2022-2023 第二十七届智能体大赛 - 游戏文档

Edition: 0410-FINAL

一、游戏规则简介

本游戏属于**塔防游戏**。双方玩家建设防御塔、使用辅助技能来防守己方基地，还需要升级基地来增强己方攻势，最终破坏掉对方基地以取得最终胜利。本游戏的核心在于双方的“工蚁”机器人并不直接被玩家所操纵，而是通过**信息素**介入的寻路算法（参见“‘工蚁’寻路算法”一节）自主适应地图环境。面对自适应的“工蚁”，如何动态调整策略、平衡进攻与防御的资金投入是决定胜负的关键。

本游戏是**回合制游戏**，双方玩家按照顺序依次传达自己操作后，游戏逻辑会进行一回合的结算，并告知双方玩家回合结算的结果。回合数到达上限时，会根据双方玩家基地的剩余血量等标准判断胜负。

受自然界的蚂蚁启发，双方玩家所拥有的”工蚁“机器人将会根据信息素寻路算法自动尝试攻入地方基地。通过投入资金升级基地，我们可以**优化生产流水线**来提升“工蚁”机器人的组装速度，也可以**列装高级护甲**提升“工蚁”机器人的生命值上限。

为了防御敌方“工蚁”的进攻，我们可以建造多种多样的防御塔，如“一炮一个”的**重型加农炮**，专注攻速的**轻机枪**，还有经典的AOE**迫击炮**。注意，信息素算法加持的”工蚁“可能会很快适应你的防线，而且由于新建防御塔的价格是**指数级增长**的，因此需要审慎选择建造点位和升级路线。

你也可以使用各类**超级武器**为自己的进攻或防御创造有利条件，比如造成高额范围伤害的**闪电风暴**和瘫痪敌人防御塔的**EMP轰炸**。虽然效果非常强力，但是他们使用代价不菲、冷却时间很长，利用它们在关键时刻一击制胜吧！

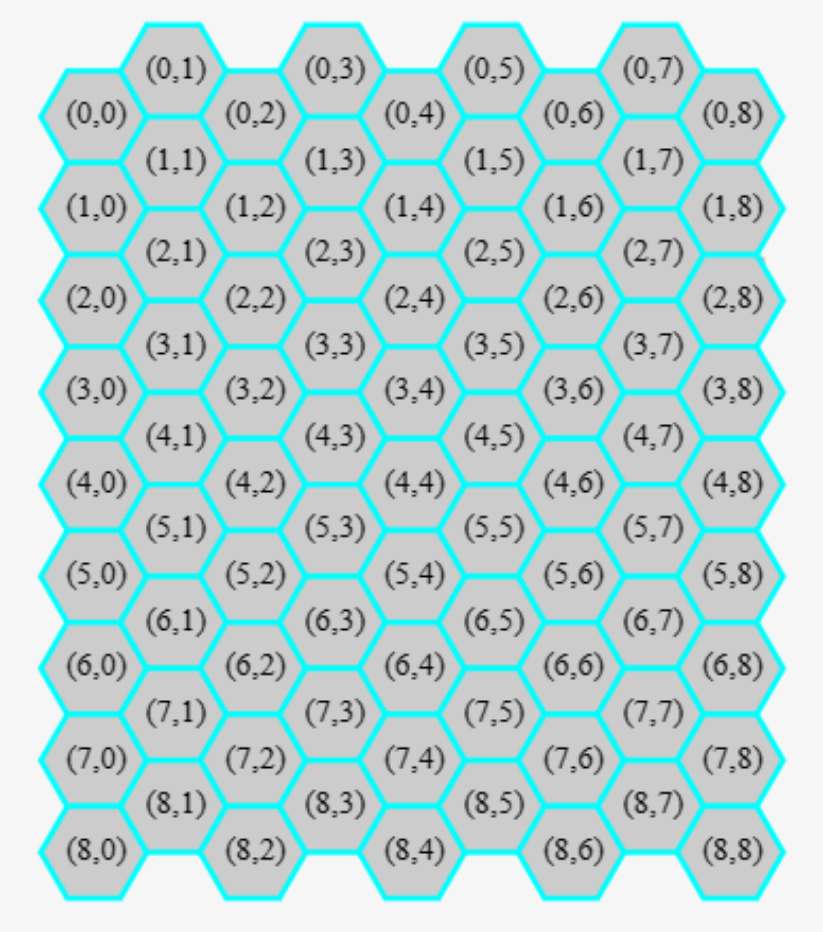
二、游戏规则详解

1. 基本数据

- 本游戏回合数从**0**开始，双方分别操作一次算作一回合。回合数到达512时结束游戏。
- “工蚁”具有年龄。年龄等于**当前回合数-生成回合数**。当年龄大于32时，蚂蚁死亡/消失。
- 规定AI每回合运行时间上限为**1秒**。

2. 游戏地图与坐标

本游戏采用六边形格点地图，采用“even-q”坐标系，如下图所示：



每个格点都有六个方向，计算相邻坐标的算法如下所示。需要额外注意的就是本坐标系在奇数列和偶数列需要分别处理。

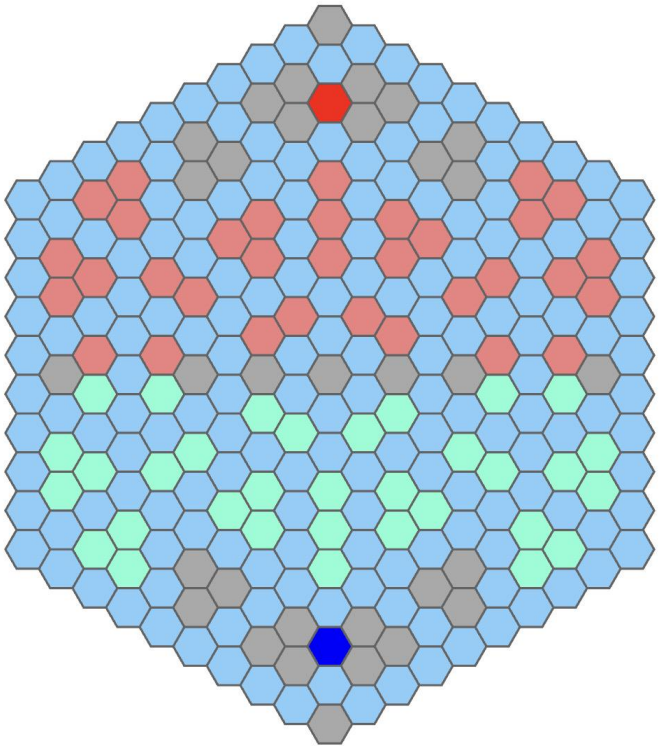
六个下标的顺序为右上、上、左上、左下、下、右下。

```
direction_difference = [[
    # even columns
    [0,  1], [-1, 0], [0, -1],
    [1, -1], [ 1, 0], [1,  1],
],[
    # odd columns
    [-1,  1], [-1,  0], [-1, -1],
    [ 0, -1], [ 1,  0], [ 0,  1],
]]

def calculate_neighbor(pos, direction):
    diff = direction_difference[place % 2][direction]
    return [pos[0] + diff[0], pos[1] + diff[1]]
```

本游戏的地图是一个边长为10的正六边形，也就是到中心点(9,9)的距离小于等于9的所有点的集合，如下图所示。其中：

- 红色网格(2,9)为玩家0的基地
- 深蓝色网格(16,9)为玩家1的基地
- 粉色区域是玩家0可以放置防御塔的区域
- 淡绿色区域是玩家1可以建造防御塔的区域
- 浅蓝色网格（以及两个基地）为蚂蚁可移动的区域，而其余颜色是蚂蚁不可移动的区域。



下面是不可移动区域的坐标列表：

```
{
    {6, 1}, {7, 1}, {9, 1}, {11, 1}, {12, 1}, {4, 2}, {6, 2}, {8, 2},
    {9, 2}, {11, 2}, {13, 2}, {4, 3}, {5, 3}, {13, 3}, {14, 3}, {6, 4},
```

```
    {8, 4}, {9, 4}, {11, 4}, {3, 5}, {4, 5}, {7, 5}, {9, 5}, {11, 5},
    {14, 5}, {15, 5}, {3, 6}, {5, 6}, {12, 6}, {14, 6}, {2, 7}, {5, 7},
    {6, 7}, {8, 7}, {9, 7}, {10, 7}, {12, 7}, {13, 7}, {16, 7}, {1, 8},
    {2, 8}, {7, 8}, {10, 8}, {15, 8}, {16, 8}, {0, 9}, {4, 9}, {5, 9},
    {6, 9}, {9, 9}, {12, 9}, {13, 9}, {14, 9}, {18, 9}, {1, 10}, {2, 10},
    {7, 10}, {10, 10}, {15, 10}, {16, 10}, {2, 11}, {5, 11}, {6, 11}, {8, 11},
    {9, 11}, {10, 11}, {12, 11}, {13, 11}, {16, 11}, {3, 12}, {5, 12}, {12, 12},
    {14, 12}, {3, 13}, {4, 13}, {7, 13}, {9, 13}, {11, 13}, {14, 13}, {15, 13},
    {6, 14}, {8, 14}, {9, 14}, {11, 14}, {4, 15}, {5, 15}, {13, 15}, {14, 15},
    {4, 16}, {6, 16}, {8, 16}, {9, 16}, {11, 16}, {13, 16}, {6, 17}, {7, 17},
    {9, 17}, {11, 17}, {12, 17}
}
```

下面是粉色区域的坐标列表：

```
{
    {6, 1}, {7, 1}, {4, 2}, {6, 2}, {8, 2}, {4, 3}, {5, 3}, {6, 4},
    {8, 4}, {7, 5}, {5, 6}, {5, 7}, {6, 7}, {8, 7}, {7, 8}, {4, 9},
    {5, 9}, {6, 9}, {7, 10}, {5, 11}, {6, 11}, {8, 11}, {5, 12}, {7, 13},
    {6, 14}, {8, 14}, {4, 15}, {5, 15}, {4, 16}, {6, 16}, {8, 16}, {6, 17}, {7, 17}
}
```

下面是淡绿色区域的坐标列表：

```
{
    {11, 1}, {12, 1}, {9, 2}, {11, 2}, {13, 2}, {13, 3}, {14, 3}, {9, 4},
    {11, 4}, {11, 5}, {12, 6}, {10, 7}, {12, 7}, {13, 7}, {10, 8}, {12, 9},
    {13, 9}, {14, 9}, {10, 10}, {10, 11}, {12, 11}, {13, 11}, {12, 12}, {11, 13},
    {9, 14}, {11, 14}, {13, 15}, {14, 15}, {9, 16}, {11, 16}, {13, 16}, {11, 17}, {12,
17}
}
```

3. 经济系统

- 玩家初始具有50金币。
- 每回合**结束**时，给予双方玩家1金币。
- **在结算阶段**，本回合每使用防御塔或超级武器杀死一只对方蚂蚁，按照对方蚂蚁的等级，等级1的蚂蚁给予3金币，等级2的蚂蚁给予5金币，等级3的蚂蚁给予7金币。本回合每有一只蚂蚁到达敌方基地，给予5金币。因寿命死亡的蚂蚁不对金币产生影响。

4. 防御塔

- **建造价格**：建造**新的**防御塔的价格为15×2^i，其中i为当前存在的己方防御塔的数量。或者说，建造第一个防御塔的金额为15，建造第二座防御塔的价格为30，建造第三座防御塔的价格为60，以后每座防御塔的建造价格均翻倍。
- **升级价格**：防御塔一共分为三个等级，共13种类。等级1升级到等级2需要60金币，等级2升级到等级3需要200金币。

- **降级、拆除返还**：降级、拆除防御塔会返还 80% 的建造花销，即等级3降级为等级2返还160金，等级2降级为等级1返还48金，拆除等级1返还12×2^i金，其中i为**拆除之后**己方防御塔的数量。
- **防御塔攻击**：每种类防御塔都有固定的伤害、攻击间隔、攻击次数、攻击方式、攻击范围。*攻击间隔为K，攻击次数为M*意为防御塔内置CD为0时，可以进行M次攻击，若攻击到任何“工蚁”，则重置CD为K，否则CD不变。用代码表示为：

  ```
  tower.cd = max(0, tower.cd - 1);
  if (tower.cd == 0) {
    if (has_targets_in_range(tower)) {
        for (i of 0..M) tower.attack();
        tower.cd = K;
    }
  }
  ```

  还需指出：建造、升级、降级防御塔都会令内置CD为攻击间隔K。
- **索敌逻辑**：选取范围内血量大于0且非己方的敌人，按照与防御塔的距离为主键、“工蚁”的ID为附键升序排列。顺序越靠前攻击优先级越高。用代码表示为：

  ```
  targets = ants_in_range(tower.x, tower.y, tower.range)
              .filter((ant) => ant.hp > 0 && ant.player != tower.player)
              .sort((a, b) => a_dist == b_dist ? a.id - b.id : a_dist - b_dist)
  // a_dist = distance([tower.x, tower.y], [ant.x, ant.y])
  ```

- 若非特殊说明，默认的攻击方式就是对优先级最高的“工蚁”造成一定的伤害。即：

  ```
  bool attack() {
    targets = find_targets();
    if (targets.empty()) return false;
    targets[0].hp -= this.atk;
    return true;
  }
  ```

- **AOE攻击**：部分防御塔拥有“AOE”(Area of effect)攻击属性。若为“范围为R的AOE攻击”，即为对优先级最高的“工蚁”及到其所在格距离不大于R的格子中所有“工蚁”均造成伤害。**这有可能影响到本来不处于攻击范围内的敌方“工蚁”。**
- **防御塔数据列表**

类型ID	名称	伤害	间隔	范围	攻击方式	上级防御塔
0	Basic	5	2	2	Default	无
1	Heavy	20	2	2	Default	0
11	Heavy+	35	2	3	Default	1
12	Ice	15	2	2	Default，但会“冻结”造成伤害的蚂蚁一回合	1
13	Cannon	50	3	3	Default	1
2	Quick	6	1	3	Default	0
21	Quick+	8	0.5	3	Default	2
22	Double	7	1	4	最多可以攻击优先级前二的目标	2
23	Sniper	15	2	6	Default	2
3	Mortar	16	4	3	范围为1的AOE	0
31	Mortar+	35	4	4	范围为1的AOE	3
32	Pulse	30	3	2	同时攻击范围内所有目标	3
33	Missile	45	6	5	范围为2的AOE	3

5. 超级武器

- 选手可以使用一定的金币，在指定的位置发射超级武器。超级武器具有一定的冷却时间。
- 超级武器列表：

ID	名称	花费	冷却	效果
1	闪电风暴 Lightning Storm	150	100	令指定位置**范围为3**的区域内进入“闪电风暴”状态，**持续20回合**。每回合对范围内的**所有**敌方“工蚁”造成100伤害。
2	EMP轰炸 EMP Blaster	150	100	令指定位置**范围为3**的区域内陷入**“电磁脉冲干扰”**状态，**持续20回合**。敌方无法在有“电磁脉冲干扰”的区域内建造、升级、降级防御塔（超级武器不受影响）。在“电磁脉冲干扰”区域内的敌方防御塔无法攻击，内置CD不变。
3	引力护盾 Deflectors	100	50	令指定位置**范围为3**的区域内进入**“引力护盾”**状态，**持续10回合**。在“引力护盾”区域内的己方“工蚁”免疫单次小于自身最大生命值50%的伤害。对于大于等于自身最大生命值50%的伤害不影响。
4	紧急回避 Emergency Evasion	100	50	**立刻**给予指定位置**范围为3**的区域内的所有我方“工蚁”**2层“紧急回避”**。每层“紧急回避”可以抵消一次防御塔造成的伤害（优先于“引力护盾”的效果进行结算）。不会过期。

6. 基地

- 每方基地在游戏开始时有50血量，每当一只敌方“工蚁”移动到基地，则会减少1血量。当有一方基地血量不大于0时游戏立即结束。
- 基地可以进行如下升级：
- **优化生产流水线**：当前回合被K整除时，都会在基地处建造一只“工蚁”。初始为1级。

	1级	2级	3级
K	4	2	1

- **列装高级护甲**：产生“工蚁”机器人的最大生命值。注意这一属性升级时，已产生的“工蚁”最大生命值**不会改变**。初始为1级。

	1级	2级	3级
最大生命	10	25	50

- **花费**：1级升级为2级：200。2级升级为3级：250。无法降级。
- 一回合只能对基地进行一种升级。

7. “工蚁”寻路算法

- **可移动区域**：如“游戏地图与坐标”一节中所示的浅蓝色区域都是可移动区域，另外，“工蚁”不会走**回头路**（即本回合不会以上回合所在的位置为移动目标）。
- **信息素**：信息素是寻路算法的核心。
    - 每个网格都存在双方玩家独立的信息素数值

- **初始值**：游戏开始时每一格的信息素为$\tau_0+\epsilon,\tau_0=10,\epsilon\sim U(-2,2)$（即为[-2,2]区间的均匀分布，由随机数种子生成），每格独立，双方独立
- **更新**：
  - 当有"工蚁"**攻入敌方基地**，它所走过的路径上的所有点的信息素都会$\Delta\tau_1=+10$
  - 当有"工蚁"因为**生命值耗尽而死亡**，它所走过的路径上的所有点的信息素都会$\Delta\tau_2=-5$
  - 当有"工蚁"因为**年龄过大而死亡**，它所走过的路径上的所有点的信息素都会$\Delta\tau_3=-3$
  - 注意，上述更新对于"工蚁"重复经过的点都只更新一次。
  - **每回合**更新信息素时，按照以下公式进行：$\tau'_P=\lambda\tau_P+(1-\lambda)\tau_0+\sum_k\Delta\tau_P^{(k)}$。其中$\lambda=0.97$为信息素衰减比例。$\Delta\tau_P^{(k)}$为第$k$只"工蚁"对点$P$产生的信息素贡献/变化。
- **目标吸引度**："工蚁"的目标是攻入敌方基地
  - 设"工蚁"现在的位置为P，相邻的位置共有六个$P_d,d=0,1,\cdots,5$。那么对应方向移动的**吸引度**为
  - $\eta_d=\eta(\mathrm{dist}(P_d)-\mathrm{dist}(P))=\eta(\Delta D)=\begin{cases}1.25,\Delta D=-1\\1.00,\Delta D=0\\0.75,\Delta D=1\end{cases}$
  - 其中$\mathrm{dist}(P)$为点P到敌方基地的距离。即向靠近敌方基地的方向移动要更具有**吸引力**。
- **寻路算法**：
  - 设六个方向的相邻点的坐标分别为$P_d,d=0,1,\cdots,5$
  - 有效性向量：$\vec{v}=(v(P_0),\cdots,v(P_5)),v(P)=\begin{cases}1,P\text{可移动}\\0,P\text{不可移动}\end{cases}$
  - 信息素向量：$\vec{\tau}=(\tau_{P_0},\cdots,\tau_{P_5})$
  - 吸引度向量：$\vec{\eta}=(\eta_0,\cdots,\eta_5)$
  - 移动向量：$\vec{P}=\vec{v}\cdot\vec{\tau}\cdot\vec{\eta}$
  - 最终移动方向：$d=\mathrm{maxidx}(\vec{P})$
  - 若存在$\vec{P}$内元素相同的情况，优先取信息素更高的方向。若还相同，取较小的方向值。

8. 胜负判定

- 大本营剩余血量多者，胜。如果剩余血量相同：
- 击败对方蚂蚁数多者，胜。如果击败蚂蚁数相同：
- 使用超级武器少者，胜。如果使用超级武器次数相同：
- AI用时少者，胜。如果用时相同：
- 先手胜。

9. 结算流程

1. 玩家操作
   1. 等待玩家0的操作，若玩家0程序崩溃、运行超时、返回了不符合协议的数据、执行了非法的操作，那么立刻判负。
   2. 执行玩家0的操作，如建造、升级防御塔，升级基地、使用超级武器等。
   3. 然后对玩家1执行同样的步骤
2. 回合结算
   1. 结算闪电风暴
   2. 按照建造顺序结算防御塔攻击。
   3. 若本回合造成敌方"工蚁"死亡（生命值变为非正数），则按照死亡的"工蚁"的等级获得金币。

4. 如果蚂蚁的年龄大于最大年龄，即**当前回合数-蚂蚁生成回合数大于32**，标记为老死。
5. 按照生成顺序结算蚂蚁移动。
    1. 已死亡的蚂蚁不可以移动。
    2. 被冻结的蚂蚁这回合无法移动，并解除冻结状态。
    3. 如果蚂蚁移动到敌方基地内，那么立刻减少敌方基地血量，并判断是否降为0，若降为0，游戏结束。
6. 结算信息素并移除已死亡、已到达的蚂蚁；到达的蚂蚁结算金币收益。
7. 按照基地等级尝试生成蚂蚁。
8. 双方金币+1
9. 回合数+1
10. 若回合数等于512，执行胜负判断，结束游戏

- **可移动区域**：如“游戏地图与坐标”一节中所示的浅蓝色区域都是可移动区域，另外，“工蚁”不会走**回头路**（即本回合不会以上回合所在的位置为移动目标）。
- **信息素**：信息素是寻路算法的核心。
    - 每个网格都存在双方玩家独立的信息素数值
    - **初始值**：游戏开始时每一格的信息素为$\tau_0+\epsilon,\tau_0=10,\epsilon\sim U(-2,2)$（即为[-2,2]区间的均匀分布，由随机数种子生成），每格独立，双方独立
    - **更新**：
        - 当有”工蚁“**攻入敌方基地**，它所走过的路径上的所有点的信息素都会$\Delta\tau_1=+10$
        - 当有”工蚁“因为**生命值耗尽而死亡**，它所走过的路径上的所有点的信息素都会$\Delta\tau_2=-5$
        - 当有”工蚁“因为**年龄过大而死亡**，它所走过的路径上的所有点的信息素都会$\Delta\tau_3=-3$
        - 注意，上述更新对于“工蚁”重复经过的点都只更新一次。
        - **每回合**更新信息素时，按照以下公式进行：$\tau'_P=\lambda\tau_P+(1-\lambda)\tau_0+\sum_k\Delta\tau_P^{(k)}$。其中$\lambda=0.97$为信息素衰减比例。$\Delta\tau_P^{(k)}$为第$k$只“工蚁”对点$P$产生的信息素贡献/变化。
    - **目标吸引度**：“工蚁”的目标是攻入敌方基地
        - 设“工蚁”现在的位置为P，相邻的位置共有六个$P_d,d=0,1,\cdots,5$。那么对应方向移动的**吸引度**为
        - $\eta_d=\eta(\mathrm{dist}(P_d)-\mathrm{dist}(P))=\eta(\Delta D)=\begin{cases}1.25,\Delta D=-1\\1.00,\Delta D=0\\0.75,\Delta D=1\end{cases}$
        - 其中$\mathrm{dist}(P)$为点P到敌方基地的距离。即向靠近敌方基地的方向移动要更具有**吸引力**。
- **寻路算法**：
    - 设六个方向的相邻点的坐标分别为$P_d,d=0,1,\cdots,5$
    - 有效性向量：$\vec{v}=(v(P_0),\cdots,v(P_5)),v(P)=\begin{cases}1,P\text{可移动}\\0,P\text{不可移动}\end{cases}$
    - 信息素向量：$\vec{\tau}=(\tau_{P_0},\cdots,\tau_{P_5})$
    - 吸引度向量：$\vec{\eta}=(\eta_0,\cdots,\eta_5)$
    - 移动向量：$\vec{P}=\vec{v}\cdot\vec{\tau}\cdot\vec{\eta}$
    - 最终移动方向：$d=\mathrm{maxidx}(\vec{P})$
    - 若存在$\vec{P}$内元素相同的情况，优先取信息素更高的方向。若还相同，取较小的方向值。

8. 胜负判定

- 大本营剩余血量多者，胜。如果剩余血量相同：
- 击败对方蚂蚁数多者，胜。如果击败蚂蚁数相同：
- 使用超级武器少者，胜。如果使用超级武器次数相同：
- AI用时少者，胜。如果用时相同：
- 先手胜。

9. 结算流程

1. 玩家操作
    1. 等待玩家0的操作，若玩家0程序崩溃、运行超时、返回了不符合协议的数据、执行了非法的操作，那么立刻判负。
    2. 执行玩家0的操作，如建造、升级防御塔，升级基地、使用超级武器等。
    3. 然后对玩家1执行同样的步骤
2. 回合结算
    1. 结算闪电风暴
    2. 按照建造顺序结算防御塔攻击。
    3. 若本回合造成敌方”工蚁“死亡（生命值变为非正数），则按照死亡的”工蚁“的等级获得金币。
    4. 如果蚂蚁的年龄大于最大年龄，即**当前回合数-蚂蚁生成回合数大于32**，标记为老死。
    5. 按照生成顺序结算蚂蚁移动。
        1. 已死亡的蚂蚁不可以移动。
        2. 被冻结的蚂蚁这回合无法移动，并解除冻结状态。
        3. 如果蚂蚁移动到敌方基地内，那么立刻减少敌方基地血量，并判断是否降为0，若降为0，游戏结束。
    6. 结算信息素并移除已死亡、已到达的蚂蚁；到达的蚂蚁结算金币收益。
    7. 按照基地等级尝试生成蚂蚁。
    8. 双方金币+1
    9. 回合数+1
    10. 若回合数等于512，执行胜负判断，结束游戏

三、选手AI编写指南

1. 输入输出

选手AI可以从标准输入流**直接读取**来自评测系统的信息。

但是，选手AI通过标准输入流向评测系统返回信息时，需要在发送的信息之前添加一个**四字节大端序**整数，代表信息的长度。如选手想要输出以下信息：

```
2
1 2 3 4 5
```

这条信息的长度为11，其十六进制数据表示为（其中第二行的`_`为空格）：

```
HEX : 32 0A 31 20 32 20 33 20 34 20 35
TEXT:  1 \n  1  _  2  _  3  _  4  _  5
```

因此需要在数据包前面加上11的四字节大端序表示`00 00 00 0B`，最终结果为：

```
HEX : 00 00 00 0B 32 0A 31 20 32 20 33 20 34 20 35
TEXT:  0  0  0 11  1 \n  1  _  2  _  3  _  4  _  5
```

选手AI可以通过向标准错误流输出信息进行调试，在Saiblo平台上评测完成后，会提供标准错误流产生的信息。请注意最终提交时请尽量减少调试信息的输出，因为这可能会占用大量运行时间。

2. 评测流程

- 总体评测流程如下
  - 平台启动双方玩家AI程序，并向双方玩家发送初始化信息
  - 每一回合均按照以下流程顺序、循环执行
    1. 等待先手玩家操作
    2. 平台接收到先手玩家操作后，进行验证和执行，如果成功完成，将操作转发给后手玩家
    3. 等待后手玩家操作
    4. 平台接收到后手玩家操作后，进行验证和执行，如果成功完成，将操作转发给先手玩家
    5. 游戏逻辑进行一回合的结算，如果回合正常结束，则将局面信息通过平台分别发送给两位玩家。如果游戏结束，那么会向平台汇报游戏结果，终止评测流程。
- 对于**先手玩家**的AI，执行流程如下：
  - 接受初始化信息，判断自己是先手玩家
  - 每一回合中：
    1. 进行决策，发送自己的操作
    2. 等待接受后手玩家的操作
    3. 等待接受局面信息
- 对于**后手玩家**的AI，执行流程如下：
  - 接受初始化信息，判断自己是后手玩家
  - 每一回合中：
    1. 等待接受先手玩家的操作
    2. 进行决策，发送自己的操作
    3. 等待接受局面信息

3. 游戏初始化信息

一行两个整数`K M`，`K == 0`代表自己为先手玩家P0，`K == 1`代表自己为后手玩家P1；`M`为随机数种子，用于计算初始局面的信息素，计算算法如下：

```
// C++ Version
```

```cpp
unsigned long long lcg_seed;
unsigned long long lcg(){
    lcg_seed = (25214903917 * lcg_seed) & ((1ll << 48) - 1);
    return lcg_seed;
}

void init_pheromon(unsigned long long M){
    lcg_seed = M;
    for(int i = 0; i < 2; i++)
        for(int j = 0; j < MAP_SIZE; j++)
            for(int k = 0; k < MAP_SIZE; k++)
                pheromone[i][j][k] = lcg() * pow(2, -46) + 8;
}
```

```python
# Python Version

lcg_seed = 0
def lcg():
    global lcg_seed
    lcg_seed = (25214903917 * lcg_seed) & ((1 << 48) - 1)
    return lcg_seed

def init_pheromon(M):
    global lcg_seed
    lcg_seed = M
    for i in [0,1]:
        for j in range(0, MAP_SIZE):
            for k in range(0, MAP_SIZE):
                pheromone[i][j][k] = lcg() * pow(2, -46) + 8
```

4. 玩家操作信息

第一行一个整数N，代表操作数。接下来N行，每行表示一个操作，第一个整数T代表操作类型，后面需要根据操作类型提供一些参数。操作类型和参数如下：

- 建造防御塔：
  - 格式：`11 x y` 在`(x, y)`处建造防御塔
  - 例子：`11 11 1` 在`(11, 1)`处建立一座新的防御塔
- 升级防御塔：
  - 格式：`12 towerId towerTypeId` 将ID为`towerId`的防御塔升级为`towerTypeId`所代表的防御塔类型
  - 例子：`12 1 31` 将ID为`1`的防御塔升级为`31`类型，也即`Mortar+`
- 降级防御塔：
  - 格式：`13 towerId` 降级ID为`towerId`的防御塔。如果已经是`Basic`防御塔，那么会将其**拆除**。
  - 例子：`13 1` 降级ID为`1`的防御塔。如果已经是`Basic`防御塔，那么会将其**拆除**。

- 超级武器：
  - 格式：`21 x y` `22 x y` `23 x y` `24 x y` 分别在`(x, y)`处部署ID为1/2/3/4的超级武器。
  - 例子：`21 9 9` 在`(9, 9)`的位置释放“闪电风暴”
- 基地升级：
  - 格式：`31` 代表升级“生产流水线（出兵速度）”。`32`升级“高级装甲（最大生命）”。
  - 例子：`31` 升级“生产流水线（出兵速度）”

可能的非法操作/限制包括：

- 操作类型不合法
- 建造、升级防御塔、使用超级武器、基地升级所需的金币不足
- 建造防御塔、使用超级武器的位置非法
- 升级、降级防御塔所指定的防御塔ID不存在
- 升级防御塔所指定的防御塔类型不存在
- 升级防御塔只能升级到下一等级，如不可以直接从`Basic`直接升级到`Heavy+`，也不允许升级到另一条线路上，如`Quick`升级为`Heavy+`
- 一回合之内，只能对一个ID的防御塔进行一次操作，也只能对基地进行一次操作。即不允许连续升级、不允许建造后直接升级、不允许升级后降级等。
- 使用的超级武器还在冷却
- 指定的基地升级已经到最高等级

如下是一个完整的操作信息的示例：

```
5
11 11 1
12 1 31
13 2
21 9 9
31
```

5. 局面信息

第一行一个整数`R`，表示回合数。

第二行一个整数`N_1`，代表场上总防御塔数量。接下来`N_1`行，每行6个整数，分别为`id player x y type cd`，即每个防御塔的ID、归属、坐标、类型、攻击CD。

接下来一个整数`N_2`，代表场上总“工蚁”数量。接下来`N_2`行，每行8个整数，分别为`id player x y hp lv age state`，即每只“工蚁”的ID、阵营、坐标、当前生命、等级、 当前寿命、当前状态。

- 蚂蚁等级分别为：0/1/2
- 当前状态：蚂蚁一共有5种状态，对应 0-4

```
enum State {
    Alive,   // Still alive
    Success, // Reach the other camp
    Fail,    // No HP
    TooOld,  // Too old
    Frozen   // Frozen
};
```

接下来两个整数`G0 G1`，分别代表先后手玩家的剩余金币。

接下来两个整数`HP0 HP1`，分别代表先后手玩家的剩余基地血量。

四、AI SDK使用指南

为了方便各位编写游戏AI，我们设计了Python和C++的AI SDK供大家使用。

- Python SDK：
  - GitHub仓库：https://github.com/saiblo/saiblo-antwar-sdk-python
  - 使用文档：https://saiblo.github.io/saiblo-antwar-sdk-python/
- C++ SDK：
  - GitHub仓库：https://github.com/saiblo/saiblo-antwar-sdk-cpp
  - 使用文档：https://saiblo.github.io/saiblo-antwar-sdk-cpp/
- Logic THU Gitlab仓库：
  - https://git.tsinghua.edu.cn/agent-logic/ant_game

五、回放文件格式说明

Saiblo网站的对局支持下载回放文件，回放文件格式为json，文件为一个json数组，每一个元素代表一个回合信息：

```
[
    {
        "seed": 5, // 随机数种子，仅在第一回合出现此参数
        "op0": [ // player0 操作数组
            { //
                "args": -1, // 操作参数，没有设为-1
                "id": -1, // 目标id参数
                "pos": { // 位置参数
                    "x": 6,
                    "y": 4
                },
                "type": 11 // 操作类型
            }
```

```
        ],
        "op1": [ //player1 操作
        ],
        "round_state": { // 回合信息
            "anthpLv": [ // 大本营高级装甲升级
                0,
                0
            ],
            "ants": [ // 所有蚂蚁信息列表
                {
                    "age": 0, // 蚂蚁年龄
                    "hp": 10, // 生命值
                    "id": 0,
                    "level": 0, // 蚂蚁等级
                    "move": -1, // 蚂蚁本回合移动方向，  -1为未移动
                    "player": 0,
                    "pos": {
                        "x": 2,
                        "y": 9
                    },
                    "status": 0 // 蚂蚁状态，参考局面信息说明
                }
            ],
            "camps": [ // 大本营血量
                50,
                50
            ],
            "coins": [ // 玩家金币
                6,
                6
            ],
            "error": "", // 结束错误信息，仅对局异常结束时返回该字段，若出现异常情况会显示具体错误信息
            "message": "[,]", // 结束信息，为字符串类型，表示玩家结束时状态
            "pheromone": [], // 信息素数组，大小为[2][MAP_SIZE][MAP_SIZE]
            "speedLv": [ // 大本营生产流水线等级
                0,
                0
            ],
            "towers": [ // 防御塔增量信息，即只包含回合内新建/等级状态发生变化的塔
                {
                    "cd": 1, // 防御塔cd
                    "id": 0, // 防御塔cd
                    "player": 0, // 所属玩家
                    "pos": { // 防御塔坐标
                        "x": 6,
                        "y": 4
                    },
                    "type": 0 // 防御塔类型
```

```
                },
            ],
            "winner": -1 // 游戏结果，表示为玩家id，未结束时为-1
        }
    }
]
```