Олимпиада СПбГУ
по информатике
2023/24 учебного года

24fin0419 – Сидельников Владислав Денисович

A	B	C	D	E	F	sum
100	100	40	35	20	65	360

Task A

```
from functools import lru_cache

@lru_cache
def rec(pos):
    if pos == n:
        return 1
    if pos > n:
        return 0
    return rec(pos + a) + rec(pos + b)

a, b, n = map(int, input().split())
ans = rec(a) * a + rec(b) * b
print(ans)
```

Task B

```
#include <map>
#include <cmath>
#include <queue>
#include <algorithm>
#include <vector>
#include <iostream>
#include <set>
using namespace std;
#define all(x) x.begin(), x.end()
#pragma warning (disable : 4996)


int getdist(int x, int y, int z) {
```

```
    if (x == 0 && y == 0 && z == 0) return 0;

    if (x == 0 || z == 0) return abs(y);
    else if (y == 0) return abs(x);

    int part;
    if (x > 0 && y > 0 && z < 0) part = 0;
    else if (x < 0 && y > 0 && z < 0) part = 1;
    else if (x < 0 && y > 0 && z > 0) part = 2;
    else if (x < 0 && y < 0 && z > 0) part = 3;
    else if (x > 0 && y < 0 && z > 0) part = 4;
    else part = 5; //if (x > 0 && y < 0 && z < 0)

    int dist;
    if (part == 0) {
        dist = y;
        z += y;
        y = 0;
    }
    if (part == 1) {
        dist = -z;
        y += z;
        z = 0;
    }
    if (part == 2) {
        dist = z;
        x += z;
        z = 0;
    }
    if (part == 3) {
        dist = -y;
        z += y;
        y = 0;
    }
    if (part == 4) {
        dist = z;
        y += z;
        z = 0;
    }
    if (part == 5) {
        dist = -z;
        x += z;
        z = 0;
    }
    if (x == 0 || z == 0) return abs(y) + dist;
    else if (y == 0) return abs(x) + dist;
}

void convert(int part1, int n, int& cx1, int& cy1, int& cz1) {
    if (part1 == 0) {
        cx1 += n;
        cy1 += n;
        cz1 -= 2 * n;
```

```
    }
    if (part1 == 1) {
        cx1 -= n;
        cy1 += 2 * n;
        cz1 -= n;
    }
    if (part1 == 2) {
        cx1 -= 2 * n;
        cy1 += n;
        cz1 += n;
    }
    if (part1 == 3) {
        cx1 -= n;
        cy1 -= n;
        cz1 += 2 * n;
    }
    if (part1 == 4) {
        cx1 += n;
        cy1 -= 2 * n;
        cz1 += n;
    }
    if (part1 == 5) {
        cx1 += 2 * n;
        cy1 -= n;
        cz1 -= n;
    }
}

int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    int n, x1, y1, z1, x2, y2, z2;
    cin >> n;
    cin >> x1 >> y1 >> z1 >> x2 >> y2 >> z2;
    vector<int> dists;

    for (int part1 = 0; part1 < 6; part1++) {
        for (int part2 = 0; part2 < 6; part2++) {
            int cx1 = x1, cy1 = y1, cz1 = z1, cx2 = x2, cy2 = y2, cz2 =
   z2;
            convert(part1, n, cx1, cy1, cz1);
            convert(part2, n, cx2, cy2, cz2);
            int x = cx2 - cx1, y = cy2 - cy1, z = cz2 - cz1;
            dists.push_back(getdist(x, y, z));
        }
    }
    int mini = 1e9;
    for (int dist : dists) mini = min(mini, dist);
    cout << mini << endl;

    return 0;
}
```

Task C

```
#include <map>
#include <cmath>
#include <queue>
#include <algorithm>
#include <vector>
#include <iostream>
#include <set>
using namespace std;
#define all(x) x.begin(), x.end()
#pragma warning (disable : 4996)

int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    string p, s;
    int n;
    cin >> p >> n >> s;
    if (p == "first") {
        cout << s << endl;
    }
    else {
        string t;
        cin >> t;
        string comb = s + "#" + t + t;
        vector<int> z(comb.size());
        int l = 0, r = -1;
        z[0] = comb.size();
        for (int i = 1; i < comb.size(); i++) {
            if (r >= i) {
                z[i] = min(r - i, z[i - l]);
            }
            while (comb[i + z[i]] == comb[z[i]]) z[i]++;
            if (i + z[i] - 1 > r) {
                l = i;
                r = i + z[i] - 1;
            }
        }
        for (int i = s.size() + 1; i < comb.size(); i++) {
            if (z[i] == s.size()) {
                cout << "YES" << endl;
                return 0;
            }
        }
        cout << "NO" << endl;

    }
    return 0;
}
```

Task D

```
import random
n = int(input())
s = ">"
while s != "!":
    i, j = random.randint(1, n), random.randint(1, n)
    print(min(i, j), max(i, j))
    s = input()
```

Task E

```
#include <map>
#include <cmath>
#include <queue>
#include <algorithm>
#include <vector>
#include <iostream>
#include <set>
using namespace std;
using ll = long long;
#define all(x) x.begin(), x.end()
#pragma warning (disable : 4996)

int n, q, di, t, dit;

struct Node {
    vector<int> neis;
    int start;
};

int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cin >> n >> q;
    vector<vector<int>> g(n);
    vector<Node> nodes(n);
    int qi;
    for (int i = 0; i < n; i++) {
        cin >> t;
        for (int j = 0; j < t; j++) {
            cin >> dit; dit--;
            nodes[i].neis.push_back(dit);
        }
    }
    vector<int> ans(1e6, 1);
    int curv = 0;
    for (int I = 1; I < 1e5 + 10; I++) {
        int start = nodes[curv].start;
        nodes[curv].start = (nodes[curv].start + 1) % nodes[curv].neis.
   size();
```

```
        curv = nodes[curv].neis[start];
        ans[I] = curv + 1;
    }
    for (int i = 0; i < q; i++) {
        cin >> qi;
        cout << ans[qi] << " ";
    }
    return 0;
}
```

Task F

```
#include <map>
#include <cmath>
#include <queue>
#include <algorithm>
#include <vector>
#include <iostream>
#include <set>
using namespace std;
using ll = long long;
#define all(x) x.begin(), x.end()
#pragma warning (disable : 4996)

const int MAXC = 1e7;

struct Domino {
    int x1, y1, x2, y2;
    ll prod;
};

int c;
vector<Domino> bestdominos;
ll maxans = 0;
int h, w;

vector<vector<int>> mapi(50, vector<int>(50));
vector<Domino> dominos;
ll curans = 0;
set<pair<int, int>> available;

bool canbeplaced(Domino& d1) {
    pair<int, int> c11 = make_pair(d1.x1, d1.y1);
    pair<int, int> c12 = make_pair(d1.x2, d1.y2);

    int k = 0;
    for (int i = 0; i < dominos.size(); i++) {
        Domino d2 = dominos[i];
        pair<int, int> c21 = make_pair(d2.x1, d2.y1);
        pair<int, int> c22 = make_pair(d2.x2, d2.y2);
        if (c++ > MAXC) return false;
```

```
        if (c11 == c21 || c11 == c22 || c12 == c21 || c12 == c22) {
            if (d1.prod > d2.prod) {
                curans -= d2.prod;
                vector<Domino> newdominos;
                for (int j = 0; j < i; j++) newdominos.push_back(dominos[
    j]);
                for (int j = i + 1; j < dominos.size(); j++) newdominos.
    push_back(dominos[j]);
                dominos = newdominos;
                available.insert({ d2.y1, d2.x1 });
                available.insert({ d2.y2, d2.x2 });
                return true;
            }
            return false;
        }
    }
    return true;
}

bool tryGenerateDomino() {
    int r = rand() % available.size();
    pair<int, int> c1;
    int I = 0;
    for (pair<int, int> sub : available) {
        I++;
        c1 = { sub.second, sub.first };
        if (I > r) break;
    }

    r = rand() % 2;
    pair<int, int> c2;
    if (r == 0) {
        c2 = { (c1.first + 1) % w, c1.second };
    }
    else if (r == 1) {
        c2 = { c1.first, (c1.second + 1) % h };
    }
    Domino d = { c1.first, c1.second, c2.first, c2.second, 1LL * mapi[d.
    y1][d.x1] * mapi[d.y2][d.x2] };
    bool iscorrect = canbeplaced(d);
    if (iscorrect) {
        curans += d.prod;
        dominos.push_back(d);
        available.erase({ d.y1, d.x1 });
        available.erase({ d.y2, d.x2 });
        if (curans > maxans) {
            maxans = curans;
            bestdominos = dominos;
        }
    }
    return iscorrect;
}
```

```
void print(vector<Domino>& dominos) {
    vector<vector<char>> ans(h, vector<char>(w, '.'));
    for (Domino d : dominos) {
        if (d.y1 == d.y2) {
            ans[d.y1][d.x1] = 'L';
            ans[d.y1][d.x2] = 'R';
        }
        else {
            ans[d.y1][d.x1] = 'T';
            ans[d.y2][d.x1] = 'B';
        }
    }
    for (int i = 0; i < h; i++) {
        for (int j = 0; j < w; j++) {
            cout << ans[i][j];
        }
        cout << "\n";
    }
    cout << endl;
}

int main()
{
    srand(4);
    ios_base::sync_with_stdio(false);
    cin.tie(0);
    cin >> h >> w;
    for (int i = 0; i < h; i++) {
        string s;
        cin >> s;
        for (int j = 0; j < w; j++) {
            mapi[i][j] = s[j] - '0';
            available.insert({ i, j });
        }
    }

    while (true) {
        if (c++ > MAXC) break;
        bool flag = false;
        for (int i = 0; i < 500; i++) {
            if (available.size() == 0) break;
            flag = tryGenerateDomino();
            if (c++ > MAXC || flag) break;
        }
        if (!flag && c++ < MAXC) {
            vector<Domino> newdominos;

            int count = rand() % ((dominos.size() + 1) * 3 / 4);
            vector<int> forshuffle;
            for (int i = 0; i < dominos.size(); i++) {
                forshuffle.push_back(i);
            }
            random_shuffle(all(forshuffle));
```

```
            for (int i = 0; i < count; i++) {
                newdominos.push_back(dominos[forshuffle[i]]);
            }
            for (int i = count; i < dominos.size(); i++) {
                Domino d = dominos[forshuffle[i]];
                curans -= d.prod;
                available.insert({ d.y1, d.x1 });
                available.insert({ d.y2, d.x2 });
            }
            dominos = newdominos;
        }
    }

    cout << maxans << endl;
    print(bestdominos);
    return 0;
}
/*2 6
317233
229870*/
```