Олимпиада СПбГУ
по информатике
2023/24 учебного года

24fin0229 – Королев Егор Юрьевич

A	B	C	D	E	F	sum
100	100	40	35	20	70	365

Task A

```
def go(cur, end, state):
    for i in [a, b]:
#         print(' ' * len(state), f"cur {cur}; plus {i}; next {cur + i};
    end {end}")
        if cur + i == end:
#             print(' ' * len(state), f"OK")
            global RES
            RES += 1
#             print(state)
        if cur + i < n:
            go(cur + i, end, state + [cur + i])

def go2(x):
    global L
    if x >= n and x != end:
        return 0
    if x < 0:
        return 0
    if L[x] is not None:
        return L[x]
    L[x] = go2(x - a) + go2(x - b)
    return L[x]

a, b, n = map(int, input().split())
RES = 0
for begin in range(n):
    start = begin
    end = begin + n
```

```
    L = [None for _ in range(end + 1)]
    for i in range(begin):
        L[i] = 0
    L[begin] = 1
    RES += go2(end)
#    print(L)

print(RES)
```

Task B

```
def get_dist():
    dist_z = z2 - z1
    delta_z = abs(z2 - z1)
    pos_a = x1
    pos_b = x1 - dist_z
    pos_a, pos_b = min(pos_a, pos_b), max(pos_a, pos_b)
#    print('--')
#    print(x2, y2, z2)
    if pos_a <= x2 <= pos_b:
        return delta_z
    else:
        return delta_z + min(abs(x2 - pos_a), abs(x2 - pos_b))

def minus(a, b):
    return (a[0] - b[0], a[1] - b[1], a[2] - b[2])

def plus(a, b):
    return (a[0] + b[0], a[1] + b[1], a[2] + b[2])

n = int(input())
x1, y1, z1 = map(int, input().split())
a, b, c = map(int, input().split())

res = 10 ** 18

x2, y2, z2 = a, b, c
res1 = get_dist()
res = min(res, res1)

x2, y2, z2 = plus(minus((a, b, c), (n, 0, -n)), (-n, n, 0))
res1 = get_dist()
res = min(res, res1)

x2, y2, z2 = plus(minus((a, b, c), (-n, n, 0)), (n, 0, -n))
res1 = get_dist()
res = min(res, res1)

x2, y2, z2 = plus(minus((a, b, c), (0, n, -n)), (n, -n, 0))
res1 = get_dist()
res = min(res, res1)

x2, y2, z2 = plus(minus((a, b, c), (n, -n, 0)), (0, n, -n))
```

```
res1 = get_dist()
res = min(res, res1)

x2, y2, z2 = plus(minus((a, b, c), (0, n, -n)), (-n, 0, n))
res1 = get_dist()
res = min(res, res1)

x2, y2, z2 = plus(minus((a, b, c), (-n, 0, n)),(0, n, -n))
res1 = get_dist()
res = min(res, res1)

print(res)
```

Task C

```
cmd = input()
if cmd == 'first':
    n = int(input())
    s = input()
    print(s)
else:
    n = int(input())
    a = input()
    b = input()
    res = False
    for i in range(n):
        res = True
        for j in range(n):
            if a[(i + j) % n] != b[j]:
                res = False
                break
        if res:
            break


    if res:
        print("YES")
    else:
        print("NO")
```

Task D

```
import sys

def swap(i, j):
    global id2pos
    global status

    real_i = id2pos[i]
    real_j = id2pos[j]
    print(int(real_i + 1), int(real_j + 1))
    sys.stdout.flush()
```

```
    status = input()
    if status == '!':
        quit()
    id2pos[i] = real_j
    id2pos[j] = real_i

    for x in range(n):
        pos2id[(n - 1 + id2pos[x]) % n] = x
        id2pos[x] = (n - 1 + id2pos[x]) % n


n = int(input())
id2pos = [i for i in range(n)]
pos2id = dict()
for i in range(n):
    pos2id[i] = i

id2x = [0 for _ in range(n)]
val2id = [0 for _ in range(n)]

status = '!'

cmp_a = 0
cmp_b = 1
while True:
    ra = id2pos[cmp_a]
    rb = id2pos[cmp_b]
    if rb != (ra + 1) % n:
        rp = (ra + 1) % n
        swap(cmp_b, pos2id[rp])
    while id2pos[cmp_a] != 0:
        swap(0, 0)
    swap(0, 0)
    if status == '<':
        pass
    else:
        id2x[cmp_a] += 1

    cmp_b += 1
    cmp_b %= n
    if cmp_b == cmp_a:
        cmp_a += 1
        cmp_b = (cmp_a + 1) % n
        if cmp_a == n:
            break

start_pos = [0 for _ in range(n)]
for i in range(n):
    start_pos[i] = n - id2x[i]


last_p = [0 for _ in range(n)]
for i in range(n):
```

```
    last_p[id2pos[i]] = n - id2x[i]

cur_p = [n - i for i in range(n)]

id2pos = [i for i in range(n)]

if True:
    X = 0
    for j in range(1, n + 1):
        pos_now = cur_p.index(j)
        pos_last = last_p.index(j)
        if pos_now == pos_last:
            continue
        last_p[pos_now], last_p[pos_last] = last_p[pos_last], last_p[
    pos_now]
        swap(pos_now, pos_last)


for _ in range(n + 1):
    swap(0, 0)
```

Task E

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

using namespace std;

int main() {
    int n, q;
    cin >> n >> q;
    vector<vector<int>> L;
    vector<int> Q;
    vector<int> R;
    vector<int> I;
    L.resize(n);
    I.resize(n, 0);
    for (int i =0; i < n; i++){
        int k;
        cin >> k;
        L[i].reserve(k);
        for (int j =0; j < k; j++) {
            int x;
            cin >> x;
            L[i].push_back(x);
        }
    }
    Q.reserve(q);
    for (int j =0; j < q; j++) {
        int x;
        cin >> x;
        Q.push_back(x);
    }
```

```
    R.resize(q);
    int t =0;
    int qi = 0;
    int X = 1;
    while (true) {
        if (t == Q[qi]) {
            R[qi] = X;
            qi++;
            if (qi == q){
                break;
            }
        }
        int xi = X-1;
        X = L[xi][I[xi]];
        I[xi]++;
        I[xi] %= L[xi].size();
        t++;
    }

    for (int j =0; j < q; j++) {
        cout << R[j] << ' ';
    }

}
```

Task F

```
def S1(bx = 0, by = 0):
    r = 0
    f = [['.' for _ in range(w)] for _ in range(h)]
    for x in range(bx, w + bx, 2):
        for y in range(by, h + by, 2):
            y1 = (y + 1) % h
            x1 = (x + 1) % w
            a = L[y][x] * L[y][x1] + L[y1][x] * L[y1][x1]
            b = L[y][x] * L[y1][x] + L[y][x1] * L[y1][x1]
            if a > b:
                r += a
                f[y][x] = 'L'
                f[y][x1] = 'R'
                f[y1][x] = 'L'
                f[y1][x1] = 'R'
            else:
                r += b
                f[y][x] = 'T'
                f[y1][x] = 'B'
                f[y][x1] = 'T'
                f[y1][x1] = 'B'
    global RES, FIELD
    if r > RES:
        RES = r
        FIELD = f
```

```python
h, w = map(int, input().split())
L = []
for _ in range(h):
    L.append([int(x) for x in input()])

RES = 0
FIELD = []

S1()
S1(1, 1)

print(RES)
for i in FIELD:
    print(*i, sep='')
```