穿越沙漠的策略研究

摘要

本文分析了“穿越沙漠”游戏每关卡的最佳路线和行为。根据动态规划和剪枝算法给出了确定天气情况下的最优策略，并讨论了不同天气概率下的游戏策略与期望收益。

针对第一问，本文首先将地图简化成构造邻接矩阵，并建立了基于动态规划的剪枝算法，给出最佳的游戏策略。第一关的最佳游戏收益为 10470 元，共挖矿 7 天，交易两次，分别在 9 号区域和 15 号区域休息 1 天和 3 天。第二关的最佳游戏收益为 12730 元。分别在第 4 天于 11 号区休息、在第 7 天于 28 号区休息。在 30 号区域的矿山挖矿 6 次，在 55 号区域的矿山挖矿 7 次。

针对第二问，首先进行地图简化，并根据天气概率确立价值函数。求解得到第三关的最优路线为 1→5→6→13。在高温天气发生的概率大于 20%时，期望收益最大的决策为在起点购买 54 箱水和 54 箱食物。在高温天气发生的概率为 10%时，期望收益最大的决策为在起点购买 39 箱水和 43 箱食物，有 99.58%的概率走到终点，期望收益为 9424.18 元。

第二问中对第四关进行求解讨论，得到在沙暴概率为 5%时，玩家的最大期望收益为 13673.2 元，沙暴的发生概率每上升 5%，玩家的最大期望收益下降约 1000 元。可以在第四关取得最大收益的策略是直接前往矿山挖矿或根据前进路上的天气情况先前往村庄购买物资后前往矿山，之后在矿山和村庄之间往返，最终在前往终点时额外留下一到两天的时间防止游戏失败。

第三问第五关先运用第一关的算法求解出十条最优路线，并通过进一步的筛选确定八条最优路线。接着根据不完全信息静态博弈理论，求解出在等概率选择路线下的博弈战略式，进而求解出该组博弈中的纳什均衡组合，由于存在四组纳什均衡组合，故采用混合策略，求出最优策略为 45 箱水，57 箱食物，并 50%概率沿 1→5→5→6→13 的路径前往终点，50%概率经 1→4→4→6→13 的路径前往终点，期望收益为 9260 元。在双人合作获得全局最优解的情况下，总收益为 19045 元。

第三问中第六关进行三人博弈，且玩家仅知道当天天气与行动结束后他人的状态。本文认为玩家在制定策略时，可以将他人看做敌方，进而构造蒙特卡洛树，本文为第六关进行了模拟对抗，在三人均有不同的决策性格时，最终的收益总和为 30437.5 元。

关键词: 动态规划　纳什均衡 剪枝算法　蒙特卡洛树

一、问题重述

1.1 背景知识

1.1.1 问题引入

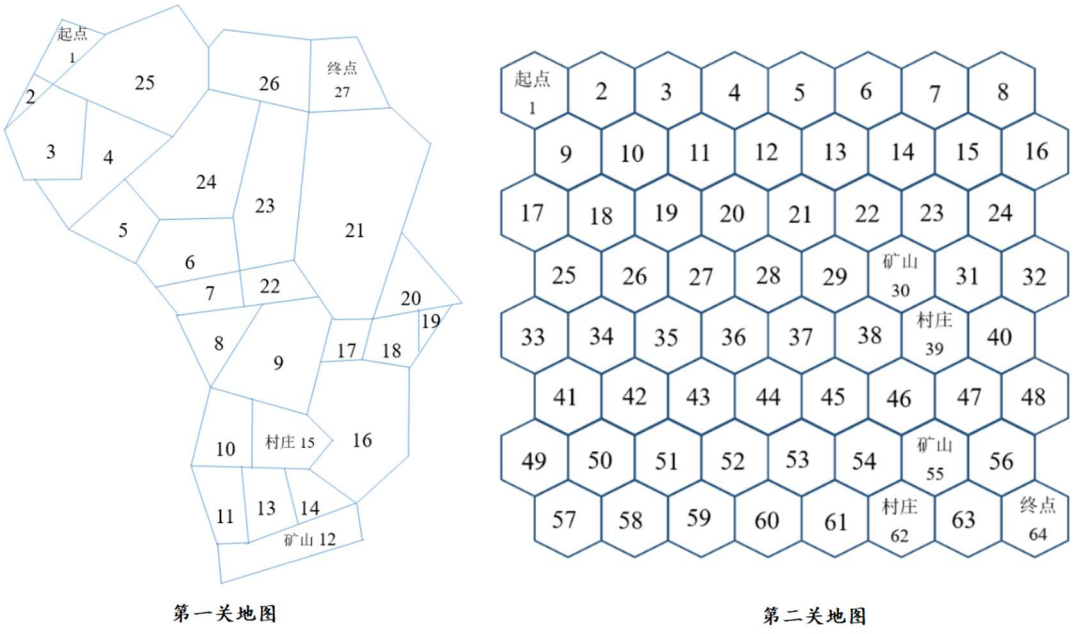
在“穿越沙漠”游戏中，玩家凭借一张地图，利用初始资金购买一定数量的水和食物，从起点出发，在沙漠中行走。途中会遇到不同的天气：“晴朗”、“高温”、“沙暴”，沙暴日必须在原地停留。玩家也可在矿山、村庄补充资金或资源，但拥有的水和食物的质量之和不能超过负重上限。

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。本文希望给出在不同条件下玩家的最优游戏策略。

1.2 具体问题

1.2.1 问题一

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，试给出一般情况下玩家的最优策略。求解附件中的“第一关”和“第二关”。

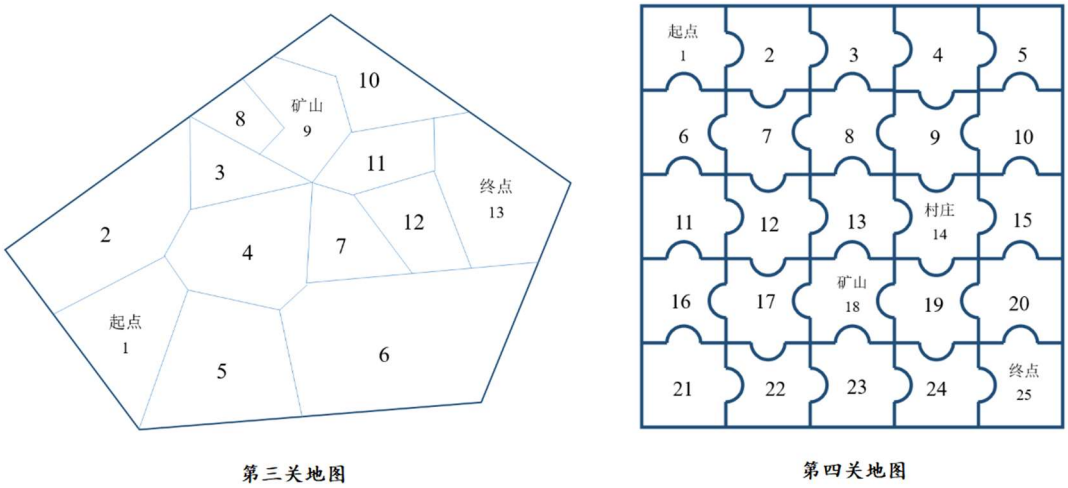


第一关地图　　第二关地图

图表 1：第一关和第二关的地图

1.2.2 问题二

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。



第三关地图　　第四关地图

图表 2：第三关和第四关地图

1.2.3 问题三

增加约束条件：有 n 名玩家时，玩家的行为和位置有可能会导致资源消耗数量增加、挖矿收益减少或购买物资价额上涨。讨论以下问题：

（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。

其中第五关地图和第三关相同，第六关地图和第二关相同。

二、问题分析

2.1 针对问题一的分析与对策

问题一是在已知未来每天的天气和地图的基础上，给出一个人到达终点时，使得所获钱财最多的最优策略。本文首先构造出用于评判策略优劣的目标函数，并将地图进行适当的简化，只保留关键点，即起点，终点，矿山和村庄的相对位置关系，使其转换为无向图。运用动态规划的算法将每一天的目标函数与后一天的目标函数构造联系，列出方程，并在给定约束条件的基础上从终点往起点递推，即可求出最优策略。

2.2 针对问题二的分析与对策

问题二是在问题一的基础上减少了可以知道每天天气的条件，玩家仅可知道行动当天的天气并决定接下来的行进路线。先分析第三关，第三关是无沙暴天气仅有高温和晴朗天气，运用贝叶斯算法解决该问，本文首先将晴天概率设为变量则可得知高温天气概率，再通过变步长搜索法求解出挖不挖矿，怎么挖矿的临界概率，并给出每种情况的最优策略；第四关则是在第三关的基础增加了较少的沙暴天气，本文通过查找真实情况下沙漠的天气，给出沙暴天气的概率，并在此基础运用解第三关的做法求解最优策略。

2.3 针对问题三的分析与对策

问题三增添了多人玩法，多人会导致资源的额外消耗，先分析第五关，由于没有村庄，因此，物资是单调减少的，可以对问题进行简化，进而求出单人条件下的最优路径；由于是

玩家共有俩人，两个人的竞争合作并存，因此运用博弈论对其进行进一步的分析与求解，最终得到最优策略。问题三第六关中，由于三名玩家存在相互博弈，且将来天气未知，因此玩家可以构造蒙特卡洛树，将其他二人视为敌方，对每一步早成的结果用效用函数评估，并根据自身性格和游戏策略进行剪枝，选择最有利的策略。建模完成后可以进行仿真，寻找最优的决策人格以及带来的期望收益。

三、模型的假设

1. 穿越某个区域所用时间与示意图中该区域的面积无关。
2. 玩家的决策是理性的。
3. 玩家的性格在单局游戏中不改变。
4. 第五关玩家以到达终点为目标购买物资。

四、符号说明

符号	定义	单位
w	玩家当前负重总量	千克
d	当前游戏天数	天
g	玩家当前拥有金钱总量	元
e_i	第i号玩家的当前位置	-
A_i	第i关地图的邻接矩阵	-
$earn$	玩家挖矿的到的基础收益	元
$cost$	玩家在购买物资的花费	元

五、模型的建立与求解

5.1 问题一的分析与求解

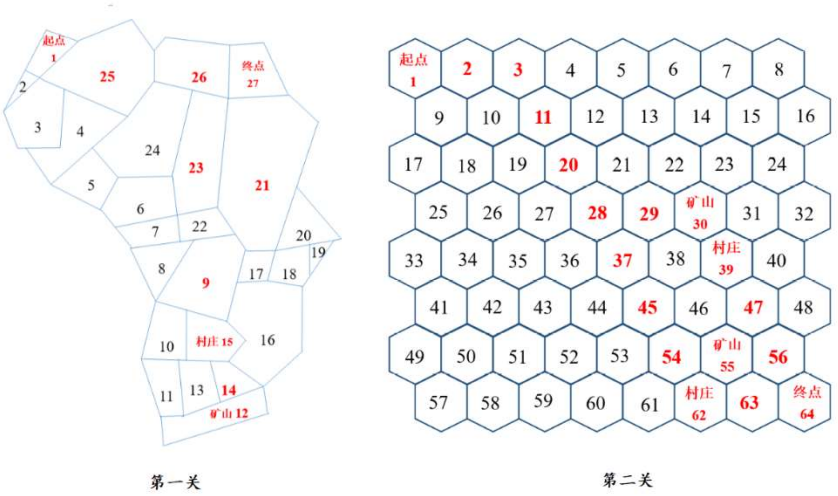
5.1.1 建模前的准备

5.1.1.1 简化地图

附件中提供了各个关卡的地图，其中起点、村庄、矿山和终点是特殊节点，其余节点为等价节点。

由于移动需要消耗资源，由假设 3 玩家决策是理性的，因此在制定优化策略时仅需考虑部分节点。由此可以对附件中提供的地图进行一定程度上的简化。

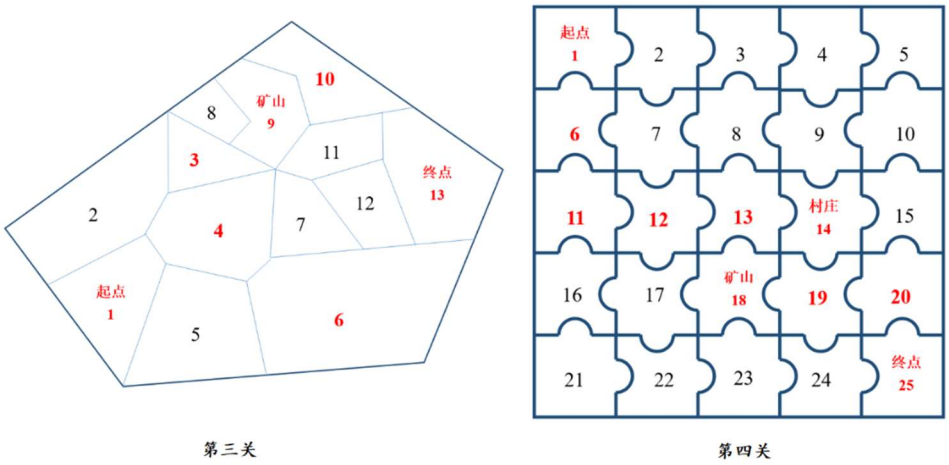
在第一二问中，玩家单人游戏，如前往地图边缘的节点则会对玩家造成不利，因此仅需考虑在特殊节点间移动的最短距离所途径的节点即可。



第一关　　第二关

图表 3：对第一关和第二关的地图进行简化

运用 Dijkstra 最短路算法对地图进行简化，上图中，标红色加粗区域为需要考虑的关键区域，黑色区域为在进行理性决策时不会考虑的区域。由此可以将第一张地图的 27 个节点简化为 9 个节点，将第二张地图的 64 个节点简化为 18 个节点，减少运算复杂度。



第三关　　第四关

图表 4：对第三关和第四关的地图进行简化

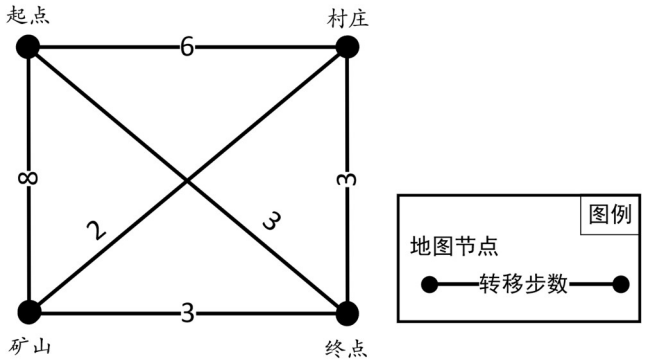
同理，对第三张和第四张的地图进行简化，分别将考虑的节点数从 13 点降低到 7 点，从 25 点降低到 10 点。

但在第三问需要考虑多人游戏时，由于与他人一同采矿、购买物资、具有相同的行动路线时均会造成负面影响，因此在处理第三问时不应对地图进行简化，玩家应适当前往其他的的区域以避免对自己造成负面影响。

5.1.1.2 构造邻接矩阵与可达矩阵

由 5.1.1.1 对地图的简化，可以将地图抽象为无向图$G =< V, E >$，其中V为关键节点，E为节点之间的路径，(v_i, v_j)的权值w_{ij}为从地图上的v_i点走到v_j点所需途径的区域数。

以第一张地图为例，可以将其抽象为下图:



图表 5：地图网络结构

由于游戏地图本身是一种无向图，对于任意无向图$G=<V,E>$，可以构造临接矩阵A：

$$A_{ij}=\begin{cases} w_{ij}\ , & (v_i,v_j)\in E(G) \\ \infty\ , & (v_i,v_j)\notin E(G) \end{cases}$$

则单人模式游戏时，第一张地图的邻接矩阵如下：

$$\begin{pmatrix} 0 & 6 & 8 & 3 \\ 6 & 0 & 2 & 3 \\ 8 & 2 & 0 & 5 \\ 3 & 3 & 5 & 0 \end{pmatrix}$$

其中i和j从 1 至 4 分别是起点，村庄，矿山，终点。

同理可以构造出第二关地图的邻接矩阵，如下所示：

$$\begin{pmatrix} 0 & 7 & 8 & 9 & 10 & 11 \\ 7 & 0 & 1 & 3 & 4 & 4 \\ 8 & 1 & 0 & 2 & 3 & 3 \\ 9 & 3 & 2 & 0 & 1 & 2 \\ 10 & 4 & 3 & 1 & 0 & 2 \\ 11 & 4 & 3 & 2 & 2 & 0 \end{pmatrix}$$

其中i和j从 1 至 6 分别是起点，矿山 30，村庄 29，矿,55，村庄 62，终点。

但在多人游戏时，为了防止其他玩家的位置和行为对自己造成不利影响，在理性决策的情况下玩家不会仅考虑特殊节点之间最短的路径。因此不应在第三问中对问题进行简化。故第三问应当使用原地图。原地图同样是无向图$G=<V,E>$，同样可以对原地图构造邻接矩阵，等价于各个节点之间边的权重都是 1，定义如下：

$$A_{ij}=\begin{cases} 1\ , & (v_i,v_j)\in E(G) \\ 0\ , & (v_i,v_j)\notin E(G) \end{cases}$$

5.1.1.3 确立价值函数

由于本题的游戏目标是在规定时间内到达终点的同时获得足够多的钱，因此钱是我们所关注的重点，本文建立价值函数：

$$g(e,d,z_1,z_2)$$

其中 g 表示在第 d 天，点 e 处的剩余资金，$z1$，$z2$ 分别表示此时剩余的食物和水的箱数。

5.1.2 模型的确定

5.1.2.1 目标函数的确定

本题的游戏目标是在规定时间内到达终点的同时获得足够多的钱，显然可得，当购买的物资被充分利用时，到达终点时剩余的食物和水的箱数均为 0，故目标函数应为:

$$\max g\left(e,d,0,0\right)$$

其中 e 为终点。

5.1.2.2 约束条件的确定

确立路线时应保证该条路线的成功率，即在到达终点的路途任一天，应满足食物和水的箱数不低于 0，即:

$$z_1 \geq 0, z_2 \geq 0$$

其次，本题限制了一个人的承重上限为 1200kg，故食物和水的箱数不宜过高，其满足:

$$3z_1 + 2z_2 \leq 1200$$

本题天数上限是 30，故:

$$d \leq 30$$

5.1.2.3 模型的汇总

由上文，可汇总得到本问的模型，即:

$$\max g\left(e,d,0,0\right)$$

$$s.t.\begin{cases} z_1 \geq 0, z_2 \geq 0 \\ 2z_1 + 3z_2 \leq 1200 \\ d \leq 30 \end{cases}$$

5.1.3 模型的求解算法

5.1.3.1 动态规划算法

本文运用动态规划算法求解此问，动态规划算法的核心是求解递推方程，先推导价值函数的递推方程；由于玩家能进行的操作共有 4 种，分别为原地停留，行进一格，在矿场挖矿和在村庄购买物资，下面，本文就具体 4 种方式分别表示价值函数。[1]

当原地停留时，玩家位置不变，物资消耗为 1 份基础消耗量，相较于前一天，价值函数满足:

$$g = g\left(e, d-1, z_1 + dz_1\left(d-1\right), z_2 + dz_2\left(d-1\right)\right)$$

其中

$$dz_1\left(d-1\right)，\ dz_2\left(d-1\right)$$

分别表示在 *d-1* 天天气的状况下的一份基础消耗量。

当玩家选择行进一格时，价值函数满足:

$$g = g\left(e_i, d-1, z_1 + 2dz_1\left(d-1\right), z_2 + 2dz_2\left(d-1\right)\right)$$

其中 *ei* 表示玩家是从 *ei* 点移动。

当玩家选择在矿场挖矿时，价值函数满足:

$$g = g\left(e, d-1, z_1 + 3dz_1\left(d-1\right), z_2 + 3dz_2\left(d-1\right)\right) + earn$$

当玩家选择在村庄购买物资时，价值函数满足：

$$g = g\left(e, d, z_1 - \mathrm{z}_3, z_2 - z_4\right) - \cos\mathrm{t}\left(z_3, z_4\right)$$

其中 *z3*，*z4* 分别表示购买的水和食物的箱数，*cost（z3，z4）* 表示购买所花费的钱财。由于目标是最大化钱财，因此在此基础上，可以得到关于价值函数的递推表达式：

$$g\left(e, d, z_1, z_2\right) = \max\left\{\begin{array}{l} g\left(e, d-1, z_1 + dz_1\left(d-1\right), z_2 + dz_2\left(d-1\right)\right), \\ g\left(e_i, d-1, z_1 + 2dz_1\left(d-1\right), z_2 + 2dz_2\left(d-1\right)\right), \\ g\left(e, d-1, z_1 + 3dz_1\left(d-1\right), z_2 + 3dz_2\left(d-1\right)\right) + earn, \\ g\left(e, d, z_1 - \mathrm{z}_3, z_2 - z_4\right) - \cos t\left(z_3, z_4\right) \end{array}\right\}$$

在此递推表达式的基础上，由起点向终点递推，即可得到最优策略。

5.1.3.2 算法的剪枝优化

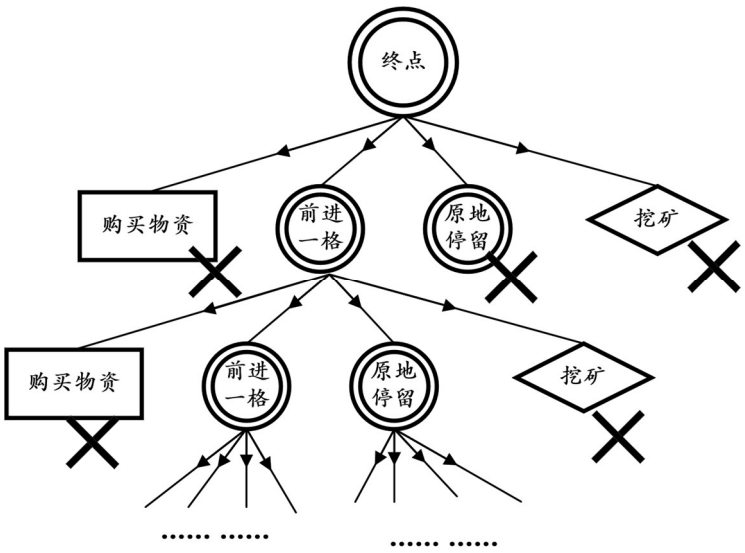
为了减小算法复杂度，增加对目标函数的限制条件达到剪枝目的，具体为：

当玩家在起点和村庄购买物资时，开销必须小于等于资金，即：

$$\cos\mathrm{t}\left(z_3, z_4\right) \le g$$

在此递推表达式的基础上，由终点向起点递推，即可得到最优策略。

又由题目要求，玩家到达终点后游戏结束，故在上一步的决策中仅有可能选择“前进一格”行为，其他行为被剪枝。以此递归，在终点的前一个节点不存在村庄和矿山，故购买物资和挖矿行为被剪枝。剪枝的过程可以加速深度较深的决策树的评估更新速度，同时也使得决策更加符合游戏逻辑。决策过程可由下图简要表示：

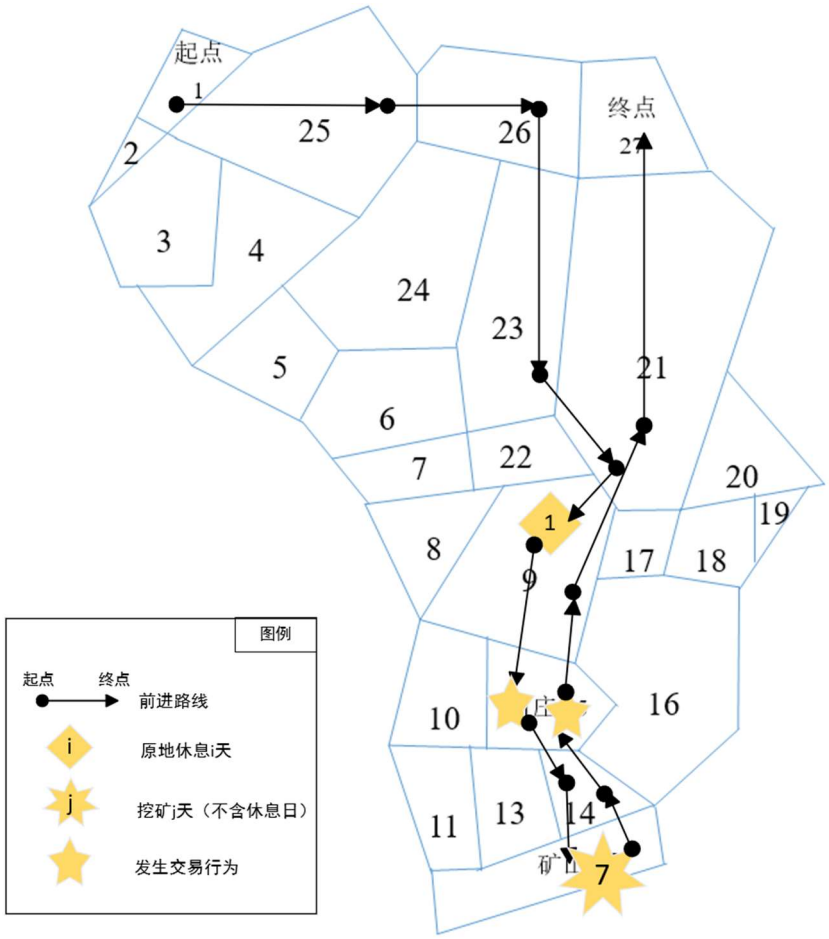


图表 6：反向递归决策树与剪枝

5.1.4 模型的求解结果及分析:

5.1.4.1 第一关和第二关求解

基于 5.1.3 提出的算法，求解出第一关的最优行进路线如下，最终收益为 10470 元：



其中共挖矿 7 天，交易两次，分别在 9 号区域和 15 号区域休息 1 天和 3 天。具体数值依据题目要求填已写在附件中。

同理对第二关进行求解，可得最优路线的最终收益为 12730 元。其中分别在第 4 天于 11 号区休息、在第 7 天于 28 号区休息。在 30 号区域的矿山挖矿 6 次，在 55 号区域的矿山挖矿 7 次。

两关求解的具体数值均依据题目要求填已写在支撑材料，同时放在附录中。

5.1.4.1 一般策略分析

在两关的最优路线中，玩家均是按照最短路前往矿山和村庄，前进途中除非遇到沙暴，否则尽量不休息。在第一关中需根据自己的水和食物量决定沙暴日是否挖矿，在第二关中沙暴日选择挖矿仍然可以获得更好的收益，又因为靠近村庄容易得到补给，故在第二关可以尽可能多地选择挖矿。

5.2 问题二的分析与求解

5.2.1 模型的准备

5.2.1.1 简化地图

与 5.1.1.1 方法相同，先将地图简化，则第三张地图的邻接矩阵如下：

$$\begin{pmatrix} 0 & 3 & 3 \\ 3 & 0 & 2 \\ 3 & 2 & 0 \end{pmatrix}$$

其中i和j从 1 至 3 分别是起点，矿山，终点。

同理可以构造出第二关地图的邻接矩阵，如下所示：

$$\begin{pmatrix} 0 & 5 & 5 & 8 \\ 5 & 0 & 2 & 3 \\ 5 & 2 & 0 & 3 \\ 8 & 3 & 3 & 0 \end{pmatrix}$$

其中i和j从 1 至 4 分别是起点，矿山，村庄，终点。

5.2.1.2 用概率量化天气情况

由于问题二中天气的未知性，用每天某种天气可能出现的概率来将其量化，以如下矩阵表示

$$P_{1\times 3} = \begin{pmatrix} p_1 & p_2 & p_3 \end{pmatrix}$$

其中 *p1* 表示晴朗天气概率，*p2* 表示高温天气概率，*p3* 表示沙暴天气概率，则该矩阵满足

$$\sum_{i=1}^{3} p_i = 1 \;\; p_i \geq 0$$

5.2.1.3 价值函数的确立

与第一问不同的是，由于天气情况的不可知，故购买物资时会相对多的采购以保证能有较大成功率到达终点，但显然购买足够多的物资保证能度过第三关不是最优方案，因此，第三关的策略在极端天气条件下是有一定失败率的，为了衡量失败率给最终策略带来的影响，本文在第一问价值函数的基础上引入惩罚量 *pun*，用以惩罚过于贪心的策略导致的失败，故，本问价值函数应为：

$$g(e,d,z_1,z_2) = mg - (1-m)\,pun$$

其中 *m* 为衡量成功失败的 0-1 变量，成功时 *m* 取 1，否则取 0。

下面确定成功失败的判别标准，失败的情况有如下 3 种，即

食物消耗殆尽，即：

$$z_1 < 0$$

水消耗殆尽，即：

$$z_2 < 0$$

天数超过最大限制，即：

$$d > d_{\max}$$

负重超过最大限制，即：

$$2z_1 + 3z_2 > 1200$$

相对多的采购以保证能有较大成功率到达终点，故到达终点时的剩余物资不一定为 0，玩家到达终点时会退回一部分钱，满足如下等式：

$$g(e,d,0,0)=g(e,d,z_1,z_2)+sell(z_1,z_2)$$

其中 e 为终点，

$$sell(z_1,z_2)=\frac{10z_1+5z_2}{2}$$

至此，确立了本问的价值函数。

5.2.2 模型的建立

在以上准备的基础上，我们确立了本问的模型为：

$$\max g(e,d,0,0)$$

其中 e 为终点。

5.2.3 模型的求解算法

5.2.3.1 动态规划算法

本文运用动态规划算法求解此问，动态规划算法的核心是求解递推方程，先推导价值函数的递推方程；由于玩家能进行的操作共有 4 种，分别为原地停留，行进一格，在矿场挖矿和在村庄购买物资，考虑到天气的随机性，增加实用用贝叶斯算法求解价值函数；下面，本文就具体 4 种方式分别表示价值函数。

当原地停留时，玩家位置不变，物资消耗为 1 份基础消耗量，相较于后一天，价值函数满足：

$$g=\sum_{i=1}^{3}p_i g\left(e,d+1,z_1-dz_1(i),z_2-dz_2(i)\right)$$

其中

$$dz_1(i)，\ dz_2(i)$$

与天气相对应，分别表示第 i 种天气的状况下的一份基础消耗量。

当玩家选择行进一格时，价值函数满足：

$$g=\sum_{i=1}^{3}p_i g\left(e_i,d+1,z_1-2dz_1(i),z_2-2dz_2(i)\right)$$

其中 *ei* 表示玩家去往 *ei* 点。

当玩家选择在矿场挖矿时，价值函数满足：

$$g=\sum_{i=1}^{3}p_i g\left(e_i,d+1,z_1-3dz_1(i),z_2-3dz_2(i)\right)+earn$$

当玩家选择在村庄购买物资时，价值函数满足：

$$g=g(e,d,z_1-z_3,z_2-z_4)-\cos t(z_3,z_4)$$

其中 *z3，z4* 分别表示购买的水和食物的箱数，*cost（z3，z4）* 表示购买所花费的钱财。

由于目标是最大化钱财，因此在此基础上，可以得到关于价值函数的递推表达式：

$$g(e,d,z_1,z_2)=\max\begin{Bmatrix}\sum_{i=1}^{3}p_i g(e,d+1,z_1-dz_1(i),z_2-dz_2(i)),\\ \sum_{i=1}^{3}p_i g(e_i,d+1,z_1-2dz_1(i),z_2-2dz_2(i)),\\ \sum_{i=1}^{3}p_i g(e_i,d+1,z_1-3dz_1(i),z_2-3dz_2(i))+earn,\\ g(e,d,z_1-z_3,z_2-z_4)-\cos t(z_3,z_4)\end{Bmatrix}$$

在此递推表达式的基础上，由终点向起点递推，即可得到最优策略。

5.2.4 问题二第三关的分析和求解

5.2.4.1 策略分析

在天气未知的情况下，玩家需要面对不同的概率事件天气事件，并准备合适的物资，以最大化自己的期望收益。

由题目信息，前 10 天内不会出现沙暴天气，且游戏截止日期为十天，因此仅需考虑高温天气发生的概率。

第三关地图较为简单，本文首先随机若干天气，使用问题一的模型进行模拟决策，可以发现在第三关地图中，几乎每种情况下去矿山挖矿都是较劣解，而通过最短路 1->5->6->13 直接前往终点是更优的组合。

5.2.4.2 灵敏度分析

虽然最优路径已经得到规划，但玩家需要面对不确定的天气因素，并考虑不同的水和食物所能获得的成功率与游戏收益的权衡。

因此本文讨论了不同的高温天气发生概率，以及不同游戏收益期望值的成功可能性，以检验游戏策略的稳定性。

在 10%的高温可能性下，不同的携带水和食物量策略，以及获胜的可能性如下:

表 1：不同游戏策略的获胜可能与收益期望

获胜概率	平均收益(元)	起点水量(箱)	起点食物量(箱)	期望(元)
94.89%	9558.06	27	33	9069.64
99.58%	9481.52	39	43	9441.7
99.87%	9436.44	42	44	9424.18
100%	9406.88	54	54	9406.88

表 1 中，平均收益指在随机仿真中，玩家在终点的最终收益的平均值。期望指游戏成功率与平均收益的乘积，反应了该策略的期望收益。上表中，最大的期望为 9441.7 元，因此在高温发生概率为 10%时，选择 39 箱水，43 箱食物，经1 → 5 → 6 → 13的路径直接前往终点是更优的策略。游戏

对 10%，20%，30%，40% ... 90%的高温发生概率进行仿真，选取最大的期望策略，统计如下表:

表 2：不同高温概率下得到最大期望的游戏策略

高温概率(%)	最大期望(元)	起点水量(箱)	起点食物量(箱)
10	9441.7	39	43
20	9382	54	54
30	9358	54	54
40	9334	54	54
50	9310	54	54
60	9286	54	54
70	9262	54	54
80	9238	54	54
90	9214	54	54

由表 2 可见，当高温发生的概率达到 20%以及以上时，取得最大游戏期望收益的策略，均是在起点即购买 54 箱水和 54 箱食物，连续撑住 3 天高温，保守地经由$1 \rightarrow 5 \rightarrow 6 \rightarrow 13$到达终点。

本关卡由于终点较近，矿山较远，且天气不定，故选取保守的策略更容易获得最佳期望收益。

5.2.5 问题二第四关的分析和求解

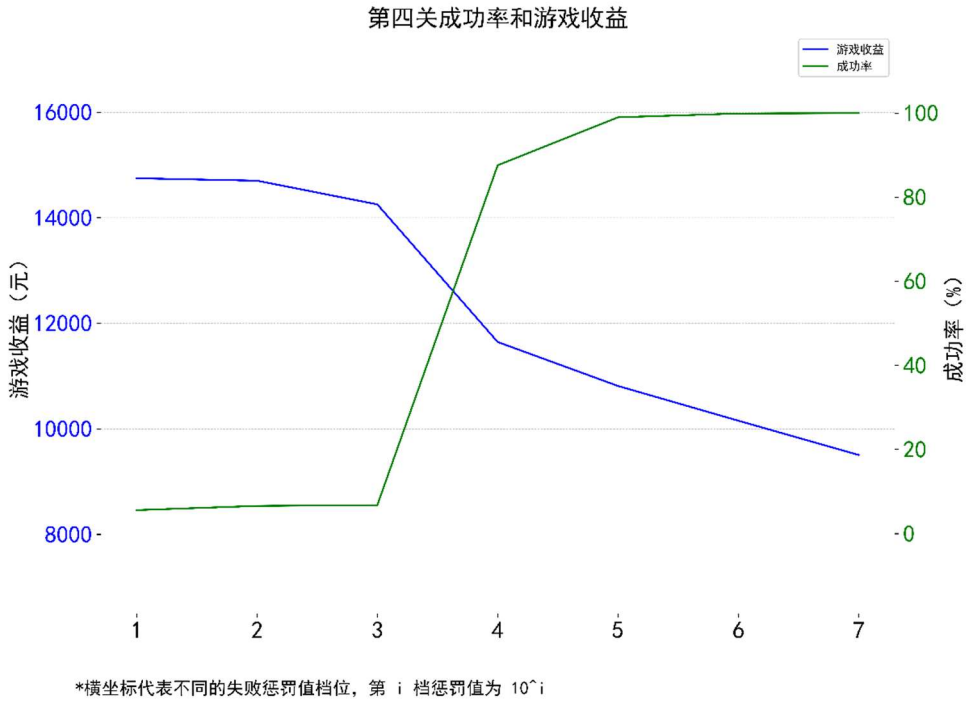
5.2.5.1 策略分析

在天气未知的情况下，玩家需要面对不同的概率事件，因此可以计算不同期望收益的成功率，并根据玩家的个人性格，如贪心或者保守，去选择游戏策略。

本文在问题一动态规划模型的基础上，增加游戏失败的惩罚值，若游戏失败则玩家失去对应档位的金钱，收益期望值也随之降低。针对不同性格与天气组合进行仿真，即可得出期望收益和对应的成功率。

5.2.5.2 求解结果和讨论

在沙暴概率为 20%的情况下，玩家在根据上文的决策策略进行理性游戏时，有 100%的可能性期望赚到 9506.2 元。有 87.6%的概率赚到 11647.4 元。在玩家采取极端贪心的策略情况下，有 5.6%的可能性赚到 14750 元，但由于成功率低于 10%，不建议玩家采用。



图表 7：第四关游戏收益与对应的成功率

由上图可见，期望收益在 14000 元以上的游戏策略很难成功（成功率小于 10%），而在期望收益下降到 11000 元以下时，成功率则大于 90%。因此玩家可以根据自己性格进行选择保守或贪心的游戏策略，如期望收益在 11000 元左右，则游戏有较大概率成功。

5.2.5.3 灵敏度分析

上图是在沙暴发生概率为 20%的情况下，讨论玩家不同的贪心程度导致的，加入惩罚项的期望值对应的成功率改变情况。下面将改变沙暴概率进行灵敏度分析，检验游戏策略是否稳定。

设第i种沙暴发生模式下，玩家在采取第j级别的贪心策略时，期望收益为E_{ij}，成功率为p_{ij}，则第i种沙暴发生模式下，最理智的优化期望收益可以表示为$Exp_i = max_j\{p_{ij} \times E_{ij}\}$，即寻找成功概率尽可能高同时期望收益也尽可能大的性格策略。

分别设置沙暴概率为 5%，10%，15%，20%进行仿真，观察该参数对最理智的优化期望收益的影响，结果如下表：

表 4：不同沙暴概率下最大期望收益

沙暴概率	最大期望收益(元)
5%	13673.2
10%	12639.1
15%	11569.2
20%	10695

由表可见，随着沙暴发生的概率增大，最大的期望收益下降，但下降过程较为平缓，每上调 5%的沙暴发生概率，约损失 1000 元收益。

可见游戏策略本身不易受到参数改变的影响，算法较为稳定。

仿真过程中发现玩家离开矿山时携带的水和食物的量对是否能完成游戏有较大影响。因此本文对不同的沙暴概率下玩家离开矿山所携带的最低食物和水进行了仿真分析，得到如下数据：

表 3：不同沙暴概率下成功游戏时离开矿山平均携带物资

沙暴概率(%)	离开矿山水量(箱)	离开矿山食物量(箱)
20	84	79
15	76.9	71.7
10	68.1	64.3
5	58.4	55.33

玩家在进行决策时应尽量将水和食物的量控制在上表的结果附近，以获得游戏胜利。

5.3 问题三分析与求解

5.3.1 问题三第五关的分析与求解

5.3.1.1 模型准的备

5.3.1.1.1 模型的假设

由于第五关有有 2 名玩家，为了避免路线相撞导致物资消耗殆尽导致的无法到达终点的情况，本文假设俩人均足够保守，做最悲观的打算，即携带刚好数量的食物和水，保证俩人即使路线完全重合依旧能到达终点。

5.3.1.1.2 单人最优路径的求解

在找到本关卡策略之前，本文先求解当玩家数量等于 1 时，最优的十条路线，此时该问题与第一关相同，运用第一关的模型，可以解得最优的十条路线，并对其编号，如下表所示：

路径\天数	0	1	2	3	4	5	6
1	1	5	5	6	13		
2	1	4	4	6	13		
3	1	1	1	5	6	13	
4	1	1	1	4	6	13	
5	1	5	5	5	6	13	
6	1	4	4	4	6	13	
7	1	4	4	7	11	13	
8	1	5	6	13			
9	1	4	6	13			
10	1	2	2	3	9	10	13

5.3.1.1.2 最优路径的筛选

将 5.3.1.1 的假设代入到所求得的 10 条路径中，可以求得 10 条路径的收益，此时的收益为俩人路径完全不重复时，走该条路的最大收益，如下表所示：

路径	1	2	3	4	5
收益	9370	9370	9315	9315	9315
路径	6	7	8	9	10
收益	9315	9205	9265	9265	9040

再考虑俩人路径完全重复时的收益，此时的收益为走该条路的最小收益，如下表所示:

路径	1	2	3	4	5
收益	9205	9205	9150	9150	9150
路径	6	7	8	9	10
收益	9150	8985	9020	9020	8765

可以看出路径 7,10 的最大收益分别为 9205 和 9040 元，而走路径 1,2 的最小收益为 9205 元，因此我们可以将路径 7,10 舍弃，留下剩下八条路径，并对其重新编号，如下表所示:

路径\天数	0	1	2	3	4	5
1	1	5	5	6	13	
2	1	4	4	6	13	
3	1	1	1	5	6	13
4	1	1	1	4	6	13
5	1	5	5	5	6	13
6	1	4	4	4	6	13
7	1	5	6	13		
8	1	4	6	13		

5.3.1.2 不完全信息静态博弈模型的建立

由本问已知条件可得知，俩名玩家的博弈类型为不完全信息静态博弈，根据前文所做的模型的准备，可以得到这组博弈的战略式表述，如下表所示:

路径\路径	1	2	3	4	5	6	7	8
1	(9205,9205)	(9315,9315)	(9370,9315)	(9370,9315)	(9315,9260)	(9370,9315)	(9315,9210)	(9370,9265)
2	(9315,9315)	(9205,9205)	(9370,9315)	(9370,9315)	(9370,9315)	(9315,9260)	(9370,9265)	(9315,9210)
3	(9315,9370)	(9315,9370)	(9150,9150)	(9260,9260)	(9205,9205)	(9260,9260)	(9315,9265)	(9315,9265)
4	(9315,9370)	(9315,9370)	(9260,9260)	(9150,9150)	(9260,9260)	(9205,9205)	(9315,9265)	(9315,9265)
5	(9260,9315)	(9315,9370)	(9205,9205)	(9205,9205)	(9150,9150)	(9260,9260)	(9260,9210)	(9315,9265)
6	(9315,9370)	(9260,9315)	(9260,9260)	(9260,9260)	(9260,9260)	(9150,9150)	(9315,9265)	(9260,9210)
7	(9210,9315)	(9265,9370)	(9265,9315)	(9205,9205)	(9210,9260)	(9265,9315)	(9020,9020)	(9210,9210)
8	(9265,9370)	(9210,9315)	(9265,9315)	(9265,9315)	(9265,9315)	(9210,9260)	(9210,9210)	(9020,9020)

由此我们建立起第五关的不完全信息静态博弈模型。

5.3.1.3 混合策略纳什均衡求解博弈模型

假设每名玩家选择路径 1-8 均为等概率的，接下来求解该组博弈中的最优组合，即为纳什均衡组合。[2]

先考虑其中一名玩家，运用贝叶斯算法解得其走每组路径的收益期望，如下表所示

路径	**1**	**2**	**3**	**4**
收益期望	9328.75	9328.75	9266.875	9266.875
路径	**5**	**6**	**7**	**8**
收益期望	9253.125	9253.125	9213.75	9213.75

由于俩名玩家是等价的，因此另一名玩家的收益期望也为该组表格，故我们可求得该组博弈的纳什均衡组合有四个，分别为玩家 *a* 走路径 1，玩家 *b* 走路径 1；玩家 *a* 走路径 1，玩家 *b* 走路径 2；玩家 *a* 走路径 2，玩家 *b* 走路径 1；玩家 *a* 走路径 2，玩家 *b* 走路径 2。

再运用混合策略，即玩家的最优策略应为，50%概率走路径 1，50%概率走路径 2。

此时玩家应在起点处购买 45 箱水，57 箱食物，期望收益为 9260 元。

5.3.2 问题三第六关的分析与求解

5.3.2.1 模型的建立

5.3.2.1.1 三人博弈模型

多名玩家同时穿越沙漠，本质上可以抽象为多智能体博弈模型。可以将情景抽象为$M = \{P, S, A, T_r, T_e, U\}$,其中 P 代表参与游戏的三名玩家的集合，可以不失一般性的定义为$\{MIN, MAX\}$，定义MAX为采取本文策略的玩家，MIN为其他两位玩家；$A\ (p, s)$代表对于P中元素在 S 下的可行方案集合。$Tr(s, a)$: 表示智能体在进行动作后状态的转移。$Te(s)$为布尔值，判断在 *s* 条件下模型是否结束运行。*U(s)*表示 *s* 的效用值，不失一般性地可以定义为$U(s)$越高则对MAX玩家最有利。[3]

玩家仅知道当天天气状况，且在当天行动后均知道其余玩家当天的行动方案和剩余资源，因此博弈问题从完全信息博弈变为信息对称的非完全信息博弈。且不同于一般的博弈模型存在先手、后手，本模型中各玩家同时决策行动。

5.3.2.1.2 蒙特卡洛树建立

由于玩家需要根据他人的位置和方案进行决策，因此本题适用于蒙特卡洛树模型，构造三人博弈树。由于天数限制为 30 天，故通常的决策树不同，此三人博弈树的最大深度为 30。

根据本题特殊性，不存在先手、后手，因此玩家的最佳决策，是在每天行动后更新决策树，根据游戏先验知识的评估函数进行剪枝，之后根据玩家的策略目标进行选择。如在另一名玩家金钱丰富，水和食物不足，下一步将到达村庄时，可以判断他将进入村庄交易，由先验知识：双人同时在村庄交易时价格变为四倍，同时根据游戏中对玩家性格的判断，进而决定是否要减除在村庄交易的决策树分支。

5.3.2.1.3 模型求解

由于树的深度较大，因此有必要采取剪枝以减少每次更新树的时间代价，但对于深度较高的树，传统 *Alpha-Beta* 剪枝已无法显著降低运算规模。因此在蒙特卡洛树进行剪枝时，需要良好的效用函数和先验知识进行指导。[4]

本文基于仿真程序总结了先验知识，且在仿真过程中发现在玩家采取不同的性格策略时，效用函数和先验知识并不相同，以最保守的游戏者为例，剪枝算法的先验规则有：

1、 不与其他玩家同时在矿山挖矿

2、 除非物资难以维持两次沙暴日，不与其他玩家同时在村庄交易

3、 在后 15 天不前往与终点的城市距离大于 10 的节点区域

基于先验知识，保守玩家最倾向于沿地图边缘前往终点，并避免采矿和交易。

对于最贪心的游戏者而言，剪枝策略则不同：

1、 不考虑沙暴概率高于 20%的情况

2、 在物资足够 4 步生存时，尽量多的选择挖矿

对于玩家决策的普适性剪枝策略，则是在 4 步内均有明显亏损的节点即可剪枝，因亏损已难以补足。

5.3.2.1.4 数据仿真

基于蒙特卡洛树模型，本文随机生成一套天气，模拟三名性格不同的玩家，以验证游戏策略正确性，为保证仿真的有效性，三名玩家的性格各不相同。

玩家 1 的性格贪心，他会乐观地选择当前最优地策略，而在极大概率上忽略其他玩家不同决策而带来地悲观情况，做出最高收益地决定。本文如此设计玩家的性格参数，是为了尽量避免囚徒困境带来的全局劣解，使得总有一个玩家能勇敢地做出使得收益能够尽量最大化的高风险决定。[5]

玩家 2 的性格保守，他会悲观地估计其它玩家地选择，从而在极大概率上做出风险最低地较优决定。本文这样设计的原因是为了与玩家 1 的性格进行明显的区分，体现性格参数对游戏中决定的影响，同时这样一个角色也能够降低玩家 1 进行高收益决策时的风险，使得仿真的答案能够逼近全局最优的结果。

玩家 3 的性格参数设定在玩家 1、玩家 2 之间，他做出极高风险与极低风险的决策的概率都不大，而是对风险较低而收益较高的决策较为青睐。这样一个玩家则能够较为真实地反映一般玩家可能的选择，也使得玩家群体性格的分布较为均匀，提供了大量玩家 1 与玩家 2 不会做出的选择以供后续的参考研究，也使得本文的答案能够逼近全局最优的结果。

5.3.2.1.5 仿真结果分析

本文对三人的决策进行仿真，三人最终的收益总和为 30437.5 元，仿真结果细节保存在附件“问题六的数值模拟.xlsx”中。

在仿真中，玩家 2 由于悲观地估计将来，因此选择了在起点等待一天后直接前往终点，这样可以避免与其他玩家共同行动对水的消耗大，同时也避免了一起交易导致的价格上涨。玩家 2 最终的收益为 7795 元。

贪心的玩家 1 和较为中立理智的玩家 3 在游戏进行中轮流挖矿和购买物资，最终玩家 1 获得了 11882.5 元的收益，玩家 3 获得了 10760 元的收益，均高于玩家 2 采取保守策略所得。但值得指出玩家 1 和玩家 3 并不常常可以坚持到游戏结束，如在前往终点的路上多次遇到沙暴会导致游戏失败。

玩家 3 的收益 10760 元更为接近理智决策下的期望收益，在其他 100 局仿真中，玩家 3 的成功率为 92%，收益期望值为 9899.2 元。

六、模型的评价

6.1 模型的优点

本文针对问题 1 的模型先对地图做了简化，再通过动态规算法和智能剪枝算法在能求解出全局最优解的情况下大大减小了算法复杂度，并可以运用于任意一张这类地图，具有普适性，可移植性强。

针对问题 2 的模型则运用贝叶斯算法将天气的随机性合理地量化，且结果表明模型对天气概率不是过于敏感，体现了模型的稳定性；惩罚项的存在让过于贪心的策略被淘汰，保证了最优策略的高成功率，体现了模型的完备性。

针对第五关的模型通过路径筛选大大减少了运算量，且运用混合策略可显著提高收益的期望。

针对第六关的模型考虑了多种决策性格，可以模拟出大部分玩家的决策情况，策略带来的收益稳定，且对参数的敏感度不高，具有稳定性的优点。

6.2 模型的缺点

针对第五关的模型有俩个强假设，使得该模型的运用具有一定狭隘性。

针对第五关的模型分析了不同性格玩家采取的策略，但在开放式对抗时，难以确定一条绝对正确的策略，玩家可能需要在不同的性格中切换以获得最大收益。

虽然本文模型已经采取了大量的剪枝判断，但由于树的深度较高，因此程序仍然对空间消耗大，具有一定的算法复杂度。

6.3 模型的改进

由 6.2 提到的模型的缺点，可以对针对第五关的模型进行如下改进:将玩家选择路径的概率为等概率这个假设舍去，构造玩家选择路径的概率分布，即运用贝叶斯纳什平衡代替纳什平衡，以求取更优的策略，但模型复杂度会相应提高。

七、参考文献

[1]]Elsadany, A. A. & Awad, A. M. (2016), 'Dynamical analysis of a delayed monopoly game with a log-concave demand function.', Oper. Res. Lett. 44 (1), 33-38.

[2]李保明. 效用、风险与纳什均衡选择[D].山东大学,2000

[3]许霄. 面向CGF战术决策的蒙特卡洛树搜索方法研究[D].国防科技大学,2018

[4] Nijssen, J. (Pim) A. M. and Winands, Mark H. M.. "Monte Carlo Tree Search for the Hide-and-Seek Game Scotland Yard.." IEEE Trans. Comput. Intellig. and AI in Games 4 , no. 4 (2012): 282-294.

[5] 马志琴. 基于囚徒困境的空间演化博弈机制研究[D].天津理工大学,2013.

附录

第一问 Result：

第一关				
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	26	5780	146	309
3	23	5780	136	295
4	23	5780	126	285
5	21	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
25				
26				
27				
28				
29				
30				

第二关				
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	11	5300	88	367
4	11	5300	78	357
5	20	5300	68	343
6	28	5300	52	331
7	28	5300	42	321
8	29	5300	32	307
9	30	5300	16	295
10	39	3190	211	283
11	39	3020	218	273
12	30	3020	202	261
13	30	4020	187	240
14	30	5020	163	222
15	30	6020	139	204
16	30	7020	115	186
17	30	8020	85	156
18	30	9020	55	126
19	39	5730	196	200
20	47	5730	180	188
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	63	12730	16	12
30	64	12730	0	0

问题一C++代码

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

int n,m,n2=0;
int a[20][20],mymap[100],name[50];

int kind[50],begin,end;

int max_weight,money0,max_day,money_gain;
```

```
int water_weight,water_price;
int food_weight,food_price;
int weather_cost[5][5];
int weather[35];

int num_vi,num_mi;

#define foodmax ((int)(max_weight/food_weight))
#define watermax ((int)(max_weight/water_weight))
#define maxnum (1000000000)

int f[32][20][402][602],ans=-maxnum,ansx,ansy,ansz,ansmoney,max_temp2[602];

ifstream file("data2.txt");
ofstream file2("ans2.txt");

int main(){
	file>>max_weight>>money0>>max_day>>money_gain;
	file>>water_weight>>water_price;
	file>>food_weight>>food_price;

	file>>weather_cost[0][0]>>weather_cost[0][1];
	file>>weather_cost[1][0]>>weather_cost[1][1];
	file>>weather_cost[2][0]>>weather_cost[2][1];

	file>>n>>m;
	for (int i=1;i<=m;i++){
		int xx,yy;
		file>>xx>>yy;

		if(mymap[xx]==0){
			n2++;
			mymap[xx]=n2;
			name[n2]=xx;
			xx=n2;
		}
		else xx=mymap[xx];

		if(mymap[yy]==0){
			n2++;
			mymap[yy]=n2;
			name[n2]=yy;
			yy=n2;
		}
```

```
            else yy=mymap[yy];

            a[xx][yy]=a[yy][xx]=1;
        }

        file>>begin>>end;
        begin=mymap[begin];
        end=mymap[end];

        file>>num_vi;
        for(int i=1;i<=num_vi;i++){
            int xx;
            file>>xx;
            kind[mymap[xx]]=1;
        }

        file>>num_mi;
        for(int i=1;i<=num_mi;i++){
            int xx;
            file>>xx;
            kind[mymap[xx]]=2;
        }

        for(int i=1;i<=max_day;i++){
            file>>weather[i];
        }

        for(int day=max_day;day>=0;day--){
            for(int place=1;place<=n;place++){
                for(int i=0;i<=foodmax;i++)max_temp2[i]=-maxnum;
                int max_temp=-maxnum;
                for(int water=watermax;water>=0;water--){
                    max_temp=max_temp2[foodmax];
                    max_temp-=water_price*4;
                    for(int food=foodmax;food>=0;food--){
                        max_temp-=food_price*4;
                        max_temp2[food]-=water_price*4;
                        max_temp=max(max_temp,max_temp2[food]);
                        max_temp=max(max_temp,-maxnum);
                        if(day==max_day&&place!=end){f[day][place][water][food]=-
maxnum;continue;}

        if(food*food_weight+water*water_weight>max_weight){f[day][place][water][food]=-
maxnum;continue;}
```

```
                    if(day==max_day){
                        f[day][place][water][food]=water*water_price+food*food_price;
                        continue;
                    }

                    f[day][place][water][food]=-maxnum;
                    if(day==0&&money0-water*water_price-food*food_price<0)continue;
                    if(day==0&&place!=begin)continue;


    if(place==end)f[day][place][water][food]=water*water_price+food*food_price;

                    if((water<weather_cost[weather[day+1]][0] ||
food<weather_cost[weather[day+1]][1])&&kind[place]!=1)continue;
                    if(kind[place]==1){
                        f[day][place][water][food]=max(max_temp,f[day][place][water][food]);
                        if(water<weather_cost[weather[day+1]][0] ||
food<weather_cost[weather[day+1]][1])continue;
                    }

                    if(kind[place]==2 && water>=3*weather_cost[weather[day+1]][0] &&
food>=3*weather_cost[weather[day+1]][1]){
                        f[day][place][water][food]=max(f[day][place][water][food],
                        f[day+1][place][water-3*weather_cost[weather[day+1]][0]][food-
3*weather_cost[weather[day+1]][1]]+2*money_gain);
                    }

                    if(weather[day+1]==2){
                        f[day][place][water][food]=max(f[day][place][water][food],
                        f[day+1][place][water-weather_cost[2][0]][food-weather_cost[2][1]]);
                    }
                    else{
                        for(int place2=1;place2<=n;place2++){
                            if(place2==place)
                                f[day][place][water][food]=max(f[day][place][water][food],
                                f[day+1][place2][water-
weather_cost[weather[day+1]][0]][food-weather_cost[weather[day+1]][1]]);
                            else if(a[place][place2]==1 &&
                            water>=2*weather_cost[weather[day+1]][0] &&
food>=2*weather_cost[weather[day+1]][1])
                                f[day][place][water][food]=max(f[day][place][water][food],
                                f[day+1][place2][water-
2*weather_cost[weather[day+1]][0]][food-2*weather_cost[weather[day+1]][1]]);
                        }
```

```
                    }


    if(kind[place]==1&&f[day][place][water][food]>max_temp)max_temp=f[day][place][water][food];
                    if(kind[place]==1)max_temp2[food]=max_temp;
                    if(day==0&&f[day][place][water][food]>-maxnum/2){
                        f[day][place][water][food]+=2*(money0-water*water_price-
food*food_price);
                        if(f[day][place][water][food]>ans){
                            ans=f[day][place][water][food];
                            ansx=place;
                            ansy=water;
                            ansz=food;
                        }
                    }
                }
            }
        }
    }
    file2<<ans/2<<endl;
    ansmoney=money0-ansy*water_price-ansz*food_price;
    ans-=ansmoney*2;
    file2<<"day=0\tplace="<<name[ansx]<<"\twater="<<ansy<<"\tfood="<<ansz<<"\tmoney="<<ans
money<<endl;

    for(int day=1;day<=max_day;day++){
        for(int place=1;place<=n;place++){
            if(place==ansx && ansy>=weather_cost[weather[day]][0] &&
ansz>=weather_cost[weather[day]][1]){
                if(f[day][place][ansy-weather_cost[weather[day]][0]][ansz-
weather_cost[weather[day]][1]]==ans){
                    ansx=place;
                    ansy=ansy-weather_cost[weather[day]][0];
                    ansz=ansz-weather_cost[weather[day]][1];
                    ans=f[day][ansx][ansy][ansz];

    file2<<"day="<<day<<"\tplace="<<name[ansx]<<"\twater="<<ansy<<"\tfood="<<ansz<<"\tmone
y="<<ansmoney<<endl;
                    if(kind[place]!=1)goto xxx;
                }
                else if(kind[place]==2 && ansy>=3*weather_cost[weather[day]][0] &&
ansz>=3*weather_cost[weather[day]][1] &&
                f[day][place][ansy-3*weather_cost[weather[day]][0]][ansz-
3*weather_cost[weather[day]][1]]+money_gain*2==ans){
```

```
                    ansx=place;
                    ansy=ansy-3*weather_cost[weather[day]][0];
                    ansz=ansz-3*weather_cost[weather[day]][1];
                    ans=f[day][ansx][ansy][ansz];
                    ansmoney+=money_gain;

    file2<<"day="<<day<<"\tplace="<<name[ansx]<<"\twater="<<ansy<<"\tfood="<<ansz<<"\tmone
y="<<ansmoney<<endl;
                    if(kind[place]!=1)goto xxx;
                }
            }
            if(a[place][ansx]==1 && ansy>=2*weather_cost[weather[day]][0] &&
ansz>=2*weather_cost[weather[day]][1]){
                if(f[day][place][ansy-2*weather_cost[weather[day]][0]][ansz-
2*weather_cost[weather[day]][1]]==ans){
                    ansx=place;
                    ansy=ansy-2*weather_cost[weather[day]][0];
                    ansz=ansz-2*weather_cost[weather[day]][1];
                    ans=f[day][ansx][ansy][ansz];

    file2<<"day="<<day<<"\tplace="<<name[ansx]<<"\twater="<<ansy<<"\tfood="<<ansz<<"\tmone
y="<<ansmoney<<endl;
                    if(kind[place]!=1)goto xxx;
                }
            }
            if(kind[place]==1&&ansx==place){
                for(int water=watermax;water>=ansy;water--){
                    for(int food=foodmax;food>=ansz;food--){
                        if(food==ansz&&water==ansy)continue;
                        if(food*food_weight+water*water_weight>max_weight)continue;
                        if(f[day][ansx][water][food]-(water-ansy)*water_price*4-(food-
ansz)*food_price*4==ans){
                            ansmoney-=(water-ansy)*water_price*2+(food-
ansz)*food_price*2;
                            ansx=place;
                            ansy=water;
                            ansz=food;
                            ans=f[day][ansx][ansy][ansz];

    file2<<"day="<<day<<"\tplace="<<name[ansx]<<"\twater="<<ansy<<"\tfood="<<ansz<<"\tmone
y="<<ansmoney<<endl;
                            goto xxx;
                        }
```

```
                        }
                    }
                }
            }
xxx:;
    }
    ansmoney+=ansy*water_price/2+ansz*food_price/2;
    file2<<ansmoney;
}
```

问题二_求解期望与模拟（c++）：

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

int n,m,n2=0;
int a[20][20],mymap[100],name[50];

int kind[50],begin,end;

int max_weight,money0,max_day,money_gain;
int water_weight,water_price;
int food_weight,food_price;
int weather_cost[5][5];
double weather_per[5];

int num_vi,num_mi;

#define foodmax ((int)(max_weight/food_weight))
#define watermax ((int)(max_weight/water_weight))
#define maxnum (100000.0)
#define round 1000

double random(double l,double h)
{
    return l+(h-l)*(double)rand()/(double)(RAND_MAX);
}

double f[32][12][402][602],ans=-maxnum,ansmoney,max_temp2[602][3];

int ansx,ansy,ansz;

ifstream file("data4.txt");
```

```
ofstream file2("ans4_3.txt");

int main(){
	file>>max_weight>>money0>>max_day>>money_gain;
	file>>water_weight>>water_price;
	file>>food_weight>>food_price;

	file>>weather_cost[0][0]>>weather_cost[0][1];
	file>>weather_cost[1][0]>>weather_cost[1][1];
	file>>weather_cost[2][0]>>weather_cost[2][1];

	file>>n>>m;
	for (int i=1;i<=m;i++){
		int xx,yy;
		file>>xx>>yy;

		if(mymap[xx]==0){
			n2++;
			mymap[xx]=n2;
			name[n2]=xx;
			xx=n2;
		}
		else xx=mymap[xx];

		if(mymap[yy]==0){
			n2++;
			mymap[yy]=n2;
			name[n2]=yy;
			yy=n2;
		}
		else yy=mymap[yy];

		a[xx][yy]=a[yy][xx]=1;
	}

	file>>begin>>end;
	begin=mymap[begin];
	end=mymap[end];

	file>>num_vi;
	for(int i=1;i<=num_vi;i++){
		int xx;
		file>>xx;
		kind[mymap[xx]]=1;
```

```
	}

	file>>num_mi;
	for(int i=1;i<=num_mi;i++){
		int xx;
		file>>xx;
	//	kind[mymap[xx]]=2;
	}

	file>>weather_per[0]>>weather_per[1]>>weather_per[2];

	for(int day=max_day;day>=0;day--){
		for(int place=1;place<=n;place++){
			for(int i=0;i<=foodmax;i++)max_temp2[i][0]=max_temp2[i][1]=max_temp2[i][2]=-
maxnum;
			double max_temp[3]={-maxnum,-maxnum,-maxnum};
			for(int water=watermax;water>=0;water--){
				max_temp[0]=max_temp2[foodmax][0];
				max_temp[0]-=water_price*4;
				max_temp[1]=max_temp2[foodmax][1];
				max_temp[1]-=water_price*4;
				max_temp[2]=max_temp2[foodmax][2];
				max_temp[2]-=water_price*4;

				for(int food=foodmax;food>=0;food--){

					max_temp[0]-=food_price*4;
					max_temp2[food][0]-=water_price*4;
					max_temp[0]=max(max_temp[0],max_temp2[food][0]);
					max_temp[0]=max(max_temp[0],-maxnum);

					max_temp[1]-=food_price*4;
					max_temp2[food][1]-=water_price*4;
					max_temp[1]=max(max_temp[1],max_temp2[food][1]);
					max_temp[1]=max(max_temp[1],-maxnum);

					max_temp[2]-=food_price*4;
					max_temp2[food][2]-=water_price*4;
					max_temp[2]=max(max_temp[2],max_temp2[food][2]);
					max_temp[2]=max(max_temp[2],-maxnum);

					if(day==max_day&&place!=end){f[day][place][water][food]=-
maxnum;continue;}
```

```
if(food*food_weight+water*water_weight>max_weight){f[day][place][water][food]=-
maxnum;continue;}
                    if(day==max_day){

f[day][place][water][food]=(double)(water*water_price+food*food_price);
                        continue;
                    }

                    f[day][place][water][food]=-maxnum;
                    if(day==0&&money0-water*water_price-food*food_price<0)continue;
                    if(day==0&&place!=begin)continue;


if(place==end)f[day][place][water][food]=(double)(water*water_price+food*food_price);


                {
                    double f_temp[3];
                    for(int today_weather=0;today_weather<=2;today_weather++){

                        f_temp[today_weather]=f[day][place][water][food];

                        if((water<weather_cost[today_weather][0]                    ||
food<weather_cost[today_weather][1])&&kind[place]!=1)continue;

                        if(kind[place]==1){

f_temp[today_weather]=max(max_temp[today_weather],f_temp[today_weather]);
                            if(water<weather_cost[today_weather][0]                    ||
food<weather_cost[today_weather][1])continue;
                        }

                        if(kind[place]==2 && water>=3*weather_cost[today_weather][0] &&
food>=3*weather_cost[today_weather][1]){
                            f_temp[today_weather]=max(f_temp[today_weather],
                            f[day+1][place][water-3*weather_cost[today_weather][0]][food-
3*weather_cost[today_weather][1]]+(double)(2*money_gain));
                        }

                        if(today_weather==2){
                            f_temp[today_weather]=max(f_temp[today_weather],
                            f[day+1][place][water-weather_cost[2][0]][food-
weather_cost[2][1]]);
                        }
```

```
                    else{
                        for(int place2=1;place2<=n;place2++){
                            if(place2==place)
                                f_temp[today_weather]=max(f_temp[today_weather],
                                f[day+1][place2][water-
weather_cost[today_weather][0]][food-weather_cost[today_weather][1]]);
                            else if(a[place][place2]==1 &&
                            water>=2*weather_cost[today_weather][0]          &&
food>=2*weather_cost[today_weather][1])
                                f_temp[today_weather]=max(f_temp[today_weather],
                                f[day+1][place2][water-
2*weather_cost[today_weather][0]][food-2*weather_cost[today_weather][1]]);
                        }
                    }

    if(kind[place]==1&&f_temp[today_weather]>max_temp[today_weather])max_temp[today_weat
her]=f_temp[today_weather];

    if(kind[place]==1)max_temp2[food][today_weather]=max_temp[today_weather];
                }

                  if(f_temp[0]>=-maxnum/1.2){

    f[day][place][water][food=f_temp[0]*weather_per[0]+f_temp[1]*weather_per[1]+f_temp[2]*w
eather_per[2];
                }
                else
                    f[day][place][water][food]=-maxnum;
            }

                if(day==0&&f[day][place][water][food]>-maxnum/1.2){
                    f[day][place][water][food]+=(double)(2*(money0-water*water_price-
food*food_price));
                    if(f[day][place][water][food]>ans){
                        ans=f[day][place][water][food];
                        ansx=place;
                        ansy=water;
                        ansz=food;
                    }
                }
            }
        }
    }
```

```
	file2<<ans/2.0<<endl;
	ansmoney=money0-ansy*water_price-ansz*food_price;
	ans-=ansmoney*2;
	file2<<"day=0\tweather=--
\tplace="<<name[ansx]<<"\twater="<<ansy<<"\tfood="<<ansz<<"\tmoney="<<ansmoney<<endl<<en
dl;

	srand((unsigned)time(0));
	int suss=round;
	for(int i=1;i<=round;i++){
		int place=ansx,water=ansy,food=ansz;
		double money=ansmoney;

		for(int j=1;j<=max_day;j++){
			double temp=random(0.0,1.0);
			int today_weather;
			if(temp<=weather_per[0])today_weather=0;
			else if(temp<=weather_per[0]+weather_per[1])today_weather=1;
			else today_weather=2;

			int temp_x=-1,temp_y=water,temp_z=food;
			double temp_max=-maxnum/1.2,temp_m=money;

			if(kind[place]==2 && water>=3*weather_cost[today_weather][0] &&
food>=3*weather_cost[today_weather][1]&&
			f[j][place][water-3*weather_cost[today_weather][0]][food-
3*weather_cost[today_weather][1]]+(double)(2*money_gain)>temp_max){
				temp_max=f[j][place][water-3*weather_cost[today_weather][0]][food-
3*weather_cost[today_weather][1]]+(double)(2*money_gain);
				temp_x=place;
				temp_y=water-3*weather_cost[today_weather][0];
				temp_z=food-3*weather_cost[today_weather][1];
				temp_m=money+(double)(money_gain);
			}

			if(water>=weather_cost[today_weather][0] &&
food>=weather_cost[today_weather][1] &&
			f[j][place][water-weather_cost[today_weather][0]][food-
weather_cost[today_weather][1]]>temp_max){
				temp_max=f[j][place][water-weather_cost[today_weather][0]][food-
weather_cost[today_weather][1]];
				temp_x=place;
				temp_y=water-weather_cost[today_weather][0];
				temp_z=food-weather_cost[today_weather][1];
```

```
                temp_m=money;
            }

            if(today_weather==2)goto xxx;

            if(kind[place]==1){
                for(int place2=1;place2<=n;place2++){
                    if(a[place2][place]==1){
                        for(int
k1=max(water,2*weather_cost[today_weather][0]);k1<=watermax;k1++){
                            for(int
k2=max(food,2*weather_cost[today_weather][1]);k2*food_weight+k1*water_weight<=max_weight;k
2++){
                                if(f[j][place2][k1-2*weather_cost[today_weather][0]][k2-
2*weather_cost[today_weather][1]]
                                -(k1-water)*4*water_price-(k2-
food)*4*food_price>temp_max){
                                    temp_max=f[j][place2][k1-
2*weather_cost[today_weather][0]][k2-2*weather_cost[today_weather][1]]
                                    -(k1-water)*4*water_price-(k2-food)*4*food_price;
                                    temp_x=place2;
                                    temp_y=k1-2*weather_cost[today_weather][0];
                                    temp_z=k2-2*weather_cost[today_weather][1];
                                    temp_m=money-(k1-water)*2*water_price-(k2-
food)*2*food_price;
                                }
                            }
                        }
                    }
                }
                goto xxx;
            }

            if(water>=2*weather_cost[today_weather][0]                    &&
food>=2*weather_cost[today_weather][1]){
                for(int place2=1;place2<=n;place2++){
                    if(a[place2][place]==1 &&
                    f[j][place2][water-2*weather_cost[today_weather][0]][food-
2*weather_cost[today_weather][1]]>temp_max){
                        temp_max=f[j][place2][water-
2*weather_cost[today_weather][0]][food-2*weather_cost[today_weather][1]];
                        temp_x=place2;
                        temp_y=water-2*weather_cost[today_weather][0];
                        temp_z=food-2*weather_cost[today_weather][1];
```

```
                                        temp_m=money;
                                    }
                                }
                    }

xxx:        if(temp_x==-1){

                        file2<<"day="<<j<<"\tweather="<<today_weather<<"\t";
                        file2<<"游戏失败"<<endl<<endl;
                        suss--;
                        break;
                    }
                    else{
                        place=temp_x;
                        water=temp_y;
                        food=temp_z;
                        money=temp_m;

    file2<<"day="<<j<<"\tweather="<<today_weather<<"\tplace="<<name[place]<<"\twater="<<wat
er<<"\tfood="<<food<<"\tmoney="<<money<<endl;
                    }
                    if(place==end){
                        money+=food*food_price/2.0+water*water_price/2.0;
                        file2<<"money="<<money<<endl;
                        break;
                    }
                }
                file2<<endl;
            }
            file2<<endl;
            file2<<"success_rate="<<(double)suss/(double)round<<endl;
}
```

问题二_观察惩罚值与沙暴概率对结果的影响

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

int n,m,n2=0;
int a[20][20],mymap[100],name[50];

int kind[50],begin,end;

int max_weight,money0,max_day,money_gain;
int water_weight,water_price;
```

```
int food_weight,food_price;
int weather_cost[5][5];
double weather_per[5];

int num_vi,num_mi;

#define foodmax ((int)(max_weight/food_weight))
#define watermax ((int)(max_weight/water_weight))
#define round 1000

double random(double l,double h)
{
	return l+(h-l)*(double)rand()/(double)(RAND_MAX);
}

double f[32][12][402][602],ans,ansmoney,max_temp2[602][3];

int ansx,ansy,ansz;

ifstream file("data4.txt");
ofstream file2("ans4_2_沙暴概率 0.05.txt");

int main(){
	file>>max_weight>>money0>>max_day>>money_gain;
	file>>water_weight>>water_price;
	file>>food_weight>>food_price;

	file>>weather_cost[0][0]>>weather_cost[0][1];
	file>>weather_cost[1][0]>>weather_cost[1][1];
	file>>weather_cost[2][0]>>weather_cost[2][1];

	file>>n>>m;
	for (int i=1;i<=m;i++){
		int xx,yy;
		file>>xx>>yy;

		if(mymap[xx]==0){
			n2++;
			mymap[xx]=n2;
			name[n2]=xx;
			xx=n2;
		}
		else xx=mymap[xx];
```

```
            if(mymap[yy]==0){
                n2++;
                mymap[yy]=n2;
                name[n2]=yy;
                yy=n2;
            }
            else yy=mymap[yy];

            a[xx][yy]=a[yy][xx]=1;
        }

        file>>begin>>end;
        begin=mymap[begin];
        end=mymap[end];

        file>>num_vi;
        for(int i=1;i<=num_vi;i++){
            int xx;
            file>>xx;
            kind[mymap[xx]]=1;
        }

        file>>num_mi;
        for(int i=1;i<=num_mi;i++){
            int xx;
            file>>xx;
            kind[mymap[xx]]=2;
        }

        file>>weather_per[0]>>weather_per[1]>>weather_per[2];

        for(double maxnum=10.0;maxnum<=10000000.0;maxnum*=10){

        ans=-maxnum;


        for(int day=max_day;day>=0;day--){
            for(int place=1;place<=n;place++){
                for(int i=0;i<=foodmax;i++)max_temp2[i][0]=max_temp2[i][1]=max_temp2[i][2]=-
maxnum;
                double max_temp[3]={-maxnum,-maxnum,-maxnum};
                for(int water=watermax;water>=0;water--){
                    max_temp[0]=max_temp2[foodmax][0];
                    max_temp[0]-=water_price*4;
```

```
                    max_temp[1]=max_temp2[foodmax][1];
                    max_temp[1]-=water_price*4;
                    max_temp[2]=max_temp2[foodmax][2];
                    max_temp[2]-=water_price*4;

                    for(int food=foodmax;food>=0;food--){

                        max_temp[0]-=food_price*4;
                        max_temp2[food][0]-=water_price*4;
                        max_temp[0]=max(max_temp[0],max_temp2[food][0]);
                        max_temp[0]=max(max_temp[0],-maxnum);

                        max_temp[1]-=food_price*4;
                        max_temp2[food][1]-=water_price*4;
                        max_temp[1]=max(max_temp[1],max_temp2[food][1]);
                        max_temp[1]=max(max_temp[1],-maxnum);

                        max_temp[2]-=food_price*4;
                        max_temp2[food][2]-=water_price*4;
                        max_temp[2]=max(max_temp[2],max_temp2[food][2]);
                        max_temp[2]=max(max_temp[2],-maxnum);

                        if(day==max_day&&place!=end){f[day][place][water][food]=-
maxnum;continue;}

    if(food*food_weight+water*water_weight>max_weight){f[day][place][water][food]=-
maxnum;continue;}
                        if(day==max_day){

    f[day][place][water][food]=(double)(water*water_price+food*food_price);
                            continue;
                        }

                        f[day][place][water][food]=-maxnum;
                        if(day==0&&money0-water*water_price-food*food_price<0)continue;
                        if(day==0&&place!=begin)continue;


    if(place==end)f[day][place][water][food]=(double)(water*water_price+food*food_price);


                    {
                        double f_temp[3];
                        for(int today_weather=0;today_weather<=2;today_weather++){
```

```
                    f_temp[today_weather]=f[day][place][water][food];

                    if((water<weather_cost[today_weather][0] ||
food<weather_cost[today_weather][1])&&kind[place]!=1)continue;

                    if(kind[place]==1){

    f_temp[today_weather]=max(max_temp[today_weather],f_temp[today_weather]);
                        if(water<weather_cost[today_weather][0] ||
food<weather_cost[today_weather][1])continue;
                    }

                    if(kind[place]==2 && water>=3*weather_cost[today_weather][0] &&
food>=3*weather_cost[today_weather][1]){
                        f_temp[today_weather]=max(f_temp[today_weather],
                        f[day+1][place][water-3*weather_cost[today_weather][0]][food-
3*weather_cost[today_weather][1]]+(double)(2*money_gain));
                    }

                    if(today_weather==2){
                        f_temp[today_weather]=max(f_temp[today_weather],
                        f[day+1][place][water-weather_cost[2][0]][food-
weather_cost[2][1]]);
                    }
                    else{
                        for(int place2=1;place2<=n;place2++){
                            if(place2==place)
                                f_temp[today_weather]=max(f_temp[today_weather],
                                f[day+1][place2][water-
weather_cost[today_weather][0]][food-weather_cost[today_weather][1]]);
                            else if(a[place][place2]==1 &&
                            water>=2*weather_cost[today_weather][0] &&
food>=2*weather_cost[today_weather][1])
                                f_temp[today_weather]=max(f_temp[today_weather],
                                f[day+1][place2][water-
2*weather_cost[today_weather][0]][food-2*weather_cost[today_weather][1]]);
                        }
                    }

    if(kind[place]==1&&f_temp[today_weather]>max_temp[today_weather])max_temp[today_weat
her]=f_temp[today_weather];

    if(kind[place]==1)max_temp2[food][today_weather]=max_temp[today_weather];
```

```
                        }

                          if(f_temp[0]>=-maxnum/1.2){

    f[day][place][water][food]=f_temp[0]*weather_per[0]+f_temp[1]*weather_per[1]+f_temp[2]*w
eather_per[2];
                        }
                        else
                            f[day][place][water][food]=-maxnum;
                    }

                        if(day==0&&f[day][place][water][food]>-maxnum/1.2){
                            f[day][place][water][food]+=(double)(2*(money0-water*water_price-
food*food_price));
                            if(f[day][place][water][food]>ans){
                                ans=f[day][place][water][food];
                                ansx=place;
                                ansy=water;
                                ansz=food;
                            }
                        }
                    }
                }
            }
    }
    //file2<<ans/2.0<<endl;
    ansmoney=money0-ansy*water_price-ansz*food_price;
    //ans-=ansmoney*2;
    //file2<<"day=0\tweather=--
\tplace="<<name[ansx]<<"\twater="<<ansy<<"\tfood="<<ansz<<"\tmoney="<<ansmoney<<endl<<en
dl;

    srand((unsigned)time(0));
    int suss=round,left_water_min=maxnum,left_food_min=maxnum;
    for(int i=1;i<=round;i++){
        int place=ansx,water=ansy,food=ansz;
        double money=ansmoney;

        for(int j=1;j<=max_day;j++){
            double temp=random(0.0,1.0);
            int today_weather;
            if(temp<=weather_per[0])today_weather=0;
            else if(temp<=weather_per[0]+weather_per[1])today_weather=1;
            else today_weather=2;
```

```
int temp_x=-1,temp_y=water,temp_z=food;
double temp_max=-maxnum/1.2,temp_m=money;

if(kind[place]==2 && water>=3*weather_cost[today_weather][0] &&
food>=3*weather_cost[today_weather][1]&&
f[j][place][water-3*weather_cost[today_weather][0]][food-
3*weather_cost[today_weather][1]]+(double)(2*money_gain)>temp_max){
temp_max=f[j][place][water-3*weather_cost[today_weather][0]][food-
3*weather_cost[today_weather][1]]+(double)(2*money_gain);
temp_x=place;
temp_y=water-3*weather_cost[today_weather][0];
temp_z=food-3*weather_cost[today_weather][1];
temp_m=money+(double)(money_gain);
}

if(water>=weather_cost[today_weather][0] &&
food>=weather_cost[today_weather][1] &&
f[j][place][water-weather_cost[today_weather][0]][food-
weather_cost[today_weather][1]]>temp_max){
temp_max=f[j][place][water-weather_cost[today_weather][0]][food-
weather_cost[today_weather][1]];
temp_x=place;
temp_y=water-weather_cost[today_weather][0];
temp_z=food-weather_cost[today_weather][1];
temp_m=money;
}

if(today_weather==2)goto xxx;

if(kind[place]==1){
for(int place2=1;place2<=n;place2++){
if(a[place2][place]==1){
for(int
k1=max(water,2*weather_cost[today_weather][0]);k1<=watermax;k1++){
for(int
k2=max(food,2*weather_cost[today_weather][1]);k2*food_weight+k1*water_weight<=max_weight;k
2++){
if(f[j][place2][k1-2*weather_cost[today_weather][0]][k2-
2*weather_cost[today_weather][1]]
-(k1-water)*4*water_price-(k2-
food)*4*food_price>temp_max){
temp_max=f[j][place2][k1-
2*weather_cost[today_weather][0]][k2-2*weather_cost[today_weather][1]]
```

```
                                        -(k1-water)*4*water_price-(k2-food)*4*food_price;
                                        temp_x=place2;
                                        temp_y=k1-2*weather_cost[today_weather][0];
                                        temp_z=k2-2*weather_cost[today_weather][1];
                                        temp_m=money-(k1-water)*2*water_price-(k2-
food)*2*food_price;
                                    }
                                }
                            }
                        }
                        goto xxx;
                }

                if(water>=2*weather_cost[today_weather][0]                              &&
food>=2*weather_cost[today_weather][1]){
                            for(int place2=1;place2<=n;place2++){
                                if(a[place2][place]==1 &&
                                f[j][place2][water-2*weather_cost[today_weather][0]][food-
2*weather_cost[today_weather][1]]>temp_max){
                                    temp_max=f[j][place2][water-
2*weather_cost[today_weather][0]][food-2*weather_cost[today_weather][1]];
                                    temp_x=place2;
                                    temp_y=water-2*weather_cost[today_weather][0];
                                    temp_z=food-2*weather_cost[today_weather][1];
                                    temp_m=money;
                                }
                            }
                }

xxx:        if(temp_x==-1){

                        //file2<<"day="<<j<<"\tweather="<<today_weather<<"\t";
                        //file2<<"游戏失败"<<endl<<endl;
                        suss--;
                        break;
                }
                else{
                        if(kind[place]==2&&temp_y==water-3*weather_cost[today_weather][0]){
                            left_food_min=min(left_food_min,temp_z);
                            left_water_min=min(left_water_min,temp_y);
                        }
                        place=temp_x;
                        water=temp_y;
```

```
                    food=temp_z;
                    money=temp_m;


    //file2<<"day="<<j<<"\tweather="<<today_weather<<"\tplace="<<name[place]<<"\twater="<<w
ater<<"\tfood="<<food<<"\tmoney="<<money<<endl;
                }
                if(place==end){
                    money+=food*food_price/2.0+water*water_price/2.0;
                    //file2<<"money="<<money<<endl;
                    break;
                }
                if(j==max_day)suss--;
            }
    //     file2<<endl;
    }
    file2<<endl;
    file2<<"success_rate="<<(double)suss/(double)round<<"\tE(money)="<<ans/2.0<<
    "\t                惩                罚                值
="<<maxnum<<"\tsuccess_rate*E(money)="<<(double)suss/(double)round*ans/2.0<<endl;
    //file2<<"day=0\tweather=--
\tplace="<<name[ansx]<<"\twater="<<ansy<<"\tfood="<<ansz<<"\tmoney="<<ansmoney<<endl;
    file2<<"left_water_min="<<left_water_min<<"\tleft_food_min="<<left_food_min<<endl;
    }
}
```

问题三（c++）

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

int begin,end;

int max_weight,money0,max_day,money_gain2;
int water_weight,water_price;
int food_weight,food_price;
int weather_cost[5][5];
int weather[12];
int path[12][10];
int food_cost[12][12],water_cost[12][12],money_cost_min[12][12];
double money_gain[12][12];


#define foodmax ((int)(max_weight/food_weight))
#define watermax ((int)(max_weight/water_weight))
```

```
#define maxnum (10000.0)

double random(double l,double h)
{
	return l+(h-l)*(double)rand()/(double)(RAND_MAX);
}

int ansx,ansy,ansz;

ifstream file("data5.txt");
ofstream file2("ans5.txt");

int main(){
	file>>max_weight>>money0>>max_day>>money_gain2;
	file>>water_weight>>water_price;
	file>>food_weight>>food_price;

	file>>weather_cost[0][0]>>weather_cost[0][1];
	file>>weather_cost[1][0]>>weather_cost[1][1];
	file>>weather_cost[2][0]>>weather_cost[2][1];

	file>>begin>>end;

	for(int i=1;i<=max_day;i++){
		file>>weather[i];
	}

	for(int i=1;i<=10;i++){
		int j=0;
		file>>path[i][j];
		while(path[i][j]!=end){
			j++;
			file>>path[i][j];
		}
	}

	for(int i=1;i<=10;i++)
	    for(int j=1;j<=10;j++)
	        food_cost[i][j]=water_cost[i][j]=0;

	for(int i=1;i<=10;i++){
		for(int j=1;j<=10;j++){
			int k=0;
			while(path[i][k]!=end && k<=max_day){
```

```
                if(path[i][k]==path[i][k+1]){
                    food_cost[i][j]+=weather_cost[weather[k+1]][1];
                    water_cost[i][j]+=weather_cost[weather[k+1]][0];
                }
                else{
                    food_cost[i][j]+=2*weather_cost[weather[k+1]][1];
                    water_cost[i][j]+=2*weather_cost[weather[k+1]][0];
                    if(path[j][k+1]==path[i][k+1]&&path[j][k]==path[i][k]){
                        food_cost[i][j]+=2*weather_cost[weather[k+1]][1];
                        water_cost[i][j]+=2*weather_cost[weather[k+1]][0];
                    }
                }
                k++;
            }
        }
}


file2<<"water_cost"<<endl;
for(int i=1;i<=10;i++){
    for(int j=1;j<=10;j++){
        file2<<water_cost[i][j]<<"\t";
    }
    file2<<endl;
}
file2<<endl;

file2<<"food_cost"<<endl;
for(int i=1;i<=10;i++){
    for(int j=1;j<=10;j++){
        file2<<food_cost[i][j]<<"\t";
    }
    file2<<endl;
}
file2<<endl;

file2<<"money_cost_min"<<endl;
for(int i=1;i<=10;i++){
    for(int j=1;j<=10;j++){
        money_cost_min[i][j]=water_cost[i][j]*water_price+food_cost[i][j]*food_price;
        file2<<money_cost_min[i][j]<<"\t";
    }
    file2<<endl;
}
```

```
        file2<<endl;

        file2<<"money_gain"<<endl;
        for(int i=1;i<=10;i++){
                for(int j=1;j<=10;j++){
                        money_gain[i][j]=(double)(money0-
money_cost_min[i][i])+(double)(money_cost_min[i][i]-money_cost_min[i][j])/2.0;
                        file2<<money_gain[i][j]<<"\t";
                }
                file2<<endl;
        }
        file2<<endl;

        file2<<"water_cost_转置"<<endl;
        for(int i=1;i<=10;i++){
                for(int j=1;j<=10;j++){
                        file2<<water_cost[j][i]<<"\t";
                }
                file2<<endl;
        }
        file2<<endl;

        file2<<"food_cost_转置"<<endl;
        for(int i=1;i<=10;i++){
                for(int j=1;j<=10;j++){
                        file2<<food_cost[j][i]<<"\t";
                }
                file2<<endl;
        }
        file2<<endl;

        file2<<"money_cost_min_转置"<<endl;
        for(int i=1;i<=10;i++){
                for(int j=1;j<=10;j++){
                        file2<<money_cost_min[j][i]<<"\t";
                }
                file2<<endl;
        }
        file2<<endl;

        file2<<"money_gain_转置"<<endl;
        for(int i=1;i<=10;i++){
                for(int j=1;j<=10;j++){
                        file2<<money_gain[j][i]<<"\t";
```

```
            }
            file2<<endl;
        }
        file2<<endl;
}
```