**旱地 q1 s1**

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

// 作物种类编号

int zhong[16] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 };

// 地块分类

int di\_leixing[27] = { 0, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3 };

// 地块面积

double di[27] = { 0, 80, 55, 35, 72, 68, 55, 60, 46, 40, 28, 25, 26, 55, 44, 50, 25, 60, 45, 35, 20, 15, 13, 15, 18, 27, 20 };

// 2023年种植记录

int ji\_1[27][16]; // 2023年各地块的种植信息，数组存储某地块的种植作物

// 每个农作物不超过的销量

int z\_xiao[16] = { 0, 57000, 21850, 22400, 33040, 9875, 170840, 132785, 71400, 30000, 12500, 1500, 35100, 36000, 14000, 10000 };

// 不同地块的产量调整系数

double shanpo\_changliang\_xishu = 0.95; // 山坡地的产量系数

double gandi\_changliang\_xishu = 1.05; // 干旱地的产量系数

// 记录每个地块上次种植豆类作物的年份

int last\_douli\_year[27] = { 0 }; // 0表示从未种植过

// 判断是否为豆类作物，豆类作物为编号1-5

bool is\_douli(int crop\_id) {

return crop\_id >= 1 && crop\_id <= 5;

}

// 计算地块的产量系数

double get\_changliang\_xishu(int di\_id) {

if (di\_leixing[di\_id] == 1)

return gandi\_changliang\_xishu; // 干旱地

if (di\_leixing[di\_id] == 3)

return shanpo\_changliang\_xishu; // 山坡地

return 1.0; // 梯田

}

// 计算某一作物在某一地块的实际产量

double calc\_changliang(int di\_id, int crop\_id, double changliang\_per\_mu) {

double xishu = get\_changliang\_xishu(di\_id);

return di[di\_id] \* changliang\_per\_mu \* xishu;

}

// 判断是否满足销量约束

bool meet\_sales\_constraint(double total\_changliang, int crop\_id) {

return total\_changliang <= z\_xiao[crop\_id];

}

// 检查是否能种植（避免与去年的作物相同）

bool can\_plant(int di\_id, int crop\_id) {

// 去年的作物不能与今年种植的相同

return ji\_1[di\_id][crop\_id] == 0; // 2023年没有种植该作物，则可以种植

}

// 随机化作物选择顺序

vector<int> get\_random\_crop\_order() {

vector<int> crops = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 };

random\_shuffle(crops.begin(), crops.end()); // 打乱作物编号顺序

return crops;

}

// 解决问题的函数

void solve() {

double total\_area = 1201; // 总耕地面积

vector<vector<int>> plan\_2024(27, vector<int>(16, 0)); // 2024年的种植计划

vector<vector<int>> plan\_2025(27, vector<int>(16, 0)); // 2025年的种植计划

// 假设每种作物的亩产量（单位：产量/亩）

vector<double> changliang\_per\_mu = { 0, 380, 475, 380, 330, 395, 760, 950, 380, 600, 500, 105, 2850, 2100, 400, 500 };

// 2024年种植计划

for (int di\_id = 1; di\_id <= 26; ++di\_id) {

vector<int> crops\_order = get\_random\_crop\_order();

bool planted\_douli = false;

for (int crop\_id : crops\_order) {

if (can\_plant(di\_id, crop\_id)) {

if (is\_douli(crop\_id) && last\_douli\_year[di\_id] <= 2021) { // 优先种植豆类作物

double changliang = calc\_changliang(di\_id, crop\_id, changliang\_per\_mu[crop\_id]);

if (meet\_sales\_constraint(changliang, crop\_id)) {

plan\_2024[di\_id][crop\_id] = 1; // 安排种植

last\_douli\_year[di\_id] = 2024; // 记录豆类种植年份

planted\_douli = true;

break;

}

}

}

}

// 如果2024年没有种植豆类，再按随机顺序选择

if (!planted\_douli) {

for (int crop\_id : crops\_order) {

if (can\_plant(di\_id, crop\_id)) {

double changliang = calc\_changliang(di\_id, crop\_id, changliang\_per\_mu[crop\_id]);

if (meet\_sales\_constraint(changliang, crop\_id)) {

plan\_2024[di\_id][crop\_id] = 1; // 安排种植

if (is\_douli(crop\_id))

last\_douli\_year[di\_id] = 2024; // 更新豆类种植年份

break;

}

}

}

}

}

// 2025年种植计划

for (int di\_id = 1; di\_id <= 26; ++di\_id) {

vector<int> crops\_order = get\_random\_crop\_order();

bool planted\_douli = false;

for (int crop\_id : crops\_order) {

if (can\_plant(di\_id, crop\_id)) {

if (is\_douli(crop\_id) && last\_douli\_year[di\_id] <= 2022) { // 三年内必须种植一次豆类

double changliang = calc\_changliang(di\_id, crop\_id, changliang\_per\_mu[crop\_id]);

if (meet\_sales\_constraint(changliang, crop\_id)) {

plan\_2025[di\_id][crop\_id] = 1;

last\_douli\_year[di\_id] = 2025;

planted\_douli = true;

break;

}

}

}

}

// 如果2025年没有种植豆类，再按随机顺序选择

if (!planted\_douli) {

for (int crop\_id : crops\_order) {

if (can\_plant(di\_id, crop\_id)) {

double changliang = calc\_changliang(di\_id, crop\_id, changliang\_per\_mu[crop\_id]);

if (meet\_sales\_constraint(changliang, crop\_id)) {

plan\_2025[di\_id][crop\_id] = 1;

if (is\_douli(crop\_id))

last\_douli\_year[di\_id] = 2025;

break;

}

}

}

}

}

// 输出2024和2025年的种植方案

cout << "2024年种植方案：" << endl;

for (int di\_id = 1; di\_id <= 26; ++di\_id) {

cout << "地块 " << di\_id << ": ";

for (int crop\_id = 1; crop\_id <= 15; ++crop\_id) {

if (plan\_2024[di\_id][crop\_id]) {

cout << "种植作物 " << crop\_id << " ";

}

}

cout << endl;

}

cout << "2025年种植方案：" << endl;

for (int di\_id = 1; di\_id <= 26; ++di\_id) {

cout << "地块 " << di\_id << ": ";

for (int crop\_id = 1; crop\_id <= 15; ++crop\_id) {

if (plan\_2025[di\_id][crop\_id]) {

cout << "种植作物 " << crop\_id << " ";

}

}

cout << endl;

}

}

int main() {

// 初始化2023年的种植数据，假设数据来源外部输入

// ji\_1[地块编号][作物编号] = 1 表示2023年在某块地种了某作物

ji\_1[1][6] = 1;

ji\_1[2][7] = 1;

ji\_1[3][7] = 1;

ji\_1[4][1] = 1;

ji\_1[5][4] = 1;

ji\_1[6][8] = 1;

ji\_1[7][6] = 1;

ji\_1[8][2] = 1;

ji\_1[9][3] = 1;

ji\_1[10][4] = 1;

ji\_1[11][5] = 1;

ji\_1[12][8] = 1;

ji\_1[13][6] = 1;

ji\_1[14][8] = 1;

ji\_1[15][9] = 1;

ji\_1[16][10] = 1;

ji\_1[17][1] = 1;

ji\_1[18][7] = 1;

ji\_1[19][14] = 1;

ji\_1[20][15] = 1;

ji\_1[21][11] = 1;

ji\_1[22][12] = 1;

ji\_1[23][1] = 1;

ji\_1[24][13] = 1;

ji\_1[25][6] = 1;

ji\_1[26][3] = 1;

solve(); // 运行问题解决函数

return 0;

}

**旱地 q1 s2**

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

//干旱地情况二

// 作物种类编号

int zhong[16] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 };

// 地块分类

int di\_leixing[27] = { 0, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3 };

// 地块面积

double di[27] = { 0, 80, 55, 35, 72, 68, 55, 60, 46, 40, 28, 25, 26, 55, 44, 50, 25, 60, 45, 35, 20, 15, 13, 15, 18, 27, 20 };

// 2023年种植记录

vector<vector<int>> ji\_1(27, vector<int>(16, 0)); // 2023年各地块的种植信息

// 每个农作物不超过的销量

int z\_xiao[16] = { 0, 57000, 21850, 22400, 33040, 9875, 170840, 132785, 71400, 30000, 12500, 1500, 35100, 36000, 14000, 10000 };

// 不同地块的产量调整系数

double shanpo\_changliang\_xishu = 0.95; // 山坡地的产量系数

double gandi\_changliang\_xishu = 1.05; // 干旱地的产量系数

// 判断是否为豆类作物，豆类作物为编号1-5

bool is\_douli(int crop\_id) {

return crop\_id >= 1 && crop\_id <= 5;

}

// 计算地块的产量系数

double get\_changliang\_xishu(int di\_id) {

if (di\_leixing[di\_id] == 1)

return gandi\_changliang\_xishu; // 干旱地

if (di\_leixing[di\_id] == 3)

return shanpo\_changliang\_xishu; // 山坡地

return 1.0; // 梯田

}

// 计算某一作物在某一地块的实际产量

double calc\_changliang(int di\_id, int crop\_id, double changliang\_per\_mu) {

double xishu = get\_changliang\_xishu(di\_id);

return di[di\_id] \* changliang\_per\_mu \* xishu;

}

// 计算利润：销量在上限内的部分按正常价格销售，超出部分按50%价格销售

double calc\_profit(double total\_changliang, int crop\_id, double unit\_price) {

double max\_sales = z\_xiao[crop\_id]; // 作物的最大销量

if (total\_changliang <= max\_sales) {

return total\_changliang \* unit\_price; // 全部按正常价格销售

}

else {

double normal\_profit = max\_sales \* unit\_price; // 正常销量部分利润

double excess\_profit = (total\_changliang - max\_sales) \* unit\_price \* 0.5; // 超出部分按50%价格销售

return normal\_profit + excess\_profit; // 总利润

}

}

// 检查是否能种植（避免与去年的作物相同）

bool can\_plant(int di\_id, int crop\_id, const vector<vector<int>>& previous\_plan) {

return ji\_1[di\_id][crop\_id] == 0 && previous\_plan[di\_id][crop\_id] == 0;

}

// 随机化作物选择顺序

vector<int> get\_random\_crop\_order() {

vector<int> crops = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 };

random\_shuffle(crops.begin(), crops.end()); // 打乱作物编号顺序

return crops;

}

// 检查是否三年内必须种植豆类作物

bool must\_plant\_douli(int di\_id, const vector<vector<int>>& plan\_2024, const vector<vector<int>>& plan\_2023) {

// 检查2023年和2024年是否都没有种植豆类作物

for (int crop\_id = 1; crop\_id <= 5; ++crop\_id) {

if ((ji\_1[di\_id][crop\_id] == 1 || plan\_2024[di\_id][crop\_id] == 1)) {

return false; // 如果2023年或2024年已经种植过豆类作物，不需要强制

}

}

return true; // 如果2023年和2024年都没有种植豆类作物，则必须在2025年种植

}

void solve() {

double total\_area = 1201; // 总耕地面积

vector<vector<int>> plan\_2024(27, vector<int>(16, 0)); // 2024年的种植计划

vector<vector<int>> plan\_2025(27, vector<int>(16, 0)); // 2025年的种植计划

vector<int> douli\_sales(6, 0); // 记录豆类作物的累计销售量

// 假设每种作物的亩产量（单位：产量/亩）

vector<double> changliang\_per\_mu = { 0, 380, 475, 380, 330, 395, 760, 950, 380, 600, 500, 105, 2850, 2100, 400, 500 };

// 假设每种作物的单位价格（单位：元/斤）

vector<double> price\_per\_kg = { 0, 2.2, 6.7, 7.3, 5.9, 5.9, 2.9, 2.5, 5.8, 5.3, 6.8, 36.7, 1.2, 2.3, 4.5, 2.8 };

// 遍历地块，为2024年优先安排种植豆类作物

for (int di\_id = 1; di\_id <= 26; ++di\_id) {

vector<int> crops\_order = get\_random\_crop\_order();

double best\_profit = -1;

int best\_crop = -1;

bool planted\_douli = false;

// 优先种植豆类作物

for (int crop\_id : crops\_order) {

if (is\_douli(crop\_id) && can\_plant(di\_id, crop\_id, plan\_2024)) {

double changliang = calc\_changliang(di\_id, crop\_id, changliang\_per\_mu[crop\_id]);

double profit = calc\_profit(changliang, crop\_id, price\_per\_kg[crop\_id]);

// 检查豆类作物是否还可以继续种植（销售没有超出限制）

if (douli\_sales[crop\_id] + changliang <= z\_xiao[crop\_id]) {

douli\_sales[crop\_id] += changliang;

if (profit > best\_profit) {

best\_profit = profit;

best\_crop = crop\_id;

planted\_douli = true;

}

}

}

}

// 如果豆类作物没有被选择，选择其他利润最高的作物

if (!planted\_douli) {

for (int crop\_id : crops\_order) {

if (!is\_douli(crop\_id) && can\_plant(di\_id, crop\_id, plan\_2024)) {

double changliang = calc\_changliang(di\_id, crop\_id, changliang\_per\_mu[crop\_id]);

double profit = calc\_profit(changliang, crop\_id, price\_per\_kg[crop\_id]);

if (profit > best\_profit) {

best\_profit = profit;

best\_crop = crop\_id;

}

}

}

}

if (best\_crop != -1) {

plan\_2024[di\_id][best\_crop] = 1;

}

}

// 2025年种植计划

for (int di\_id = 1; di\_id <= 26; ++di\_id) {

vector<int> crops\_order = get\_random\_crop\_order();

double best\_profit = -1;

int best\_crop = -1;

// 检查是否需要强制种植豆类作物

if (must\_plant\_douli(di\_id, plan\_2024, ji\_1)) {

// 强制种植豆类作物之一（随机选择豆类作物）

for (int crop\_id : crops\_order) {

if (is\_douli(crop\_id)) {

best\_crop = crop\_id;

break;

}

}

}

else {

// 否则选择利润最高的作物

for (int crop\_id : crops\_order) {

if (can\_plant(di\_id, crop\_id, plan\_2024)) {

double changliang = calc\_changliang(di\_id, crop\_id, changliang\_per\_mu[crop\_id]);

double profit = calc\_profit(changliang, crop\_id, price\_per\_kg[crop\_id]);

if (profit > best\_profit) {

best\_profit = profit;

best\_crop = crop\_id;

}

}

}

}

if (best\_crop != -1) {

plan\_2025[di\_id][best\_crop] = 1;

}

}

// 输出2024年的种植方案

cout << "2024年种植方案：" << endl;

for (int di\_id = 1; di\_id <= 26; ++di\_id) {

cout << "地块 " << di\_id << ": ";

for (int crop\_id = 1; crop\_id <= 15; ++crop\_id) {

if (plan\_2024[di\_id][crop\_id]) {

cout << "种植作物 " << crop\_id << " ";

}

}

cout << endl;

}

// 输出2025年的种植方案

cout << "2025年种植方案：" << endl;

for (int di\_id = 1; di\_id <= 26; ++di\_id) {

cout << "地块 " << di\_id << ": ";

for (int crop\_id = 1; crop\_id <= 15; ++crop\_id) {

if (plan\_2025[di\_id][crop\_id]) {

cout << "种植作物 " << crop\_id << " ";

}

}

cout << endl;

}

}

int main() {

// 初始化2023年的种植数据，假设数据来源外部输入

ji\_1[1][6] = 1;

ji\_1[2][7] = 1;

ji\_1[3][7] = 1;

ji\_1[4][1] = 1;

ji\_1[5][4] = 1;

ji\_1[6][8] = 1;

ji\_1[7][6] = 1;

ji\_1[8][2] = 1;

ji\_1[9][3] = 1;

ji\_1[10][4] = 1;

ji\_1[11][5] = 1;

ji\_1[12][8] = 1;

ji\_1[13][6] = 1;

ji\_1[14][8] = 1;

ji\_1[15][9] = 1;

ji\_1[16][10] = 1;

ji\_1[17][1] = 1;

ji\_1[18][7] = 1;

ji\_1[19][14] = 1;

ji\_1[20][15] = 1;

ji\_1[21][11] = 1;

ji\_1[22][12] = 1;

ji\_1[23][1] = 1;

ji\_1[24][13] = 1;

ji\_1[25][6] = 1;

ji\_1[26][3] = 1;

solve(); // 运行问题解决函数

return 0;

}

**旱地 q2**

#include <bits/stdc++.h>

#include <random> // 用于随机数生成器

#include <chrono> // 用于chrono库函数

using namespace std;

typedef long long ll;

// 随机数生成器，使用chrono库获取当前时间作为种子

mt19937 rng(chrono::steady\_clock::now().time\_since\_epoch().count());

// 作物种类编号

int zhong[16] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15};

// 地块分类

int di\_leixing[27] = {0, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3};

// 地块面积

double di[27] = {0, 80, 55, 35, 72, 68, 55, 60, 46, 40, 28, 25, 26, 55, 44, 50, 25, 60, 45, 35, 20, 15, 13, 15, 18, 27, 20};

// 2023年种植记录

int ji\_1[27][16]; // 2023年各地块的种植信息，数组存储某地块的种植作物

// 每个农作物不超过的销量（2023年基准销量）

int z\_xiao[16] = {0, 57000, 21850, 22400, 33040, 9875, 170840, 132785, 71400, 30000, 12500, 1500, 35100, 36000, 14000, 10000};

// 不同地块的产量调整系数

double shanpo\_changliang\_xishu = 0.95; // 山坡地的产量系数

double gandi\_changliang\_xishu = 1.05; // 干旱地的产量系数

// 记录每个地块上次种植豆类作物的年份

int last\_douli\_year[27] = {0}; // 0表示从未种植过

// 随机生成在指定范围内的浮点数

double random\_double(double min\_val, double max\_val) {

uniform\_real\_distribution<double> dist(min\_val, max\_val);

return dist(rng);

}

// 判断是否为豆类作物，豆类作物为编号1-5

bool is\_douli(int crop\_id) {

return crop\_id >= 1 && crop\_id <= 5;

}

// 计算地块的产量系数

double get\_changliang\_xishu(int di\_id) {

if (di\_leixing[di\_id] == 1)

return gandi\_changliang\_xishu; // 干旱地

if (di\_leixing[di\_id] == 3)

return shanpo\_changliang\_xishu; // 山坡地

return 1.0; // 梯田

}

// 计算某一作物在某一地块的实际产量

double calc\_changliang(int di\_id, int crop\_id, double changliang\_per\_mu) {

double xishu = get\_changliang\_xishu(di\_id);

return di[di\_id] \* changliang\_per\_mu \* xishu;

}

// 判断是否满足销量约束

bool meet\_sales\_constraint(double total\_changliang, int crop\_id, double current\_sales\_limit) {

return total\_changliang <= current\_sales\_limit;

}

// 检查是否能种植（避免与去年的作物相同）

bool can\_plant(int di\_id, int crop\_id) {

// 去年的作物不能与今年种植的相同

return ji\_1[di\_id][crop\_id] == 0; // 2023年没有种植该作物，则可以种植

}

// 随机化作物选择顺序

vector<int> get\_random\_crop\_order() {

vector<int> crops = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15};

random\_shuffle(crops.begin(), crops.end()); // 打乱作物编号顺序

return crops;

}

// 解决问题的函数

void solve() {

double total\_area = 1201; // 总耕地面积

vector<vector<int>> plan\_2024(27, vector<int>(16, 0)); // 2024年的种植计划

vector<vector<int>> plan\_2025(27, vector<int>(16, 0)); // 2025年的种植计划

// 假设每种作物的亩产量（单位：产量/亩），考虑年变化

vector<double> changliang\_per\_mu = {0, 380, 475, 380, 330, 395, 760, 950, 380, 600, 500, 105, 2850, 2100, 400, 500};

// 更新未来每年的预期销售量和亩产量

vector<double> sales\_growth\_rate(16, 0.0); // 每种作物的年销量增长率

vector<double> cost\_growth\_rate(16, 0.05); // 每种作物的种植成本年增长率，固定为5%

// 设置销量增长率

for (int crop\_id = 1; crop\_id <= 15; ++crop\_id) {

if (crop\_id == 6 || crop\_id == 7) {

sales\_growth\_rate[crop\_id] = random\_double(0.05, 0.10); // 编号6和7作物年增长率在5%到10%之间

} else {

sales\_growth\_rate[crop\_id] = random\_double(-0.05, 0.05); // 其他作物年销量在±5%之间

}

}

// 2024年种植计划

for (int di\_id = 1; di\_id <= 26; ++di\_id) {

vector<int> crops\_order = get\_random\_crop\_order();

bool planted\_douli = false;

for (int crop\_id : crops\_order) {

if (can\_plant(di\_id, crop\_id)) {

// 更新2024年的亩产量，考虑年变化

double changliang\_per\_mu\_2024 = changliang\_per\_mu[crop\_id] \* (1.0 + random\_double(-0.1, 0.1));

// 更新2024年的销量限制，考虑增长率

double sales\_limit\_2024 = z\_xiao[crop\_id] \* (1.0 + sales\_growth\_rate[crop\_id]);

if (is\_douli(crop\_id) && last\_douli\_year[di\_id] <= 2021) { // 优先种植豆类作物

double changliang = calc\_changliang(di\_id, crop\_id, changliang\_per\_mu\_2024);

if (meet\_sales\_constraint(changliang, crop\_id, sales\_limit\_2024)) {

plan\_2024[di\_id][crop\_id] = 1; // 安排种植

last\_douli\_year[di\_id] = 2024; // 记录豆类种植年份

planted\_douli = true;

break;

}

}

}

}

// 如果2024年没有种植豆类，再按随机顺序选择

if (!planted\_douli) {

for (int crop\_id : crops\_order) {

if (can\_plant(di\_id, crop\_id)) {

double changliang\_per\_mu\_2024 = changliang\_per\_mu[crop\_id] \* (1.0 + random\_double(-0.1, 0.1));

double sales\_limit\_2024 = z\_xiao[crop\_id] \* (1.0 + sales\_growth\_rate[crop\_id]);

double changliang = calc\_changliang(di\_id, crop\_id, changliang\_per\_mu\_2024);

if (meet\_sales\_constraint(changliang, crop\_id, sales\_limit\_2024)) {

plan\_2024[di\_id][crop\_id] = 1; // 安排种植

if (is\_douli(crop\_id))

last\_douli\_year[di\_id] = 2024; // 更新豆类种植年份

break;

}

}

}

}

}

// 2025年种植计划

for (int di\_id = 1; di\_id <= 26; ++di\_id) {

vector<int> crops\_order = get\_random\_crop\_order();

bool planted\_douli = false;

for (int crop\_id : crops\_order) {

if (can\_plant(di\_id, crop\_id)) {

// 更新2025年的亩产量和销量限制

double changliang\_per\_mu\_2025 = changliang\_per\_mu[crop\_id] \* (1.0 + random\_double(-0.1, 0.1));

double sales\_limit\_2025 = z\_xiao[crop\_id] \* (1.0 + sales\_growth\_rate[crop\_id] \* 2); // 第二年增长两年

if (is\_douli(crop\_id) && last\_douli\_year[di\_id] <= 2022) { // 三年内必须种植一次豆类

double changliang = calc\_changliang(di\_id, crop\_id, changliang\_per\_mu\_2025);

if (meet\_sales\_constraint(changliang, crop\_id, sales\_limit\_2025)) {

plan\_2025[di\_id][crop\_id] = 1;

last\_douli\_year[di\_id] = 2025;

planted\_douli = true;

break;

}

}

}

}

// 如果2025年没有种植豆类，再按随机顺序选择

if (!planted\_douli) {

for (int crop\_id : crops\_order) {

if (can\_plant(di\_id, crop\_id)) {

double changliang\_per\_mu\_2025 = changliang\_per\_mu[crop\_id] \* (1.0 + random\_double(-0.1, 0.1));

double sales\_limit\_2025 = z\_xiao[crop\_id] \* (1.0 + sales\_growth\_rate[crop\_id] \* 2);

double changliang = calc\_changliang(di\_id, crop\_id, changliang\_per\_mu\_2025);

if (meet\_sales\_constraint(changliang, crop\_id, sales\_limit\_2025)) {

plan\_2025[di\_id][crop\_id] = 1;

if (is\_douli(crop\_id))

last\_douli\_year[di\_id] = 2025;

break;

}

}

}

}

}

// 输出2024和2025年的种植方案

cout << "2024年种植方案：" << endl;

for (int di\_id = 1; di\_id <= 26; ++di\_id) {

cout << "地块 " << di\_id << ": ";

for (int crop\_id = 1; crop\_id <= 15; ++crop\_id) {

if (plan\_2024[di\_id][crop\_id]) {

cout << "种植作物 " << crop\_id << " ";

}

}

cout << endl;

}

cout << "2025年种植方案：" << endl;

for (int di\_id = 1; di\_id <= 26; ++di\_id) {

cout << "地块 " << di\_id << ": ";

for (int crop\_id = 1; crop\_id <= 15; ++crop\_id) {

if (plan\_2025[di\_id][crop\_id]) {

cout << "种植作物 " << crop\_id << " ";

}

}

cout << endl;

}

}

int main() {

// 初始化2023年的种植数据，假设数据来源外部输入

// ji\_1[地块编号][作物编号] = 1 表示2023年在某块地种了某作物

ji\_1[1][6] = 1;

ji\_1[2][7] = 1;

ji\_1[3][7] = 1;

ji\_1[4][1] = 1;

ji\_1[5][4] = 1;

ji\_1[6][8] = 1;

ji\_1[7][6] = 1;

ji\_1[8][2] = 1;

ji\_1[9][3] = 1;

ji\_1[10][4] = 1;

ji\_1[11][5] = 1;

ji\_1[12][8] = 1;

ji\_1[13][6] = 1;

ji\_1[14][8] = 1;

ji\_1[15][9] = 1;

ji\_1[16][10] = 1;

ji\_1[17][1] = 1;

ji\_1[18][7] = 1;

ji\_1[19][14] = 1;

ji\_1[20][15] = 1;

ji\_1[21][11] = 1;

ji\_1[22][12] = 1;

ji\_1[23][1] = 1;

ji\_1[24][13] = 1;

ji\_1[25][6] = 1;

ji\_1[26][3] = 1;

solve(); // 运行问题解决函数

return 0;

}

**水地q1 s1**

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

// 作物种类编号

int zhong[19] = { 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18 };

// 地块分类

int di\_leixing[33] = { 0,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,3,3,3 };

// 地块面积

double di[33] = { 0,15,10,14,6,10,12,22,20 };

void \_\_int()

{

for (int i = 9; i < 33; i++)di[i] = 0.6;

}

// 2023年种植记录

int ji\_1[33][19]; // 2023年各地块的种植信息，数组存储某地块的种植作物

// 每个农作物不超过的销量

int z\_xiao[19] = { 0,36240,26880,6240,30000,36210,45360,900,2610,3480,3930,4500,35480,13050,2850,1200,3300,1620,1800 };

// 不同地块的产量调整系数

double shui\_changliang\_xishu = 0.8; // 山坡地的产量系数

double zhier\_changliang\_xishu = 0.9; // 干旱地的产量系数

// 判断是否为豆类作物，豆类作物为编号1-3

bool is\_douli(int crop\_id) {

return crop\_id >= 1 && crop\_id <= 3;

}

// 计算地块的产量系数

double get\_changliang\_xishu(int di\_id) {

if (di\_leixing[di\_id] == 1) return shui\_changliang\_xishu; // 干旱地

if (di\_leixing[di\_id] == 3) return zhier\_changliang\_xishu; // 山坡地

return 1.0; // 梯田

}

// 计算某一作物在某一地块的实际产量

double calc\_changliang(int di\_id, int crop\_id, double changliang\_per\_mu) {

double xishu = get\_changliang\_xishu(di\_id);

return di[di\_id] \* changliang\_per\_mu \* xishu;

}

// 判断是否满足销量约束

bool meet\_sales\_constraint(double total\_changliang, int crop\_id) {

return total\_changliang <= z\_xiao[crop\_id];

}

// 检查是否能种植（避免与去年的作物相同）

bool can\_plant(int di\_id, int crop\_id, const vector<vector<int>>& previous\_plan) {

// 去年的作物不能与今年种植的相同

return ji\_1[di\_id][crop\_id] == 0 && previous\_plan[di\_id][crop\_id] == 0; // 2023年和上一年的计划没有种植该作物

}

// 随机化作物选择顺序

vector<int> get\_random\_crop\_order() {

vector<int> crops = { 13,16,6,1,10,18,17,14,11,3,12,9,5,7,2,8,15,4 };

random\_shuffle(crops.begin(), crops.end()); // 打乱作物编号顺序

return crops;

}

// 解决问题的函数

void solve() {

vector<vector<int>> plan\_2024(33, vector<int>(19, 0)); // 2024年的种植计划

vector<vector<int>> plan\_2025(33, vector<int>(19, 0)); // 2025年的种植计划

// 假设每种作物的亩产量（单位：产量/亩）

vector<double> changliang\_per\_mu = { 0,3600,2400,3600,2400,3000,8000,3300,3000,4000,4500,5000,4000,15000,5000,2000,12000,6000,6600 };

// 遍历地块

for (int di\_id = 1; di\_id <= 32; ++di\_id) {

double used\_area = 0.0; // 已使用的地块面积

// 获取随机化的作物顺序

vector<int> crops\_order = get\_random\_crop\_order();

// 遍历随机化后的作物，安排种植

for (int crop\_id : crops\_order) {

// 确保不与去年种植的作物相同

if (can\_plant(di\_id, crop\_id, plan\_2024) && used\_area < di[di\_id]) {

double changliang = calc\_changliang(di\_id, crop\_id, changliang\_per\_mu[crop\_id]);

if (meet\_sales\_constraint(changliang, crop\_id)) {

plan\_2024[di\_id][crop\_id] = 1; // 安排种植

used\_area += di[di\_id] / 2; // 假设每块地可以种两种作物，分配一半面积给作物

if (used\_area >= di[di\_id]) break; // 地块面积满了

}

}

}

}

// 2025年重复类似操作

for (int di\_id = 1; di\_id <= 32; ++di\_id) {

double used\_area = 0.0; // 已使用的地块面积

vector<int> crops\_order = get\_random\_crop\_order();

for (int crop\_id : crops\_order) {

if (can\_plant(di\_id, crop\_id, plan\_2025) && used\_area < di[di\_id]) {

double changliang = calc\_changliang(di\_id, crop\_id, changliang\_per\_mu[crop\_id]);

if (meet\_sales\_constraint(changliang, crop\_id)) {

plan\_2025[di\_id][crop\_id] = 1;

used\_area += di[di\_id] / 2; // 每块地最多种两种作物

if (used\_area >= di[di\_id]) break; // 地块面积满了

}

}

}

}

// 输出2024和2025年的种植方案

cout << "2024年种植方案：" << endl;

for (int di\_id = 1; di\_id <= 32; ++di\_id) {

cout << "地块 " << di\_id << ": ";

for (int crop\_id = 1; crop\_id <= 18; ++crop\_id) {

if (plan\_2024[di\_id][crop\_id]) {

cout << "种植作物 " << crop\_id << " ";

}

}

cout << endl;

}

cout << "2025年种植方案：" << endl;

for (int di\_id = 1; di\_id <= 32; ++di\_id) {

cout << "地块 " << di\_id << ": ";

for (int crop\_id = 1; crop\_id <= 18; ++crop\_id) {

if (plan\_2025[di\_id][crop\_id]) {

cout << "种植作物 " << crop\_id << " ";

}

}

cout << endl;

}

}

int main() {

\_\_int();

// 初始化2023年种植记录...

ji\_1[1][4] = 1;

ji\_1[2][12] = 1;

ji\_1[3][5] = 1;

ji\_1[4][6] = 1;

ji\_1[5][1] = 1;

ji\_1[6][2] = 1;

ji\_1[7][0] = 1;

ji\_1[8][0] = 1;

ji\_1[9][2] = 1;

ji\_1[10][8] = 1;

ji\_1[11][9] = 1;

ji\_1[12][10] = 1;

ji\_1[13][12] = 1;

ji\_1[14][11] = 1;

ji\_1[15][3] = 1;

ji\_1[16][3] = 1;

ji\_1[17][2] = 1;

ji\_1[18][1] = 1;

ji\_1[19][1] = 1;

ji\_1[20][6] = 1;

ji\_1[21][5] = 1;

ji\_1[22][13] = 1;

ji\_1[23][14] = 1;

ji\_1[24][15] = 1;

ji\_1[25][16] = 1; ji\_1[25][17] = 1;

ji\_1[26][9] = 1; ji\_1[26][10] = 1;

ji\_1[27][1] = 1;

ji\_1[28][3] = 1;

ji\_1[29][8] = 1; ji\_1[29][5] = 1;

ji\_1[30][13] = 1; ji\_1[30][6] = 1;

ji\_1[31][12] = 1; ji\_1[31][14] = 1;

ji\_1[32][18] = 1; ji\_1[32][7] = 1;

solve(); // 运行问题解决函数

return 0;

}

**水地 q1 s2**

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

// 作物种类编号

int zhong[19] = { 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18 };

// 地块分类

int di\_leixing[33] = { 0,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,3,3,3 };

// 地块面积

double di[33] = { 0,15,10,14,6,10,12,22,20 };

void \_\_int()

{

for (int i = 9; i < 33; i++)di[i] = 0.6;

}

// 2023年种植记录

int ji\_1[33][19]; // 2023年各地块的种植信息，数组存储某地块的种植作物

// 每个农作物不超过的销量（2023年基准销量）

int z\_xiao[19] = { 0,36240,26880,6240,30000,36210,45360,900,2610,3480,3930,4500,35480,13050,2850,1200,3300,1620,1800 };

// 每个作物的初始售价（假设单位为100元/单位产量）

double price[19] = { 0, 10, 8, 15, 12, 9, 7, 20, 18, 22, 25, 30, 28, 35, 40, 45, 50, 55, 60 };

// 不同地块的产量调整系数

double shui\_changliang\_xishu = 0.8; // 山坡地的产量系数

double zhier\_changliang\_xishu = 0.9; // 干旱地的产量系数

// 判断是否为豆类作物，豆类作物为编号1-3

bool is\_douli(int crop\_id) {

return crop\_id >= 1 && crop\_id <= 3;

}

// 计算地块的产量系数

double get\_changliang\_xishu(int di\_id) {

if (di\_leixing[di\_id] == 1) return shui\_changliang\_xishu; // 干旱地

if (di\_leixing[di\_id] == 3) return zhier\_changliang\_xishu; // 山坡地

return 1.0; // 梯田

}

// 计算某一作物在某一地块的实际产量

double calc\_changliang(int di\_id, int crop\_id, double changliang\_per\_mu) {

double xishu = get\_changliang\_xishu(di\_id);

return di[di\_id] \* changliang\_per\_mu \* xishu;

}

// 计算某一作物在某一地块的销售利润

double calc\_profit(int crop\_id, double total\_changliang) {

double regular\_sales = min(total\_changliang, (double)z\_xiao[crop\_id]); // 正常价格销售部分

double discounted\_sales = max(0.0, total\_changliang - z\_xiao[crop\_id]); // 超出限制的部分

// 总利润 = 正常销售部分的收入 + 超出部分的收入（50%降价）

double profit = (regular\_sales \* price[crop\_id]) + (discounted\_sales \* price[crop\_id] \* 0.5);

return profit;

}

// 检查是否能种植（避免与去年的作物相同）

bool can\_plant(int di\_id, int crop\_id, const vector<vector<int>>& previous\_plan) {

// 去年的作物不能与今年种植的相同

return ji\_1[di\_id][crop\_id] == 0 && previous\_plan[di\_id][crop\_id] == 0; // 2023年和上一年的计划没有种植该作物

}

// 随机化作物选择顺序

vector<int> get\_random\_crop\_order() {

vector<int> crops = { 13,16,6,1,10,18,17,14,11,3,12,9,5,7,2,8,15,4 };

random\_shuffle(crops.begin(), crops.end()); // 打乱作物编号顺序

return crops;

}

// 解决问题的函数

void solve() {

vector<vector<int>> plan\_2024(33, vector<int>(19, 0)); // 2024年的种植计划

vector<vector<int>> plan\_2025(33, vector<int>(19, 0)); // 2025年的种植计划

// 假设每种作物的亩产量（单位：产量/亩）

vector<double> changliang\_per\_mu = { 0,3600,2400,3600,2400,3000,8000,3300,3000,4000,4500,5000,4000,15000,5000,2000,12000,6000,6600 };

// 遍历地块

for (int di\_id = 1; di\_id <= 32; ++di\_id) {

double used\_area = 0.0; // 已使用的地块面积

double max\_profit = -1; // 最大利润

int best\_crop = -1; // 最佳作物选择

// 获取随机化的作物顺序

vector<int> crops\_order = get\_random\_crop\_order();

// 遍历随机化后的作物，安排种植

for (int crop\_id : crops\_order) {

// 确保不与去年种植的作物相同

if (can\_plant(di\_id, crop\_id, plan\_2024) && used\_area < di[di\_id]) {

double changliang = calc\_changliang(di\_id, crop\_id, changliang\_per\_mu[crop\_id]);

double profit = calc\_profit(crop\_id, changliang);

if (profit > max\_profit) {

max\_profit = profit;

best\_crop = crop\_id;

}

}

}

// 安排种植最大利润的作物

if (best\_crop != -1 && used\_area < di[di\_id]) {

plan\_2024[di\_id][best\_crop] = 1; // 安排种植

used\_area += di[di\_id] / 2; // 假设每块地可以种两种作物，分配一半面积给作物

}

}

// 2025年重复类似操作

for (int di\_id = 1; di\_id <= 32; ++di\_id) {

double used\_area = 0.0; // 已使用的地块面积

double max\_profit = -1; // 最大利润

int best\_crop = -1; // 最佳作物选择

vector<int> crops\_order = get\_random\_crop\_order();

for (int crop\_id : crops\_order) {

if (can\_plant(di\_id, crop\_id, plan\_2025) && used\_area < di[di\_id]) {

double changliang = calc\_changliang(di\_id, crop\_id, changliang\_per\_mu[crop\_id]);

double profit = calc\_profit(crop\_id, changliang);

if (profit > max\_profit) {

max\_profit = profit;

best\_crop = crop\_id;

}

}

}

// 安排种植最大利润的作物

if (best\_crop != -1 && used\_area < di[di\_id]) {

plan\_2025[di\_id][best\_crop] = 1;

used\_area += di[di\_id] / 2; // 每块地最多种两种作物

}

}

// 输出2024和2025年的种植方案

cout << "2024年种植方案：" << endl;

for (int di\_id = 1; di\_id <= 32; ++di\_id) {

cout << "地块 " << di\_id << ": ";

for (int crop\_id = 1; crop\_id <= 18; ++crop\_id) {

if (plan\_2024[di\_id][crop\_id]) {

cout << "种植作物 " << crop\_id << " ";

}

}

cout << endl;

}

cout << "2025年种植方案：" << endl;

for (int di\_id = 1; di\_id <= 32; ++di\_id) {

cout << "地块 " << di\_id << ": ";

for (int crop\_id = 1; crop\_id <= 18; ++crop\_id) {

if (plan\_2025[di\_id][crop\_id]) {

cout << "种植作物 " << crop\_id << " ";

}

}

cout << endl;

}

}

int main() {

\_\_int();

// 初始化2023年种植记录...

ji\_1[1][4] = 1;

ji\_1[2][12] = 1;

ji\_1[3][5] = 1;

ji\_1[4][6] = 1;

ji\_1[5][1] = 1;

ji\_1[6][2] = 1;

ji\_1[7][0] = 1;

ji\_1[8][0] = 1;

ji\_1[9][2] = 1;

ji\_1[10][8] = 1;

ji\_1[11][9] = 1;

ji\_1[12][10] = 1;

ji\_1[13][12] = 1;

ji\_1[14][11] = 1;

ji\_1[15][3] = 1;

ji\_1[16][3] = 1;

ji\_1[17][2] = 1;

ji\_1[18][1] = 1;

ji\_1[19][1] = 1;

ji\_1[20][6] = 1;

ji\_1[21][5] = 1;

ji\_1[22][13] = 1;

ji\_1[23][14] = 1;

ji\_1[24][15] = 1;

ji\_1[25][16] = 1; ji\_1[25][17] = 1;

ji\_1[26][9] = 1; ji\_1[26][10] = 1;

ji\_1[27][1] = 1;

ji\_1[28][3] = 1;

ji\_1[29][8] = 1; ji\_1[29][5] = 1;

ji\_1[30][13] = 1; ji\_1[30][6] = 1;

ji\_1[31][12] = 1; ji\_1[31][14] = 1;

ji\_1[32][18] = 1; ji\_1[32][7] = 1;

solve(); // 运行问题解决函数

return 0;

}

**水地 q2**

#include <bits/stdc++.h>

#include <random> // 用于随机数生成器

#include <chrono> // 用于chrono库函数

using namespace std;

typedef long long ll;

// 随机数生成器，使用chrono库获取当前时间作为种子

mt19937 rng(chrono::steady\_clock::now().time\_since\_epoch().count());

// 作物种类编号

int zhong[19] = { 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18 };

// 地块分类

int di\_leixing[33] = { 0,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,3,3,3 };

// 地块面积

double di[33] = { 0,15,10,14,6,10,12,22,20 };

void \_\_int()

{

for (int i = 9; i < 33; i++)di[i] = 0.6;

}

// 2023年种植记录

int ji\_1[33][19]; // 2023年各地块的种植信息，数组存储某地块的种植作物

// 每个农作物不超过的销量（2023年基准销量）

int z\_xiao[19] = { 0,36240,26880,6240,30000,36210,45360,900,2610,3480,3930,4500,35480,13050,2850,1200,3300,1620,1800 };

// 不同地块的产量调整系数

double shui\_changliang\_xishu = 0.8; // 山坡地的产量系数

double zhier\_changliang\_xishu = 0.9; // 干旱地的产量系数

// 随机生成在指定范围内的浮点数

double random\_double(double min\_val, double max\_val) {

uniform\_real\_distribution<double> dist(min\_val, max\_val);

return dist(rng);

}

// 判断是否为豆类作物，豆类作物为编号1-3

bool is\_douli(int crop\_id) {

return crop\_id >= 1 && crop\_id <= 3;

}

// 判断是否为蔬菜类作物，假设编号4-10为蔬菜类作物

bool is\_vegetable(int crop\_id) {

return crop\_id >= 4 && crop\_id <= 10;

}

// 计算地块的产量系数

double get\_changliang\_xishu(int di\_id) {

if (di\_leixing[di\_id] == 1) return shui\_changliang\_xishu; // 干旱地

if (di\_leixing[di\_id] == 3) return zhier\_changliang\_xishu; // 山坡地

return 1.0; // 梯田

}

// 计算某一作物在某一地块的实际产量

double calc\_changliang(int di\_id, int crop\_id, double changliang\_per\_mu) {

double xishu = get\_changliang\_xishu(di\_id);

return di[di\_id] \* changliang\_per\_mu \* xishu;

}

// 判断是否满足销量约束

bool meet\_sales\_constraint(double total\_changliang, int crop\_id, double current\_sales\_limit) {

return total\_changliang <= current\_sales\_limit;

}

// 检查是否能种植（避免与去年的作物相同）

bool can\_plant(int di\_id, int crop\_id, const vector<vector<int>>& previous\_plan) {

// 去年的作物不能与今年种植的相同

return ji\_1[di\_id][crop\_id] == 0 && previous\_plan[di\_id][crop\_id] == 0; // 2023年和上一年的计划没有种植该作物

}

// 随机化作物选择顺序

vector<int> get\_random\_crop\_order() {

vector<int> crops = { 13,16,6,1,10,18,17,14,11,3,12,9,5,7,2,8,15,4 };

random\_shuffle(crops.begin(), crops.end()); // 打乱作物编号顺序

return crops;

}

// 解决问题的函数

void solve() {

vector<vector<int>> plan\_2024(33, vector<int>(19, 0)); // 2024年的种植计划

vector<vector<int>> plan\_2025(33, vector<int>(19, 0)); // 2025年的种植计划

// 假设每种作物的亩产量（单位：产量/亩）

vector<double> changliang\_per\_mu = { 0,3600,2400,3600,2400,3000,8000,3300,3000,4000,4500,5000,4000,15000,5000,2000,12000,6000,6600 };

// 更新未来每年的预期销售量和亩产量

vector<double> sales\_growth\_rate(19, 0.0); // 每种作物的年销量增长率

vector<double> price\_growth\_rate(19, 0.0); // 蔬菜类作物的年销售价格增长率

vector<double> cost\_growth\_rate(19, 0.05); // 每种作物的种植成本年增长率，固定为5%

// 设置销量增长率和销售价格增长率

for (int crop\_id = 1; crop\_id <= 18; ++crop\_id) {

sales\_growth\_rate[crop\_id] = random\_double(-0.05, 0.05); // 每种作物的年销量在±5%之间变化

if (is\_vegetable(crop\_id)) {

price\_growth\_rate[crop\_id] = random\_double(0.03, 0.07); // 蔬菜类作物的年价格增长率在3%到7%之间

}

}

// 遍历地块

for (int di\_id = 1; di\_id <= 32; ++di\_id) {

double used\_area = 0.0; // 已使用的地块面积

// 获取随机化的作物顺序

vector<int> crops\_order = get\_random\_crop\_order();

// 遍历随机化后的作物，安排种植

for (int crop\_id : crops\_order) {

// 确保不与去年种植的作物相同

if (can\_plant(di\_id, crop\_id, plan\_2024) && used\_area < di[di\_id]) {

// 计算2024年的亩产量和销量限制

double changliang\_per\_mu\_2024 = changliang\_per\_mu[crop\_id] \* (1.0 + random\_double(-0.1, 0.1)); // 每年变化在±10%

double sales\_limit\_2024 = z\_xiao[crop\_id] \* (1.0 + sales\_growth\_rate[crop\_id]); // 每年变化在±5%

if (is\_vegetable(crop\_id)) {

double price\_2024 = 100 \* (1.0 + price\_growth\_rate[crop\_id]); // 假设初始价格为100，增长趋势为3%到7%

cout << "作物 " << crop\_id << " 在2024年的预期销售价格: " << price\_2024 << endl;

}

double changliang = calc\_changliang(di\_id, crop\_id, changliang\_per\_mu\_2024);

if (meet\_sales\_constraint(changliang, crop\_id, sales\_limit\_2024)) {

plan\_2024[di\_id][crop\_id] = 1; // 安排种植

used\_area += di[di\_id] / 2; // 假设每块地可以种两种作物，分配一半面积给作物

if (used\_area >= di[di\_id]) break; // 地块面积满了

}

}

}

}

// 2025年重复类似操作

for (int di\_id = 1; di\_id <= 32; ++di\_id) {

double used\_area = 0.0; // 已使用的地块面积

vector<int> crops\_order = get\_random\_crop\_order();

for (int crop\_id : crops\_order) {

if (can\_plant(di\_id, crop\_id, plan\_2025) && used\_area < di[di\_id]) {

double changliang\_per\_mu\_2025 = changliang\_per\_mu[crop\_id] \* (1.0 + random\_double(-0.1, 0.1)); // 每年变化在±10%

double sales\_limit\_2025 = z\_xiao[crop\_id] \* (1.0 + sales\_growth\_rate[crop\_id] \* 2); // 第二年增长两年

if (is\_vegetable(crop\_id)) {

double price\_2025 = 100 \* pow(1.0 + price\_growth\_rate[crop\_id], 2); // 假设初始价格为100，第二年价格增长率在3%到7%之间

cout << "作物 " << crop\_id << " 在2025年的预期销售价格: " << price\_2025 << endl;

}

double changliang = calc\_changliang(di\_id, crop\_id, changliang\_per\_mu\_2025);

if (meet\_sales\_constraint(changliang, crop\_id, sales\_limit\_2025)) {

plan\_2025[di\_id][crop\_id] = 1;

used\_area += di[di\_id] / 2; // 每块地最多种两种作物

if (used\_area >= di[di\_id]) break; // 地块面积满了

}

}

}

}

// 输出2024和2025年的种植方案

cout << "2024年种植方案：" << endl;

for (int di\_id = 1; di\_id <= 32; ++di\_id) {

cout << "地块 " << di\_id << ": ";

for (int crop\_id = 1; crop\_id <= 18; ++crop\_id) {

if (plan\_2024[di\_id][crop\_id]) {

cout << "种植作物 " << crop\_id << " ";

}

}

cout << endl;

}

cout << "2025年种植方案：" << endl;

for (int di\_id = 1; di\_id <= 32; ++di\_id) {

cout << "地块 " << di\_id << ": ";

for (int crop\_id = 1; crop\_id <= 18; ++crop\_id) {

if (plan\_2025[di\_id][crop\_id]) {

cout << "种植作物 " << crop\_id << " ";

}

}

cout << endl;

}

}

int main() {

\_\_int();

// 初始化2023年种植记录...

ji\_1[1][4] = 1;

ji\_1[2][12] = 1;

ji\_1[3][5] = 1;

ji\_1[4][6] = 1;

ji\_1[5][1] = 1;

ji\_1[6][2] = 1;

ji\_1[7][0] = 1;

ji\_1[8][0] = 1;

ji\_1[9][2] = 1;

ji\_1[10][8] = 1;

ji\_1[11][9] = 1;

ji\_1[12][10] = 1;

ji\_1[13][12] = 1;

ji\_1[14][11] = 1;

ji\_1[15][3] = 1;

ji\_1[16][3] = 1;

ji\_1[17][2] = 1;

ji\_1[18][1] = 1;

ji\_1[19][1] = 1;

ji\_1[20][6] = 1;

ji\_1[21][5] = 1;

ji\_1[22][13] = 1;

ji\_1[23][14] = 1;

ji\_1[24][15] = 1;

ji\_1[25][16] = 1; ji\_1[25][17] = 1;

ji\_1[26][9] = 1; ji\_1[26][10] = 1;

ji\_1[27][1] = 1;

ji\_1[28][3] = 1;

ji\_1[29][8] = 1; ji\_1[29][5] = 1;

ji\_1[30][13] = 1; ji\_1[30][6] = 1;

ji\_1[31][12] = 1; ji\_1[31][14] = 1;

ji\_1[32][18] = 1; ji\_1[32][7] = 1;

solve(); // 运行问题解决函数

return 0;

}