赛区评阅编号（由赛区组委会填写）：

2020 高教社杯全国大学生数学建模竞赛

承　诺　书

我们仔细阅读了《全国大学生数学建模竞赛章程》和《全国大学生数学建模竞赛参赛规则》（2019 年修订稿，以下简称为“竞赛章程和参赛规则”，可从全国大学生数学建模竞赛网站下载）。

我们完全清楚，在竞赛开始后参赛队员不能以任何方式，包括电话、电子邮件、“贴吧”、QQ 群、微信群等，与队外的任何人（包括指导教师）交流、讨论与赛题有关的问题；无论主动参与讨论还是被动接收讨论信息都是严重违反竞赛纪律的行为。

我们完全清楚，抄袭别人的成果是违反竞赛章程和参赛规则的行为；如果引用别人的成果或资料（包括网上资料），必须按照规定的参考文献的表述方式列出，并在正文引用处予以标注。

我们以中国大学生名誉和诚信郑重承诺，严格遵守竞赛章程和参赛规则，以保证竞赛的公正、公平性。如有违反竞赛章程和参赛规则的行为，我们将受到严肃处理。

我们授权全国大学生数学建模竞赛组委会，可将我们的论文以任何形式进行公开展示（包括进行网上公示，在书籍、期刊和其他媒体进行正式或非正式发表等）。

我们参赛选择的题号（从 A/B/C/D/E 中选择一项填写）：B

我们的报名参赛队号（12 位数字全国统一编号）：202017241008

参赛学校（完整的学校全称，不含院系名）：华中科技大学

参赛队员 (打印并签名) ：1. 李欣航

2. 鲁镇仪

3. 蒋瀚锐

指导教师或指导教师组负责人 (打印并签名)：贺云峰

（指导教师签名意味着对参赛队的行为和论文的真实性负责）

日期: 2020 年 09 月 12 日

（请勿改动此页内容和格式。此承诺书打印签名后作为纸质论文的封面，注意电子版论文中不得出现此页。以上内容请仔细核对，如填写错误，论文可能被取消评奖资格。）

赛区评阅编号（由赛区组委会填写）：

2020 高教社杯全国大学生数学建模竞赛

编 号 专 用 页

赛区评阅记录（可供赛区评阅时使用）：

评阅人						
备注						

送全国评阅统一编号（由赛区组委会填写）：

全国评阅随机编号（由全国组委会填写）：

（请勿改动此页内容和格式。此编号专用页仅供赛区和全国评阅使用，参赛队打印后装订到纸质论文的第二页上。注意电子版论文中不得出现此页。）

摘要

对于问题一：

对于问题二：

对于问题三第一小问：

对于问题三第二小问：

关键字： 图论　Dijkstra 算法

一、 问题重述

1.1 问题背景

在日益快节奏的生活与时代的背景下，人们玩游戏的重心也逐渐的从娱乐性质转为竞争性质。对于一款游戏来说，攻略的必要性是有目共睹的，人们越来越离不开攻略了。

对于这样一款益智游戏《穿越沙漠》来说，游戏内容是十分有趣的。玩家在游戏中凭借着一张地图和其他的一些信息，就需要利用自己的智慧来完成一个又一个的选择最终到达游戏的胜利。

1.2 问题的提出

该文章主要是研究在《穿越沙漠》这款游戏内，在不同的六个关卡中，分别提出不同要求，不同情况下的时候，玩家所应该做的最优策略。玩家在游戏开始时拥有一张地图，他的目标是首先在规定时间内到达终点，中途不能够使得水或食物已耗尽，并且需要能够使得最终所保留的资金尽可能多。

游戏本身每天有各种地点和属性，关于自然的属性分别有地点所代表区域，例如村庄，矿山；还有天气，天气分成“晴朗”、“高温”、“沙暴”这三种；关于玩家所操纵人物与人物所携带的资源的属性，分别有基础消耗量，行走消耗量，其他消耗量，水，食物以及在矿山中所能获得的资源等等。

在问题一当中，这是一个单机游戏，并且给出了所有的天气状况，题目要求得到一般化结果，并给出在此情况下求出玩家所应该做出的在第一关与第二关的最优策略。

在问题二当中，假设更为合理，也即玩家只知道当天的天气情况，也更加贴近现实生活。题目要求先求出一般化结果，然后在此条件下得到玩家在第三关与第四关的最优策略。

在问题三当中，游戏从单机变成了联机多人竞技，一共 n 名玩家。若当天任意 $k(2 \leq k \leq n)$ 名玩家均是从 A 走到 B，那么他们每个人消耗量快速增加，变成基础消耗量的 $2k$ 倍。若这 k 名玩家同时在矿山中挖矿，则每日挖矿消耗与单机时不变，仍为基础消耗量的 3 倍，但所获得的收益大幅度减少，仅为基础收益的 $\frac{1}{k}$。若这 k 名玩家同一天在同一村庄中购买物品，则价格再次翻倍，也即变为原来的 4 倍。

在问题三第一小问当中，事先已知每天的天气情况，每人须最初制定好决策，并且策略不可更改，题目要求得到一般化结果，并最终运用在第五关上。

在问题三第二小问当中，玩家需要动态的进行下一时刻的决策方案。要求得到关于此种情况的一般化结果，并最终运用在第六关上。

二、 问题分析

2.1 问题一的分析

问题一

2.2 问题二的分析

问题二

2.3 问题三的分析

在问题三当中，

三、 模型假设

- 认为每个地区作为一个图上的顶点进行简化处理。
- 将两两相邻的地区化作图种对应顶点的边进行简化处理。

四、 符号说明

表 **1**　符号说明

符号	含义	单位
$w_{\max}$	玩家的负载重量 (waight) 上限	kg
m_{init}	玩家在特定关卡的初始资金	元 (￥人民币)
c_w	玩家在特定关卡某一天气下的基础水份 (water) 消耗量	箱
c_f	玩家在特定关卡某一天气下的基础食物 (foods) 消耗量	箱
w	当天的天气 (weather)	无
m	某一特定关卡的基础收益	元 (￥人民币)
m_w	特定关卡中每箱水的基准价格	元 (￥人民币)
m_f	特定关卡中每箱食物的基准价格	元 (￥人民币)
w_w	每箱水的重量	kg
w_f	每箱食物的重量	kg

五、 地图的建模

由于地图当中每个地区的大小跟他们的形状无关，于是可以是做一个顶点，两两相邻的区域是相邻的，可以视作一条边。每天仅能最终从一个顶点移到另一个相邻的顶点上。最终，此地图被化简成为无向图进行处理。

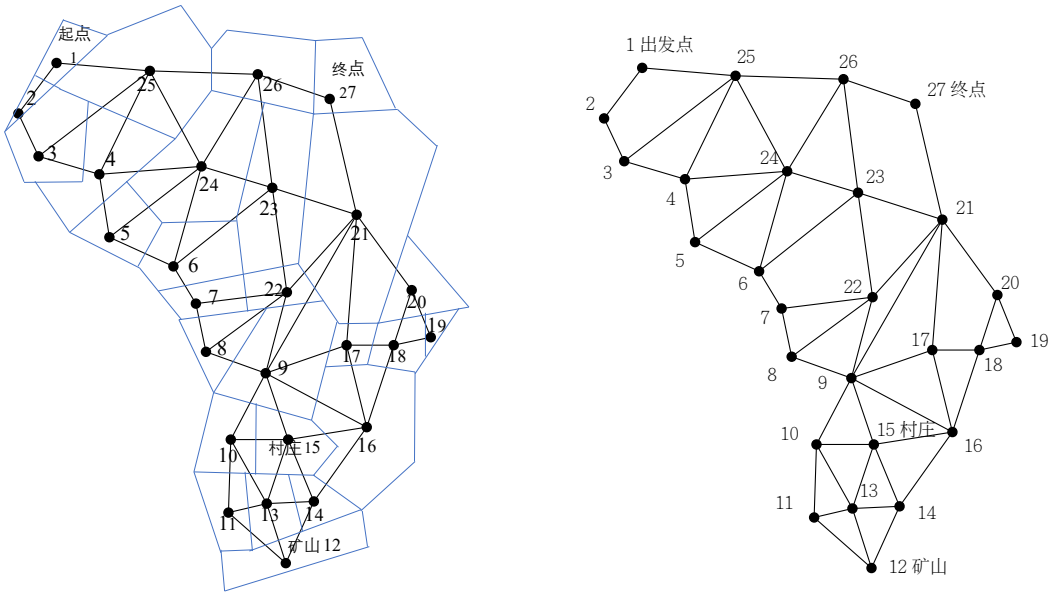

图 **1**　关卡一建模示意图

记这个图为 G，此图 G 当中的所有顶点的集合记为 V，每个地区所对应的顶点分别为 $v_i \quad (1 \leq i \leq |V|)$。图 G 当中的所有边所形成的集合记为 E，每相邻的两个地区所对应形成的边记为 $e_i \quad (1 \leq i \leq |E|)$，并且每个 e_i 对应着两个顶点，即 $e_i = (v_j, v_k)$，其中 (v_j, v_k) 是无序点对。

5.1 单机玩家选择的策略

游戏起点的地方所对应图的点为 v_1，游戏结束的地方所对应图的点为 $v_{|V|}$。下面不妨设 $n = |V|$，也即此点为 v_n。

假设村庄的分布编号为 $\{a_1, a_2, \cdots, a_t\}$，矿山的编号分别为 $\{b_1, b_2, \cdots, b_s\}$，利用 Dijkstra 算法，能够得到每一个矿山分别到起始点和终点的距离 $d(v_1, v_{b_i})$ 和 $d(v_{b_i}, v_n)$。

先采用剪枝策略，对以下几种情况进行剪枝，若对于 (v_1, v_{b_i}) 所形成的长度为 $d(v_1, v_{b_i})$ 的路线中恰好存在村庄在其路中，也即路径为 $v_1 e_1 \cdots v_{a_i} \cdots, v_{b_j}$，则将此枝条保留，其余的情况则不予考虑，进行删除。同理，在形成长度为 $d(v_{b_i}, v_n)$ 的路径 $l = v_{b_j} e_1 \cdots v_{a_i} \cdots v_n$ 当中若存在 $v_{b_i} \in l$，则其余部分为无用节点，即可删除。

例如对与关卡一，我们可以通过上述剪枝最终得到如下的图：

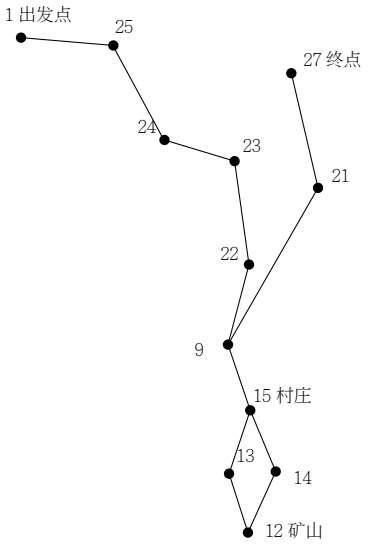

图 **2**　关卡一剪枝后示意图

之后进行深度搜索遍历所有情况即可。

当具体到情况一的时候，跟上面那幅图一致，这是一幅有向图，在之前的剪枝当中同时把原无向图当作每条边都是双向的有向图进行剪枝。最终得到 7 种合乎逻辑的结果，这些情况全部是当你想要去采矿所能采取的在剔除了枝条后的结果。运用编程语言

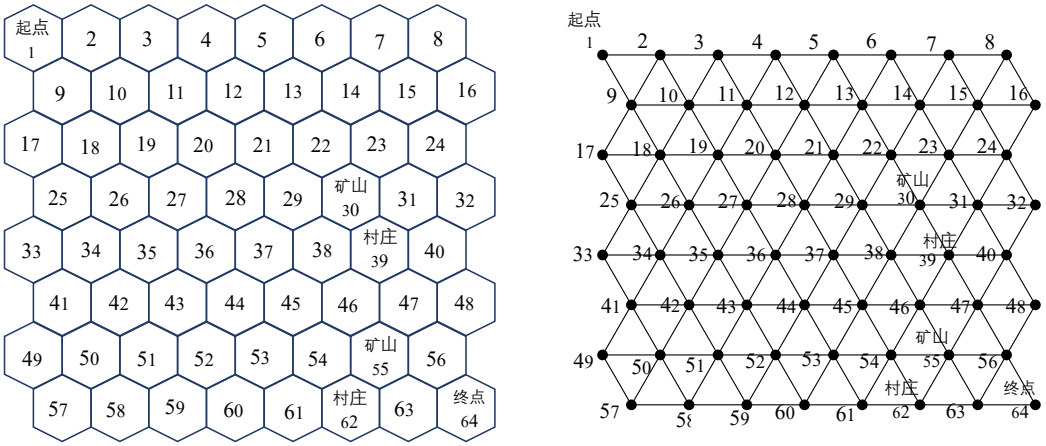

图 **3**　关卡二建模示意图

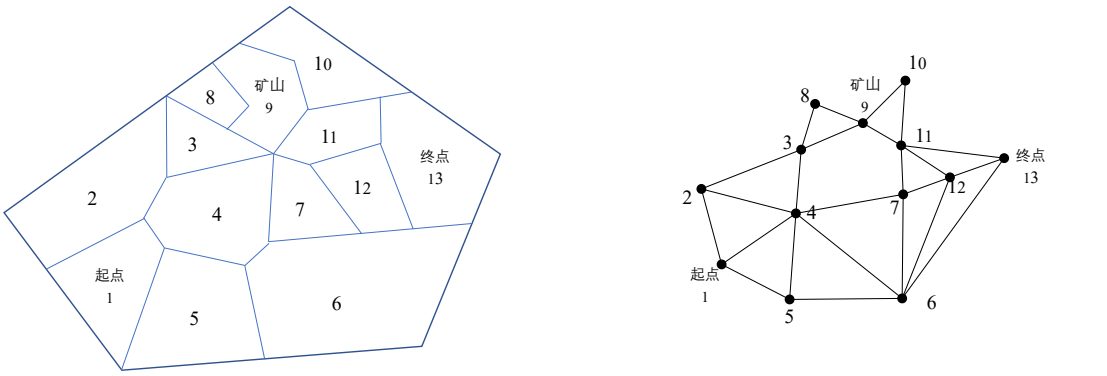

图 **4**　关卡三五建模示意图

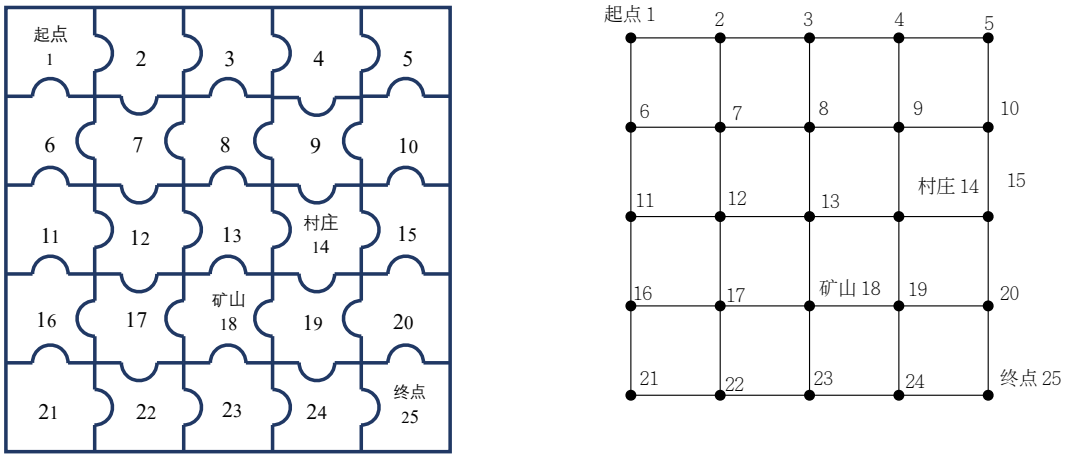

图 **5**　关卡四六建模示意图

六、 模型总结

6.1 模型改进

6.2 模型的优缺点

模型的优点：

1 问题模型

2 问题一模型优点：

3 问题三模型优点：

4 问题四模型优点：

模型的缺点：

1 问题一模型缺点：

2 问题三模型缺点：

3 问题四模型缺点：

附录 A　软件版本

Python 3.8

Visual Studio 2019 community

附录 B　源程序

第一关，第二关解题程序–C++ 源代码

```
// game12.cpp
//

#include <iostream>
#include <vector>
#include <algorithm>
#include <functional>

#define _STD ::std::

enum class weather {
   heat,
   serenity,
   sand,
};

//record the ligature situation
static const _STD vector<_STD vector<int>> map_one = {
   {},                        //0, mean nothing, ignore it.
   {2, 25},                   //1
   {1, 3},                    //2
   {2, 4, 25},                //3
   {3, 5, 24, 25},            //4
   {4, 6, 24},                //5
   {5, 7, 23, 24},            //6
   {6, 8, 22},                //7
   {7, 9, 22},                //8
   {8, 10, 15, 16, 17, 21, 22}, //9
   {9, 11, 13, 15},           //10
   {10, 12, 13},              //11
   {11, 13, 14},              //12
   {10, 11, 12, 14, 15},      //13
   {12, 13, 15, 16},          //14
   {9, 10, 13, 14, 16},       //15
   {9, 14, 15, 17, 18},       //16
   {9, 16, 18, 21},           //17
```

```
    {16, 17, 19, 20},        //18
    {18, 20},                //19
    {18, 19, 21},            //20
    {9, 17, 20, 22, 23, 27}, //21
    {7, 8, 9, 21, 23},       //22
    {6, 21, 22, 24, 26},     //23
    {4, 5, 6, 23, 25, 26},   //24
    {1, 3, 4, 24, 26},       //25
    {23, 24, 25, 27},        //26
    {21, 26},                //27
};


//record the ligature situation
static const _STD vector<_STD vector<int>> map_one_simple = {
    {},                      //0, mean nothing, ignore it.
    {25},                  //1
    {},                   //2
    {},               //3
    {},            //4
    {},               //5
    {},            //6
    {},               //7
    {},               //8
    {15,21}, //9
    {},           //10
    {},             //11
    {14},             //12
    {12},      //13
    {15},          //14
    {13,9},       //15
    {},        //16
    {},           //17
    {},          //18
    {},                 //19
    {},             //20
    {27},    //21
    {9},         //22
    {22},       //23
    {23},     //24
    {24},         //25
    {},          //26
    {},                 //27
};


const _STD vector<int> map_one_village = { 15 };
```

```
const _STD vector<int> map_one_mine = { 12 };

const _STD vector<enum weather> map_weather = {
   weather::heat, // nothing

   weather::heat, // the first day
   weather::heat,
   weather::serenity,
   weather::sand,
   weather::serenity,
   weather::heat,
   weather::sand,
   weather::serenity,
   weather::heat,
   weather::heat,

   weather::sand,
   weather::heat,
   weather::serenity,
   weather::heat,
   weather::heat,
   weather::heat,
   weather::sand,
   weather::sand,
   weather::heat,
   weather::heat,

   weather::serenity,
   weather::serenity,
   weather::heat,
   weather::serenity,
   weather::sand,
   weather::heat,
   weather::serenity,
   weather::serenity,
   weather::heat,
   weather::heat,
};

int Dijkstra(int x, int y, int size)
{
   _STD vector<int> ans(size, 1000000);
   ans[x] = 0;
   for (int i = 1; i < size; i++) {
      for (int j = 1; j < size; j++) {
         if (ans[j] == 1000000) {
            continue;
```

```
            }
            for (auto m : map_one[j]) {
                if (ans[j] + 1 < ans[m]) {
                    ans[m] = ans[j] + 1;
                }
            }
        }
    }
    return ans[y];
}


void game(const _STD vector<int>& area, const _STD vector<int>& mine);
void calculate(int money, int water, int foods, const _STD vector<int>& area);

int main() {
    _STD cout << "地图一中起点到采石场最短距离为: "
        << Dijkstra(1, 12, map_one.size()) << _STD endl;
    _STD cout << "地图一中采石场到终点最短距离为: "
        << Dijkstra(12, 27, map_one.size()) << _STD endl;
    _STD vector<_STD vector<int>> area_count = {
        //0 1 2  3  4  5  6  7 8 9  10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
            29 30
        {1,25,24,23,23,22,9,
            9,15,13,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,14,15,9 ,21,27},
        {1,25,24,23,23,22,9,
            9,15,13,12,14,15,13,12,12,12,12,12,12,12,12,12,12,12,12,14,15,9 ,21,27},
        {1,25,24,23,23,22,9,
            9,15,13,12,12,14,15,13,12,12,12,12,12,12,12,12,12,12,12,14,15,9 ,21,27},
        {1,25,24,23,23,22,9,
            9,15,13,12,12,12,14,15,13,12,12,12,12,12,12,12,12,12,12,14,15,9 ,21,27},
        {1,25,24,23,23,22,9,
            9,15,13,12,12,12,12,14,15,13,13,13,12,12,12,12,12,12,12,14,15,9 ,21,27},
        {1,25,24,23,23,22,9,
            9,15,13,12,12,12,12,12,14,15,15,15,13,12,12,12,12,12,12,14,15,9 ,21,27},
        {1,25,24,23,23,22,9,
            9,15,13,12,12,12,12,12,12,14,14,14,15,13,12,12,12,12,12,14,15,9 ,21,27},
        {1,25,24,23,23,22,9,
            9,15,13,12,12,12,12,12,12,12,12,12,14,15,13,12,12,12,12,14,15,9 ,21,27},
        {1,25,24,23,23,22,9,
            9,15,13,12,12,12,12,12,12,12,12,12,12,14,15,13,12,12,12,14,15,9 ,21,27},
    };
    _STD cout << area_count[0].size() << _STD endl;
    for (auto area : area_count) {
        calculate(4980, 98, 453, area);
        _STD cout << _STD endl;
    }
```

```
}


class play {
public:
   play(const _STD vector<_STD vector<int>>& map, const _STD vector<enum weather>& weather,
      const _STD vector<int>& mine, const _STD vector<int>& village) {
      this->map = map;
      this->weather = weather;
      this->day = weather.size() - 1;

   }

   void dfs(int day, int area, int water, int foods, int money, _STD vector<int> a) {
      if (water < 0 || foods < 0 || money < 0) {
         return;
      }
      a.push_back(area);
      switch (map_weather[day])
      {
      case weather::heat:
         dfs(day + 1, area, water - 8, foods - 6, money, a);
         break;
      case weather::sand:
         dfs(day + 1, area, water - 10, foods - 10, money, a);
         break;
      case weather::serenity:
         dfs(day + 1, area, water - 5, foods - 7, money, a);
         break;
      default:
         break;
      }
      for (auto x : this->map[area]) {
         if ([&]()->bool {
            for (auto t : this->mine) {
               if (t == x) {
                  return true;
               }
            }
            return false;
            }()) {
            switch (map_weather[day])
            {
            case weather::heat:
               dfs(day + 1, area, water - 24, foods - 18, money + 1000, a);
               break;
```

```
            case weather::sand:
               dfs(day + 1, area, water - 30, foods - 30, money + 1000, a);
               break;
            case weather::serenity:
               dfs(day + 1, area, water - 15, foods - 21, money + 1000, a);
               break;
            default:
               break;
            }
         }
         else {
            switch (map_weather[day])
            {
            case weather::heat:


               break;

            case weather::sand:


               break;
            case weather::serenity:


               break;
            default:
               break;
            }
         }

      }
   }

   _STD vector<_STD vector<int>> map;
   _STD vector<enum weather>weather;
   _STD vector<int> mine;
   _STD vector<int> village;
   int day;
};


void calculate(int money, int water, int foods, const _STD vector<int>& area) {
   for (int i = 1; i <= 30; ++i) {
      switch (map_weather[i])
      {
      case weather::heat:
```

```
            if (area[i] == 12) {
                // 开始挖矿
                if (area[i - 1] == 12) {
                    water -= 8 * 3;
                    foods -= 6 * 3;
                    money += 1000;
                }
                else {
                    water -= 8 * 2;
                    foods -= 6 * 2;
                }
            }
            else {
                water -= 8 * 2;
                foods -= 6 * 2;
            }
            break;

        case weather::sand:
            if (area[i] == 12) {
                if (area[i - 1] == 12) {
                    water -= 10 * 3;
                    foods -= 10 * 3;
                    money += 1000;
                }
                else {
                    water -= 10;
                    foods -= 10;
                }
            }
            else {
                water -= 10;
                foods -= 10;
            }
            break;

        case weather::serenity:
            if (area[i] == 12) {
                if (area[i - 1] == 12) {
                    water -= 5 * 3;
                    foods -= 7 * 3;
                    money += 1000;
                }
                else {
                    water -= 5 * 2;
                    foods -= 7 * 2;
                }
```

```
            }
            else {
               water -= 5 * 2;
               foods -= 7 * 2;
            }
            break;

         default:
            break;
         }
         _STD cout << i << "water " << water <<
            " foods " << foods << " money " << money
            << _STD endl;
      }
      _STD cout << "最终结果为: " << money + water * 10 + foods * 20 << _STD endl;
   }



   void game(const _STD vector<int>& area, const _STD vector<int>& mine) {
      int water = 0, foods = 0;
      for (int i = 1; i <= 30; ++i) {
         switch (map_weather[i])
         {
         case weather::heat:
            if (area[i] == 12) {
               if (area[i - 1] == 12) {
                  water += 8 * 3;
                  foods += 6 * 3;
               }
            }
            else {
               water += 8;
               foods += 6;
            }
            break;

         case weather::sand:
            if (area[i] == 12) {
               if (area[i - 1] == 12) {
                  water += 10 * 3;
                  foods += 10 * 3;
               }
            }
            else {
               water += 10;
               foods += 10;
```

```
            }
            break;

        case weather::serenity:
            if (area[i] == 12) {
                if (area[i - 1] == 12) {
                    water += 5 * 3;
                    foods += 7 * 3;
                }
            }
            else {
                water += 5;
                foods += 7;
            }
            break;

        default:
            break;
        }
    }
    _STD cout << "water needs :" << water
        << "\nfoods needs :" << foods << _STD endl;
}
```