基于规划模型与贪心算法的农作物种植策略

摘要

本文讨论农作物种植策略优化问题，建立了农作物种植优化模型，使用非线性规划，贪心策略，分治思想等方法对乡村的农作物种植方案进行了求解，得到了 2024-2030 年最优的种植策略和相应的经济收益预测。通过对不同情景下的农作物产量和销售情况进行模拟，得出了一些切实可行的种植建议。

针对问题一：本文根据 2023 年农作物的种植情况，针对多余产量的两种处理方案：1）滞销处理；2）按 2023 年 50%售价销售，探求利润最大化目标的最优种植策略。首先，采用分治思想，将土地划分为四类：平旱地、梯田、山坡地适合种植粮食作物（水稻除外），水浇地适合种植水稻或双季蔬菜，大棚则适宜种植蔬菜和菌类作物。在问题一的第（1）小问中，利用线性规划算法求解各类土地的种植方案，得出在滞销情况下的最佳种植策略，显著减少浪费并降低利润损失。针对第（2）小问，采用非线性优化与贪心算法，逐步选择利润最高的作物，最终确定多余产量按 50%售价销售时的最佳种植方案。结果显示，当滞销造成浪费时，总收益与 23 年的总收益比较有所下降，而降价 50%销售多余部分时，损失相对较小。依此得出，降价处理多余产量能够有效减少收益损失。

针对问题二：考虑可能出现的销售量变化，亩产量的变化以及成本的变化，我们利用问题一的模型，添加随机因子，通过多次随机模拟，我们以最后取平均值来替换我们的结果。结果表明，在考虑销售量波动的情形下，适度增加粮食作物的种植比例，能够有效提高整体收益，降低风险。此外，通过比较不同作物的种植策略，我们发现，粮食类作物的稳定性高于蔬菜类作物，因而在不确定性较大的市场条件下，优先选择粮食类作物能够更好地抵御市场波动的影响。综上所述，优化种植方案需综合考虑市场变化与作物特性，以实现长期的经济收益。

针对问题三：本问题进一步考虑了农作物之间的可替代性和互补性，以及销售量、销售价格、种植成本之间的相关性。基于问题 2 的基础，我们增加一些约束条件例如：销售量、销售价格、种植成本之间具有相关性等。综合考虑了不同作物的相互影响，以提出更为精确和可靠的农作物种植建议。最终的最优种植策略应综合考虑市场变化、气候影响和作物特性，旨在实现长期的经济收益和可持续发展。

关键词：农作物种植；非线性规划；贪心策略；分治思想

一、问题的重述

1.1 研究背景

中国耕地面积广袤、人口众多，自古以来都是“以农为本”的农业大国。研究表明健康的土壤有利于农业的可持续发展，因为它本身是个生态系统，系统平衡时有一定的自我修复功能，然而系统能否平衡与人类的耕作模式息息相关[1]。改变传统的种植观念，使农民更加科学合理地种植农作物，能提高农作物产量，促进农业发展[2]。所以合理规划土地资源，制定科学的种植方式的具有重要的现实意义。在此前提下，我们提出因地制宜的科学种植策略，即在不同土地类型、不同季节选择不同种类的农作物进行种植，以求充分利用有限的耕地资源发展乡村种植业，实现种植利润最大化，促进乡村经济可持续发展。

1.2 题目信息

- 每种作物在同一地块上不能重茬种植。
- 每三年中每个地块（大棚）的所有土地至少种植一次豆类作物，如果当年只种植该地块的部分土地，在未来两年内需保证该地块剩余部分土地至少种植一次豆类作物（且该豆类作物不与前一年种植的豆类作物相同）。
- 每一地块（大棚）每年可以混合种植不同种类的作物。
- 未来作物种植亩数、种植成本、亩产量和销售价格与 2023 年基本维持稳定，在此基础上进行不大于10% 的波动。
- 附件给出乡村目前可用耕地情况和种植农作物种类、2023 年种植农作物种植情况和销售利润的统计结果。

1.3 问题的提出

本文将要解决以下几个问题：

问题一：在农作物亩产量、种植成本、预期销售量和销售价格与 2023 年保持稳定的情况下，对于某作物的实际产量超过预期销量（供大于求）的多余部分将有以下 2 种方案进行处理：

方案(1)滞销处理；

方案(2)多余部分按 2023 年售价的50%进行售卖；

我们需分别给出在上述两种方案下的最优种植方案，实现利润最大化，并将结果填入所给附件。

问题二：在 2023 年给定数据下，各种农作物的预期销售量、亩产量、种植成本和销售价格发生了一定波动，建立数学模型，给出该乡村 2024~2030 年农作物的最优种植方案。

问题三：在问题二的基础上，考虑各种农作物之间存在一定的可替代性和互补性，预期销售量与销售价格、种植成本之间存在相关性，通过模拟数据求解该

乡村 2024—2030 最优种植策略，并将所得结果与问题二的比较分析。

二、问题的分析

对本文提出的最有种植策略问题，我们逐一做出如下分析：

2.1 问题一分析

本问要求根据 2023 年农作物的种植情况，针对多余的产量，分别按照两种方案处理：1）滞销处理；2）按 2023 年 50%售价销售，并给出这两种情况下的最佳方案。我们将**种植利润值**作为衡量最佳方案的标准，**首先**采用分治的思想，将所有土地按照所给的土地类型和适宜种植的作物种类进行划分，划分类型为图 1 中 4 个红框部分。这四类结合附件 1 中内容显示：**平旱地、梯田、山坡地**只适于种值粮食作物（水稻除外）；**水浇地单季**可种植单季作物（水稻）；**双季种植时无特殊情况土地，**水浇地第一季种植多种蔬菜作物（除大白菜和红/白萝卜），普通大棚第一季种植多种蔬菜作物（除大白菜和红/白萝卜），智慧大棚两季都种植多种蔬菜（除大白菜和红/白萝卜）；**双季种植时存在特殊种植情况土地，**水浇地第二季只能种大白菜、红/白萝卜，普通大棚第二季只能种植菌类作物；。

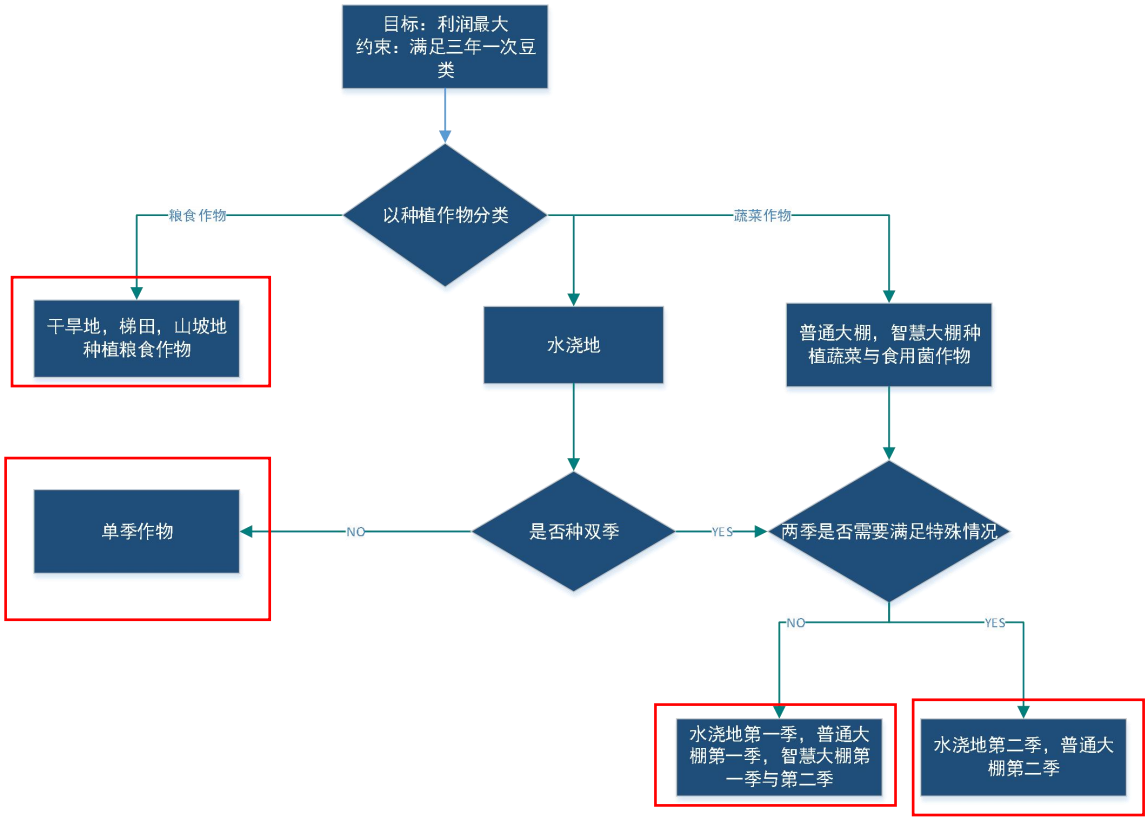

图 1 种植方案流程图

针对问题一第（1）小问，我们采用线性规化算法，在约束条件下求出 4 类情况下的利润最大时作物种植情况，然后将 4 类情况整合得出最佳方案；**针对题**

一第（2）小问，我们采用非线性优化和贪心算法在约束条件下，每一次都选择种植产生最大利润的作物，最终的到最佳方案。

2.2 问题二分析

在本问题中，我们需要为乡村制定 2024 至 2030 年的最优农作物种植方案，综合考虑未来销售量、亩产量、种植成本和销售价格的不确定性。小麦和玉米的销售量预计将以 5%至 10%的年增长率上升，而其他农作物的销售量相较于 2023 年则可能有±5%的波动。此外，亩产量受气候影响，每年可能波动±10%，这要求我们在模型中设置安全边际以应对不确定性。种植成本预计每年增长约 5%，因此需在制定种植方案时考虑这一因素以保障农民的盈利能力。粮食类作物的销售价格相对稳定，而蔬菜类作物每年有 5%的增长潜力，适合优先种植。相比之下，食用菌的销售价格则呈下降趋势，特别是羊肚菌每年下降约 5%，因此需谨慎选择是否种植。所以我们在第一问的模型下，总目标内需要添加随机因子，贪心策略则要考虑是否能够保证所种植的作物能够盈利，最终得到最佳方案。

2.3 问题三分析

本问题中，我们需要与现实接轨，考虑一些实际的因素，例如：各种农作物之间可能存在一定的可替代性和互补性，预期销售量与销售价格、种植成本之间存在一定的相关性等，例如，考虑可替代性的因素，我们可以用一种成本更低的作物来替代另一种高成本的作物，从而达到利润最大化的目标。而在得出预期销售量和销售价格、销售成本之间的相关性后，我们就可以根据销售价格和成本的波动及时地调整种植量，从而实现利润的最大化。因此，在问题二的基础上，我们要结合上述这些因素，在模型中添加更多的约束条件。

三、模型假设

针对本文提出的问题，我们做了如下模型假设：

1. 假设前一年的作物种植产量即为本年的预期销售量（例如 2024 年预期销量为 2023 年各作物生产量，以此类推）。

2. 假设种植成本种植过程中所有成本，包括劳动力、施肥、购种、打药等成本。

3. 假设超产打折销售部份每次都能完全售尽。

4. 假设农作物的未来的预期销售量、种植成本、亩产量和销售价格与题目要求情况完全相符

四、符号说明

本文常用符号见下表，其它符号见文中说明.

符号	符号说明
i	种植第i块地（i=1, 2, …, 54）.
j	表示种植作物编号（j=1, 2, …, 46）.
t	表示所处年份, 以 2023 为$t=0$, 2024 为$t=1$, 以此类推.
k	表示种植季节，对于第一季和单季时$k=1$，第二季$k=2$.
w_i	第i块地作物销售所得总利润.
x_{ijt}	表示在第i年于第j块地种植第种作物时所得亩产值.
x_{ijtk}	表示在第t年第k季于第i块地种植第j种作物时所得亩产.
β_j	表示第j中作物的在前一年总产量
I_{ijt}	定性变量，若第t年于第i块地种植第j种作物时$I_{ijt}=1$, 否则$I_{ijt}=0$.

五、作物种植数据统计分析

我们根据题目所给附件 2 中 2023 年各农作物亩产量、成本、销售单价和种植面积计算出 2023 年各作物总利润、总产量和总种植亩数。按照食用菌、粮食和蔬菜三种作物种类统计出如表 1、2、3 所示：

表 1 2023 年食用菌类作物统计数据表

作物名称	2023 作物利润（元）	2023 作物产量（斤）	23 年种植亩数
白灵菇	270000	18000	1.8
榆黄菇	512100	9000	1.8
香菇	133200	7200	1.8
羊肚菌	378000	4200	4.2

表 1 显示食用菌类作物种类少，总种植面积很少，未超过 10 亩。食用菌类作物中，榆黄菇种植亩数处于最低档，仅 1.8 亩，但作物利润巨大，虽然其产量仅为白灵菇的一半，但作物利润超过 50 万元，约为白灵菇所得利润的 2 倍，这表明该地非常适宜发展榆黄菇种植业。而羊肚菌虽然种植亩数为 4.2 亩，占食用菌作物种植面积的近一半，但其亩产量仅为 4200 斤，产量最低的情况下作物利

润位居第二，表明市场销售情况较好，能够以较为优异的价格出售，适宜未来继续发展。

表 2 2023 年粮食类作物统计数据表

作物名称	2023 作物利润（元）	2023 作物产量（斤）	23 年种植亩数
小麦	498040	170840	222
玉米	225000	132785	135
谷子	560825	71400	185
高粱	160000	30000	50
黍子	84750	12500	25
荞麦	54750	1500	15
南瓜	39650	35100	13
红薯	81000	36000	18
莜麦	63000	14000	35
大麦	28000	10000	20
水稻	118440	21000	42
黑豆	145475	21850	46
红豆	163800	22400	60
黄豆	256800	57000	147
绿豆	197680	33040	96
爬豆	57906.25	9875	25

文献显示，中国粮食的种植主体是自主经营的小规模农户。尽管大规模农业劳动力的非农迁移导致了对“谁来种粮”的担忧，但小农所特有的行为机理及其所诱致的农业种植的“趋粮化”[3]，根据国家政策和社会因素等影响，我国乡村地区大部分农民具有很强的“趋粮性”，因为一方面粮食作物容易种，不必担心市场，可用来避险。另一方面国家对于种植粮食的土地会按亩给予补贴，鼓励种粮。因此表 2 中数据显示四大主要粮食作物（小麦、玉米、谷子、黄豆）种植面积均在 100 亩以上不足为奇，这四大粮食作物的总体种植面积之和已经占到了总体耕地面积的近 60%，属于基础型种植业。小麦当年总种植亩数为 222 亩，超过整个乡村耕地的 1/6，产生利润近 50 万，表明它在该华北乡村农作物种植业中处于重要地位；第二大粮食作物为谷子，其种植面积较小，但作物单价高，市场行情好，谷子总体利润很高。粮食作物的种植面积差距较大，除四大主要粮食作物外，其余粮食作物种植面积基本上维持在 30 亩左右，用来平衡土壤环境，充分利用土地微量元素。

表 3 2023 年蔬菜类作物统计数据表

作物名称	2023 作物利润（元）	2023 作物产量（斤）	23 年种植亩数
白萝卜	237500	100000	25
包菜	23889	3930	0.9
菠菜	5310	900	0.3

菜花	17538	3480	0.9
大白菜	315000	150000	30
刀豆	168000	26880	13.2
红萝卜	111000	36000	12
黄瓜	93765	13050	0.9
黄心菜	7923	1620	0.3
豇豆	268528	36240	11.8
空心菜	16170	3300	0.3
辣椒	7980	1200	0.6
茄子	237624	45360	6.9
芹菜	8280	1800	0.3
青椒	13098	2610	0.9
生菜	15120	2850	0.6
土豆	82500	30000	15
西红柿	197093	36210	14.9
小青菜	187392	35480	10.9
油麦菜	20700	4500	0.9
芸豆	38592	6240	1.8

蔬菜类作物属于经济作物，由于其种植相较粮食作物更为麻烦，需要长期照顾，且没有国家政策支持，无固定销路，一旦未及时卖出或当年种植产量大于市场需求价格大幅下跌，农民就会破产，因此蔬菜类作物种植面积通常较小。根据表 3 可以看出，2023 年该乡村蔬菜类作物种植面积未超过 1 亩的有近 10 种，只有冬季能够生长的大白菜，红/白萝卜种植面积相对较大。

根据图 2，图 3 可看出，在该华北山区谷子、榆黄菇和小麦种植总利润位居前三，远高于其他农作物产生利润。而作物产量中小麦、大白菜、玉米和白灵菇柱状图一骑绝尘，远高于其他作物，大部分作物产量相对稳定，年产量在 4000 斤附近波动。

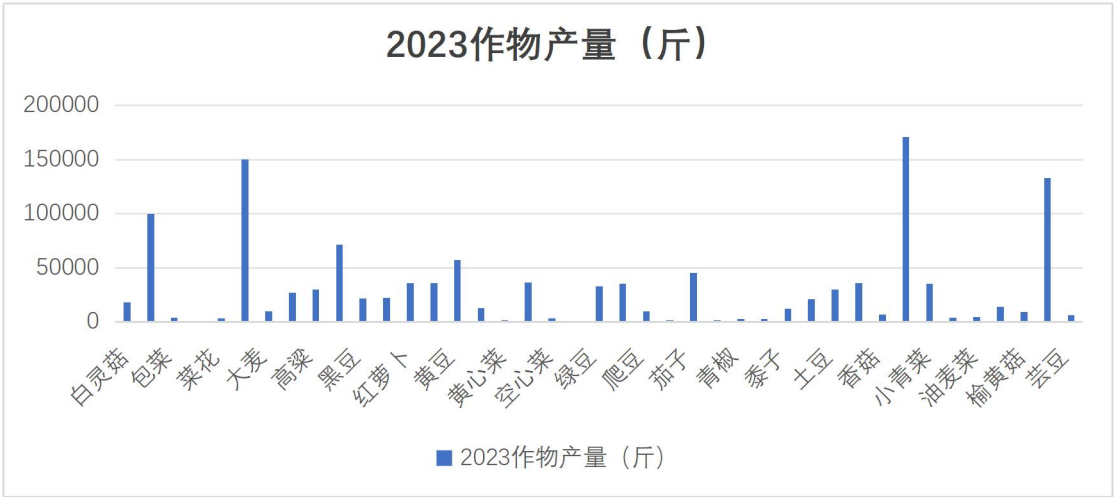

图 2 2023 年作物总产量柱状图

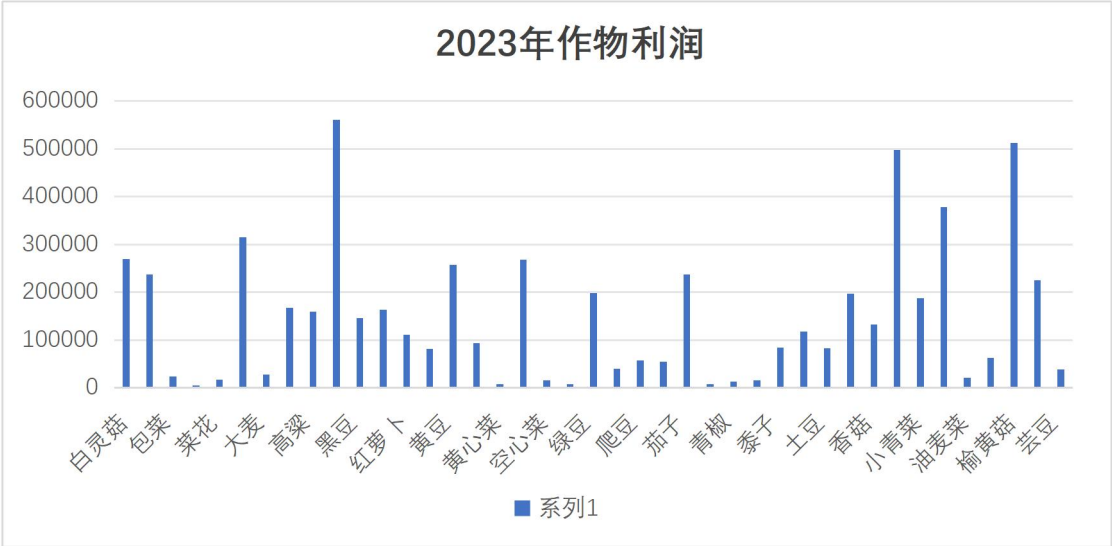

图 3 2023 年作物总利润柱状图

六、问题一建模与求解

6.1 超产作物滞销处理时最优种植策略

根据附件数据和问题要求，要在 54 块上分配 41 种作物，涉及变量为 41×54，总量高达 2000 余个，为减少变量，简化模型，我们采用分治的思想，按照每块地适宜种植的作物大类不同，将问题划分为四个子问题（即下文对应 6.1.1——6.1.4）分别进行建立线性规划。

选择每块土地上的利润作为衡量是否为最优化的变量，建立它关于种植亩数的线性规划的目标函数，然后根据题目和附件建立相应的约束条件求解，最后将四个子问题汇总得到本方案下的一年的最优种植策略。

6.1.1 平旱地、梯田和山坡地最优种植策略

本方案针对产量过剩部分采用滞销策略，为使利润最大，在建模时我们考虑尽可能少的滞销浪费，因此约束条件除题目给予的硬性条件外添加“当年各作物种植产量不大于前一年的产量（除豆类外）”这一条件。上述条件中之所以除去豆类作物，是因为 2023 年豆类作物种植面积小于总面积的 1/3，而题目要求每块地 3 年必须种一次豆类，所以上述条件不能包含豆类作物。产量恰好等于预期销售量，最终不会产生剩余。然后将利润 w_i 作为因变量，建立利润与种植亩数 y_{ijt} 的目标函数，如下式(1)所示：

$$w_i = \sum_{i=1}^{26}\sum_{j=1}^{15}(\text{销售单价}\times\text{种植面积}\times\text{亩产}-\text{成本单价}\times\text{种植面积}) \tag{1}$$

根据题意，我们需要满足的硬性约束条件包括：

- **在不同种类的土地类型和不同时节上只能种植特定的农作物。**例如平旱地、梯田和山坡地只能种值除水稻之外的粮食作物；在普通大棚第二季只能种植食用菌。

$$\text{即}\begin{cases} x_{ijt}=0 & \forall j>15 \\ x_{ijt}\neq 0 & \forall 1\le j\le 15 \end{cases}$$

当作物能够种植于该耕地上时，亩产不为 0，否则亩产恒为 0.

- **每种作物在同一地块（含大棚）都不能连续重茬种植。**即 $I_{ijt}+I_{ij(t+1)}\le 1$ 表示连续两年作物无重复情况，$I_{ijtk}+I_{ijt(k+1)}\le 1$ 表示同一年连续两季作物无重复情况。
- **每个地块的所有土地三年内必须种植一次豆类作物。**我们采用 $I_{ij(t-1)}+I_{ijt}+I_{ij(t+1)}\ge 1$ 表示连续三年间至少有一次 I 值不为 0，说明至少种植了一次豆类。
- **当年种某植产量小于的前一年的产量。**我们用 $\sum_{i=1}^{26}x_{ijt}<\beta_j$ 表示每块地某种作物的总产量小于销售量。
- **考虑每季的种植地不能太分散，单个地块每种作物在单个地块（含大棚）种植的面积不宜太小，**因为附件中所给数据最小种植单位为 0.1，所以选择让最小种植面积不小于 0.1，我们用 $y_{ijt}\ge 0.1$ 表示。

用变量表示目标函数为：

$$\max w_i = \sum_{i=1}^{26}\sum_{j=1}^{15}(b_{ijt}y_{ijt}x_{ijt}-a_{ijt}y_{ijt}) = \sum_{i=1}^{26}\sum_{j=1}^{15}(b_{ijt}x_{ijt}-a_{ijt})y_{ijt} \tag{2}$$

其中 b_{ijt} 第 t 年第 i 块地第 j 种作物的销售单价（元/斤），a_{ijt} 为第 t 年第 i 块地第 j 种作物的种植成本（元/亩），x_{ijt} 为第 t 年第 i 块地第 j 种作物的亩产量（斤/亩），y_i 为第 t 年第 i 块地第 j 种作物的种植面积（亩），w 为销售 6 总利润（元）。上式（2）中 b_i, a_i, x_i 均与 2023 年统计值保持一致，可看作常量，所以目标函数为利润值 w 与自变量种植面积 y_i 的目标函数。

约束条件为：

$$s.t.\begin{cases} i=1,\cdots,26 \\ x_{ijt}=0 & \forall j>15 \\ x_{ijt}\neq 0 & \forall 1\le j\le 15 \\ \sum_{i=1}^{26}x_{ijt}<\beta_j \\ I_{ijt}+I_{ij(t+1)}\le 1 \\ I_{ijtk}+I_{ijt(k+1)}\le 1 \\ I_{ij(t-1)}+I_{ijt}+I_{ij(t+1)}\ge 1 \\ y_{ijt}\ge 0.1 \end{cases} \tag{3}$$

通过求解上述目标函数，得到平旱地、梯田、山坡地的作物种植情况，详见result1_1.xlsx.

6.1.2 水浇地第一季、普通大棚第一季、智慧大棚两季最优种植策略

从图 1 中可以看出在水浇地上的情况要分两类（单季或两季种植）进行讨论，但由 23 年数据分析可以得出在水浇地上种植蔬菜的亩利润显著高于种植水稻，因此，我们不考虑单季种植水稻的情况，优先考虑在水浇地种植蔬菜作物，如果最终蔬菜预期产量已达预期销售量且仍存在多余空地，再于空地种植水稻（例如表 4 所示）。

以下是优先种植蔬菜时建立的线性规划模型，我们可参考 5.1.2 中平旱地等 3 种耕地种植策略的求解方案，给出相似的目标函数和约束条件，只是 i 和 j 的编号范围发生变化。

因此目标函数为：

$$w_i = \sum_{i=27}^{54}\sum_{j=17}^{34}(b_{ijt}y_{ijt}x_{ijt} - a_{ijt}y_{ijt}) = \sum_{i=27}^{54}\sum_{j=17}^{34}(b_{ijt}x_{ijt} - a_{ijt})y_{ijt} \tag{5}$$

约束条件只需将式(3)中 i 的范围改成 $i \in (27,54)$，j 的范围改成 $j \in (17,34)$ 即可。最终利用贪心算法，选择亩产利润最大的作物在各土地资源上进行种植，即为利润最大情况下 2024 年水浇地第一季、普通大棚第一季、智慧大棚两季的种植策略。

2024 年水浇地种植结果如下表 4 所示，经过求解我们发现 D1—D5、D7—D8 种植蔬菜时实际产量已达蔬菜的预期销售量，还剩余 D6 这一地块，于是就在这一地块上选择种植单季水稻。

表 4 水浇地 2024 年部分种植策略表

土地编号	作物名称	亩数
D1	西红柿	14
D2	土豆	10
D3	茄子	6
D4	土豆	5
D5	小青菜	10
D6	水稻	12
D7	豇豆	22
D8	刀豆	20

6.1.3 普通大棚第二季最优种植策略

1.数据分析

根据附件 1,乡村的现有耕地和乡村的种植作物等数据得知共有 16 块普通大棚为 F1—F16，普通大棚第二季只能种植食用菌类，每块土地的面积均为 0.6 亩，种植情况为榆黄菇占用土地 F1—F3、香菇 F4—F6、白灵菇 F7—F9，分别各占用 3 块普通大棚，羊肚菌占用 F10—F16 共 7 块普通大棚。

2.模型建立

由题意可知，若种植产量超过预期销售量，则会由于丢弃或打折产生损耗，因此要保证最大利润的方式为各食用菌种植情况与 2023 年相同，而且需保证种植方式不重茬，也就表明 2024 年羊肚菌只能种植在 F1—F9 前 9 个大棚中的 7 个，实际上可以任选其中 7 个，不必连续，但为了方便讨论，我们选择将羊肚菌分配给 F1—F7 前 7 个大棚，将榆黄菇分配给 F8—F10 号大棚，香菇分配给 F11—F13 号大棚，白灵菇分配给 F14—F16 号大棚。下一年（2025 年）按照 2023 年种植情况进行种植，两年一循环，2026 年按照上述 2024 年分配情况进行。既能保证种植利润最大，又能满足不重茬的约束条件。

6.1.4 水浇地第二季最优种植策略

水浇地第二季只能种植大白菜、白萝卜和红萝卜三种，作物数量少，且水浇地只有 8 块数目也相对较少。因此我们直接采用逻辑推理和枚举方式，对这三种作物种植情况进行推导模拟，三年一循环，我们只需求出 2024 年和 2025 年种植策略即可得出所有要求的种植策略。

下面以 2024 年种植为例进行说明。

据表 4 可知 2024 年第一季中在 D6 种植了单季作物水稻,因此第二季只能种植剩余 7 块，与此同时，为满足不连续重茬，需考虑 2023 年水浇地第二季种植情况，据附件 2 所给数据整理可得下表 5：

表 5 2023 年水浇地第二季部分种植情况

土地名称	作物名称	种植亩数
D1	白萝卜	15
D2	大白菜	10
D3	大白菜	14
D4	大白菜	6
D5	白萝卜	10
D6	红萝卜	12
D7	水稻	22
D8	水稻	20

为实现不重茬且利润最大，滞销量最小，根据 2023 年数据红萝卜需种植 12 亩，白萝卜需种植 25 亩，大白菜需种植 30 亩。然后对这三种作物种植地进行分配，实际可以混种，只需避开上一年种植过的土地即可，但是，本文为方便

讨论，我们尽可能的将每种作物集中种植于几块土地，经过推理演绎得出 2024 年和 2025 年种植策略分别为下表 6、7 所示：

表 6 2024 年水浇地第二季种植情况

土地名称	大白菜	白萝卜	红萝卜
D1	10	0	0
D2	0	0	0
D3	0	0	12
D4	0	0	0
D5	0	3	0
D6	0	0	0
D7	0	22	0
D8	20	0	0

表 7 2025 年水浇地第二季种植情况

土地名称	大白菜	白萝卜	红萝卜
D1	0	0	12
D2	0	0	0
D3	0	0	0
D4	0	5	0
D5	0	0	0
D6	10	0	0
D7	20	0	0
D8	0	20	0

6.2 超产作物按 2023 年销售价格的 50% 降价出售最优种植策略

此方案下与 6.1 情况类似，只是针对剩余产品采用打折销售，因此也采用分治的思想，先将问题分为 4 个子问题分别求解然后进行整合。

6.2.1 平旱地、梯田和山坡地最优种植策略

该方案下整体情况与 5.1.1 背景条件无甚区别，但由于剩余的作物采用五折售价销售，因此整体目标函数发生变化。而整体约束条件无较大变化，只需删去销量约束即可。

经过考虑我们的目标函数理论模型为分段函数，如下所示：

$$\begin{cases} \text{利润=实际产量}\times\text{销售单价成本}-\text{成本} & \text{实际产量}<\text{预期产量} \\ \text{利润=预期产量}\times\text{销售单价+超量打折后总价}-\text{成本} & \text{实际产量}\geq\text{预期产量} \end{cases}$$

用数学表达式表示为：

$$\max w_i = \min(x_{ijt}y_{ijt},\beta_j)b_{ijt} + \frac{b_{ijt}}{2}\max(x_{ijt}y_{ijt}-\beta_j,0) - \sum_{i=1}^{26}\sum_{j=1}^{15}a_{ijt}x_{ijt} \tag{6}$$

约束条件为：

$$s.t\quad\begin{cases} i=1,\cdots,26 \\ x_{ijt}=0 \qquad \forall j>15 \\ x_{ijt}\neq 0 \qquad \forall 1\le j\le 15 \\ I_{ijt}+I_{ij(t+1)}\le 1 \\ I_{ijtk}+I_{ijt(k+1)}\le 1 \\ I_{ij(t-1)}+I_{ijt}+I_{ij(t+1)}\ge 1 \end{cases} \tag{7}$$

如此便可求出 2024 年在方案 2 的情况下的最优种植策略，之后的 2025 年在 2024 年的基础上，满足约束条件进行迭代得出该年的种植策略，以此类推，直至得出 2024—2030 这 7 年的最优种植策略。

6.2.2 水浇地第一季、普通大棚第一季和智慧大棚两季最优种植策略

需要满足的约束条件和目标函数相同，因此只需修改 i,j 的变量范围即可。

目标函数为：

$$\max w_i=\min(x_{ijt}y_{ijt},\beta_j)b_{ijt}+\frac{b_{ijt}}{2}\max(x_{ijt}y_{ijt}-\beta_j,0)-\sum_{i=1}^{26}\sum_{j=1}^{15}a_{ijt}x_{ijt} \tag{8}$$

约束条件为只需将 i 的范围改为 $i\in(27,54)$， j 的范围改为 $j\in(17,34)$.

6.2.3 普通大棚第二季最优种植策略

根据分析，我们可知榆黄菇亩产利润 284500 元，香菇亩产利润 74000 元，白灵菇亩产利润 150000 元，羊肚菌亩产利润 90000 元；如果作物产量有剩余打 5 折进行出售，降价后榆黄菇亩产利润为 140750 元，甚至依然大于其他 3 种食用菌类作物未降价前亩产利润，所以要在满足不重茬种植的情况下，实现利润的最大化，应考虑尽可能多的种植榆黄菇，已知 2023 榆黄菇种植地数为 3 块，所以 2024 年可以选择在除上年种植的剩余 13 个普通大棚上全部种植榆黄菇，剩余 3 个曾经种植榆黄菇的大棚，选择全部种植降价后利润第二大的白灵菇。

表 8 食用菌类作物种植策略

利润（元/亩）	降价后利润	23 年种植亩数	23 年种植地块数	24 年预计种植地块数
284500	140750	1.8	3	13
74000	36000	1.8	3	0
150000	70000	1.8	3	3
90000	40000	4.2	7	0

6.2.4 水浇地第二季最优种植策略

在此方案中，如果产量超过了预期销售量，那么多余的产品作物打五折售卖。

而我们经过计算分析发现，即便是打折后销售，大白菜、白萝卜、红萝卜这三种蔬菜依然有利可图，因此我们只需在 6.1.4 所得结果的基础上，只在其中所有空白地即表格中填“0”部分选择利润最大的作物进行种植（并且需满足不重茬）。

七、问题二建模与求解

问题二中要求在综合考虑题目提出的各作物亩产、预期销量、种植成本、销售价格分别有些许波动的情况下，建立数学模型，给出最佳种植策略。由于本题并未给出如果当年产量超过了预期售量，多余部分将如何销售，因此针对该问题本文中假设多余部分全部打折处理。

对于种植策略，我们依然采用分治思想，将所有耕地适宜种植何类作物进行划分，划分后可清晰看出平旱地、梯田、山坡地适宜种植粮食作物（除水稻），普通大棚第一季、水浇地第一季、智慧大棚两季适宜种植多种蔬菜作物，普通大棚第二季适宜种植食用菌类作物。

建立线性优化模型，对各类型土地作物种植情况分别优化求解，最终给出策略结论。

在分块处理前，首先对本题中给出的条件涉及到的变量进行定义和设置。

- **预期销售量**

对于小麦和玉米，假设预期销售量年增长率为θ，作物 i 第 t 年预期销售量为β_{jt}则：

$$\begin{aligned}\beta_{6,t}&=\beta_{6,0}\times(1+\theta)^t\\ \beta_{7,t}&=\beta_{7,0}\times(1+\theta)^t\end{aligned}\quad \theta\in(0.05,0.10)$$

对于其他农作物，假设预期销售量年增长率为α，则：

$$\beta_{j,t}=\beta_{j,0}\times(1+\alpha)^t$$

- **亩产量**

假设农作物亩产量增长率为ε，则：

$$x_{jt}=x_{j,0}(1+\varepsilon)\quad \alpha\in(-0.05,0.05)$$

- **种植成本**

假设农作物种植成本增长率为δ，则：

$$a_{jt}=a_{j,0}(1+\delta)^t$$

- **销售价格**

对于粮食作物，价格基本稳定于 2023 年，即$b_{jt}=b_{j,0}$

对于蔬菜作物，价格逐年增加 5%，即$b_{jt}=b_{j,0}(1+0.05)^t$

对于食用菌类作物，假设价格变化率为γ，则:$b_{jt}=b_{j,0}(1-\gamma)\quad \gamma\in(0.01,0.05)$

7.1 平旱地、梯田、山坡地最优种植策略

在本文中我们对所有发生波动的变量增加一个因子，除此之外整体思路与第

一问类似，依然采用线性规划算法对耕地种植情况进行建模计算。需要注意的是虽然从总体上看目标函数和约束条件没有变化，实际由于变化因子的加入，种植面积、亩产量等相应变量所表示意义均与第一问不同。

目标函数为：

$$\max w_i = \sum_{i=1}^{26}\sum_{j=1}^{15}(b_{ijt}y_{ijt}x_{ijt} - a_{ijt}y_{ijt}) = \sum_{i=1}^{26}\sum_{j=1}^{15}(b_{ijt}x_{ijt} - a_{ijt})y_{ijt}$$

约束条件：

$$s.t\begin{cases} i = 1,\cdots,26 \\ x_{ijt} = 0 \qquad \forall j > 15 \\ x_{ijt} \neq 0 \qquad \forall 1 \le j \le 15 \\ \sum_{i=1}^{26} x_{ijt} < \beta_j \\ I_{ijt} + I_{ij(t+1)} \le 1 \\ I_{ijtk} + I_{ijt(k+1)} \le 1 \\ I_{ij(t-1)} + I_{ijt} + I_{ij(t+1)} \ge 1 \\ y_{ijt} \ge 0.1 \end{cases}$$

由于小麦和玉米的预期销售量整体趋势一直呈现增长状态，所以如果种植了二者，他们的最低可种植面积不断增加。

经过上述目标函数和约束条件求解得到平旱地、梯田、山坡地的种植策略，部分结果如下表 9 所示，其中“\”表示该地未混种：

表 9 平旱地和部分梯田种植策略

土地编号	作物名称 1	亩数	作物名称 2	亩数
A1	黍子	53	荞麦	27
A2	谷子	18	荞麦	37
A3	爬豆	11	谷子	24
A4	绿豆	48	莜麦	24
A5	黄豆	22	红薯	46
A6	黑豆	18	大麦	30
B1	高粱	20	大麦	40
B2	小麦	46	\	\
B3	玉米	13	谷子	27
B4	荞麦	28	\	\
B5	南瓜	25	\	\

7.2 多种蔬菜种植时最优种植策略

水浇地两季种植时第一季、普通大棚第一季、智慧大棚两季均适宜种植除大白菜、红/白萝卜之外的多种蔬菜，因此我们将三者作为同种耕地类型进行讨论。本文建立关于[利润—种植面积]的目标函数，建模求解出在最大利润下的各地作

物种植情况。思路同问题 1 中 6.2 类似，只是由于在问题 2 情境下，原来的变量增加了一个因子。所以目标函数仍为：

$$w_i = \sum_{i=27}^{54}\sum_{j=17}^{34}(\text{销售单价}\times\text{种植面积}\times\text{亩产}-\text{成本单价}\times\text{种植面积})$$

变量表示目标函数：

$$w_i = \sum_{i=27}^{54}\sum_{j=17}^{34}(b_{ijt}y_{ijt}x_{ijt} - a_{ijt}y_{ijt}) = \sum_{i=27}^{54}\sum_{j=17}^{34}(b_{ijt}x_{ijt} - a_{ijt})y_{ijt}$$

约束条件同样只需将式(3)中 i 的范围改成 $i \in (27,54)$，j 的范围改成 $j \in (17,34)$ 即可。

7.3 普通大棚第二季最优种植策略

在问题 2 所给条件下，羊肚菌每年的售价都以 5%的比率下降，剩余菌类销售价格每年降幅在 1%~5%范围，所以我们逐年计算各作物的销售价格并计算出对应总利润，然后利用贪心策略依次选择最大利润的作物于各大棚内种植。在种植时，只需注意不重茬即可。

7.4 水浇地第二季最优种植策略

在这种情况下我们的大体思路与模型与第一问第一种情况下（过产滞销）水浇地的种植策略相同，但需要注意的是，各种作物都因受市场条件影响，种植成本平均每年增长 5%左右，但水稻作为粮食作物，其销售价格基本稳定，且蔬菜作物的销售价格有增长的趋势，平均每年增长 5% 左右，故水稻在此处种植时考虑的优先级应比第一问第一种情况下更低。

通过代码的迭代运算，我们发现在 2028 年以后，种植水稻所带来的利润并不如蔬菜作物每年销售价格增长所带来的利润，故在 2029 年开始，我们认为应该放弃种植水稻。而在大白菜、白萝卜以及红萝卜这三种蔬菜作物的比较中，我们发现大白菜带来的利润最大，白萝卜次之，红萝卜最后，故我们优先考虑满足大白菜的销量需求，最后考虑红萝卜的销量需求。

7.5 模型检验——敏感度分析

我们选取了黄豆作为敏感性分析的例子，对 2024 年到 2030 年的黄豆种植量在分别取销量、亩产、销售价格变化率最小、中间、最大的情况下做灵敏性分析。

分析结果如图 4 所示，据图可看出在变化率不同的各种情况下，黄豆种植量的变化趋势大致上相同，故我们可以得出的结论是，变化率的变化对于作物种植量趋势的影响不大。

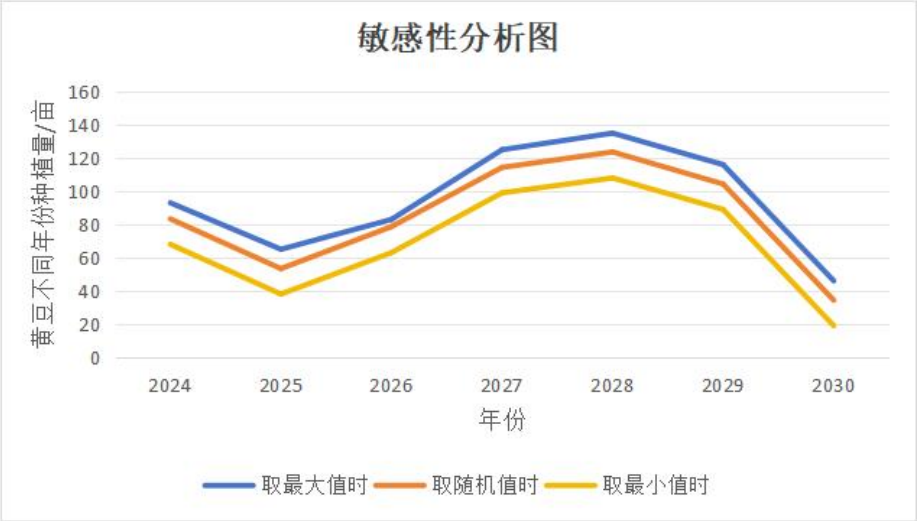

图 4 灵敏性分析图

八、问题三建模与求解

8.1 相关性分析

表 10 种植成本与种植量的线性回归分析

种植成本与种植量的线性回归分析			
	B	t	显著性
（常量）	49.249	4.773	0
种植成本	-0.009	-2.546	0.015

从表 10 中我们可以看到未标准化回归系数 B 为-0.009，这意味着每增加 1 元的种植成本，种植量预计减少 0.009 亩。t 值为-2.546，显著性为 0.015，显著性值小于 0.05，表示这个系数在统计上是显著的，意味着种植成本对种植量有负向的显著影响。

表 11 销售单价与种植量的线性回归分析

销售单价与种植量的线性回归分析			
	B	t	显著性
（常量）	36.404	3.85	0
销售单价	-0.473	-1.008	0.32

从表 11 中我们可以看到未标准化回归系数 B 为-0.473，表示销售单价每增加 1 元，种植量预计会减少 0.473 亩，也就是说，销售单价与种植量之间是负相关关系。t 值为-1.008，显著性为 0.32（大于 0.05），说明销售单价对种植量的影响在统计上不显著，也就是说，我们不能非常确定销售单价对种植量有显著的影响。

8.2 考虑相关性和可替代性时求解最优策略

问题 3 在问题 2 的基础上进一步考虑了农作物之间的可替代性和互补性，以及销售量、销售价格、种植成本之间的相关性。因此我们以问题 2 模型作为基础，用种植亩数之间的等价关系描述可替代性和相关性。

假如两种作物存在可替代性，则种植一亩作物 m 可减少种植 n 作物的面积为：

$$y_{n,t} = \nu y_{mt}$$

假如两种作物存在互补性，说明两种作物之间存在相互促进的关系，假设作物 m 是玉米，作物 n 是大豆。这两种作物在生长过程中存在互补关系：大豆具有固氮能力，可以为土壤提供氮素；而玉米则能遮荫降温，减少土壤水分蒸发，有利于大豆的生长。卖家通过种植玉米和大豆的间作模式，可以充分利用土地资源，提高总产量和利润。我们将二者的互补关系简单的描述为种植一亩作物 m 可增加 n 作物的面积为：

$$y_{n,t} = \mu y_{m,t}$$

问题 2 中目标函数为最大化利润函数，表示为：

$$\max w_i = \sum_i \sum_j (b_{ijt} y_{ijt} x_{ijt} - a_{ijt} y_{ijt}) = \sum_i \sum_j (b_{ijt} x_{ijt} - a_{ijt}) y_{ijt}$$

在上述基础上，考虑添加互补性和可替代性的影响因子，修改最大化利润函数为：

$$\max w' = \max w_i + \sum_{m,n} R_{m,n} \cdot F(y_{m,t}, y_{n,t})$$

其中用 $R_{m,n}$ 表示作物 m 与 n 相互关系，当 $R_{m,n} > 1$ 认为有可替代性，当 $R_{m,n} < 1$ 时认为二者存在互补性，例如我们通过附件 2 中 2023 年统计数据得到油麦菜和生菜的 R 值为 1.87，黄瓜和生菜的 R 值为 1.32，说明油菜对生菜的可替代性比黄瓜强。$F(y_{m,t}, y_{n,t})$ 是一个函数，表示作物 m 与作物 n 之间的互动效应，比如总体效益增加。

约束条件与问题一相同，为：

$$s.t \begin{cases} i = 1, \cdots, 41 \\ x_{ijt} = 0 \\ x_{ijt} \neq 0 \\ \sum_i x_{ijt} < \beta_j \\ I_{ijt} + I_{ij(t+1)} \leq 1 \\ I_{ijtk} + I_{ijt(k+1)} \leq 1 \\ I_{ij(t-1)} + I_{ijt} + I_{ij(t+1)} \geq 1 \\ y_{ijt} \geq 0.1 \end{cases}$$

得到约束条件和目标函数后我们利用线性规划算法对模型求解，然后将本问的种植策略与问题 2 的结果进行对比，分析二者的相关性，从而查明相互性和可替代性对最优种植策略的影响如何。

九、模型评价

非线性规划模型的优点：

1、适用于更广泛的问题：非线性规划能够处理具有非线性关系的现实世界问题，例如经济学、工程学和科学中的复杂问题。许多实际问题中的目标函数或约束条件是非线性的，非线性规划提供了更合适的建模方式。

2、灵活性高： 由于可以包含非线性函数，非线性规划能够处理线性规划无法处理的情况，如非线性成本函数、非线性生产函数、复杂的物理模型等。

3、更好的精确性： 在某些情况下，线性化模型只能给出问题的近似解，而非线性规划能够提供更精确的解。这对某些问题而言可能是至关重要的，如工程设计中的最优化问题。

4、局部最优解： 非线性规划允许找到局部最优解。这在多峰值或复杂地形的优化问题中可能是有用的，因为局部最优解在某些问题中已经足够好。

本文不足：

1、问题二中的随机因子变量可变，最后的结果有误差。

2、在对于作物单价以平均值来处理，有误差。

十、参考文献

[1] 乔金亮, 健康的土壤离不开科学耕作, https://theory.gmw.cn/2021-08/17/content_35086484.htm , 2024.9.6
[2] 李金芳. 农业推广在农业发展中的重要性及应用要点[J]. 世界热带农业信息, 2022, (05) :79-80.
[3] 罗必良，张露，仇童伟. 小农的种粮逻辑——40 年来中国农业种植结构的转变与未来策略[J]. 南方经济，2018，37(8)：1-28.

十一、附录

1.附录清单

附录一：23 年粮食作物种植总情况
附录二：picture
附录三：相关性
附录四：种植情况 pro
附录五：求解问题一干旱地类第一种情况的代码：q1_han_s1
附录六：求解问题一干旱地类第二种情况的代码：q1_han_s2
附录七：求解问题一水浇地第一种情况的代码：q1_shui_s1
附录八：求解问题一水浇地第二种情况的代码：q1_shui_s2
附录九：求解问题二干旱地类的代码：q2_han
附录十：求解问题二水浇地类的代码：q2_shui
附录十一：result1_1.xlsx
附录十二：result1_2.xlsx
附录十三：result2.xlsx

2.附录内容

附录一：23 年粮食作物种植总情况

2023 年作物种植情况.xlsx

附录二：picture

相关图片.xlsx

附录三：相关性

线性回归分析表和皮尔逊系数矩阵.xlsx

附录四：种植情况 pro

在题目条件的基础上加入了利润等信息汇总.xlsx

附录五：求解问题一干旱地类第一种情况的代码：q1_han_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;
}
```

附录六：求解问题一干旱地类第二种情况的代码：q1_han_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, 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;
}
```

附录七：求解问题一水浇地第一种情况的代码：q1_shui_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] = { 0,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,3,3,3 };

// 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] =
{ 0,36240,26880,6240,30000,36210,45360,900,2610,3480,3930,4500,35480,13050,2850,1200,33
00,1620,1800 };

// 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 =
{ 0,3600,2400,3600,2400,3000,8000,3300,3000,4000,4500,5000,4000,15000,5000,2000,12000,60
00,6600 };

    // 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;
}
```

附录八：求解问题一水浇地第二种情况的代码：q1_shui_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] =
{ 0,36240,26880,6240,30000,36210,45360,900,2610,3480,3930,4500,35480,13050,2850,1200,33
00,1620,1800 };

// 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;
}
```

附录九：求解问题二干旱地类的代码：q2_han

```
#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;
}
```

附录十：求解问题二水浇地类的代码：q2_shui

```
#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] = { 0,1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,3,3,3 };

// 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] =
{ 0,36240,26880,6240,30000,36210,45360,900,2610,3480,3930,4500,35480,13050,2850,1200,33
00,1620,1800 };

// 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 =
{ 0,3600,2400,3600,2400,3000,8000,3300,3000,4000,4500,5000,4000,15000,5000,2000,12000,60
00,6600 };

    // 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;
}
```

附录十一：result1_1.xlsx

result1_1.xlsx

附录十二：result1_2.xlsx

result1_2.xlsx

附录十三：result2.xlsx

result2.xlsx