穿越沙漠问题

摘 要

本文针对沙漠掘金问题进行了研究，在不同的条件之下给出了多种简化方法以及决策模型，以针对最多金钱路线进行探究。

针对问题 1，考虑确定信息，本文利用资源无剩余、最短路径策略、无向图简化、多余边去除的简化方法和晴朗天气不停留、最多挖矿、补货贪心的通用策略，以及动态规划最小花费下界策略来进行路径选择。

针对问题 2，考虑不确定天气情况，对于沙漠地图进行最初路线规划一致性、无向图简化、多余边去除、晴朗天气不停留、预决策进行信息决策简化。同时利用天气概率，提出基于天气概率的直达终点最短花费模型，得到高温下在花费和天气信息下的决策行为，并结合玩家决策倾向综合考虑各种因素最优策略的玩家最终决策。对于挖矿行为，本文提出基于逃离阈值的挖矿停止指导策略，以玩家设定逃离阈值（因为资源缺少而中途结束游戏的概率）计算村庄补货、前往终点的逃离时机。

针对问题 3，我们做出最短路径简化，以基于混合 Nash 模型给出博弈决策形式，在具体情况之下，同时对于多人挖矿情形给出基于资源相似度的考虑法则和提早补货的决策。

本文模型方法多样，涵盖有等效简化、降维简化、综合指导策略、定量决策判定等思想，同时原理直观、最终形式直接，符合一般直觉。缺点是无法给予通用的定量策略。

关键词：无向图；动态规划；概率模型；博弈论

一、问题重述

1.1 问题背景

本问题的背景为如下小游戏：

玩家拥有地图，可利用初始资金购买一定数量的水和食物，从起点出发，在沙漠中行走。游戏目标是在规定时间内到达终点，并保留尽可能多的资金。

游戏规则如下：

（1）基本设置：以天为基本时间单位，游戏的开始时间为第 0 天，玩家位于起点。玩家必须在截止日期或之前到达终点，到达终点后该玩家的游戏结束。

（2）资源约束：每天玩家拥有的水和食物质量之和不能超过负重上限。若未到达终点而水或食物已耗尽，视为游戏失败。

（3）天气约束：“晴朗”、“高温”、“沙暴”三种状况之一。

（4）玩家行动：每天玩家可从地图中的某个区域到达与之相邻的另一个区域，也可在原地停留（沙暴日必须在原地停留）。玩家在矿山停留时，可通过挖矿获得资金，挖矿一天获得的资金量称为基础收益。到达矿山当天不能挖矿。沙暴日也可挖矿。

（6）玩家的行动与其资源消耗量：

原地停留一天	基础消耗量（与天气有关）
行走一天	基础消耗量×2
挖矿一天	基础消耗量×3

（7）物资购买价格与退回价格：

起点处购买物资（仅可买一次）	基准价格
村庄处购买物资	基准价格×2
终点处退回剩余物资	基准价格×1/2

1.2 问题提出

根据游戏的不同设定，需要解决以下问题：

1. 假设只有一名玩家，在整个游戏时段内每天天气状况事先全部已知，给出一般情况下玩家的最优策略，并求解附件中的“第一关”和“第二关”。

2. 假设只有一名玩家，玩家仅知道当天的天气状况，可据此决定当天的行动方案，试给出一般情况下玩家的最佳策略，并对附件中的“第三关”和“第四关”进行具体讨论。

3. 现有 n 名玩家，他们有相同的初始资金，且同时从起点出发。玩家的行动与其单人资源消耗量如下：

原地停留一天	基础消耗量（与天气有关）
单独行走一天	基础消耗量×2
任意$k\left(2\le k\le n\right)$名玩家**一同行走**（均从区域 A 走到区域 B）	基础消耗量×$2k$
挖矿一天	基础消耗量×3

若单人挖矿，则获得的资金为基础收益；若任意$k\left(2\le k\le n\right)$名玩家在同一矿山挖矿，则他们中的任一位获得的资金是基础收益×$\frac{1}{k}$；

多玩家情况下，物资购买价格与退回价格如下：

起点处购买物资（仅可买一次）	基准价格
单独在村庄处购买物资	基准价格×2
任意$k\left(2\le k\le n\right)$名玩家在**同一**村庄购买物资	基准价格×4
终点处退回剩余物资	基准价格×1/2

（1）假设在整个游戏时段内每天天气状况事先全部已知，每名玩家的行动方案需在第0天确定且此后不能更改。试给出一般情况下玩家应采取的策略，并对附件中的“第五关”进行具体讨论。

（2）假设所有玩家仅知道当天的天气状况，从第1天起，每名玩家在当天行动结束后均知道其余玩家当天的行动方案和剩余的资源数量，随后确定各自第二天的行动方案。试给出一般情况下玩家应采取的策略，并对附件中的“第六关”进行具体讨论。

二、问题分析

2.1 问题一分析

问题一中，地图可被简化成由起点、终点、村庄和矿山这些关键点组成的无向图。目前考虑两种不同的策略：

(1) 不经过矿山：玩家对所有天气状况均已知，故可利用动态规划得到 t 天内从起点到指定区域 d 处的最小花费$C[t,d]$，加以对剩余资源数量和资源限重的限制，获得不经过矿山到达终点的最佳路径。

(2) 经过矿山或村庄等特定点：对玩家行动进行模拟，玩家以最短路径从起点到矿山或村庄、从矿山或村庄到终点，在矿山停留挖矿的时间和在村庄购买物资的以资源限重为约束条件，加以贪心策略而获得该条件下的最佳路径。

2.2 问题二分析

问题二中，玩家仅知道当天的天气状况，可据此决定当天的行动方案，主要考虑以下两个策略的构建：

(1) 初始状态时路线和资源的购买方案；

(2) 路途中基于天气概率、所处位置和所剩物资的行动方案。

现将路线规划分为两种：直接前往终点模式和挖矿赚钱模式。

对于直接前往终点模式，玩家在起点处备满路径中需要的物资，尽量减少购买物资的消费，即减少路上资源的消耗，获取最佳策略；对于挖矿赚钱模式，玩家有无法到达终点的风险，这里需设置游戏失败的额外损失，。

对于具体的第三关和第四关地图与参数设置，需要先判断其属于前往终点模式或挖矿赚钱模式，再采取基于概率的直达终点最短花费决策模型或基于逃离阈值的挖矿停止策略。

2.3 问题三分析

针对问题 3，我们做出最短路径简化，以基于混合 Nash 模型给出博弈决策形式，在具体情况之下，同时对于多人挖矿情形给出基于资源相似度的考虑法则和提早补货的决策。

三、模型假设与符号说明

3.1 模型的假设

(1) 问题 1 和问题 2 中对“最优策略”的定义为：使得玩家游戏成功且游戏结束时持有资金最多的策略。

(2) 其他具体假设见各题部分。

3.2 符号的说明

表 1 符号的说明

符号	说明	单位
$W_{\max}$	负重上限	kg
I	开始时的初始资金	元
$wage$	挖矿时的基础收益	元/天
$m_{水}, m_{食}$	每箱水/食物的质量	kg
$price_{水}, price_{食}$	每箱水/食物的基准价格	元/箱
$cost_{水}(weather),$	$weather$ =晴朗，晴朗天气的基础消耗量	箱

	weather =高温，高温天气的基础消耗量	
	weather =沙暴，沙暴天气的基础消耗量	
cost (*weather*)	某天气下水和食物基础消耗量的基准总价	元
$C\ [t_s, t_e, d]$	由第 t_s 天到 t_e 天经过 d 次转移的最小花费	元
dis(s, d)	从 s 处到 d 处的最短路径中转移次数	
p_i, p_j, p_k	p_i, p_j, p_k 分别表示晴朗、高温、沙暴天气概率	

四、模型建立与求解

4.1 问题一

问题一中所述的“最优策略”为让游戏成功且结束时持有资金最多的策略。游戏结束时，玩家持有资金 = 初始资金 - 总支出 + 总收入，其中总支出由玩家的资源消耗量决定，如结论 1 所述：

结论 1：最优策略使得玩家资源在游戏结束时不会有剩余。

证明：这里使用反证法。假设策略 A 为最优策略，且使得玩家资源在结束时有剩余，则总存在策略 A'，在策略 A 的最后一次购买环节（无论在起点或在村庄），不购买策略 A 所剩余的资源。这样策略 A' 依然可以让游戏成功，且相比原策略 A，在总收入不变的情况下减少了总支出，玩家结束时拥有的资金更多，因此策略 A' 优于策略 A，与假设矛盾，故结论 1 成立。

在问题一中，为了让游戏结束时持有资金更多，我们考虑两种行走策略：

(1) 以最短路径去终点：虽然总收入 $=0$，但总支出达到最小；

(2) 去终点的路上在矿山挖矿：虽然路上耗费资源更多，总支出会比(1)大，但总收入 $\neq 0$，有可能让最终持有资金 $>$ 初始资金。

在实施两个策略前，先要对地图化简。

4.1.1 地图化简

问题一的两个地图可以看作是无向连通图，选取起点、终点、村庄和矿山这四类关键点，典型的行进路线如下图 1 所示，路线中的非关键点省略：

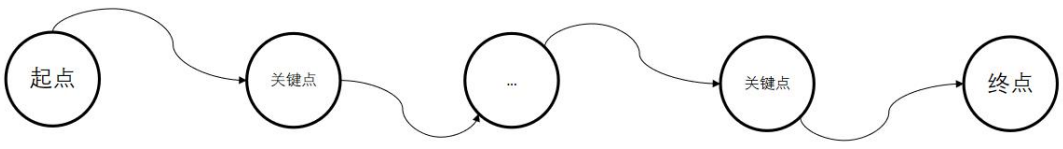

图 1 略去非关键点的最优行进路线

结论 2：关键点之间的最短路径必为最佳策略的组成部分。

证明：关键点之间的最短路径为需要移动次数最少的路径。整个沙漠天气都相同，意味着同一天移动策略相同时，$cost_{水}(weather)$, $cost_{食}(weather)$均相同。对于任意非最短路径，在采取和最短路径相同的移动策略后无法到达关键点，仍需要额外的移动消耗量，故最短路径是最佳策略的组成部分。

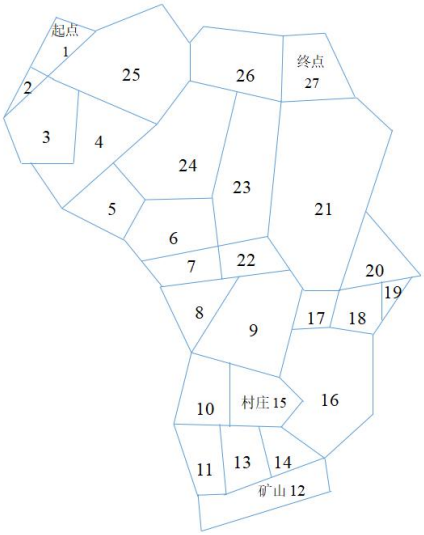

图 2(a) 第一关地图

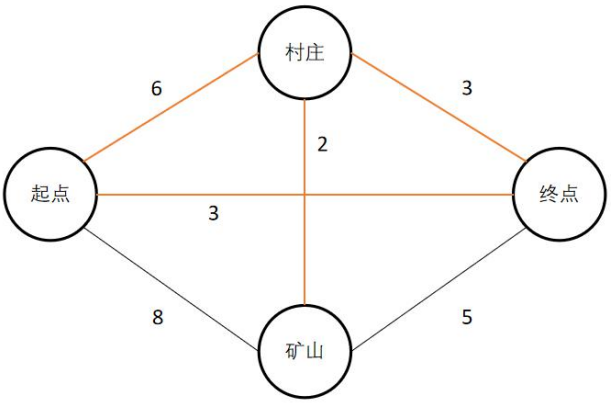

图 2(b) 第一关地图的关键点之间的完全图

因此非关键点位置就可以从无向图中省略，我们可以将无向图转化为仅有关键点的完全图，边的权值为点之间的最短路径长度，以“第一关”的地图为例，由多块区域组成的图简化成了关键点之间的完全图，如图 2(a)和图 2(b)。

现对完全图进行简化，简化规则为：假设该边两端是关键点 A 和 B，若该边不为 A 和 B 之间的最短路径则去除。如图 3 所示，该边为多余边，可以删除的理由如下：

这里使用反证法。假设策略 A 为最优策略，策略中含有从关键点 A 到 B 的边，则总存在策略 A'，从 A 到 B 的路径中选择经由 C，且 A→C→B 路径更短。这样策略 A' 在天气固定和总收入不变的情况下移动的次数更少，减少了总支出，玩家结束时拥有的资金更多，因此策略 A' 优于策略 A，与假设矛盾，故最优策略中从 A 到 B 一定会经由 C，关键点 A 和 B 之间的边不会被使用，所以去除它完全不影响策略规划。

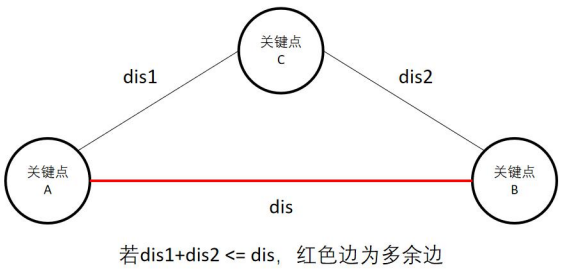

图 3 多余边示意图

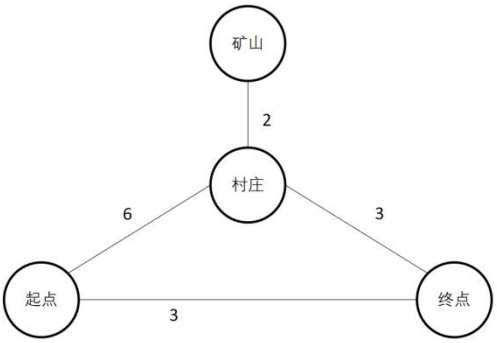

图 4 第一关的地图简化图

如图 2(b)，非多余的边已被标红，提取出来可得到最终简化的地图（图 4）。

类似地，如图 5(a)(b)所示，第二关的地图也可进行如下简化。

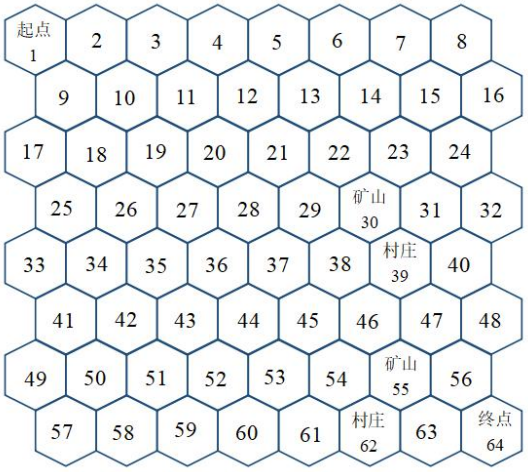

图 5(a) 第二关的地图

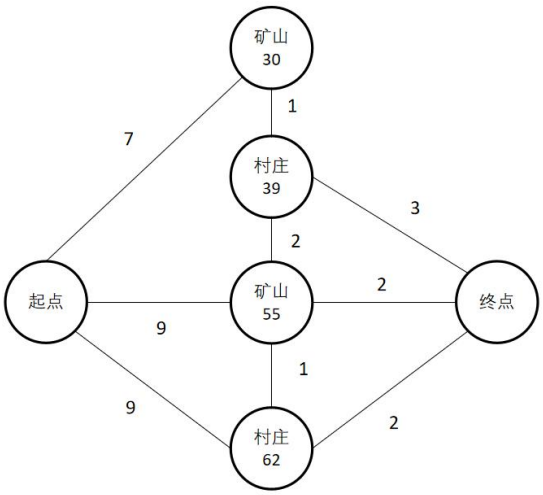

图 5(b) 第二关的地图简化图

4.1.2 通用策略：贪心

贪心策略总是做出在当前看来是最优的选择，即贪心策略并不是从整体上加以考虑，它所做出的选择只是局部最优解。考察本题“第一关”“第二关”的参数设定（如下表 2），作出以下约定：

表 2 “第一关”“第二关”的参数设置

负重上限		1200千克	初始资金		10000元
截止日期		第30天	基础收益		1000元
资源	每箱质量（千克）	基准价格（元/箱）	基础消耗量（箱）		
			晴朗	高温	沙暴
水	3	5	5	8	10
食物	2	10	7	6	10

(1) 晴朗天气不停留：

由上表可知，无论是水或是食物，晴朗天气的基础消耗量 $cost_{水}()$, $cost_{食}()$ 最小，因此在移动步数一定的情况下，尽可能在晴朗天气下进行移动而非停留。

(2) 最多挖矿原则：

根据(1)，若玩家处在矿山且不行走，则必为挖矿。现考虑在矿山的高温和沙暴天气，高温或沙暴天气下所消耗资源的基准总价为：

$$cost(weather) = price_{水} * cost_{水}(weather) + price_{食} * cost_{食}(weather)$$

其中 *weather* 为高温或沙暴。

现考察在沙暴中是否挖矿，观察到表 2 满足以下关系：

$$3 * cost(\text{沙暴}) \leqslant 3 * cost(\text{高温}) + cost(\text{沙暴})$$

即表明在矿山赚一次基础收益，在沙暴挖矿的成本低于停留一次再挖矿，且在沙暴中挖矿比不挖矿赚一次基础收益更快，无论在时间上还是在资源商，沙暴中挖矿均优于不挖矿。

因此对于本题第一关和第二关，将挖矿置于最多策略，无论天气如何，在资源足够的情况下一定挖矿，直到资源恰好可支撑前往村庄或终点。

(3) 最多补货原则：

本题的补货符合 01 背包问题，在表 2 中出现绝对性价比占优的“食物”，即 $m_{水} > m_{食}$， $price_{水} < price_{食}$，因此考虑到村庄补货价格翻倍，我们在起点处总希望更多地购买“食物”，以降低在起点处和村庄处的总支出；此外，当起点购买的资源会被用完的情况下，我们需要在起点尽量装满背包，即最大化利用背包限重 $W_{\max}$。

4.1.3 指导策略：动态规划

从一般情况下考虑，最佳路径的寻找可以分为两步：

(1) 在无限资源的前提下，分别找不经过矿山和经过矿山的路径；

(2) 考察以上路径所需要的资源，若无法直达，则需经过村庄。

下面通过动态规划给出最小花费下界策略：

设由第 t_s 天到第 t_e 天经过 d 次转移的最小花费 $C[t_s, t_e, d]$，则在第 t_e-1 天玩家可选择停留或转移，故状态转移方程为：

$C[t_s,t_e,d]$ = min{ $C[t_s,t_e-1,d-1]$ + 2 * $cost$ ($weather$), $C[t_s,t_e-1,d]$ + $cost$ ($weather$)}，其中 $weather$ 为第 t_e 天的天气（且不为沙暴）；

初始化为：$C[t_s,t_s,0] = 0$, $C[t_s,t_s+1,1] = 2*cost(weather)$，其中 $weather$ 为第 t_s+1 天的天气。

在第 t_s 天到第 t_e 天中，设初始位置为 s，结束位置为 e：由 4.1.1 节的结论 2 可知，最佳策略包含关键点之间的最短路径 $dis(s,e)$。

若初始位置 s 到结束位置 e 会经过村庄，则可以通过上述解出的 C，求出让这 d 次转移花费最小时到达村庄的时间 t_i：

$$\min_{t_s<t_i<t_e}\{C[t_s,t_i,dis(s,\text{村})]+C[t_i,t_d,dis(\text{村},e)]\}$$

若初始位置 s 到结束位置 e 会经过矿山，则求出让这 d 次转移花费最小时在矿山所呆的时间 t_i， t_j，即第 t_i+ 1 天到第 t_j 天在矿山挖矿：

$$\min_{t_s<t_i<t_j<t_e}\{C[t_s,t_i,dis(s,\text{矿})]+M[t_i+1,t_j]+C[t_j,t_d,dis(\text{矿},e)]\}$$

其中 $M[t_i,t_j]=\sum_{t=t_i}^{t_j-1}[3*cost(weather_t)-wage]$，$weather_t$ 为第 t 天的天气。

在第一关和第二关的设定中，解得 C 空间在 $t_s=0$ 的投影如下：

	0	1	2	3	4	5	6	7	8	9	10	11
0	0	1048575	1048575	1048575	1048575	1048575	1048575	1048575	1048575	1048575	1048575	1048575
1	100	200	1048675	1048675	1048675	1048675	1048675	1048675	1048675	1048675	1048675	1048675
2	200	300	400	1048775	1048775	1048775	1048775	1048775	1048775	1048775	1048775	1048775
3	295	390	490	590	1048870	1048870	1048870	1048870	1048870	1048870	1048870	1048870
4	445	540	640	740	1049020	1049020	1049020	1049020	1049020	1049020	1049020	1049020
5	540	635	730	830	930	1049115	1049115	1049115	1049115	1049115	1049115	1049115
6	640	735	830	930	1030	1130	1049215	1049215	1049215	1049215	1049215	1049215
7	790	885	980	1080	1180	1280	1049365	1049365	1049365	1049365	1049365	1049365
8	885	980	1075	1170	1270	1370	1470	1049460	1049460	1049460	1049460	1049460
9	985	1080	1175	1270	1370	1470	1570	1670	1049560	1049560	1049560	1049560
10	1085	1180	1275	1370	1470	1570	1670	1770	1870	1049660	1049660	1049660
11	1235	1330	1425	1520	1620	1720	1820	1920	2020	1049810	1049810	1049810
12	1335	1430	1525	1620	1720	1820	1920	2020	2120	2220	1049910	1049910
13	1430	1525	1620	1715	1810	1910	2010	2110	2210	2310	2410	1050005
14	1530	1625	1720	1815	1910	2010	2110	2210	2310	2410	2510	2610
15	1630	1725	1820	1915	2010	2110	2210	2310	2410	2510	2610	2710
16	1730	1825	1920	2015	2110	2210	2310	2410	2510	2610	2710	2810
17	1880	1975	2070	2165	2260	2360	2460	2560	2660	2760	2860	2960
18	2030	2125	2220	2315	2410	2510	2610	2710	2810	2910	3010	3110
19	2130	2225	2320	2415	2510	2610	2710	2810	2910	3010	3110	3210
20	2230	2325	2420	2515	2610	2710	2810	2910	3010	3110	3210	3310
21	2325	2420	2515	2610	2705	2800	2900	3000	3100	3200	3300	3400
22	2420	2515	2610	2705	2800	2895	2990	3090	3190	3290	3390	3490
23	2520	2615	2710	2805	2900	2995	3090	3190	3290	3390	3490	3590
24	2615	2710	2805	2900	2995	3090	3185	3280	3380	3480	3580	3680
25	2765	2860	2955	3050	3145	3240	3335	3430	3530	3630	3730	3830
26	2865	2960	3055	3150	3245	3340	3435	3530	3630	3730	3830	3930
27	2960	3055	3150	3245	3340	3435	3530	3625	3720	3820	3920	4020
28	3055	3150	3245	3340	3435	3530	3625	3720	3815	3910	4010	4110
29	3155	3250	3345	3440	3535	3630	3725	3820	3915	4010	4110	4210
30	3255	3350	3445	3540	3635	3730	3825	3920	4015	4110	4210	4310

图 6 动态规划求解出的 $C[0]$（截图只反映了前 11 列），即 $t_s=0$

其中 $C[0,i,j]$表示第 0 天到第 i 天经过 j 次转移的花费，红色区域表示不可达，蓝色区域表示转移可实现，设所有可实现区域（蓝色区域）集合为 U。

在第一关中，从起点到终点的最短距离为 $j=3$，在上图该列中可以找到最小值为 590，即第一关直接前往终点（不考虑矿山挣钱）的花费最小是 590；在第二关中，从起点到终点的最短距离为 $j=11$，在上图该列中可以找到最小值为 2610，即第二关直接前往终点（不考虑矿山挣钱）的花费最小是 2610。

动态规划得到的矩阵可起到指导作用，对于特定 t_s，可以观察到 $C[t_s]$的每一列由上至下递增，即

$$C[t_s,t_e,d]\geq C[t_s,t_e-1,d],\ (t_s,t_e-1,d)\in U$$

其中$(t_s,t_e,d)\in U$ 表示第 t_s 天到第 t_e 天可以完成 d 次转移。

由此我们可以看出尽可能快地到达目的地等价于消耗量最小策略，因此得到指导策略：在非沙暴天气绝不停留。

4.1.4 对第一关和第二关的具体分析

对第一关进行如下分析：

(1) 如 4.1.1 节的图 2 所示，简化地图，去除多余边。

(2) 利用动态规划求出不考虑矿山挣钱，直接前往终点的最优路径，即 1→25→26→27，3 天连续走，需要在起点处购买 590 元的资源（42 箱水，38 箱食物，如图 6 所示）。

(3) 现考虑经过矿山，去矿山需要 8 次转移，回矿山需要 5 次转移，则至少第 9 天才可以在矿山开始挖矿，总共剩下最多 17 天挖矿，来回村庄需要 4 天，且起点→矿山和矿山→终点途中均会经过村庄，于是我们分两种情况讨论：在矿山挖矿的时间段为 1 段或 2 段。

(4) 考虑去矿山的两轮挖矿，即在矿山挖矿一段时间后，资源耗尽之后去村庄进行补给，再挖矿一段时间后返回终点，该方案最终资金数为 9800 元。

日期	天气	行动	所在区域	耗水量(kg)	耗食量(kg)	余水量(kg)	余食量(kg)	剩余资金数	背包重量(kg)
0			1			0+540=540	0+660=660	10000-4200=5800	0+1200=1200
1	高温	行走	25	-48	-24	492	636	5800	1128
2	高温	行走	24	-48	-24	444	612	5800	1056
3	晴朗	行走	23	-30	-28	414	584	5800	998
4	沙暴	停留	23	-30	-20	384	564	5800	948
5	晴朗	行走	22	-30	-28	354	536	5800	890
6	高温	行走	9	-48	-24	306	512	5800	818
7	沙暴	停留	9	-30	-20	276	492	5800	768
8	晴朗	行走	15村庄	-30	-28	246+489=735	464	5800-1630=4170	710+489=1199
9	高温	行走	14	-48	-24	687	440	4170	1127
10	高温	行走	12矿山	-48	-24	639	416	4170	1055
11	沙暴	挖矿	12矿山	-90	-60	549	356	4170+1000=5170	905
12	高温	挖矿	12矿山	-72	-36	477	320	5170+1000=6170	797
13	晴朗	挖矿	12矿山	-45	-42	432	278	6170+1000=7170	710
14	高温	挖矿	12矿山	-72	-36	360	242	7170+1000=8170	602
15	高温	挖矿	12矿山	-72	-36	288	206	8170+1000=9170	494
16	高温	挖矿	12矿山	-72	-36	216	170	9170+1000=10170	386
17	沙暴	挖矿	12矿山	-90	-60	126	110	10170+1000=11170	236
18	沙暴	停留	12矿山	-30	-20	96	90	11170	186
19	高温	行走	14	-48	-24	48	66	11170	114
20	高温	行走	15村庄	-48	-24	0+471=471	42+280=322	11170-4370=6800	42+751=793
21	晴朗	行走	14	-30	-28	441	294	6800	735
22	晴朗	行走	12矿山	-30	-28	411	266	6800	677
23	高温	挖矿	12矿山	-72	-36	339	230	6800+1000=7800	569
24	晴朗	挖矿	12矿山	-45	-42	294	188	7800+1000=8800	482
25	沙暴	挖矿	12矿山	-90	-60	204	128	8800+1000=9800	332
26	高温	行走	14	-48	-24	156	104	9800	260
27	晴朗	行走	15村庄	-30	-28	126	76	9800	202
28	晴朗	行走	9	-30	-28	96	48	9800	144
29	高温	行走	21	-48	-24	48	24	9800	72
30	高温	行走	27终点	-48	-24	0	0	9800	0

(5) 考虑去矿山的一轮挖矿，之后前往终点。

在(4)(5)步骤中，除了在非矿山地区沙暴天气只能原地停留外，行动优先级为：挖矿（资源充足时）> 移动 > 停留。

最后得到最佳方案如下（一轮挖矿），最终资金数 10430 元：

日期	天气	行动	所在区域	耗水量(kg)	耗食量(kg)	余水量(kg)	余食量(kg)	剩余资金数	背包重量(kg)
0			1			0+540=540	0+660=660	10000-4200=5800	0+1200=1200
1	高温	行走	25	-48	-24	492	636	5800	1128
2	高温	行走	24	-48	-24	444	612	5800	1056
3	晴朗	行走	23	-30	-28	414	584	5800	998
4	沙暴	停留	23	-30	-20	384	564	5800	948
5	晴朗	行走	22	-30	-28	354	536	5800	890
6	高温	行走	9	-48	-24	306	512	5800	818
7	沙暴	停留	9	-30	-20	276	492	5800	768
8	晴朗	行走	15村庄	-30	-28	246+489=735	464	5800-1630=4170	710+489=1199
9	高温	行走	14	-48	-24	687	440	4170	1127
10	高温	行走	12矿山	-48	-24	639	416	4170	1055
11	沙暴	挖矿	12矿山	-90	-60	549	356	4170+1000=5170	905
12	高温	挖矿	12矿山	-72	-36	477	320	5170+1000=6170	797
13	晴朗	挖矿	12矿山	-45	-42	432	278	6170+1000=7170	710
14	高温	挖矿	12矿山	-72	-36	360	242	7170+1000=8170	602
15	高温	挖矿	12矿山	-72	-36	288	206	8170+1000=9170	494
16	高温	挖矿	12矿山	-72	-36	216	170	9170+1000=10170	386
17	沙暴	挖矿	12矿山	-90	-60	126	110	10170+1000=11170	236
18	沙暴	停留	12矿山	-30	-20	96	90	11170	186
19	高温	行走	14	-48	-24	48	66	11170	114
20	高温	行走	15村庄	-48	-24	0+108=108	42+38=80	11170-740=10430	42+146=188
21	晴朗	行走	9	-30	-28	78	52	10430	130
22	晴朗	行走	21	-30	-28	48	24	10430	72
23	高温	行走	27终点	-48	-24	0	0	10430	0

图 7 第一关的最佳方案

对第二关进行如下分析：

(1) 如 4.1.1 节的图 5 所示，简化地图，去除多余边。

(2) 利用动态规划求出不考虑矿山挣钱，直接前往终点的最优路径，即起点→村庄 62→终点，总共需要 11 次转移，连续走直到沙暴才停下，需要在起点处购买 2610 元的资源（如图 6 所示）。

(3) 本题中有 2 个矿山和 2 个村庄，且起点到矿山的最短路径不经过村庄，但矿山与村庄的往返只需要 2 天，因此我们分两种情况讨论。

(4) 考虑直接去矿山 55 挖矿（资源耗尽之后去村庄 62 补给），再去终点，该方案的资金数为 11115 元。

日期	天气	行动	所在区域	耗水量(kg)	耗食量(kg)	余水量(kg)	余食量(kg)	剩余资金数	背包重量(kg)
0			1			0+633=633	0+566=566	10000-3885=6115	0+1199=1199
1	高温	行走	9	-48	-24	585	542	6115	1127
2	高温	行走	18	-48	-24	537	518	6115	1055
3	晴朗	行走	19	-30	-28	507	490	6115	997
4	沙暴	停留	19	-30	-20	477	470	6115	947
5	晴朗	行走	27	-30	-28	447	442	6115	889
6	高温	行走	36	-48	-24	399	418	6115	817
7	沙暴	停留	36	-30	-20	369	398	6115	767
8	晴朗	行走	37	-30	-28	339	370	6115	709
9	高温	行走	45	-48	-24	291	346	6115	637
10	高温	行走	54	-48	-24	243	322	6115	565
11	沙暴	停留	54	-30	-20	213	302	6115	515
12	高温	行走	55矿山	-48	-24	165	278	6115	443
13	晴朗	挖矿	55矿山	-45	-42	120	236	6115+1000=7115	356
14	高温	挖矿	矿山62	-72	-36	48	200	7115+1000=8115	248
15	高温	行走	62村庄	-48	-24	0+717=717	176+286=462	8115-5250=2865	176+1003=1179
16	高温	行走	62村庄	-48	-24	669	438	2865	1107
17	沙暴	挖矿	55矿山	-90	-60	579	378	2865+1000=3865	957
18	沙暴	挖矿	55矿山	-90	-60	489	318	3865+1000=4865	807
19	高温	挖矿	55矿山	-72	-36	417	282	4865+1000=5865	699
20	高温	挖矿	55矿山	-72	-36	345	246	5865+1000=6865	591
21	晴朗	挖矿	55矿山	-45	-42	300	204	6865+1000=7865	504
22	晴朗	挖矿	55矿山	-45	-42	255	162	7865+1000=8865	417
23	高温	挖矿	55矿山	-72	-36	183	126	8865+1000=9865	309
24	晴朗	挖矿	55矿山	-45	-42	138	84	9865+1000=10865	222
25	沙暴	挖矿	55矿山	-90	-60	48	24	10865+1000=11865	72
26	高温	行走	62村庄	-48	-24	0+171=171	0+118=118	11865-1750=10115	0+289=289
27	晴朗	行走	55矿山	-30	-28	141	90	10115	231
28	晴朗	挖矿	55矿山	-45	-42	96	48	10115+1000=11115	144
29	高温	行走	56	-48	-24	48	24	11115	72
30	高温	行走	64终点	-48	-24	0	0	11115	0

(5) 考虑先后去矿山 30、矿山 55 挖矿，中间经过村庄 39，可以去村庄 62 补给，再去终点，该方案的资金数为 12355。

日期	天气	行动	所在区域	耗水量(kg)	耗食量(kg)	余水量(kg)	余食量(kg)	剩余资金数	背包重量(kg)
0						0+741=741	0+458=458	10000-3525=6475	0+1199=1199
1	高温	行走	9	-48	-24	693	434	6475	1127
2	高温	行走	10	-48	-24	645	410	6475	1055
3	晴朗	行走	19	-30	-28	615	382	6475	997
4	沙暴	停留	19	-30	-20	585	362	6475	947
5	晴朗	行走	20	-30	-28	555	334	6475	889
6	高温	行走	28	-48	-24	507	310	6475	817
7	沙暴	停留	28	-30	-20	477	290	6475	767
8	晴朗	行走	29	-30	-28	447	262	6475	709
9	高温	行走	30矿山	-48	-24	399	238	6475	637
10	高温	挖矿	30矿山	-72	-36	327	202	6475+1000=7475	529
11	沙暴	挖矿	30矿山	-90	-60	237	142	7475+1000=8475	379
12	高温	挖矿	30矿山	-72	-36	165	106	8475+1000=9475	271
13	晴朗	挖矿	30矿山	-45	-42	120	64	9475+1000=10475	184
14	高温	挖矿	30矿山	-72	-36	48	28	10475+1000=11475	76
15	高温	行走	39村庄	-48	-24	0+720=720	4+472=476	11475-7120=4355	4+1192=1196
16	高温	行走	47	-48	-24	672	452	4355	1124
17	沙暴	停留	47	-30	-20	642	432	4355	1074
18	沙暴	停留	47	-30	-20	612	412	4355	1024
19	高温	行走	55矿山	-48	-24	564	388	4355	952
20	高温	挖矿	55矿山	-72	-36	492	352	4355+1000=5355	844
21	晴朗	挖矿	55矿山	-45	-42	447	310	5355+1000=6355	757
22	晴朗	挖矿	55矿山	-45	-42	402	268	6355+1000=7355	670
23	高温	挖矿	55矿山	-72	-36	330	232	7355+1000=8355	562
24	晴朗	挖矿	55矿山	-45	-42	285	190	8355+1000=9355	475
25	沙暴	挖矿	55矿山	-90	-60	195	130	9355+1000=10355	325
26	高温	挖矿	55矿山	-72	-36	123	94	10355+1000=11355	217
27	晴朗	挖矿	55矿山	-45	-42	78	52	11355+1000=12355	130
28	晴朗	行走	56	-30	-28	48	24	12355	72
29	高温	行走	64终点	-48	-24	0	0	12355	0

在(4)(5)步骤时，除了非矿山地区沙暴天气只能原地停留外，行动优先级为：挖矿（资源充足时）> 移动 > 停留。

最后改进后，得到最佳方案如下：先去 62 村庄，之后在 55 矿山挖矿，中途回 62 村庄补给一次，最后直接从 55 矿山去终点。（起点—62 村—55 矿—62 村—55 矿—64 终），最终资金数为 12470 元：

日期	天气	行动	所在区域	耗水量(kg)	耗食量(kg)	余水量(kg)	余食量(kg)	剩余资金数	背包重量(kg)
0			1			0+468=468	0+732=732	10000-4440=5560	0+1200=1200
1	高温	行走	9	-48	-24	420	708	5560	1128
2	高温	行走	18	-48	-24	372	684	5560	1056
3	晴朗	行走	26	-30	-28	342	656	5560	998
4	沙暴	停留	26	-30	-20	312	636	5560	948
5	晴朗	行走	35	-30	-28	282	608	5560	890
6	高温	行走	36	-48	-24	234	584	5560	818
7	沙暴	停留	36	-30	-20	204	564	5560	768
8	晴朗	行走	44	-30	-28	174	536	5560	710
9	高温	行走	55	-48	-24	126	512	5560	638
10	高温	行走	61	-48	-24	78	488	5560	566
11	沙暴	停留	61	-30	-20	48	468	5560	516
12	高温	行走	62村庄	-48	-24	0+522=522	444	5560-1740=3820	444+522=966
13	晴朗	行走	55矿山	-30	-28	492	416	3820	908
14	高温	挖矿	55矿山	-72	-36	420	380	3820+1000=4820	800
15	高温	挖矿	55矿山	-72	-36	348	344	4820+1000=5820	692
16	高温	挖矿	55矿山	-72	-36	276	308	5820+1000=6820	584
17	沙暴	挖矿	55矿山	-90	-60	186	248	6820+1000=7820	434
18	沙暴	挖矿	55矿山	-90	-60	96	188	7820+1000=8820	284
19	高温	行走	62村庄	-48	-24	48+555=603	164+250=414	8820-4350=4470	212+805=1017
20	高温	行走	55矿山	-48	-24	555	390	4470	945
21	晴朗	挖矿	55矿山	-45	-42	510	348	4470+1000=5470	858
22	晴朗	挖矿	55矿山	-45	-42	465	306	5470+1000=6470	771
23	高温	挖矿	55矿山	-72	-36	393	270	6470+1000=7470	663
24	晴朗	挖矿	55矿山	-45	-42	348	228	7470+1000=8470	576
25	沙暴	挖矿	55矿山	-90	-60	258	168	8470+1000=9470	426
26	高温	挖矿	55矿山	-72	-36	186	132	9470+1000=10470	318
27	晴朗	挖矿	55矿山	-45	-42	141	90	10470+1000=11470	231
28	晴朗	挖矿	55矿山	-45	-42	96	48	11470+1000=12470	144
29	高温	行走	63	-48	-24	48	24	12470	72
30	高温	行走	64终点	-48	-24	0	0	12470	0

图 8 第二关的最佳方案

4.2 问题二

问题二中，玩家仅知道当天的天气状况，可据此决定当天的行动方案，因此需要解决以下两个策略的构建：

(1) 初始状态时路线和资源的购买方案；

路线规划分为两种：直接前往终点模式和挖矿赚钱模式。

对于直接前往终点模式，玩家在起点处备满路径中需要的物资，尽量减少购买物资的消费，获取最佳策略；对于挖矿赚钱模式，玩家有无法到达终点的风险，这里需设置游戏失败的额外损失，它与玩家的性格有关，需预先设定（玩家越谨慎则额外损失越高，贴近天气最差情况；玩家越冒险则额外损失越低，贴近天气最佳情况），然后计算配备的物资。

(2) 路途中基于天气概率、所处位置和所剩物资的行动方案（4.2.2 节和 4.2.3

节详述）。

4.2.1 最初路线规划一致

本题中我们约定：一旦规定路线规划模式中的其中一种，玩家中途不会更改。

(1) 若玩家采用挖矿赚钱模式，玩家会在开始备好充足的资源以供长期挖矿，使收益更高，中途因恶劣天气改变策略直接前往终点是不经济的；

(2) 若玩家采用直接前往终点模式，玩家在开始时段即备好支撑其前往终点的资源，这些资源通常不足以半路支持挖矿的，故玩家不会中途转而去挖矿。

保证决策依然遵循最短路径且不能中途改道后，问题二中的两个地图也可如4.1.1 节一样去除多余边而简化了，如图 9 和图 10 所示。

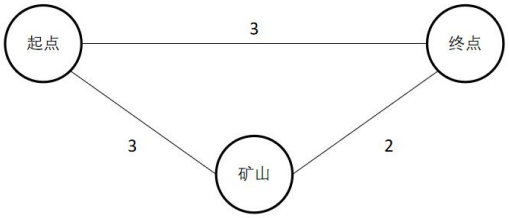

图 9 第三关的地图简化图

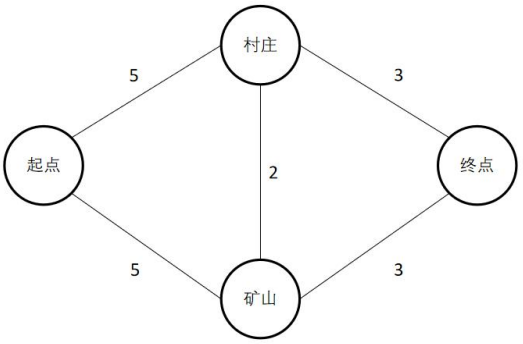

图 10 第四关的地图简化图

由于在沙漠地图、资源消耗基础量、背包容量等多变，很难在一般情况下讨论所有可能，因此，我们将在第三关、第四关的具体情况之下人工决定两种路线规划（挖矿赚钱模式或直接前往终点模式）。在路线规划给定的情况之下，给予一般性的讨论（4.2.4 节详述）。

4.2.2 直接前往终点模式

4.2.2.1 基于概率的直达终点最短花费决策模型

玩家根据以往的天气可以估算出三种天气分布的概率：设已出现晴朗天气 N_i 天，高温天气 N_j 天，沙暴天气 N_k 天，故三种天气出现概率估算为

$$p_i = \frac{N_i + 1}{N_i + N_j + N_k + 3},\quad p_j = \frac{N_j + 1}{N_i + N_j + N_k + 3},\quad p_k = \frac{N_k + 1}{N_i + N_j + N_k + 3}.$$

下面将根据概率做直接前往终点的代价估计，使得最终路线策略花费最少。沿用 4.1.2 节的晴朗天气不停留策略，我们需要考虑另外两种天气的情况：高温下可停留或行走，风暴下只能停留。只有高温的决策是不确定的，因此我们需要计算出何时在高温下停留或行走，让到达终点的花费最少。

在 t 天中，根据数学期望，晴朗天气有 $p_i * t$ 天，高温天气有 $p_j * t$ 天，沙暴

天气有 $p_k * t$ 天，现设高温下停留、高温下移动的概率分别为 $p_j * p_{j0}$，$p_j * p_{j1}$，那么在 t 天内走过的路程（移动的步数）的数学期望为：

$$d = t * \left(p_i + p_j p_{j1}\right) \qquad (*)$$

其中概率满足 $p_{j0} + p_{j1} = 1$。

走 d 步花费的数学期望如下：

$E(d) = t * (2p_i\, cost(\text{晴朗}) + p_j p_{j0}\, cost(\text{高温}) + 2p_i p_{j1}\, cost(\text{高温}) + p_k\, cost(\text{沙暴}))$，其中 t 由(*)式给定。

在已知当前气温为高温，后面还有 d 步要走情况下，停留或行走两种情况分别对应花费的数学期望如下：

$$C_0 = cost(\text{高温}) + E(d)$$

$$C_1 = 2 * cost(\text{高温}) + E(d-1)$$

其中 C_0, C_1 分别代表高温下停留时和行走时的数学期望。

现对 C_0, C_1 进行 p_{j0} 和 p_{j1} 的加权，即 $C = p_{j0}C_0 + p_{j1}C_1$，在 C 的函数表达式中，p_{j0} 和 p_{j1} 是自变量且满足 $p_{j0} + p_{j1} = 1$，将 p_{j0} 用 p_{j1} 表示，并让 C 对 p_{j1} 求导：

$$C = (1 + p_{j1})cost(\text{高温}) + (d - p_{j1})\frac{2p_i cost(\text{晴朗}) + (1 + p_{j1})p_j cost(\text{高温}) + p_k cost(\text{高温})}{p_i + p_j p_{j1}}$$

$$\frac{dC}{dp_{j1}} = -\frac{(p_i + d\,p_j)\,(2\,cost(\text{晴朗})\,p_i - cost(\text{高温})\,p_i + cost(\text{高温})\,p_j + cost(\text{沙暴})\,p_k)}{(p_i + p_j\,p_{j1})^2}$$

可见 p_{j1} 无法决定 C 的正负号，只有该式子 $2p_i\, cost(\text{晴朗}) - p_i\, cost(\text{高温}) + p_j\, cost(\text{高温}) + p_k\, cost(\text{沙暴})$ 来决定 C 随 p_{j1} 单增或单减：

$$\frac{dC}{dp_{j1}} < 0 \Leftrightarrow 2\,cost(\text{晴朗})\,p_i - cost(\text{高温})\,p_i + cost(\text{高温})\,p_j + cost(\text{沙暴})\,p_k > 0$$

若 C 随 p_{j1} 单减，则要使 C 更小，p_{j1} 趋于更大，此时作出决策为高温下行走；

$$\frac{dC}{dp_{j1}} > 0 \Leftrightarrow 2\,cost(\text{晴朗})\,p_i - cost(\text{高温})\,p_i + cost(\text{高温})\,p_j + cost(\text{沙暴})\,p_k < 0$$

若 C 随 p_{j1} 单增，则要使 C 更小，p_{j1} 趋于更小，此时作出决策为高温下停留。

4.2.2.2 合理性分析

在多项式中出现了资源消耗和天气概率，这两项决定了高温下停留或行走：

(1) $cost$(高温)越大，高温下移动消耗就越大，所以选择高温下不移动的可能越大，这与模型中上述导数趋于正相符；

(2) 晴天可能 p_i 越大，玩家根据已出现的天气得到遇到晴天的概率较高，那么当前高温下移动就会亏损过多（相对于晴天移动），因此选择高温下不移动的可能更大，对应导数为正，模型与推理相符。

4.2.2.3 玩家决策倾向模型

在本题中，花费有多种类别（取决于玩家主观考虑）：食物耗费量、水耗费量等。因此我们得出基于多种花费类别的决策概率。在每一步的决策之中，我们都要考虑最多金钱剩余量和顺利到达终点。

但是不同的玩家有不同的考量方式，冒险的玩家会更倾向于最多的金钱剩余量，而保守的玩家会倾向于尽可能稳妥的到达终点；当前物资的组合可能会出现某物资先用完的情况，因此我们需要特别注意某种物资的使用等等。总而言之，多种类别的决策模型为加权平均，而权重会基于玩家对于当前状态和自身决策倾向进行考量。

$$N = \Sigma_{i=0}^{n} \alpha_i n_i \quad , \Sigma_{i=0}^{n} \alpha_i = 1$$

其中 α_i 表示考虑各因素的权重，n_i 表示表示各种因素下的最优策略的概率，在综合考虑之后，玩家将依据加权结果进行相应的决策：

$$action(weather = hot) = \begin{cases} move & , N \geq 0.5 \\ stay & , N < 0.5 \end{cases}$$

4.2.2.4 对第三关的具体分析

首先进行预决策：选择直接前往终点模式。理由如下：

本关具有如下特点：

1. 地图中没有村庄，因此所有的资源都必须在起点处购买，故可以直接按 1x 基准价格估计消耗资源对应的价值。

2. 玩家在途中不会遇到沙暴天气。

3. 本关的矿山挖矿收益（200）相对较少。

本关的关键点较少，除了起点外，只有 1 个矿山和 1 个终点。可以将行动方案按照是否前往矿山挖矿进行分类。若不前往矿山挖矿，直接向终点迈进即可。

	关键点访问规划	说明
方案 1	起点—终点	至少须安排 3 天行走。
方案 2	起点—矿山—终点	行程开头和结尾须至少分别安排 3 天和 2 天的行走，（在满足资源约束的条件下）余下最多 5 天可安排为挖矿。

本关挖矿收益较少，需要判断前往矿山挖矿是否有利可图，从而决定方案。每天采取的行动会消耗资源，但可能带来收入（如挖矿），因此用收入 - 消耗资源对应的价值来衡量当天行动的收益。

表 3：天气与行动组合收益表

天气与行动组合	消耗资源对应价值, 1x 基准价格	收入	收益
晴朗行走	-110		-110
晴朗挖矿	-165	+200	+35
高温停留	-135		-135
高温行走	-270		-270
高温挖矿	-405	+200	-205

注：上表略去了“晴朗停留”“沙暴停留”等无意义或不可能的组合。

已知不会出现沙暴天气，假设晴天和高温的出现频率为 1:1，则可以得到每天采取行走和挖矿行动的收益期望：

表 4：晴天和高温 1:1 下每天行动收益期望

行动	收益期望
行走	-190/天
停留	-135/天
挖矿	-85/天
仅晴朗天挖矿	-50/天

由上表，在“晴天和高温的出现频率为 1:1”的前提假设下，在矿山挖矿的收益期望为负值，意味着挖矿的收入都无法填补挖矿行为自身的资源消耗期望支出，更何况前往矿山还需要付出更多行走的天数和支出。

使用该收益期望来预估两种方案；显然，“直接前往终点”（方案 1）的收益期望 ＞ “前往矿山挖矿”（方案 2）的收益期望。

步骤如下：

(1) 预决策：判断得到采取直接前往终点模式，并依据玩家的冒险程度确定购买的物资；

	1	2	3	4	5	6	7	8	9	10
	晴朗	高温	晴朗	晴朗	晴朗	晴朗	高温	高温	高温	高温
晴朗	1	1	2	3	4	5	5	5	5	5
高温	0	1	1	1	1	1	2	3	4	5
沙暴	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
晴朗	0.666667	0.5	0.6	0.666667	0.714286	0.75	0.666667	0.6	0.545455	0.5
高温	0.333333	0.5	0.4	0.333333	0.285714	0.25	0.333333	0.4	0.454545	0.5
沙暴		0	0	0	0	0	0	0	0	0
决策	前进	停留	前进	前进	前进	前进	停留	停留	停留	停留
	金钱消耗									
ci	55									
cj	135									
ck	150									

(2) 根据“基于概率的直达终点最短花费决策模型”进行具体决策：

以第五关的具体天气为例，最终的决策为：前进、停留、前进、前进，与已知天气的最优解相同。初始时刻，防止资源不足考虑最差天气（全高温）的消耗：810 元、162kg 水、108kg 食物。路途消耗：水 81kg、食物 66kg，返还剩余物资获得 172.5 元，总共剩余 9362.5 元。

4.2.3 挖矿赚钱模式

4.2.3.1 基于逃离阈值的挖矿停止策略

这种模型主要考虑挖矿过程中选择停止挖矿而前往终点的时机。天气概率沿用 4.2.2.1 节的 p_i, p_j, p_k。挖矿赚钱模式中，玩家将尽可能多的进行行动，而非停留。基于此，我们将采用 4.1.3 节的结论“在非沙暴天气绝不停留”，进行行动策略分析。

该模型以前往终点失败率与挖矿收益为参考量，做出前往终点或继续挖矿决策。玩家在挖矿模式下，对于前往终点的考量是基于有限时间下的逃离，挖矿会得到更多的金钱但是会损失更多的时间，使得逃离失败率增加。对于玩家而言，需要平衡两者，同样的，不同性格的玩家会给出不同的决策，因此我们将设定逃离失败率阈值，超过时玩家必须要前往补货或是终点。

该模型在总体上多挖矿的理念上，给予一个前往终点的时机以及村庄补货的时机。

由于该模型对应挖矿模式，玩家必定会前往矿山，则有两种方式：直接前往矿山、先去村庄补充物资再前往矿山。选择何种方式前往矿山在最初规划路线时已决定，此处只研究当玩家身处矿山时采取的策略。

这里约定执行 4.1.2 节所述最多挖矿原则，即玩家会保持挖矿策略直到因为时间原因需要前往终点。

我们将从矿山到终点的策略分为两种进行讨论：直接前往终点、通过村庄再前往终点。那么整个前往终点的路线图如下：

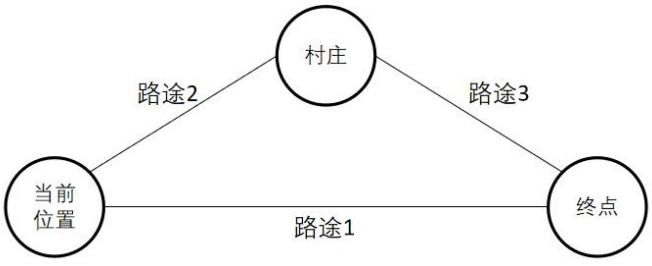

图 11 前往终点的路线图示意

对于路途 3，因为在选择挖矿赚钱模式时已做预判，且经过村庄可进行补货。我们做出假设：路途 3 不会因物资不足而导致游戏失败。

对于路途 1 与 2，我们需要考虑游戏结束的可能性：在恶劣天气之下，我们

背包中的资源无法支撑到达村庄。由于采取前往终点对应的时间紧迫，玩家会在非沙暴天气绝不停留。

在这样的策略之下，路径 k ($k = 1, 2$)上的沙暴天数 t_k 成了唯一的不确定因素，假设消耗量上限为 G（超过会导致失败），剩余天数 T，我们计算沙暴天数的阈值如下，第一式表示因资源不够而无法到达，第二式表示因所用时间超过剩余天数而无法到达。

$$t_k cost(\text{沙暴}) + 2d_k(cost(\text{晴朗}) + cost(\text{高温})) \geq G, k = 1, 2$$

$$t_k + d_k \geq T$$

其中 d_k 表示路径 k 上所需转移次数。

对于满足任意一式的最小值 t_{kn}，我们对其概率做估计。由于该种天气出现的概率满足二项分布，因此我们可以计算消耗量超过上限的情况的预估概率：

$$p_n = \Sigma_{t=t_{kn}}^{+\infty} C_{t+d_k}^{t} p_k^t (1 - p_k)^{d_k}$$

更多地，我们可以对此推广成考虑去往任意地点的决策之下，在路途中资源耗尽的概率。具体来说，得到这个概率，我们可以计算出前往终点、前往矿山、前往村庄等等的最晚行动时间，来保证玩家不会因为资源不够而游戏失败。

4.2.3.2 挖矿的撤离决策

类似 4.2.2.3 节所述，不同的玩家有不同的考量方式，但是在尽可能多挖矿的情形之下，为了避免游戏失败，在固定的资源基础消耗量和挖矿收益下玩家自身都会存在一定的逃离失败阈值 p_r。

我们通过以上的求解方法，可以看到在挖矿中途可以 p_r 决定什么时候应前往补货，在时间较少准备撤离前往终点的时候决定什么时候去村庄补货或者是前往终点等等。总之，应用此概率，我们可以在其指导之下做出撤离的指导。

4.2.3.3 对第四关的具体分析

首先进行预决策：选择前往矿山挖矿模式。理由如下：

本关具有如下特点：

(1) 资源参数和第三关一致，但矿山挖矿收益为 1000 元/天；且截止日期为第 30 天，时间相对充裕。

(2) 地图且有村庄且位于矿山附近，可以中途购买资源补给。

本关也可以将行动方案按照是否前往矿山挖矿进行分类。若不前往矿山挖矿，则直接向终点迈进即可。本关挖矿收益较少，需要判断前往矿山挖矿是否有利可图，从而决定选择的方案。

	关键点访问规划	说明
方案 1	起点—终点	至少须安排 5 + 3 天行走。

方案 2	起点—(矿山≒村庄)—终点	行程开头和结尾须至少分别安排 5 天和 3 天的行走，（在满足资源约束的条件下）余下最多 22 天可安排为在村庄和矿山之间往返以及挖矿。

类似第三关中的方法，可以得到：

表 5: 天气与行动组合收益表

天气与行动组合	资源消耗量 (kg)	消耗资源对应价值		收入	收益 (1x)	收益 (2x)
		1x 基准价格	2x 基准价格			
晴朗行走	34	-110	-220		-110	-220
晴朗挖矿	51	-165	-330	1000	835	670
高温停留	45	-135	-270		-135	-270
高温行走	90	-270	-540		-270	-540
高温挖矿	135	5	-810	1000	595	190
沙暴停留	50	-150	-300		-150	-300
沙暴挖矿	150	50	-900	1000	550	100

注：上表略去了“晴朗停留”等无意义或不可能的组合。

已知较少出现沙暴天气，假设晴天:高温:沙暴的出现频率为 2:2:1，则可以得到每天采取行动的收益期望：

表 6：晴天:高温:沙暴为 2:2:1 下每天行动收益期望

行动	资源消耗量期望 (kg)	收益期望 (1x)	收益期望 (2x)
行走	62	-190/天	-380/天
停留	46.67	-140/天	-280/天
挖矿	104.4	+682/天	+364/天
挖矿(但沙暴天停留)	84.4	+542/天	+284/天

由上表，在“晴天:高温:沙暴的出现频率为 2:2:1”的前提假设下，无论是按 1x 还是 2x 基础价格折算消耗资源支出，在矿山挖矿的收益期望均为正值，并且根据 1x 和 2x 的差距，可以分析出起点所购资源占比多的情况下，挖矿的收益期望高。

村庄和矿山之间的最短距离为 2，这决定了“从村庄购买资源，到矿山挖矿，之后再回到村庄”这样的行程中至少包含 2 + 2 天的行走。

- 行走部分的收益期望 -380 × 2 = -760 元，至少需要挖矿两次才能使整体收益为正。
- 行走部分花费 62 × 2 × 2 = 248 kg 资源；假设理想状态下，从村庄出来最多可满载 1200 kg 资源，经过跋涉后还剩余 952 kg，一般情况下足以支撑 ≥ 2 次挖矿。

因此，本关的规划方案为前往矿山挖矿，因为在沙暴较少的以及起点所购资源占比多的情况下，前往矿山挖矿有较大可能获得正收益。

4.3 问题三

4.3.1 混合 Nash 均衡模型

首先，对于全对称的 n 名玩家，都可以采取 m 种决策$D_i(i=1,2,\ldots,m)$，具体对应总决策设为$\Sigma(f_1,f_2,\ldots,f_n)$，其中$f_i$代表第 i 名玩家的决策。同时，我们设对于第 i 名玩家，该总决策对其收益为$C_i(f_1,f_2,\ldots,f_n)(i=1,2,\ldots,n)$，则在对 Nash 均衡情况之下，有以下均衡式：

$$C_i(f_1,f_2,\ldots,\bar{f}_i,\ldots,f_n)=max_{f_i=D_1}^{D_m}C_i(f_1,f_2,\ldots,f_i,\ldots,f_n),i=1,2,\ldots,n$$

但是由于实际上 Nash 均衡不一定可以得到满足，因此我们引入第 $i(i=1,2,\ldots,n)$ 位参与者采取策略$D_j(j=1,2,\ldots,m)$的概率为p_{ij}，在这样的混合 Nash 均衡模型之下，我们需要找到在最小收益的时候达到这位玩家决策策略的最佳组合，选择合适的$\boldsymbol{p_i}$使得有以下混合均衡式成立：

$$\max_{\boldsymbol{p_i}} min\ \Sigma_{j=1}^{m}p_{ij}C_i(f_1,f_2,\ldots,f_i,\ldots,f_n)\left(f_i=D_j\right),i=1,2,\ldots,n$$

4.3.2 基于最短路径决策的无向图简化

实际上，在此游戏情形中，我们由于决策（不同行动）的可能性非常多，因此我们需要进行复杂度降低，选取最有可能的情况。第一题的单人博弈之中，最优决策一定由最短路径组成。针对第（1）问的多人一次博弈的情形之下，我们同样也只考虑所有最短路径的情况，以简化无向图和决策。

4.3.3 自私人贪心策略&第五关具体分析

在无其它可行方案下，我们设定玩家一定会考虑最贪心的策略：忽略其他人可能带来的威胁，现对第五关进行具体分析：

与第三关同理，玩家不会选择前往矿山挖矿，都会选择直接前往终点，根据**“基于最短路径决策的无向图简化”**，我们得到以下有向图，最短路径有两条。

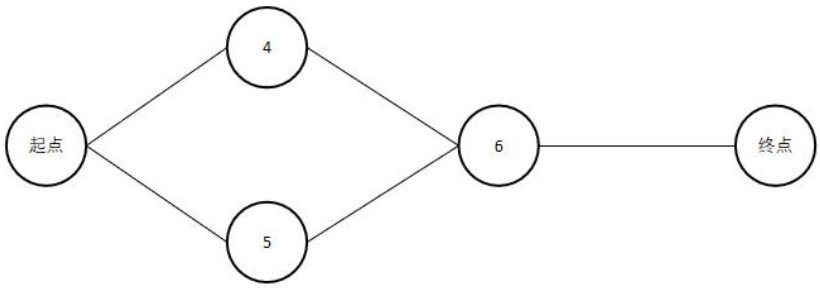

图 12 第五关无向图简化

在这样的无向图之下，我们进行两人博弈。显然，两条路实际上为完全相同的两条路。因此，在 Nash 均衡公式约束之下，我们最终得到的一定为两条相等的规划路径和物资购买策略。而在混合 Nash 均衡策略之下，可以得到两条路径选择的策略概率均为 50%。因此一般情况之下来说玩家任意走一条路即可。

在单人情况下，最优解为：行走、停留、行走、行走。花费金额 465，水 81kg、食物 66kg。对于双人博弈具体的策略，为了保证能够顺利到达终点（物资充足），玩家将会选择一直出现两人同行，即消耗量翻倍。一般解为：行走、停留、行走、行走。花费金额 930，水 162kg、食物 132kg。

4.3.4 多人挖矿博弈策略&第六关具体分析

与第四关同理，玩家选择前往矿山挖矿，由于此题最短路径非常多，因此前期玩家很难一直处于同一位置，但是在挖矿的时候，会出现同时挖矿的情况，我们主要对挖矿时候玩家的策略进行具体分析。

当多人堆积在一个矿山之时，玩家需要做出挖矿与否的决定。直观上来看，因此会出现博弈。该模型主要对此进行讨论，以找到多人挖矿的博弈选择。

基于资源相似度的考虑法则：如果处于矿山的玩家的资源量都相似，那么同时前往村庄补货的可能性大，会导致潜在的购买物资金钱消耗增大，并且会更有可能因为物资过度在前往村庄的路上游戏失败。对于同一物资区间，我们定义当资源支撑的挖矿天数大致相同的时候，玩家的资源相似度高，可以表述如下，其中G_0,G_1分别表示两个玩家现有的物资，cost 在这里表示物资花费。

$$3\left(p_i cost\left(晴朗\right)+p_j cost\left(高温\right)+p_k cost\left(沙暴\right)\right)>|G_0-G_1|$$

对应提早补货策略：当玩家发现与其它玩家的资源大致相同时，将补货日程提前一天，用“逃离阈值的挖矿停止策略”来判断离开时间。

表 7：1x 倍率购买资源对应挖矿收入

挖矿	晴朗	高温	沙暴
1 人	835	595	550
2 人	335	95	50
3 人	168	-72	-117

表 8：2x 倍率购买资源对应挖矿收入

挖矿	晴朗	高温	沙暴
1 人	670	190	100
2 人	170	-310	-400
3 人	3	-477	-567

表 9：4x 倍率购买资源对应挖矿收入

挖矿	晴朗	高温	沙暴
1 人	340	-620	-800
2 人	-160	-1120	-1300
3 人	-327	-1287	-1467

由此可以看到，初始购买资源来说，倍率为 1x，应该尽可能避开 3 人同时挖矿，而使用村庄补货的价格来进行挖矿，我们更可能在晴朗天气下进行挖矿，而其他时候倾向于单人挖矿。但是在 4x 倍率购买资源的时候，我们可以几乎毫不犹豫的亏损。这表明玩家在村庄与其它玩家相遇的时候，尽量不买物品，同时在其他位置的时候也应该尽可能地避免同时前往村庄。

五、模型评价与改进

5.1 模型的优点

本文模型方法多样，涵盖有等效简化、降维简化、综合指导策略、定量决策判定等思想，同时原理直观、最终形式直接，符合一般推测。

5.2 模型的缺点与改进方向

只是给出通用的规划，需要根据不同地图单独讨论。

参考文献

[1] 姜启源, 谢金星, 叶俊. 数学模型 (第四版) [M]. 高等教育出版社, 2011.
[2] 【算法复习】动态规划 https://www.cnblogs.com/hithongming/p/9229871.html
[3] 【动态规划】01 背包问题 https://www.cnblogs.com/mfrank/p/10533701.html
[4] 吴广谋, 吕周洋. 博弈论基础与应用[M]. 东南大学出版社, 2009.

附录

附录 1：4.1.3 节动态规划代码 calMinPath.py

```
import numpy as np

MAX = 2 ** 20 - 1

c: np.ndarray
p: np.ndarray


def calMinPath(T, D):  # 从 1 到 T 天，从 1 到 D 点的所有最少消耗路径
    # 第一、二关天气
    # 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]  # 对应天数的不同天气
    # 第五关天气
    weather = [None, 0, 1, 0, 0, 0, 0, 1,1,1,1]

    # 第一、二关花费
    # cost = [95, 100, 150]  # 三种天气的基本花费
    # 第五关花费
    cost = [55,135,150]
    global c, p
    c = np.full((T+1, T+1, D+1), MAX, dtype=np.int32) # 初始化消耗为无穷
    p = np.full_like(c, 'NoPath', dtype='object')

    for ts in range(T):
        # c[ts-1, ts-1, 1] = 0
        c[ts, ts, 0] = 0
        for te in range(ts + 1, T + 1):
            for d in range(D+1):
                if d == 0: # 必须停留
                    c[ts,te,d] = c[ts,te-1,d] + cost[weather[te]]

                    p[ts,te,d] = "stay"
                elif weather[te] == 2:  # 沙暴
                    c[ts, te, d] = c[ts, te - 1, d] + cost[weather[te]]
                    p[ts, te, d] = "stay"
                elif c[ts,te-1,d-1] + 2*cost[weather[te]] > c[ts,te-1,d] + cost[weather[te]]: # 原地呆
```

```
                    c[ts, te, d] = c[ts, te - 1, d] + cost[weather[te]]
                    p[ts, te, d] = "stay"
                else:
                    c[ts, te, d] = c[ts, te - 1, d - 1] + 2 * cost[weather[te]]  # 行走
                    p[ts, te, d] = "walk"


def printMinPath(ts, te, d):  # 打印最晚 T 到达 D 的最少消耗路径
    # 寻找最小消耗路径
    global c, p
    T_ = ts
    minC = c[ts, ts, d]
    for t in range(ts + 1, te+1):
        if minC > c[ts, t, d]:
            minC = c[ts, t, d]
            T_ = t
    print(minC)

    # 打印最小消耗路径
    path = list()
    for t in range(T_, ts, -1):  # 从结束往开始重构最优解
        if p[ts, t, d] == "NoPath":  # 从 d-1 行走至 d
            print("NoPath")
            break
        path.append(d)  # t 时刻在 d 点
        if p[ts, t, d] == "walk":  # 从 d-1 行走至 d
            d = d - 1
    path.append(0)
    # while len(path) > 0:
    #     print(path.pop())
    print("->".join(str(x) for x in path[::-1]))


if __name__ == '__main__':
    # 第一关直接前往终点
    # calMinPath(30, 3)
    # printMinPath(0, 30, 3)
    # 第二关直接前往终点
    # calMinPath(30, 11)
    # printMinPath(0, 30, 11)
    # 第五关直接前往终点
    calMinPath(10, 3)
```

```
79.      printMinPath(0, 10, 3)
```

附录 2：对游戏进行模拟的代码 model.py

```
1.  from enum import Enum
2.
3.
4.  class eWea(Enum):
5.      sandstorm = '沙暴'
6.      sunny = '晴朗'
7.      hot = '高温'
8.
9.  class eAct(Enum):
10.     stay = '停留'
11.     move = '行走'
12.     mine = '挖矿'
13.
14. class eTerrain(Enum):
15.     normal = '普通沙漠'
16.     mineland = '矿山'
17.     village = '村庄'
18.     destination = '终点'
19.
20. class AgainstGameRule(Exception):
21.     pass
22.
23. class CanNotMoveWhenMeetStorm(AgainstGameRule):
24.     pass
25.
26. class bmodel:
27.     def __init__(self):
28.         self.weight_limit = 1200
29.         self.day_limit = 30
30.         self.money = 10000
31.         self.bincome = 1000
32.
33.         self.res_meta = {
34.             '水': {'w': 3, 'p': 5},
35.             '食物': {'w': 2, 'p': 10}
36.         }
37.
38.         self.consum_meta = {
39.             '水': {'晴朗': 5, '高温': 8, '沙暴': 10},
40.             '食物': {'晴朗': 7, '高温': 6, '沙暴': 10}
41.         }
```

```
42.
43.             self.day = 0
44.             self.food_left = 0
45.             self.water_left = 0
46.
47.             self.today_action = None
48.
49.             self.weather = None
50.             self.area_no = None
51.             self.terrain = None
52.             self.mineland_day = None
53.             pass
54.
55.         @property
56.         def weight(self):
57.             return self.food_weight * self.food_left \
58.                    + self.water_weight * self.water_left
59.
60.         @staticmethod
61.         def _res_buniprice(res):
62.             return res['p']
63.
64.         @staticmethod
65.         def _res_weight(res):
66.             return res['w']
67.
68.         @staticmethod
69.         def _res_bconsum(res_c, weather):
70.             return res_c[weather.value]
71.
72.         @property
73.         def water_buniprice(self):
74.             return self._res_buniprice(self.res_meta['水'])
75.
76.         @property
77.         def water_weight(self):
78.             return self._res_weight(self.res_meta['水'])
79.
80.         def water_bconsum(self, weather):
81.                         return   self._res_bconsum(self.consum_meta[' 水
    '], weather)
82.
83.         @property
84.         def food_buniprice(self):
```

```
85.             return self._res_buniprice(self.res_meta['食物'])
86.
87.         @property
88.         def food_weight(self):
89.             return self._res_weight(self.res_meta['食物'])
90.
91.         def food_bconsum(self, weather):
92.                     return  self._res_bconsum(self.consum_meta[' 食 物
    '], weather)
93.
94.
95.         @property
96.         def today_mine_income(self):
97.             """挖矿收益"""
98.             if self.today_action is eAct.mine:
99.                 return self.bincome
100.             return 0
101.
102.         # =============== 结算
103.         def consum_under_wea(self, weather: eWea):
104.             self.water_bconsum(weather)
105.             self.food_bconsum(weather)
106.
107.
108.         def today_income(self):
109.             """今日收益"""
110.             # 挖矿?
111.             return self.today_mine_income
112.
113.         def today_outcome(self):
114.             """今日支出"""
115.             # 买东西?
116.             # water_buy
117.             # food_buy
118.             #
119.             # food_buy * self.today_food_uniprice
120.             # water_buy * self.today_water_uniprice
121.             #
122.             # water_buy * self.water_weight
123.             # food_buy * self.food_weight
124.
125.             # 每天的消耗
126.
127.         @property
```

```
128.     def today_consums_table(self):
129.         return {
130.             '水': [self.today_water_consum,
131.                    self.today_water_consum * self.today_water_unip
    rice,
132.                    self.today_water_consum * self.water_weight
133.                    ],
134.             '食物': [self.today_food_consum,
135.                     self.today_food_consum * self.today_food_unipr
    ice,
136.                     self.today_food_consum * self.food_weight
137.                     ]
138.         }
139.
140.     # 玩家经过或在村庄停留时可用剩余的初始资金或挖矿获得的资金随时购买
    水和食物，每箱价格为基准价格的 2 倍
141.     @property
142.     def can_buy(self):
143.         if self.terrain is eTerrain.village:
144.             return True
145.
146.
147.     @property
148.     def today_food_uniprice(self):
149.         return self.food_buniprice * 2
150.
151.     @property
152.     def today_water_uniprice(self):
153.         return self.water_buniprice * 2
154.
155.
156.         # 沙暴日必须原地停留
157.         # 第一天到矿山不能挖矿
158.
159.
160.     @property
161.     def today_consum_factor(self):
162.         """今日资源消耗倍率因子：取决于今日采取的行动"""
163.         mmap = {
164.             eAct.stay: 1,
165.             eAct.move: 2,
166.             eAct.mine: 3,
167.         }
168.         return mmap[self.today_action]
```

```
169.
170.     @property
171.     def today_water_consum(self):
172.         water_consum = self.water_bconsum(self.weather) * self.to
    day_consum_factor
173.         return water_consum
174.
175.     @property
176.     def today_food_consum(self):
177.         food_consum = self.food_bconsum(self.weather) * self.toda
    y_consum_factor
178.         return food_consum
179.
180.     @property
181.     def day_left(self):
182.         """剩余天数"""
183.         return self.day_limit - self.day
184.
185.     @staticmethod
186.     def format_consum(consum):
187.         quantity, cost, weight = consum
188.         return f'{quantity} 箱', f'{cost} 金', f'{weight} kg'
189.
190.     @staticmethod
191.     def sumup_res_money(consums):
192.         return sum(consums[res_name][1] for res_name in consums)
193.
194.     @property
195.     def desc(self):
196.         consums = self.today_consums_table
197.
198.                     ret = f'第 {self.day} 天 [天
    气 : {self.weather.value}] [动
    作: {self.today_action.value}]\n' + \
199.             f'[区域: {self.area_no}/{self.terrain.value}]\n' + \
200.               f'[今日消耗]:\n' + \
201.               f'水\t{self.format_consum(consums["水"])}\n' + \
202.                       f'食物\t{self.format_consum(consums["食物
    "])}\n' + \
203.               f'合计: {self.sumup_res_money(consums)} 金'
204.         # + f'资源\t数量\t价值\t质量\n' + \
205.
206.         return ret
```

```
207.
208.     def print_desc(self):
209.         print(self.desc)
210.
211.     def can_game_continue(self):
212.         if self.money >= 0:
213.             return True
214.         if self.weight_limit >= self.weight:
215.             return True
216.         if self.day_left > 0:
217.             return True
218.         if self.food_left >= 0 and self.water_left >= 0:
219.             return True
220.         if self.terrain is not eTerrain.destination:
221.             return True
222.         # 已到达终点
223.
224.         pass
225.
226.     def run_next_day(self, next_action):
227.         if not self.can_game_continue():
228.             raise RuntimeError('不满足约束，游戏无法继续，已失败')
229.
230.         # if next_action is eAct.mov
231.
232.         self.day += 1
233.         self.today_action = next_action
234.
235.         # 刷新天气、地形
236.
237.         if self.weather is eWea.sandstorm and self.today_action i
    s eAct.move:
238.             self.today_action = eAct.stay
239.             print('沙暴日强制滞留')
240.
241.         if self.today_action is eAct.move and self.terrain is eTe
    rrain.mineland:
242.             self.mineland_day = self.day
243.
244.
245.     def check_state(self):
246.         # [检查状态组合是否有效]
247.         # 检查挖矿是否非法
```

```
248.             if self.today_action is eAct.mine and self.terrain is not
    eTerrain.mineland:
249.                 raise AgainstGameRule('无法挖矿')
250.             if self.today_action is eAct.mine and self.mineland_day =
    = self.day:
251.                 raise AgainstGameRule('第一天无法挖矿')
252.             # 检查可否移动
253.             if self.weather is eWea.sandstorm and self.today_action i
    s eAct.move:
254.                 raise CanNotMoveWhenMeetStorm('沙暴天气无法移动')
255.
256.     @property
257.     def game_state(self):
258.         return {
259.             'day': self.day,
260.             'weather': self.water_weight,
261.             'weight': self.weight,
262.             'money': self.money,
263.             'area_no': self.area_no,
264.             'mineland_day': self.mineland_day,
265.
266.         }
267.
268.     @game_state.setter
269.     def game_state_set(self, state):
270.         self.day = state.get('day')
271.
272.
273.
274. weather = [None, '高温', '高温', '晴朗', '沙暴', '晴朗', '高温', '沙
    暴', '晴朗', '高温', '高温', '沙暴', '高温', '晴朗', '高温', '高温', '
    高温', '沙暴', '沙暴', '高温', '高温', '晴朗', '晴朗', '高温', '晴朗
    ', '沙暴', '高温', '晴朗', '晴朗', '高温', '高温']
275. m = bmodel()
276.
277.
278.
279. def m_set(m: bmodel, action, act_arg, weather, ):
280.     m.day = 0
281.     m.food_left = 0
282.     m.water_left = 0
283.
284.     m.today_action = None
285.     m.weather = []
```

```
286.     m.area_no = None
287.     m.terrain = None
288.     m.firstday_in_mineland = False
289. 
290. 
291. def area_no_2_terrain(area_no):
292.     mmap = {
293.         15: eTerrain.village,
294.         12: eTerrain.mineland,
295.         27: eTerrain.destination
296.     }
297.     return mmap.get(area_no, eTerrain.normal)
298. 
299. 
300. class OperaAgent:
301.     def __init__(self, m: bmodel):
302.         self.m = m
303.         self._cb_today_settle = None
304. 
305.     def go_to(self, area_no: int):
306.         """移动"""
307.         m = self.m
308.         m.day += 1
309.         m.today_action = eAct.move
310.         old_area_no = m.area_no
311.         m.area_no = area_no
312. 
313.         # 刷新天气和地形
314.         m.weather = eWea(weather[m.day])
315.         m.terrain = area_no_2_terrain(m.area_no)
316. 
317.         if m.terrain is eTerrain.mineland:
318.             old_mineland_day = m.mineland_day
319.             m.mineland_day = m.day
320. 
321.         try:
322.             m.check_state()
323.             self.today_settle()
324.         except CanNotMoveWhenMeetStorm:
325.             # 还原操作
326.             m.today_action = eAct.stay
327.             m.area_no = old_area_no
328.             if m.terrain is eTerrain.mineland:
329.                 m.mineland_day = old_mineland_day
```

```
330.                 m.terrain = area_no_2_terrain(m.area_no)  # 刷新地形
331.                 m.check_state()
332.                 self.today_settle()
333.                 # 重试
334.                 self.go_to(area_no)
335. 
336.     def stay(self):
337.         """停留"""
338.         m = self.m
339.         m.day += 1
340.         m.today_action = eAct.stay
341. 
342.         # 刷新天气和地形
343.         m.weather = eWea(weather[m.day])
344. 
345.         m.check_state()
346.         self.today_settle()
347. 
348. 
349.     def mine(self):
350.         """挖矿"""
351.         m = self.m
352.         m.day += 1
353.         m.today_action = eAct.mine
354. 
355.         # 刷新天气和地形
356.         m.weather = eWea(weather[m.day])
357. 
358.         m.check_state()
359.         self.today_settle()
360. 
361.     def today_settle(self):
362.         m.print_desc()
363.         if self._cb_today_settle:
364.             self._cb_today_settle(m, self)
365. 
366. 
367. def list_add(x, y):
368.     for i in range(len(x)):
369.         x[i] += y[i]
370. 
371. def consum_add(x, y):
372.     for key in x:
373.         list_add(x[key], y[key])
```

```
374.
375.
376.
377.
378. def main():
379.     agent = OperaAgent(m)
380.
381.
382.     res_consum_log = []
383.     action_log = []
384.     area_log = []
385.     weather_log = []
386.
387.     stat = {'水': [0,0,0], '食物': [0,0,0]}
388.
389.
390.     def log(*args):
391.         today_sonsum = m.today_consums_table
392.         # assert m.day - 1 == len(res_consum_log)
393.         res_consum_log.append(today_sonsum)
394.
395.         area_log.append(m.area_no)
396.         action_log.append(m.today_action)
397.         weather_log.append(m.weather)
398.
399.         consum_add(stat, today_sonsum)
400.
401.         fdesc = [(x, m.format_consum(stat[x])) for x in stat]
402.         print('[目前总消耗]')
403.         for x in fdesc:
404.             print("{}\t{}".format(*x))
405.
406.         money_sum = sum(stat[x][1] for x in stat)
407.         print(f'总计: {money_sum} 金')
408.         print('')
409.
410.         return stat
411.
412.     agent._cb_today_settle = log
413.
414.     def go_dd():
415.         agent.go_to(25)
416.         agent.go_to(26)
417.         agent.go_to(27)
```

```
418.
419.     def go_mineland():
420.         # agent.go_to(1)
421.         agent.go_to(25)
422.         agent.go_to(24)
423.         agent.go_to(23)
424.         agent.go_to(22)
425.         agent.go_to(9)
426.         agent.go_to(15)  # 村庄
427.         # agent.go_to(14)
428.         # agent.go_to(12)
429.
430.     def go_mine():
431.         agent._cb_today_settle = lambda x, y: x
432.         go_mineland()
433.         agent._cb_today_settle = log
434.
435.         print('=========begin======')
436.
437.         # m.day = 9
438.         # m.today_action = eAct.stay
439.         #
440.         # m.area_no = 12
441.         #
442.         # # 刷新天气和地形
443.         # m.weather = eWea(weather[m.day])
444.         # m.terrain = area_no_2_terrain(m.area_no)
445.         #
446.         # if m.terrain is eTerrain.mineland:
447.         #     m.mineland_day = m.day
448.
449.         m.check_state()
450.         agent.go_to(14)
451.         agent.go_to(12)
452.         agent.mine()  # 13
453.         agent.mine()  # 14
454.         agent.mine()  # 15
455.         agent.mine()  # 16
456.         agent.mine()
457.         agent.mine()
458.         agent.mine()
459.         agent.go_to(14)
460.         agent.go_to(15)
461.
```

```
462.         # agent.go_to(14)
463.         # agent.go_to(12)
464.         # agent.mine()  # 13
465.         # agent.mine()  # 14
466.         # agent.mine()  # 15
467.         # agent.mine()  # 16
468.         # # agent.mine()
469.         # # agent.mine()
470.         # agent.stay()
471.         # agent.stay()
472.         # agent.mine()
473.         # agent.mine()
474.         # agent.mine()
475.         #
476.         # agent.go_to(14)
477.         # agent.go_to(15)
478.
479.
480.         # agent.stay()
481.         # agent.stay()
482.
483.     # go_mineland()
484.     go_mine()
485.
486.
487.     print('区域', '水消耗 kg', '食物消耗 kg',
488.           '水消耗 箱', '食物消耗 箱',
489.           '行动', '天气')
490.     for x in range(len(res_consum_log)):
491.         s = res_consum_log[x]
492.         a = (area_log[x], -s['水'][2], -s['食物'][2],
493.              -s['水'][0], -s['食物'][0],
494.              action_log[x].value,
495.              weather_log[x].value,
496.              )
497.         print(*a, sep='\t')
498.     print('\n')
499.
500.     import pandas
501.
502.     print(res_consum_log)
503.
504.
505. if __name__ == '__main__':
```

```
506.     main()
```