[i];
    }

    for (uint32_t i = 22; i < 33; i++) {
        helpHash3Base64[i - 22] = help[i];
    }

    for (uint32_t i = 33; i < 44; i++) {
```

```
        helpHash4Base64[i - 33] = help[i];
    }

    for (uint32_t i = 44; i < 55; i++) {
        helpHash5Base64[i - 44] = help[i];
    }

    int64_t orgHash1 = decodeBase64(helpHash1Base64);
    int64_t orgHash2 = decodeBase64(helpHash2Base64);
    int64_t orgHash3 = decodeBase64(helpHash3Base64);
    int64_t orgHash4 = decodeBase64(helpHash4Base64);
    int64_t orgHash5 = decodeBase64(helpHash5Base64);

    for (int64_t shift = 0; shift <= n; shift++) {
        // std::cout << inputHash1 << " " << inputHash2 << " " <<
inputHash3
        //           << " " << inputHash4 << inputHash5 << "\n";
        // std::cout << orgHash1 << " " << orgHash2 << " " <<
orgHash3 << "
        // "
        //           << orgHash4 << orgHash5 << "\n\n";

        if (inputHash1 == orgHash1 && inputHash2 == orgHash2 &&
            inputHash3 == orgHash3 && inputHash4 == orgHash4 &&
            inputHash5 == orgHash5) {
            std::cout << "YES\n";
            return 0;
        }

        int64_t c = input[shift] - '0' + 1;

        inputHash1 =
            (inputHash1 + p1 * 3 - (c * hash1.kPowers[shift]) % p1) %
 p1;
        inputHash1 =
            (inputHash1 + (c * hash1.kPowers[shift + n]) % p1) % p1;
        inputHash2 =
            (inputHash2 + p2 * 3 - (c * hash2.kPowers[shift]) % p2) %
 p2;
        inputHash2 =
            (inputHash2 + (c * hash2.kPowers[shift + n]) % p2) % p2;
        inputHash3 =
            (inputHash3 + p3 * 3 - (c * hash3.kPowers[shift]) % p3) %
 p3;
        inputHash3 =
            (inputHash3 + (c * hash3.kPowers[shift + n]) % p3) % p3;
        inputHash4 =
            (inputHash4 + p4 * 3 - (c * hash4.kPowers[shift]) % p4) %
 p4;
        inputHash4 =
            (inputHash4 + (c * hash4.kPowers[shift + n]) % p4) % p4;
        inputHash5 =
            (inputHash5 + p5 * 3 - (c * hash5.kPowers[shift]) % p5) %
```

```
 p5;
            inputHash5 =
                (inputHash5 + (c * hash5.kPowers[shift + n]) % p5) % p5;
            orgHash1 = (orgHash1 * k1) % p1;
            orgHash2 = (orgHash2 * k2) % p2;
            orgHash3 = (orgHash3 * k3) % p3;
            orgHash4 = (orgHash4 * k4) % p4;
            orgHash5 = (orgHash5 * k5) % p5;
        }

        std::cout << "NO\n";
    }

    return 0;
}
```

Task D

```
#include <stdint.h>

#include <iostream>

int main() {
    uint32_t n;
    std::cin >> n;

    std::string lastStatus = "";

    while (lastStatus != "!") {
        uint32_t i = (1 + rand() % n);
        uint32_t j = (1 + rand() % n);

        if (rand() & 1)
            while (i == j) j = (1 + rand() % n);

        std::cout << i << " " << j << "\r\n";

        std::cin >> lastStatus;
    }

    return 0;
}
```

Task F

```
#include <stdint.h>

#include <iostream>

uint64_t weights[50][50];
char matrix[50][50];
char matrix2[50][50];
```

```
int main() {
    uint32_t h, w;
    std::cin >> h >> w;
    std::string in;

    for (uint32_t y = 0; y < h; y++) {
        std::cin >> in;

        for (uint32_t x = 0; x < w; x++) {
            weights[x][y] = in[x] - '0';
            matrix[x][y] = '.';
            matrix2[x][y] = '.';
        }
    }

    uint64_t totalWeight = 0;

    for (uint32_t y = 0; y < h; y += 2) {
        uint64_t tempWeight1 = 0;
        uint64_t tempWeight2 = 0;

        for (uint32_t x = 0; x < w; x += 2) {
            tempWeight1 += weights[x][y] * weights[x + 1][y];
            tempWeight1 += weights[x][y + 1] * weights[x + 1][y + 1];
            matrix[x][y] = 'L';
            matrix[x + 1][y] = 'R';
            matrix[x][y + 1] = 'L';
            matrix[x + 1][y + 1] = 'R';
        }

        for (uint32_t x = 0; x < w; x++) {
            tempWeight2 += weights[x][y] * weights[x][y + 1];
            matrix2[x][y] = 'T';
            matrix2[x][y + 1] = 'B';
        }

        if (tempWeight2 > tempWeight1) {
            totalWeight += tempWeight2;

            for (uint32_t x = 0; x < w; x++) {
                matrix[x][y] = matrix2[x][y];
                matrix[x][y + 1] = matrix2[x][y + 1];
            }
        } else {
            totalWeight += tempWeight1;
        }
    }

    std::cout << totalWeight << "\n";

    for (uint32_t y = 0; y < h; y++) {
        for (uint32_t x = 0; x < w; x++) {
```

```
            std::cout << matrix[x][y];
        }

        std::cout << "\n";
    }

    return 0;
}
```