穿越沙漠问题的模型与求解

摘要

《沙漠掘金》游戏是十分有名的企业内训游戏，常被用来对员工的合作、沟通、竞争、压力及冲突进行深度挖掘、分析和找到解决方法的专业游戏课程。提出合适的解决方案可以有效地量化合作的效果。本文就穿越沙漠问题中的各个关卡给出了对应的最优解和解决方案。

对于问题一，由于地图与天气已知，在有限的天数内，行动策略与购买策略均有限，故最佳策略一定存在。本文首先通过搜索最短路径、确定最优购买策略、确定挖矿天数给出了对一般情况的最优方案寻找方法。接下来以第一关和第二关为例，给出了最优的方案，通过编程搜索，得到了第一关到达终点时最多剩余资金为 10470 元，第二关到达终点时最多剩余资金为 12730 元。

对于问题二，由于未来天气情况未知，故很可能不存在对所有情况都为最佳的策略。因此，我们选择最佳策略的目标为：在确保安全到达终点的前提下，使得最终资金的期望值最高。对于第三关，为了计算消耗资金的期望值，本文引入了天气晴朗的概率 p，且每日天气相互独立。通过对 p 大小的讨论，我们给出了所有情况下的策略。当 p 较小时 ($p < \frac{27}{32}$)，选择的最优策略为在前三天不停留地沿最短路径走到终点。当 p 较大时 ($p > \frac{27}{32}$)，本文通过递推关系进行动态规划，得到了在起点处携带 54 箱水与 54 箱食物为最优购买策略；行进时如遇高温，在确保能够安全到达终点时选择停，其余情况选择走的策略。

对于问题三，由于多个玩家的初始条件相同，如果将目标函数设置成每个人自身取得最大利益，则每个人的策略将会完全相同，从而陷入囚徒困境。所以本文将目标函数设为所有玩家最终资金之和。为了达到目标，本文采用了贪心算法。通过将各个玩家进行排序，在开始时规定他们的行动准则，第 k 个玩家在已知当天天气情况下可以推知前 k-1 个玩家于当天选择的策略，他将选择在考虑到前 $k-1$ 个玩家的最优策略下可选策略中最优的。将此一般策略应用到第五关的情境中，得到了第五关的最优解。对于第六关，我们进行具体分析得到了一个较优的策略。

关键词: BF 算法、数学期望、动态规划、贪心算法

1 问题重述

1.1 问题背景

《沙漠掘金》游戏是十分有名的企业内训游戏，常被用来对员工的合作、沟通、竞争、压力及冲突进行深度挖掘、分析和找到解决方法的专业游戏课程.

1.2 问题描述

题目给出了一个“沙漠掘金”游戏的简化版，其目的是在截止日期之前，保证路上生活必需品充足且不超过负重上限的同时走到终点，并通过挖矿追求最高的获利.

游戏的地图可以看成一个无向图，每一天从每个节点可以移动到相邻的节点，也可以停留在当前节点，而移动到相邻节点的生活必需品耗费是停留在当前节点的耗费（称为基础消耗）的二倍，同时地图中某些特定的节点为矿山，在矿山处挖矿可以获取金钱，而挖矿的生活必需品耗费则是基础消耗的三倍，到达矿山当天不能挖矿. 玩家还可以在村庄购买生活必需品，而其金钱消耗则是在起点处购买生活必需品消耗的二倍，玩家可以在到达村庄当天购买生活必需品. 到达终点后，玩家可以以起点处半价的价格出售未使用完的生活必需品.

游戏还设置了天气系统，认为某时刻全图的天气相同，而基础消耗会随着天气的变化而变化，并作为已知量给出，沙暴是一种特殊的天气，在沙暴天气中，玩家不能移动.

在此基础上，题目给出了以下三个问题：

问题一 一名玩家在每日天气都已知的情况下，按照附件中给出的地图，寻找一般策略和第一关和第二关的最优购买和行动策略.

问题二 一名玩家，玩家只能知道当天的天气，按照附件中给出的地图，寻找一般策略和第三关的最佳行动策略.

问题三 多名玩家，他们的起始情况相同。k 名玩家同时在同一路线行走时消耗变为一名玩家行走的 k 倍，同时在同一矿山挖矿时收益比一名玩家挖矿缩小 k 倍，同时在同一村庄购买物资时价格变为基础消耗的四倍. 第五关所有天气已知，玩家需在初始时制定行动策略并不能更改. 第六关玩家只知道当天的天气和其他玩家的行动方案和剩余资源. 对于第五关和第六关的具体情况，需要给出最佳策略.

2 符号说明

符号	意义
t	游戏进行的天数（天）
m	玩家身上的剩余金钱（元）
a	水的基础消耗（箱）
b	食物的基础消耗（箱）
p	收益（元）
w	当前拥有的水（箱）
f	当前拥有的食物（箱）
W_f	食物重量 (kg/箱)
W_w	水的重量 (kg/箱)
p_f	食物价格 (kg/箱)
p_w	水的价格 (kg/箱)
P	背包最大负重 (kg/箱)
S	策略
K	玩家动作构成的集合

3 问题分析

3.1 问题一分析

问题一给出了地图天气关于 T 的变化以及在不同天气下的 a, b 值. 由于给出的地图本质上是一个无向图,我们首先尝试将地图画成无向图的形式,并列出其邻接矩阵 Matrix_1,Matrix_2.

其次，由于一步得到整个问题的最优解需要考虑的因素包含路径、每天的行动策略、初始点的购买量、挖矿的天数等诸多问题，同时考虑如此之多的变量对收益的影响十分困难，所以我们需要一步步将问题简化，具体简化的方式包括但不限于——减少地图的分支、限制食物与水的携带量，限制矿山的访问次数等等.

3.2 问题二分析

问题二中的玩家仅仅知道当前时刻的天气，但却并不知道其后的情况，所以我们引入不同天气出现的概率，之后使用概率方法进行计算，以到达终点后剩余资金的期望为目标函数进行优化.

3.3 问题三分析

问题三中的玩家由一人变成了多人. 但是并不能以单个玩家的利益最大作为优化的目标函数，因为每个玩家的地位在游戏中对称，当以单个玩家的利益作为优化的目标函数时，容易得到各个玩家的行动方案重合的解，造成囚徒困境. 故优化的目标函数应选取为所有玩家的最终剩余资金之和.

4 模型的建立与求解

4.1 问题一的一般策略：

已知地图与天气，在有限的天数内，行动策略与购买策略均有限. 故最佳策略一定存在.

对于较为复杂的地图，我们可通过起点、矿山、村庄、终点之间的最短路径对其进行简化. 并且根据它们之间的距离，确定可能的最佳路线和行动策略，进一步缩小最佳策略的可能范围.

而对于既定的路线和行动策略，我们可以确定最优购买方案：在确保物资始终充足且到达终点时剩余物资为 0 的前提下，在起点处购买尽可能多的食物. 由此我们可以确定任一路线和行动策略的最终资金数.

通过对已缩小的范围内的路线和行动策略进行搜索，我们可以得到最优策略（包括行动策略和购买策略）.

4.2 问题一：第一关

基于此，我们对给出的各种情况分别进行讨论.

4.2.1 情况 1: 玩家前往矿山

当玩家前往矿山挖矿时，为得到最优解，我们仍旧需要考虑诸如路径、每天的策略、初始点的购买量、挖矿的天数等问题，假如直接进行搜索仍旧会产生十分巨大的计算量.

利用附录中的 Graph_and_Path.m，我们 w 可以求得从起点到矿山的最短路径长度为 8，且存在经过村庄的长度为 8 的路径 p_1. 从矿山到终点的最短路径长度为 5，且存在经过村庄的长度为 5 的路径 p_2.

基于此，我们有如下的定理：

定理 4.1. *使得玩家收益最大的经过矿山的决策方案 S 经过的节点构成的子图一定包含于两条最短路径构成的子图 G.*

为了证明定理 4.1，我们给出下面的引理.

引理 4.1. *[1] 在一张图中，假如 $p : a \rightsquigarrow b \rightsquigarrow c$ 是 a 到 c 的最短路径，那么它的子路径 $p' : a \rightsquigarrow b$ 也是 a 到 b 的最短路径.*

同时，我们形式化地定义一个决策 S:

定义 4.1. *集合 T 是游戏时间 t 构成的集合，其中元素为整数. 集合 K 是玩家动作构成的集合，定义为*

$$K = \{\text{挖矿}, \text{移动到相邻某格}x, \text{原地不动}, \text{购买物资}\}$$

定义 4.2. 一个决策 S 是指从 $T' = [0, t] \cap T$ 到 K 的映射 $S : T' \mapsto K$，使得

$$S : \max\{T'\} \mapsto \text{移动到终点}$$

则利用引理与定义便可证明定理 4.1，详细证明见于附录.

同时，我们对于整个决策还能做出以下断言：

定理 4.2. 使得玩家收益最大的决策方案不会两次进入矿山.

定理 4.3. 在起点时需要尽可能将玩家背包装满，且玩家尽可能多地购买食物，在第一次经过村庄时仅补充水.

这条定理可以通过如下的角度得到说明：首先，为了收益尽可能大，我们选择应使得玩家在矿场挖矿时间尽可能长，故应将背包装满. 由于单位质量水的价格远低于食物，且在起点时购买物资比村庄便宜，故在能确保物资始终充足的前提下应尽可能买更多的食物.

我们还有如下断言：

断言：食物的数量存在上限，食物的数量既不能超过整条路线上的食物消耗，又需要使得在第一次经过村庄，补充水之后的水能坚持到第二次到达村庄.

定理 4.2 的证明亦见于附录. 基于以上的三条定理，我们便设计了 Best_Solution_1.py 程序，使用枚举的方法，对第一关的最优解进行计算，程序的流程图如图 1，源代码见于附件. 得到的最优解如表 3(见附录)

4.2.2 情况 2: 玩家不去矿山时

当玩家不去矿山时，无论如何选择，最终对于玩家的结果都会是亏损，但上一问中的最优解实现了 470 元的获利，故此情况下一定没有最优解.

4.3 问题一：第二关

为了使得挖矿的总天数尽可能多，我们给出如下假设.

1. 玩家需要在第三十天时达到终点.
2. 在到达矿山之前玩家不会停止移动（除非遇到沙暴）.

在这样的条件下，我们筛选出玩家想要最优解可能经过的三种路径方式：

$$\text{起点} \rightsquigarrow \text{矿山 } 30 \rightsquigarrow \text{村庄 } 39 \rightsquigarrow \text{矿山 } 30 \rightsquigarrow \text{村庄 } 39 \rightsquigarrow \text{终点} \tag{1}$$

$$\text{起点} \rightsquigarrow \text{村庄 } 62 \rightsquigarrow \text{矿山 } 55 \rightsquigarrow \text{村庄 } 62 \rightsquigarrow \text{矿山 } 55 \rightsquigarrow \text{终点} \tag{2}$$

$$\text{起点} \rightsquigarrow \text{村庄 } 39 \rightsquigarrow \text{矿山 } 55 \rightsquigarrow \text{村庄 } 62 \rightsquigarrow \text{矿山 } 55 \rightsquigarrow \text{终点} \tag{3}$$

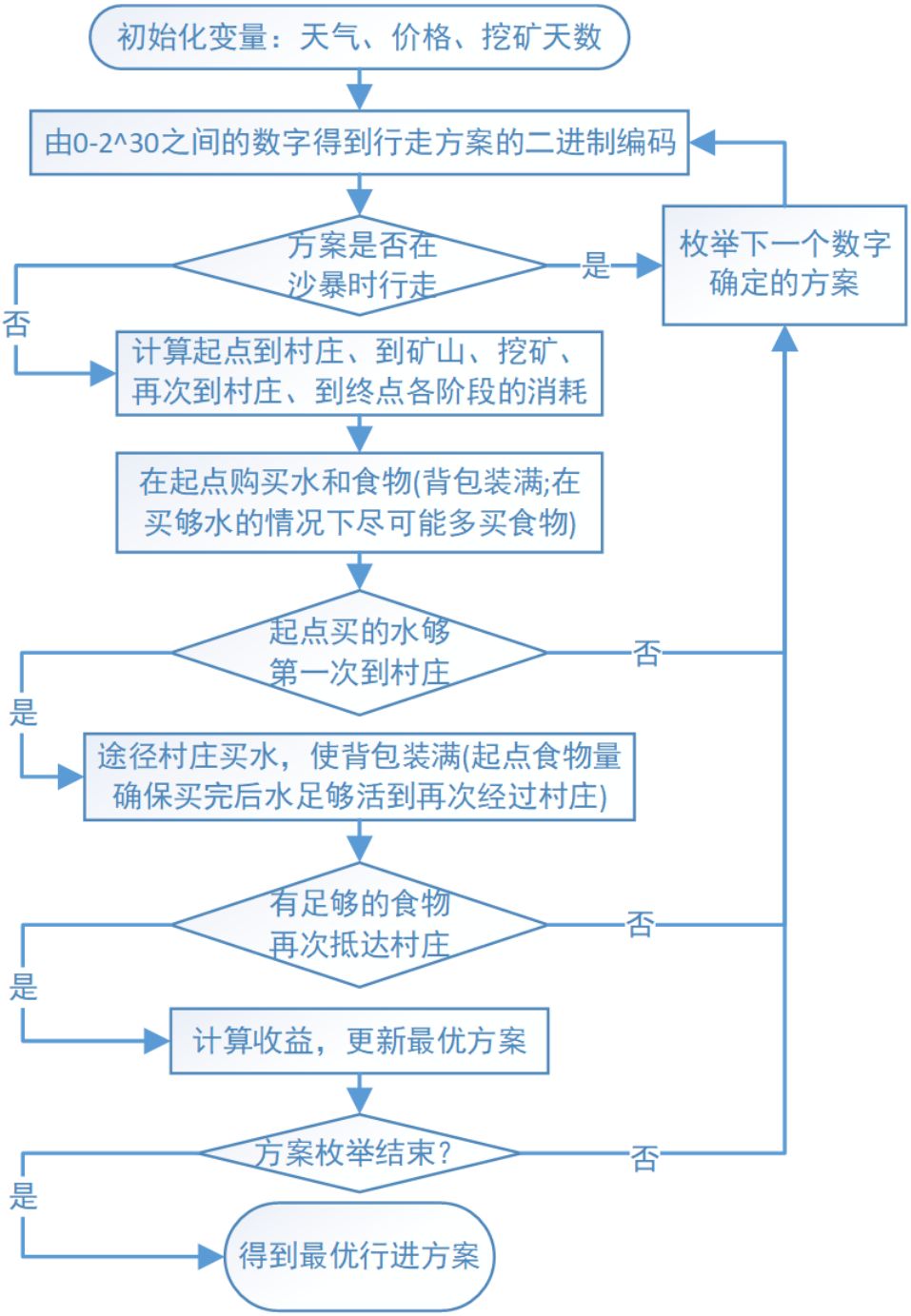

图 1: 程序 Best_Solution_1.py 的流程图

他们对应的最优解分别为:

起点 ⇝ 矿山 (9)(挖矿 10-14) ⇝ 村庄 (15) ⇝ 矿山 (16)(17 休息, 挖矿 18-26)
⇓
终点 (30) ⇜ 村庄 (27)

起点 ⇝ 村庄 (12) ⇝ 矿山 (13)(挖矿 14-18) ⇝ 村庄 (19) ⇝ 矿山 (20)(挖矿 21-28)
⇓
终点 (30)

起点 ⇝ 村庄 (11) ⇝ 矿山 (13)(挖矿 14-18) ⇝ 村庄 (19) ⇝ 矿山 (20)(挖矿 21-28)
⇓
终点 (30)

它们对应的最终资金分别是: 12345、12570、12730 元, 通过比较, 我们可见, 3 对应的最优解为最优策略. 每一天的具体数据详见附录.

4.4 问题二的一般策略:

由于未来天气情况未知, 故很可能不存在对所有情况都为最佳的策略. 因此, 我们选择最佳策略的目标为: 在确保安全到达终点的前提下, 使得最终资金的期望值最高.

利用与问题一相同的方法, 可以先缩小最佳策略的可能范围. 对于已知各天气情况的出现概率, 我们可以利用动态规划的方法, 计算在任何物资情况下, 任一位置到终点所消耗物资的价格的期望. 由此, 在每一天, 我们可以在已知当天天气情况下, 选择使得最终剩余资金期望最高的行动方案. 以此得到最优策略.

4.5 问题二: 第三关

假设之后 10 天天气晴朗的概率 p, 且每日天气相互独立. 由于玩家仅知道当天的天气状况, 故不一定存在对所有可能情形都为最优的策略. 由此, 我们希望选取的最优策略的目标是: 在保证能够安全到达终点的前提下, 使得最终资金期望最多.

定理 4.4. 最优策略可以在路径 $1 \longrightarrow 5 \longrightarrow 6 \longrightarrow 13$ 上实现.

证明. 为了证明定理 4.4, 我们分以下两种情况讨论.

- 情形 1：不经过矿山. 此时玩家没有额外的收益，而 $1 \longrightarrow 5 \longrightarrow 6 \longrightarrow 13$ 为最短的路径，必然能够实现最优策略.
- 情况 2：经过矿山，在相同的天气条件下，参考附录中 Best_Path.m 的结果，我们知道其收益一定差于情形一.

□

因此，我们只需在路径 $1 \longrightarrow 5 \longrightarrow 6 \longrightarrow 13$ 上寻找最优策略.
此外，我们还有以下断言：
断言：对于晴朗天气，玩家一定不停留.

情况 1： 当 p 较小时 ($p \leq \frac{27}{32}$).

由于玩家在晴朗天气一定不停留，只需考虑在高温天气下的行动策略. 假设玩家从一个位置走到下一个相邻位置，所消耗物资（水和食物）的基准价格期望为 E_p.

在晴朗天气下行走一天，消耗物资基准价格为 $2 \times (3 \times 5 + 4 \times 10) = 110$(元)；高温天气下停留消耗物资基准价格为行走一天 $5 \times 9 + 10 \times 9 = 135$(元)，行走一天为 270 元.

策略 1：无论什么天气均行走，此时，$E_{p1} = 110p + 270(1-p) = 270 - 160p$.

策略 2：在高温天气停留，晴朗天气行走，则有

$$E_{p2} = 110p + (135 + E_{p2})(1-p).$$

解得 $E_{p2} = \frac{135}{p} - 25$.

比较上述两种策略的期望，可知当 $p \leq \frac{27}{32}$ 时，$E_{p1} \leq E_{p2}$.

策略 1 消耗物资的价格的期望低于策略 2，且策略 1 在起点处只需购买能够确保安全到达终点的资源最小值——54 箱水、54 箱食物，策略 2 在起点购买物资一定比策略 1 多. 综上两条，策略 1 对比策略 2 更优. 故此时，选择的最优策略为在在起点购买 54 箱水和 54 箱食物，前三天不停留地沿 $1 \longrightarrow 5 \longrightarrow 6 \longrightarrow 13$ 到达终点.

情况二： 当 $p > \frac{27}{32}$ 时，在前 10 天内出现晴天天数小于等于 2 的概率为

$$P = (1-p)^{10} + \binom{10}{1}(1-p)^9 p + \binom{10}{2}(1-p)^8 p^2 < 60(1-p)^8 < 60 \times (\frac{5}{32})^8 < 2.2 \times 10^{-5}$$

此情况可忽略不计. 因而，可以认为在前 10 天内有至少 3 天晴朗，所以如此选择策略可以使得玩家走到终点.

此时，我们的方法是利用递推算法进行动态规划.

我们认为出发时所携带的食物与水有如下界限

$$2 \times 3 \times 9 \leq f_0 \leq 2 \times 10 \times 9 \tag{4}$$

$$2 \times 3 \times 9 \leq w_0 \leq 2 \times 10 \times 9 \tag{5}$$

上述二式左侧界由其行走至少三天决定，而右侧界限由玩家至多行走十天决定.

算法说明：

根据定理 4.4，我们知道玩家有唯一可以选择的路径，将其改写为

$$0 \longrightarrow 1 \longrightarrow 2 \longrightarrow 3$$

认为在 $0, 1, 2$ 三个点，每个点处耗费食物与水的总价都是点、当前的食物量、当前的水量的函数，分别记为：

$$E_0(w, f), E_1(w, f), E_2(w, f)$$

同时，我们还能得到如下递推关系：

$$E_0(w, f) = p \cdot (110 + E_1(w - 6, f - 8)) + (1 - p) \cdot \min\{270 + E_1(w - 18, f - 18), E_0(w - 9, f - 9)\} \tag{6}$$

$$E_1(w, f) = p \cdot (110 + E_2(w - 6, f - 8)) + (1 - p) \cdot \min\{270 + E_2(w - 18, f - 18), E_1(w - 9, f - 9)\} \tag{7}$$

$$E_2(w, f) = p \cdot 110 + (1 - p) \cdot \min\{270, E_2(w - 9, f - 9)\} \tag{8}$$

按照上述递推关系，我们设计了 Recurrence.py，并计算得到了 $E_0(w, f), E_1(w, f), E_2(w, f)$，我们发现对于给定的 i

$$\max\{E_i(w, f)\} - \min\{E_i(w, f)\} \leq 5 \times 10^{-1}$$

基本上其变化可以忽略不计. 这意味着当在起点购买的水和食物数量均达到 54 时，此后消耗物资价格的期望基本稳定，为了使终点处剩余资金的期望最大化，玩家只需在起点处购买 54 箱水和 54 箱食物.

在每个 E_i 之内，我们都取 $E_i(54, 54)$ 作为所有 E_i 的代表元，部分参考数据见于表 1.

接下来的问题是根据得到的金钱消耗期望决定行走策略.

根据之前的断言，当前天气为晴天时，玩家应当选择动身前往下个节点. 当前天气为高温时，则应根据 $135 + E_i, 270 + E_{i+1}$ 的相对大小来决定走或停.

表 1: 不同概率下每点处的金钱消耗期望

p	0.8	0.82	0.84	0.86	0.88	0.9	0.92	0.94	0.96	0.98	1
E_2	142	138.8	135.6	131.9	128.4	126	122.8	119.6	116.4	113.2	110
E_1	284	277.6	271.2	263.9	257.7	251.1	244.6	238.3	232.1	226	220
E_0	426	416	406.8	395.9	386.1	376.1	366.4	356.9	347.7	338.7	330
$E_0 - E_1$	142	138.4	135.6	131.9	128.4	125	121.7	118.6	115.6	112.7	110
$E_1 - E_2$	142	138.8	135.6	131.9	129.2	125.1	121.8	118.6	115.6	112.7	110

当 $135 + E_i < 270 + E_{i+1}$ 时，选择留在原地的金钱消耗期望较小，而当 $E_i + 135 > 270 + E_{i+1}$ 时，前进的金钱消耗期望较小.

化简上述不等式，发现判据为 $E_i - E_{i+1}$ 与 135 的大小关系. 观察表 1，发现当 $p > \frac{27}{32} \approx 0.84$ 时，有

$$E_i - E_{i+1} < 135$$

故当 $p > \frac{27}{32}$ 时，最优策略为：在起点购买 54 箱水与 54 箱食物. 行进时如遇高温，在确保能够安全到达终点时选择停留，其余情况选择行走.

4.5.1 第三关的结论：

对于不同的概率 p，我们均可给出玩家的最优策略：

- 当 $p \leq \frac{27}{32}$ 时，在起点购买 54 箱水与 54 箱食物，无论当前天气如何，都向行走.
- 当 $p > \frac{27}{32}$ 时，在起点购买 54 箱水与 54 箱食物. 行进时如遇高温，在确保能够安全到达终点时选择停留，其余情况选择行走.

  事实上，$p = 0.84 \approx \frac{27}{32}$ 时，$E_i - E_{i+1} \approx 135$，策略变化的界限与情况一理论计算的概率界限相吻合. 更进一步，由表 1 可见，当 p 在 $\frac{27}{32}$ 附近时，两种策略下的终点剩余资金的期望值相近，两种策略不相上下.

4.6 问题二：第四关

在本关中，由于玩家的最优路线有以下两种情况：

1. 不经过矿区，直接沿最短路走到终点.

2. 经过矿区.

由于存在经过矿山的最短路径，故只需考虑情况 (2) 中的最优策略.

在最后一次从矿山出发前往终点的过程中，若先经过村庄，则需要消耗额外花费两天时间行走，减少了挖矿的总天数. 故可以认为，玩家最后一次从矿山出发后沿最短路径前往终点.

由此，有以下两种策略：

1.

2.

为了使挖矿天数尽可能多，不妨假定玩家在第 30 天到达终点.

4.6.1 对策略 1 挖矿天数的估计：

考虑到沙漠天气较为恶劣，为了简化模型便于计算，我们假设 30 天内高温天数与晴朗天数之比不低于 1：1. 若策略 1 只经过 1 次村庄，则玩家行走的天数为 12 天，挖矿天数为 18 天. 此时，其消耗物资的总质量不少于 $6\times90+6\times34+9\times135+9\times51=2418kg > 2\times 1200kg$. 加之可能出现的沙暴天气，总消耗物资量会更多，且挖矿天数可能更少.

故策略 1 有较大的极大的可能需要经过村庄两次. 从而此时策略 1 行走的天数为 16 天，挖矿天数至多 14 天.

4.6.2 对策略 2 挖矿天数的估计：

假设连续 11 天至多出现 2 天沙暴，至少出现 3 天晴朗. 则即使沙暴天出现在挖矿的期间，如果挖 7 天矿，消耗物资的总质量总不超过 $3\times34+1\times90$（行走）$+2\times150+5\times135$（挖矿）$=1167kg < 1200kg$. 故每次挖矿至少可持续 7 天. 因此策略 2 可至少挖矿 14 天，且其只需行走 14 天.

而且，与策略 1 相比，策略 2 在起点处只需购买更少的水，因此，有更多的负重空间来购买食物. 由于食物单位质量的价格远高于水 $(10/2 > 5/3)$，故在起点处购买食物能节省更多的钱.

综上两条估计，我们有理由猜测采取策略 2 是较优的策略. 由此，对于第四关，玩家可采取的较优的策略为

并且在起点购买时，在保证水始终充足的基础上，尽可能购买更多的食物.

4.7 问题三：第五关

对于多人的情形，假如将目标函数设置成每个人自身取得最大利益，则每个人的策略将会完全相同. 故策略的目标函数为所有玩家最终资金之和. 当多名玩家在同步挖矿、移动、购买时，均会产生高于单人情况的消耗. 因此，个人理性的每名玩家，在最初时均选择最优解，反而可能会造成集体的非理性，产生“囚徒困境”. 因此，对于多人的一般情形，我们采取一种无序个人的有序化策略：

对于 n 名玩家，我们对他们进行编号 $1, 2, \cdots, n$. 第 1 名玩家，选取最优策略（如果有多条最优策略，则选取其中一个策略，使得玩家采用这个策略时，每天的行走策略都选取编号尽可能小的区域，其次尽可能不挖矿，其次尽可能不在村庄购物——这样的策略是唯一的.）；第 2 名玩家，选择在考虑第 1 名玩家策略情况下的最优策略（如果有多条，则用与第 1 名玩家相同的方法选择其中一条最优策略，也是唯一确定的）；对于第 k 名玩家 $(2 \leq k \leq n)$，选择在考虑到前 $k-1$ 名玩家策略下的最优策略.

上述策略虽然可能无法达到一般情形的整体最优解，但仍然是一个能够实现较优目标函数的策略.

故有分析过程如下：

4.7.1 玩家不去矿山

当玩家不去矿山时，由于地图与天气状况较为简单，我们直接给出最优策略为：

$$1 \longrightarrow 5 \longrightarrow \text{停留} \longrightarrow 6 \longrightarrow 13$$

此时的亏损为

$$\Delta m = \sum_{t=1}^{4} (\Delta f_t \cdot p_f + \Delta w_t \cdot p_w) = -300(\text{元})$$

4.7.2 玩家去矿山

当玩家前去矿山时，我们按照附录中的 Best_Path.m 文件生成的 str0.mat，给出最优解如下.

$$\text{起点} \longrightarrow 2 \longrightarrow \text{停留} \longrightarrow 3 \longrightarrow 9 \longrightarrow 10 \longrightarrow 13$$

此时玩家的亏损为 635(元)，不及最优路线.

4.7.3 结论

所以根据对一般情况给出的策略，第 1 名玩家选择的策略是：

$$1 \longrightarrow 5 \longrightarrow \text{停留} \longrightarrow 6 \longrightarrow 13$$

第 2 名玩家选择的策略是：

$$1 \longrightarrow \text{停留} \longrightarrow \text{停留} \longrightarrow 5 \longrightarrow 6 \longrightarrow 13$$

事实上，这个策略也是这 2 名玩家在该问题下的集体最优策略. 这点只需要比较玩家二选择此线路与选择

$$1 \longrightarrow \text{停留} \longrightarrow 4 \longrightarrow 7 \longrightarrow 12 \longrightarrow 13$$

的亏损差距，经过简单的计算，是前者亏损更少.

4.8 问题三：第六关

对于本关，最优策略的目标：在确保能够安全到达终点的前提下，使得三名玩家最终资金数之和最多.

4.8.1 一般策略：

在第 0 天，对 n 名玩家进行编号 $1, 2, \cdots, n$. 我们将采取贪心算法的思想给出 n 名玩家的策略.

对于之后的每一天，玩家 1 均采取当下使最终资金数期望最高的最优策略（行走、停留、挖矿、购买）. 如果有多个最优策略，类似 3(1) 的一般策略，我们可以用既定的方法对最优策略进行排序，并使玩家 1 采取排序后的第一条最优策略. 由此，其他玩家在已知当天天气状况和玩家 1 物资情况及位置的条件下，便可得知玩家 1 在当天的行动策略.

玩家 2 在考虑到玩家 1 当天策略的基础上，对当天可能采取的策略进行评估，并采取使最终资金数期望最高的策略.（如果期望最高的策略有多种，则同样可用既定的排序方案唯一确定，使得后面的玩家可以推知）. 以此类推，每名玩家均在考虑到前面玩家在当天的策略下，选择自己的当天最优策略.

4.8.2 第六关：

根据第六关的规则和条件，对于最优策略，我们有如下断言：

1. 由于 k 名玩家 $(1 \leq k \leq 3)$ 在挖矿时，总收益均为基准收益的 $\frac{1}{k} \times k = 1$ 倍，故每天至多有一名玩家在挖矿. 由于在矿山挖矿收益不菲（在沙暴天气挖矿一天的收益，大于高温天气行走两天消耗物资在村庄购买的价格，$1000 - 450 > 2 \times 270$），故尽可能使三名玩家挖矿的总天数尽可能多.

2. 当有 $k(k \geq 2)$ 名玩家在同一天均从区域 A 走到区域 $B(A \neq B)$，则这 k 名玩家的总消耗量为基础消耗量的 $2k^2$ 倍. 而若不考虑天气的影响和天数的限制，让其中一位玩家先在 A 地停留一天，并在下一个非沙暴天从区域 A 走到 B，此时 k 名玩家的总消耗量为基础消耗 $2(k-1)^2 + 1 + 2 = 2k^2 - 4k + 5$ 倍，为更优. 故最优策略中，应尽可能避免多名玩家在同一天内从区域 A 行动到区域 B $(A \neq B)$.

3. 由于多名玩家在同一天于村庄购买的价格为基准价格的 4 倍，故也应尽可能避免此情况.

4. 为了规避物资不足的风险，同时避免过多的浪费，三名玩家在考虑到未来可能出现的（较少的）沙暴天气下，尽可能买更少的食物.

基于上述的断言，我们可以有如下的策略：

首先，对 3 名玩家进行编号 1,2,3. 并规定每位玩家在非沙暴天气不停留（在矿山挖矿除外）.

1. 玩家 1 的行动策略：玩家 1 在起点多停留一天后沿线路

$$1 \to 6 \to 11 \to 16 \to 21 \to 22 \to 23 \to 24 \to 25$$

到达终点.

2. 玩家 2,3 的行动策略：

第一阶段: 玩家 2 沿线路 $1 \to 6 \to 11 \to 16 \to 17 \to 18$ 到达矿山，玩家 3 沿路线 $1 \to 2 \to 3 \to 4 \to 9 \to 14$ 到达村庄.（同时到达）

第二阶段：玩家 2 不停止地挖矿，直到玩家 3 沿路线 $14 \to 13 \to 18$ 到达矿山（当天仍然挖矿）. 然后玩家 2 沿路线 $18 \to 19 \to 14$ 前往村庄购买物资，再沿路线 $14 \to 13 \to 18$ 返回矿山；同时玩家 3 在矿山不停地挖矿，直到玩家 2 到达矿山. 此后，玩家 3 按上述路线前往村庄购买物资、返回矿山，玩家 2 不停地挖矿，直到玩家 3 返回矿山……如此往复.

第三阶段：当所剩天数不足以自己完成新一轮去村庄购买物资、并返回矿山挖矿获得足够收益时，玩家 2（或 3）从矿山沿路径

$$18 \to 19 \to 20 \to 25$$

直接前往终点.

3. 玩家 1 的购买策略：在起点购买物资时，玩家 1 在考虑到未来可能出现的恶劣天气下，购买确保能够安全到达终点的最小数量.

4. 玩家 2,3 的购买策略：在起点购买物资时，玩家 2,3 在考虑到未来可能出现的恶劣天气下，购买确保此后不会出现物资短缺情况下的最小数量，并将剩余的负重空间全部购买食物. 而在经过村庄购买物资时，只需在确保此后不会出现物资短缺情况下，购买尽可能少的物资.

策略的可行性：

1. 玩家 1：由于 30 天内较少出现沙暴天气，不妨假定前 12 天至多出现 3 天沙暴天气. 即使前 15 天非沙暴天均为高温天，玩家 1 采取策略消耗物资的总质量不超过 $45+8\times90+3\times50 = 915 < 1200$. 故玩家 1 可以在起点处合适地购买物资，使得确保能够安全到达终点.

2. 玩家 2,3：由于 30 天内较少出现沙暴天气，不妨假定前 9 天至多出现 2 天沙暴天气. 即使非沙暴天均为高温天，玩家 2 在第一次到达村庄所消耗物资的总质量也不超过 $7 \times 90 + 2 \times 135 + 2 \times 150 = 1200$，而玩家 3 在第一次到达村庄消耗物资必然少于玩家 2，故玩家 2,3 均可在起点处合适地购买物资，确保能够安全地第一次到达村庄.

   由于 30 天内较少出现沙暴天气，不妨假定连续 10 天中至多出现 2 天沙暴天气. 即使非沙暴天均为高温天，在从村庄出发去矿山、挖矿、再返回村庄的过程中，玩家 2（或 3）消耗物资的质量也不超过 $2\times90+4\times135+2\times150+2\times90=1200$. 故玩家 2,3 在每次经过村庄时合适地购买物资，来确保物资的充足.

策略的优点：

1. 从玩家 2 到达矿山，到玩家 2,3 中最后一名玩家离开矿山，始终有人在矿山进行挖矿，实现了挖矿收益的最大化.

2. 上述策略中，不存在有多人在同一天均从区域 A 走到区域 $B(A \neq B)$ 的情况.

3. 上述策略中，不存在有多人在同一天均在村庄购买物资的情况.

4. 上述购买方案中，每名玩家在确保能够安全到达终点的情况下，采取了最优的购买策略.

5 模型的评价与改进

5.1 问题一的评价

5.1.1 优点：

利用最短路径、最优购买策略、挖矿天数缩小状态空间的规模. 简化了搜索算法，使得时间复杂度在可以实现的范围之内；通过搜索算法，计算出了最优策略.

5.1.2 缺点：

即使经过了优化，时间复杂度仍然较高，可能存在进一步优化的空间.

5.2 问题二的评价

5.2.1 优点：

1. 利用到达终点时剩余资金的期望来衡量策略的优劣.
2. 在第三关引入不同天气出现的概率，以便计算不同策略到达终点时剩余资金的期望.
3. 在第三关采用动态规划算法寻找最优策略，并给出了不同概率下的最佳策略.

5.2.2 缺点：

第四关模型假设过强，没有解决一般情形下的最佳策略.

5.3 问题三的评价

5.3.1 优点：

1. 利用贪心算法给出多人情况的较优策略.
2. 给出的一般策略有较好的普适性，对于第五关、第六关两种具体情况，均给出了较好的策略.

5.3.2 缺点：

没有严谨地对策略是否最优进行讨论，可能存在更优的策略.

参考文献

[1] LEISERSON C E, RIVEST R L, STEIN C. Introduction to Algorithms[M]. Ed. by CORMEN T H. America: The MIT Press.

[2] 薛定宇, 陈阳泉. 高等数学问题的 MATLAB 求解[M]. Ed. by 王一玲. 北京: 清华大学出版社, 2008.

A 支撑材料列表

表 2: 支撑材料列表

MATLAB	Graph_and_Path.m
	Graph_and_Path_2.m
	stchange.m
	Matrix_1.mat
	str0.mat
	Best_Path.m
Python	Best_Solution.py
	Recurrence.py
Excel	Result.xlsx

B 证明

B.1 定理 4.1 的证明.

证明. 假设存在某一种经过矿山的决策方案，使得玩家的收益最大，且其经过节点构成的子图 G' 并不包含于某两条最短路径构成的子图 G，记这种决策方案为 S'，首先假设 S' 对应的路径如下：

$$\text{起点} \longrightarrow \text{村庄} \longrightarrow \text{矿山} \longrightarrow \text{终点} \tag{9}$$

并不含环，那么那么我们如下构造决策方案

$$S(t) = \begin{cases} \text{留在原地} & S'(t) = \text{留在原地} \\ \text{留在原地} & S \text{ 在 } t \text{ 时刻已经到达村庄或矿山但 } S' \text{ 未到达} \\ \text{在 } G \text{ 对应的路径上向前移动} & S'(t) = \text{移动} \\ \text{购买物资，数量与 } S' \text{ 相同} & S'(t) = \text{购买物资} \\ \text{挖矿} & S'(t) = \text{挖矿} \end{cases} \tag{10}$$

首先，S 一定是良定的，因为根据引理 4.1，G 对应的路径是从起点到村庄的最短路径，所以按照 S 的构造，在策略 S 之下玩家一定先于 S' 到达村庄，同理，也一定先于 S' 到达矿山.

我们记策略 S 下玩家身上的食物余量为 f，水余量为 w，S' 下则分别为 f', w'，那么关于某时刻 t 的决策产生的食物与水的变化量，我们有

$$|\Delta f(t)| \leq |\Delta f'(t)| \tag{11}$$

$$|\Delta w(t)| \leq |\Delta w'(t)| \tag{12}$$

又因为 S' 对应的路线并非最短，故一定存在某个 t 使得

$$S'(t) = \textit{移动}$$

$$S(t) = \textit{留在原地}$$

故一定存在某个 t 使得等号不成立，又由于中途每天购入的水与食物两个决策完全相同，我们就知道决策 S 产生的消耗小于 S' 产生的消耗. 故 S 对应的最终 m_0 与 S' 对应的最终 m_0' 有关系

$$m_0 > m_0' \tag{13}$$

假如 S' 对应的路径含环（即在决策 S' 之下玩家会走“回头路”），那么按照同样的构造方法，我们知道对于任意一个环结构上的决策 C'，例如：

村庄 ⇄ 矿山

我们按照与上面相同的方法构造对应的 C，则我们知道 C 对应的食物、水的消耗依旧与 C' 对应的消耗有如式 11,12 的关系，接下来进行与不含环的情况相同的分析，我们就得到了式 13.

这样，通过对含环的策略与不含环的策略进行分别讨论，我们就证明了定理 4.1. □

B.2 定理 4.2 的证明.

证明. 我们首先给出一条较为显然的引理：

引理 B.1. *当挖矿的总时间一定时，去矿山的次数越少，策略的收益越高.*

假如某个经过矿山两侧的策略 S' 想要取得比 S 更高的收益，则根据引理 B.1 其一定挖矿的天数高于 S'. 又由于从矿山至村庄再返回至少需要四天，根据 Best_Solution_1.py 的计算结果，我们知道其最早在第 30 日到达终点，其中有连续的四天在往返村庄、矿山的路上. 其中与 Best_Solution_1.py 得出的解相比多出了三个高温天气、一个沙暴天气与一个晴天. 经过观察，多出的行动实际上是 26-30 日的行走路程与任意一天的挖掘路程. 也就是说，假如将食物与水的消耗转化为金钱，则其与 Best_Solution_1.py 的金钱消耗相差 Δm 如下计算

$$\Delta m = \Delta m_{\textit{挖矿}} + \Delta m_{\textit{行走}} \geq 3 \times 95 + 2 \times 490$$

但是挖矿一天能赚的的金钱仅有 1000 元，故并不足以弥补多出的路途消耗. □

C 源程序

C.1 问题 1

C.1.1 Matlab 程序

Graph_and_Path.m[2]

```
Graph_1=graph(Matrix_1); %由邻接矩阵生成问题1的图
S=sparse(Matrix_1); %由邻接矩阵生成稀疏矩阵S
[d_1,p_1]=graphshortestpath(S,1,12);
%求得从起点到矿山的最短距离与路径
[d_2,p_2]=graphshortestpath(S,12,27);
%求得从矿山到终点的最短距离和路径
plot(Graph_1) %画出图
```

Graph_and_Path_2.m

```
A=zeros(64); %初始化
for i=1:8
    for j=1:8
        if i==1
            if j==1
                [A(1,2),A(1,9)]=deal(1);
            elseif j==8
                    [A(8,7),A(8,16),A(8,15)]=deal(1);
            else [A(j,j+1),A(j,j-1),A(j,j+8),A(j,j+7)]=deal(1);
            end
        elseif i==8
            if j==1
                [A(57,49),A(57,50),A(57,58)]=deal(1);
            elseif j==8
                [A(64,63),A(64,56)]=deal(1);
            else [A(56+j,55+j),A(56+j,57+j),A(56+j,48+j),A(56+j
               ,49+j)]=deal(1);
            end
```

```
        elseif mod(i,2)==0
            if j==1
                [A(8*(i-1)+j,8*(i-1)+j-8),A(8*(i-1)+j,8*(i-1)+j-7)
                    ,A(8*(i-1)+j,8*(i-1)+j+1),A(8*(i-1)+j,8*(i-1)+j
                    +8),A(8*(i-1)+j,8*(i-1)+j+9)]=deal(1);
            elseif j==8
                [A(8*(i-1)+j,8*(i-1)+j-8),A(8*(i-1)+j,8*(i-1)+j+8)
                    ,A(8*(i-1)+j,8*(i-1)+j-1)]=deal(1);
            else [A(8*(i-1)+j,8*(i-1)+j-1),A(8*(i-1)+j,8*(i-1)+j
                -8),A(8*(i-1)+j,8*(i-1)+j-7),A(8*(i-1)+j,8*(i-1)+j
                +1),A(8*(i-1)+j,8*(i-1)+j+8),A(8*(i-1)+j,8*(i-1)+j
                +9)]=deal(1);
            end
        else
            if j==1
                [A(8*(i-1)+j,8*(i-1)+j-8),A(8*(i-1)+j,8*(i-1)+j+8)
                    ,A(8*(i-1)+j,8*(i-1)+j+1)]=deal(1);
            elseif j==8
                [A(8*(i-1)+j,8*(i-1)+j-8),A(8*(i-1)+j,8*(i-1)+j-9)
                    ,A(8*(i-1)+j,8*(i-1)+j-1),A(8*(i-1)+j,8*(i-1)+j
                    +8),A(8*(i-1)+j,8*(i-1)+j+7)]=deal(1);
            else [A(8*(i-1)+j,8*(i-1)+j-1),A(8*(i-1)+j,8*(i-1)+j
                -8),A(8*(i-1)+j,8*(i-1)+j-9),A(8*(i-1)+j,8*(i-1)+j
                +1),A(8*(i-1)+j,8*(i-1)+j+8),A(8*(i-1)+j,8*(i-1)+j
                +7)]=deal(1);
            end
        end
    end
end
%生成邻接矩阵A
Graph_2=graph(A); %生成对应的图
S=sparse(A); %由邻接矩阵生成对应的稀疏矩阵
[d_1,p_1]=graphshortestpath(S,1,30);
```

```
[d_2,p_2]=graphshortestpath(S,1,39);
[d_3,p_3]=graphshortestpath(S,1,55);
[d_4,p_4]=graphshortestpath(S,1,62);
plot(Graph_2) %绘制图
```

Best_Path.m

```
flags=zeros(3,10);
str=zeros(1,10);
date=1;
node=[1,0,0,0,0,0,0,0,0,0];
money=zeros(1,11);
food=zeros(1,11);
water=zeros(1,11);
max_money=-1e10;
while date~=0
    if date>10 && node(date)<6
        str(date)=0;
        [money(date),food(date),water(date)]=deal(0);
        date=date-1;
        if date==0
            break
        end
        node(date+1)=0;
    elseif node(date)==6
        if max_money<money(date)+10*food(date)+20*water(date)
            max_money=money(date)+10*food(date)+20*water(date);
            str0=str;
        end
        str(date)=0;
        [money(date),food(date),water(date)]=deal(0);
        node(date)=0;
        date=date-1;
        if date==0
```

```
28              break
29          end
30      elseif node(date)~=4
31          for strategy=0:1
32              if flags(strategy+1,date)==0
33                  flags(strategy+1,date)=1;
34                  if strategy==1
35                      node(date+1)=node(date)+1;
36                  else
37                      node(date+1)=node(date);
38                  end
39                  [money(date+1),food(date+1),water(date+1)]=
                        stchange(money(date),food(date),water(date),
                        strategy,date);
40                  str(date)=strategy;
41                  date=date+1;
42                  break
43              end
44              if flags(1,date)==1 && flags(2,date)==1
45                  str(date)=0;
46                  node(date)=0;
47                  [money(date),food(date),water(date)]=deal(0);
48                  flags(:,date)=0;
49                  date=date-1;
50                  if date==0
51                      break
52                  end
53              end
54          end
55      elseif node(date)==4
56          if flags(1,date)==1 && flags(3,date)==1 && flags(2,date)
                ==0
57              str(date)=0;
```

```
58              node(date)=0;
59              [money(date),food(date),water(date)]=deal(0);
60              flags(:,date)=0;
61              date=date-1;
62              if date==0
63                  break
64              end
65          end
66          for strategy=0:2
67              if flags(strategy+1,date)==0
68                  flags(strategy+1,date)=1;
69                  if flags(:,date)==1
70                      flags(2,date)=0;
71                  end
72                  if strategy==1
73                      node(date+1)=node(date)+1;
74                  else
75                      node(date+1)=node(date);
76                  end
77                  [money(date+1),food(date+1),water(date+1)]=
                        stchange(money(date),food(date),water(date),
                        strategy,date);
78                  str(date)=strategy;
79                  date=date+1;
80                  break
81              end
82          end
83      end
84  end
```

C.1.2 Python 程序

Best_Solution_1.py

```python
Weather = [None, 1, 1, 0, 2, 0, 1, 2, 0, 1, 1,
                 2, 1, 0, 1, 1, 1, 2, 2, 1, 1,
                 0, 0, 1, 0, 2, 1, 0, 0, 1, 1]
                 # 0: 晴朗, 1: 高温, 2: 沙暴, 索引对应天数
supply_amount = [(5, 7), (8, 6), (10, 10)]  # 索引对应天气
max_mine_day = 7


def get_time_table(binary_code: str, max_mine_day: int) -> dict:
    time_table = {}
    counter = 0
    for day in range(1, 31):
        if binary_code[day] == '1':
            counter += 1
        if counter == 6:
            time_table['village_t_1'] = time_table.get('
                village_t_1', day)
        elif counter == 8:
            time_table['arrive_mine_t'] = time_table.get('
                arrive_mine_t', day)
        elif counter == 9:
            time_table['start_mine_t'] = time_table.get('
                start_mine_t', day)
        elif counter == 8 + max_mine_day:
            time_table['end_mine_t'] = time_table.get('end_mine_t'
                , day)
            time_table['walk_again_t'] = time_table.get('
                end_mine_t', day) + 1
        elif counter == 10 + max_mine_day:
            time_table['village_t_2'] = time_table.get('
                village_t_2', day)
        elif counter == 13 + max_mine_day:
            time_table['end_game_t'] = time_table.get('end_game_t'
```

```
                , day)
    return time_table


def stay_when_storm(binary_code, time_table) -> bool:
    for date in range(1, time_table['arrive_mine_t'] + 1):
        if Weather[date] == 2 and binary_code[date] == '1':
            return True
    for date in range(time_table['walk_again_t'], time_table['
       end_game_t'] + 1):
        if Weather[date] == 2 and binary_code[date] == '1':
            return True
    return False


def calc_supply_by_stage(binary_code, time_table, type):  # 箱
   type=1:food, type=0:water
    box = {}; cost = 0
    for date in range(1, time_table['village_t_1'] + 1):
        if binary_code[date] == '0':
            cost += supply_amount[Weather[date]][type]
        else:
            cost += supply_amount[Weather[date]][type] * 2
    box['start_to_village'] = cost; cost = 0
    for date in range(time_table['village_t_1'] + 1, time_table['
       arrive_mine_t'] + 1):
        if binary_code[date] == '0':
            cost += supply_amount[Weather[date]][type]
        else:
            cost += supply_amount[Weather[date]][type] * 2
    box['village_to_mine'] = cost; cost = 0
    for date in range(time_table['arrive_mine_t'] + 1, time_table[
       'end_mine_t'] + 1):
```

```
56          if binary_code[date] == '0':
57              cost += supply_amount[Weather[date]][type]
58          else:
59              cost += supply_amount[Weather[date]][type] * 3
60      box['mine'] = cost; cost = 0
61      for date in range(time_table['end_mine_t'] + 1, time_table['
            village_t_2'] + 1):
62          if binary_code[date] == '0':
63              cost += supply_amount[Weather[date]][type]
64          else:
65              cost += supply_amount[Weather[date]][type] * 2
66      box['mine_to_village'] = cost; cost = 0
67      for date in range(time_table['village_t_2'] + 1, time_table['
            end_game_t'] + 1):
68          if binary_code[date] == '0':
69              cost += supply_amount[Weather[date]][type]
70          else:
71              cost += supply_amount[Weather[date]][type] * 2
72      box['village_to_end'] = cost
73      return box
74
75
76  def main():
77      possible = []
78      for i in range(2 ** 30):
79          binary_code = '*' + bin(i)[2:].rjust(30, '0')  # 高位填充0
80          if binary_code.count('1') != max_mine_day + 13:
81              continue
82          time_table = get_time_table(binary_code, max_mine_day)
83          if stay_when_storm(binary_code, time_table):
84              continue
85          food_box = calc_supply_by_stage(binary_code, time_table,
                1)
```

```
86          water_box = calc_supply_by_stage(binary_code, time_table,
                0)
87          initial_food = int(600+food_box['start_to_village']-3*(
                water_box['village_to_mine']+water_box['mine']+
                water_box['mine_to_village'])/2)
88          initial_food = min(initial_food, sum(food_box.values())
89          initial_water = (1200 - initial_food * 2) // 3
90          left_weight = 1200 - initial_food * 2 - initial_water * 3
91          if initial_water < water_box['start_to_village']:
92              continue  # 不跳出循环说明能走到村庄
93          left_weight += (food_box['start_to_village'] * 2 +
                water_box['start_to_village'] * 3)
94          water_in_village_1 = left_weight // 3
95          left_weight -= water_in_village_1 * 3
96          if water_in_village_1 + initial_water < sum(water_box.
                values()) - water_box['village_to_end']:
97              continue  # 不跳出循环说明能再次走到村庄
98          if initial_food < sum(food_box.values()) - food_box['
                village_to_end']:
99              continue
100         water_in_village_2 = sum(water_box.values()) -
                initial_water - water_in_village_1
101         food_in_village_2 = sum(food_box.values()) - initial_food
102         revenue = 10000 + max_mine_day * 1000 - 10 * (initial_food
                + water_in_village_1 + water_in_village_2) - 5 *
                initial_water - 20 * food_in_village_2
103         possible.append((binary_code, revenue))
104         if revenue > 10000:
105             print(revenue)
106     possible.sort(key=lambda x: x[1], reverse=True)
107     for i in range(5):
108         print(possible[i])
109
```

```

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

Recurrence.py

```
P = 0.9  # 晴天的概率
INF = 1e5
m_use_e_2 = [[0.0]*180 for i in range(180)]
m_use_e_1 = [[0.0]*180 for i in range(180)]
m_use_e_0 = [[0.0]*180 for i in range(180)]


def initialize():
    for w in range(180):
        for f in range(180):
            if w < 18 or f < 18:
                m_use_e_2[w][f] = INF
            if w < 36 or f < 36:
                m_use_e_1[w][f] = INF
            if w < 54 or f < 54:
                m_use_e_0[w][f] = INF


def fill():
    for w in range(18, 180):
        for f in range(18, 180):
            m_use_e_2[w][f] = 110 * P + (1-P) * min(270, 135 +
                m_use_e_2[w-9][f-9])  # fill 2
    for w in range(36, 180):
        for f in range(36, 180):
            m_use_e_1[w][f] = P * (110 + m_use_e_2[w-6][f-8]) +
                (1-P) * min(270 + m_use_e_2[w-18][f-18], 135 +
                m_use_e_1[w-9][f-9])  # fill 1
```

```
    for w in range(54, 180):
        for f in range(54, 180):
            m_use_e_0[w][f] = P * (110 + m_use_e_1[w-6][w-8]) +
                (1-P) * min(270+m_use_e_1[w-18][f-18], 135 +
                m_use_e_0[w-9][f-9])



def main():
    initialize()
    fill()
    for i in m_use_e_0:
        for j in i:
            print(f'{j:.3f}',end=', ')
        print()

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

D 图表

表 3: 第一关的最优解

日期	所在区域	剩余资金数	剩余水量	剩余食物量
0	1	5780	178	333
1	25	5780	162	321
2	24	5780	146	309
3	23	5780	136	295
4	23	5780	126	285
5	22	5780	116	271
6	9	5780	100	259
7	9	5780	90	249
8	15	4150	243	235
9	14	4150	227	223
10	12	4150	211	211
11	12	4150	201	201
12	12	5150	177	183
13	12	6150	162	162
14	12	7150	138	144
15	12	8150	114	126
16	12	9150	90	108
17	12	9150	80	98
18	12	10150	50	68
19	12	11150	26	50
20	14	11150	10	38
21	15	10470	36	40
22	9	10470	26	26
23	21	10470	10	14
24	27	10470	0	0

表 4: 第二关的最优解

日期	所在区域	剩余资金数	剩余水量	剩余食物量
0	1	5300	130	405
1	2	5300	114	393
2	3	5300	98	381
3	4	5300	88	367
4	4	5300	78	357
5	5	5300	68	343
6	13	5300	52	331
7	13	5300	42	321
8	22	5300	32	307
9	30	5300	16	295
10	39	3190	211	283
11	39	2890	211	283
12	47	2890	195	271
13	55	2890	185	257
14	55	3890	161	239
15	55	4890	137	221
16	55	5890	113	203
17	55	6890	83	173
18	55	7890	53	143
19	62	4730	201	207
20	55	4730	185	195
21	55	5730	170	174
22	55	6730	155	153
23	55	7730	131	135
24	55	8730	116	114
25	55	9730	86	84
26	55	10730	62	66
27	55	11730	47	45
28	55	12730	32	24
29	56	12730	16	12
30	64	12730	0	0