Олимпиада СПбГУ
по информатике
2023/24 учебного года

24fin0169 – Захаров Сергей Николаевич

A	B	C	D	E	F	sum
100	100	100	35	20	71	426

Task A

```
from math import factorial

a = 0
b = 0
n = 0
ans = 0

def C(n, k) -> int:
    return factorial(n) // factorial(n - k) // factorial(k)

def solve(x: int, y: int):
    global a, b, n, ans
    v = 1
    if x == 0:
        v *= b
    elif y == 0:
        v *= a
    else:
        v = C(x + y - 1, x - 1) * a + C(x + y - 1, y - 1) * b
    ans += v


def main() -> None:
    global a, b, n, ans
    a, b, n = map(int, input().split())
    for x in range(n + 1):
        for y in range(n + 1):
            if x * a + y * b == n:
                solve(x, y)
```

```
    print(ans)


if __name__ == '__main__':
    main()
```

Task B

```
class point:
    def __init__(self, x, y, z) -> None:
        self.x = x
        self.y = y
        self.z = z


def reverse_to_x(a: point) -> None:
    a.x *= -1
    a.y *= -1
    a.z *= -1
    a.y, a.z = a.z, a.y


def reverse_to_y(a: point) -> None:
    a.x *= -1
    a.y *= -1
    a.z *= -1
    a.x, a.z = a.z, a.x


def reverse_to_z(a: point) -> None:
    a.x *= -1
    a.y *= -1
    a.z *= -1
    a.y, a.x = a.x, a.y


def simple_dist(a: point, b: point) -> int:
    v1 = abs(a.x - b.x) + abs(a.y - b.y) + abs(a.z - b.z)
    return v1 // 2


def main() -> None:
    n = int(input())
    x1, y1, z1 = map(int, input().split())
    x2, y2, z2 = map(int, input().split())
    a = point(x1, y1, z1)
    b = point(x2, y2, z2)

    best = simple_dist(a, b)
    b1 = point(x2 + 2 * n, y2 - n, z2 - n)
    b2 = point(x2 - n, y2 + 2 * n, z2 - n)
    b3 = point(x2 - n, y2 - n, z2 + 2 * n)
```

```
    b4 = point(x2 - 2 * n, y2 + n, z2 + n)
    b5 = point(x2 + n, y2 - 2 * n, z2 + n)
    b6 = point(x2 + n, y2 + n, z2 - 2 * n)

    ps = [b, b1, b2, b3, b4, b5, b6]

    best = 10 ** 18
    for p in ps:
        best = min(best, simple_dist(a, p))
    print(best)


if __name__ == '__main__':
    main()
```

Task C

```
from typing import List

class Hash:
    def __init__(self) -> None:
        self.n = 10 ** 5 + 2
        self.M = 911345721
        self.p = 987456123
        self.p_pow = [1 for i in range(self.n)]
        for i in range(1, len(self.p_pow)):
            self.p_pow[i] = self.p_pow[i - 1] * self.p
            self.p_pow[i] %= self.M

    def str_to_hash(self, s: str) -> int:
        h = 0
        for i, el in enumerate(s):
            cur = (int(el) + 1) * self.p_pow[i]
            h += cur
            h %= self.M
        return h

    def str_to_array_hash(self, s: str) -> List[int]:
        hs = [0]
        for i, el in enumerate(s):
            cur = (int(el) + 1) * self.p_pow[i + 1]
            hs.append((hs[-1] + cur) % self.M)
        return hs


def first() -> None:
    n = int(input())
    s = input()
    h = Hash()
    print(h.str_to_hash(s))


def second() -> None:
```

```
    n = int(input())
    t = input()
    s = int(input())
    h = Hash()
    ts = h.str_to_array_hash(t)

    for k in range(1, n + 1):
        cur = ts[-1] - ts[k - 1]
        cur += ts[k - 1] * h.p_pow[n]
        cur %= h.M
        cur_s = s * h.p_pow[k]
        cur_s %= h.M
        if cur == cur_s:
            print("YES")
            return
    print("NO")


def main() -> None:
    s = input()
    if s == 'first':
        first()
    else:
        second()


if __name__  == '__main__':
    main()

# 760648847
```

Task D

```
def main():
    n = int(input())
    res = ""
    while res != '!':
        if res == '>':
            print(1, 2)
        else:
            print(1, 1)
        res = input()

if __name__ == '__main__':
    main()
```

Task E

```
#include<bits/stdc++.h>
using namespace std;

int n;
```

```
vector<vector<int>> g;
vector<int> res;
vector<int> offset;

void simulate(int t, int v) {
    if (res.size() > 100000) {
        return;
    }
    res.push_back(v);
    int new_v = g[v][offset[v] % g[v].size()];
    offset[v]++;
    simulate(++t, new_v);
}

int main() {
    int q, n;
    cin >> n >> q;
    g.resize(n);
    offset.resize(n);
    for (int i = 0; i < n; i++) {
        int d;
        cin >> d;
        for (int j = 0; j < d; j++) {
            int x;
            cin >> x;
            x--;
            g[i].push_back(x);
        }
    }
    simulate(0, 0);
    while (q--) {
        int t;
        cin >> t;
        cout << res[t] + 1 << ' ';
    }
    return 0;
}
```

Task F

```
from typing import List

class Solution:
    def __init__(self, f: List[List[int]], t: int) -> None:
        self.h = len(f)
        self.w = len(f[0])
        self.f = f
        self.score = 0
        self.result = [[''' for i in range(self.w)] for j in range(self.h)
    ]
        if t == 1:
            self.build_sol_1()
        elif t == 2:
```

```
        self.build_sol_2()
    elif t == 3:
        self.build_sol_3()
    elif t == 4:
        self.build_sol_4()
    elif t == 5:
        self.build_sol_5()
    elif t == 6:
        self.build_sol_6()

def build_sol_1(self) -> None:
    for i in range(self.h):
        for j in range(0, self.w, 2):
            self.score += self.f[i][j] * self.f[i][j + 1]
            self.result[i][j] = 'L'
            self.result[i][j + 1] = 'R'

def build_sol_2(self) -> None:
    for i in range(0, self.h, 2):
        for j in range(self.w):
            self.score += self.f[i][j] * self.f[i + 1][j]
            self.result[i][j] = 'T'
            self.result[i + 1][j] = 'B'

def build_sol_3(self) -> None:
    for i in range(0, self.h, 2):
        for j in range(0, self.w, 2):
            v1 = self.f[i][j] * self.f[i][j + 1] + self.f[i + 1][j] *
 self.f[i + 1][j + 1]
            v2 = self.f[i][j] * self.f[i + 1][j] + self.f[i][j + 1] *
 self.f[i + 1][j + 1]
            self.score += max(v1, v2)
            if v1 > v2:
                self.result[i][j] = 'L'
                self.result[i][j + 1] = 'R'
                self.result[i + 1][j] = 'L'
                self.result[i + 1][j + 1] = 'R'
            else:
                self.result[i][j] = 'T'
                self.result[i + 1][j] = 'B'
                self.result[i][j + 1] = 'T'
                self.result[i + 1][j + 1] = 'B'

def build_sol_4(self) -> None:
    for i in range(1, self.h + 1, 2):
        for j in range(1, self.w + 1, 2):
            v1 = self.f[i % self.h][j % self.w] * self.f[i % self.h
][(j + 1) % self.w] + self.f[(i + 1) % self.h][j % self.w] * self.f[(i
 + 1) % self.h][(j + 1) % self.w]
            v2 = self.f[i % self.h][j % self.w] * self.f[(i + 1) %
self.h][j % self.w] + self.f[i % self.h][(j + 1) % self.w] * self.f[(i
 + 1) % self.h][(j + 1) % self.w]
            self.score += max(v1, v2)
```

```
            if v1 > v2:
                self.result[i % self.h][j % self.w] = 'L'
                self.result[i % self.h][(j + 1) % self.w] = 'R'
                self.result[(i + 1) % self.h][j % self.w] = 'L'
                self.result[(i + 1) % self.h][(j + 1) % self.w] = 'R'
            else:
                self.result[i % self.h][j % self.w] = 'T'
                self.result[(i + 1) % self.h][j % self.w] = 'B'
                self.result[i % self.h][(j + 1) % self.w] = 'T'
                self.result[(i + 1) % self.h][(j + 1) % self.w] = 'B'

def build_sol_5(self) -> None:
    for i in range(0, self.h, 2):
        for j in range(1, self.w + 1, 2):
            v1 = self.f[i % self.h][j % self.w] * self.f[i % self.h
][(j + 1) % self.w] + self.f[(i + 1) % self.h][j % self.w] * self.f[(i
 + 1) % self.h][(j + 1) % self.w]
            v2 = self.f[i % self.h][j % self.w] * self.f[(i + 1) %
self.h][j % self.w] + self.f[i % self.h][(j + 1) % self.w] * self.f[(i
 + 1) % self.h][(j + 1) % self.w]
            self.score += max(v1, v2)
            if v1 > v2:
                self.result[i % self.h][j % self.w] = 'L'
                self.result[i % self.h][(j + 1) % self.w] = 'R'
                self.result[(i + 1) % self.h][j % self.w] = 'L'
                self.result[(i + 1) % self.h][(j + 1) % self.w] = 'R'
            else:
                self.result[i % self.h][j % self.w] = 'T'
                self.result[(i + 1) % self.h][j % self.w] = 'B'
                self.result[i % self.h][(j + 1) % self.w] = 'T'
                self.result[(i + 1) % self.h][(j + 1) % self.w] = 'B'

def build_sol_6(self) -> None:
    for i in range(1, self.h + 1, 2):
        for j in range(0, self.w, 2):
            v1 = self.f[i % self.h][j % self.w] * self.f[i % self.h
][(j + 1) % self.w] + self.f[(i + 1) % self.h][j % self.w] * self.f[(i
 + 1) % self.h][(j + 1) % self.w]
            v2 = self.f[i % self.h][j % self.w] * self.f[(i + 1) %
self.h][j % self.w] + self.f[i % self.h][(j + 1) % self.w] * self.f[(i
 + 1) % self.h][(j + 1) % self.w]
            self.score += max(v1, v2)
            if v1 > v2:
                self.result[i % self.h][j % self.w] = 'L'
                self.result[i % self.h][(j + 1) % self.w] = 'R'
                self.result[(i + 1) % self.h][j % self.w] = 'L'
                self.result[(i + 1) % self.h][(j + 1) % self.w] = 'R'
            else:
                self.result[i % self.h][j % self.w] = 'T'
                self.result[(i + 1) % self.h][j % self.w] = 'B'
                self.result[i % self.h][(j + 1) % self.w] = 'T'
                self.result[(i + 1) % self.h][(j + 1) % self.w] = 'B'
```

```python
def main() -> None:
    h, w = map(int, input().split())
    f = [[0 for i in range(w)] for j in range(h)]
    for i in range(h):
        s = input()
        for j in range(w):
            f[i][j] = int(s[j])

    n = 6
    sols = []
    for t in range(1, n + 1):
        sols.append(Solution(f, t))

    best_ind = -1
    for i, el in enumerate(sols):
        if best_ind == -1 or el.score > sols[best_ind].score:
            best_ind = i

    print(sols[best_ind].score)
    for line in sols[best_ind].result:
        for el in line:
            print(el, end='')
        print()

if __name__ == '__main__':
    main()
```