Solution

2022.01

Outline

数数

数数

- 计算 $[1, R]$ 内的个数减去 $[1, L-1]$ 内的个数

数数

- 计算 $[1, R]$ 内的个数减去 $[1, L-1]$ 内的个数
- 答案最多 9 位，因此我们可以爆搜出所有的数字集合 (为了方便前导零也计算在内)，并用 2^9 的时间来判断是否符合条件

数数

- 计算 $[1, R]$ 内的个数减去 $[1, L-1]$ 内的个数
- 答案最多 9 位，因此我们可以爆搜出所有的数字集合 (为了方便前导零也计算在内)，并用 2^9 的时间来判断是否符合条件
- 有了数字集合我们可以从高位开始枚举答案

数数

- 计算 $[1, R]$ 内的个数减去 $[1, L-1]$ 内的个数
- 答案最多 9 位，因此我们可以爆搜出所有的数字集合 (为了方便前导零也计算在内)，并用 2^9 的时间来判断是否符合条件
- 有了数字集合我们可以从高位开始枚举答案
- 令数字总个数为 S，数字 i 有 c_i 个

数数

- 计算 $[1, R]$ 内的个数减去 $[1, L-1]$ 内的个数
- 答案最多 9 位，因此我们可以爆搜出所有的数字集合 (为了方便前导零也计算在内)，并用 2^9 的时间来判断是否符合条件
- 有了数字集合我们可以从高位开始枚举答案
- 令数字总个数为 S，数字 i 有 c_i 个
-

$$\frac{S!}{c_0! \times c_1! \times \cdots \times c_9!}$$

数数

- 计算 $[1, R]$ 内的个数减去 $[1, L-1]$ 内的个数
- 答案最多 9 位，因此我们可以爆搜出所有的数字集合 (为了方便前导零也计算在内)，并用 2^9 的时间来判断是否符合条件
- 有了数字集合我们可以从高位开始枚举答案
- 令数字总个数为 S，数字 i 有 c_i 个
- $$\frac{S!}{c_0! \times c_1! \times \cdots \times c_9!}$$
- 若高位已经小于 R 的限制，则后面几位的方案数可以用上式计算

数数

- 计算 $[1, R]$ 内的个数减去 $[1, L-1]$ 内的个数
- 答案最多 9 位，因此我们可以爆搜出所有的数字集合 (为了方便前导零也计算在内)，并用 2^9 的时间来判断是否符合条件
- 有了数字集合我们可以从高位开始枚举答案
- 令数字总个数为 S，数字 i 有 c_i 个
- $$\frac{S!}{c_0! \times c_1! \times \cdots \times c_9!}$$
- 若高位已经小于 R 的限制，则后面几位的方案数可以用上式计算
- 否则高位一定与 R 相同，继续枚举至下一位即可

数数

- 计算 $[1, R]$ 内的个数减去 $[1, L-1]$ 内的个数
- 答案最多 9 位，因此我们可以爆搜出所有的数字集合 (为了方便前导零也计算在内)，并用 2^9 的时间来判断是否符合条件
- 有了数字集合我们可以从高位开始枚举答案
- 令数字总个数为 S，数字 i 有 c_i 个
- $$\frac{S!}{c_0! \times c_1! \times \cdots \times c_9!}$$
- 若高位已经小于 R 的限制，则后面几位的方案数可以用上式计算
- 否则高位一定与 R 相同，继续枚举至下一位即可
- $O(\mathrm{C}_{18}^{9} \times (2^9 + 9^2))$

Outline

红与蓝

红与蓝

- 每个结点实际上有三种情况：红胜、蓝胜、先手胜

红与蓝

- 每个结点实际上有三种情况：红胜、蓝胜、先手胜
- 对于非叶结点的先手胜，它与叶子结点直接染色本质上是一样的，或者说染颜色就是先手胜

红与蓝

- 每个结点实际上有三种情况：红胜、蓝胜、先手胜
- 对于非叶结点的先手胜，它与叶子结点直接染色本质上是一样的，或者说染颜色就是先手胜
- $O(n)$ 遍历一遍树，利用儿子的信息 (每种情况的数量) 可以推出自己的情况，用根的情况即可判断出胜负

红与蓝

- 每个结点实际上有三种情况：红胜、蓝胜、先手胜
- 对于非叶结点的先手胜，它与叶子结点直接染色本质上是一样的，或者说染颜色就是先手胜
- $O(n)$ 遍历一遍树，利用儿子的信息 (每种情况的数量) 可以推出自己的情况，用根的情况即可判断出胜负
- 对于第二问 (能选哪些叶子)：

红与蓝

- 每个结点实际上有三种情况：红胜、蓝胜、先手胜
- 对于非叶结点的先手胜，它与叶子结点直接染色本质上是一样的，或者说染颜色就是先手胜
- $O(n)$ 遍历一遍树，利用儿子的信息 (每种情况的数量) 可以推出自己的情况，用根的情况即可判断出胜负
- 对于第二问 (能选哪些叶子)：
- $n \leq 1000$ 时可以直接枚举第一步选的叶子，之后再判断一遍胜负

红与蓝

红与蓝

- 设 $R(u)$ 表示 u 子树内第一步走并使得 u 为红胜的叶子集合

红与蓝

- 设 $R(u)$ 表示 u 子树内第一步走并使得 u 为红胜的叶子集合
- 设 $B(u)$ 表示 u 子树内第一步走并使得 u 从蓝胜变为先手胜的叶子集合 (这可以逼对手在 u 内跟着走一步)

红与蓝

- 设 $R(u)$ 表示 u 子树内第一步走并使得 u 为红胜的叶子集合
- 设 $B(u)$ 表示 u 子树内第一步走并使得 u 从蓝胜变为先手胜的叶子集合 (这可以逼对手在 u 内跟着走一步)
- 考虑求 $R(u)$:

红与蓝

- 设 $R(u)$ 表示 u 子树内第一步走并使得 u 为红胜的叶子集合
- 设 $B(u)$ 表示 u 子树内第一步走并使得 u 从蓝胜变为先手胜的叶子集合 (这可以逼对手在 u 内跟着走一步)
- 考虑求 $R(u)$:
- 若 u 为红胜，则 $R(u)$ 为 u 子树内的所有叶子

红与蓝

- 设 $R(u)$ 表示 u 子树内第一步走并使得 u 为红胜的叶子集合
- 设 $B(u)$ 表示 u 子树内第一步走并使得 u 从蓝胜变为先手胜的叶子集合 (这可以逼对手在 u 内跟着走一步)
- 考虑求 $R(u)$:
- 若 u 为红胜，则 $R(u)$ 为 u 子树内的所有叶子
- 若 u 为先手胜，则 $R(u)$ 包括蓝胜儿子的 $B(v)$ 与先手胜儿子的 $R(v)$

红与蓝

- 设 $R(u)$ 表示 u 子树内第一步走并使得 u 为红胜的叶子集合
- 设 $B(u)$ 表示 u 子树内第一步走并使得 u 从蓝胜变为先手胜的叶子集合 (这可以逼对手在 u 内跟着走一步)
- 考虑求 $R(u)$:
- 若 u 为红胜，则 $R(u)$ 为 u 子树内的所有叶子
- 若 u 为先手胜，则 $R(u)$ 包括蓝胜儿子的 $B(v)$ 与先手胜儿子的 $R(v)$
- $B(u)$ 类似讨论即可

红与蓝

- 设 $R(u)$ 表示 u 子树内第一步走并使得 u 为红胜的叶子集合
- 设 $B(u)$ 表示 u 子树内第一步走并使得 u 从蓝胜变为先手胜的叶子集合 (这可以逼对手在 u 内跟着走一步)
- 考虑求 $R(u)$:
- 若 u 为红胜，则 $R(u)$ 为 u 子树内的所有叶子
- 若 u 为先手胜，则 $R(u)$ 包括蓝胜儿子的 $B(v)$ 与先手胜儿子的 $R(v)$
- $B(u)$ 类似讨论即可
- 时间复杂度 $O(Tn)$

Outline

炮塔

炮塔

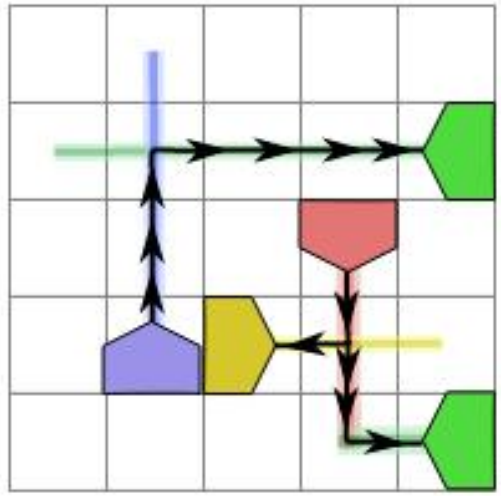
- 主要考虑如何解决轨迹不能相交

炮塔

- 主要考虑如何解决轨迹不能相交
- 注意到相交的轨迹实际上给出了一条炮塔之间的路径

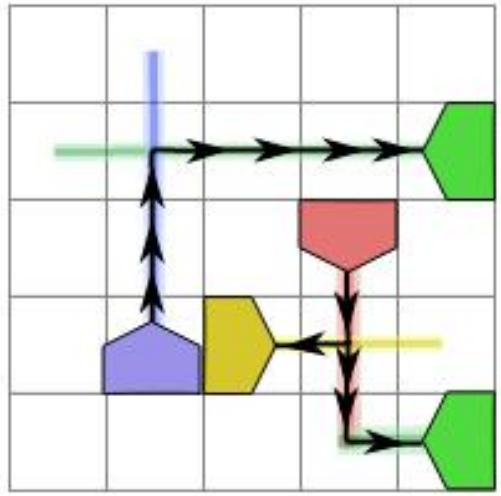
炮塔

- 主要考虑如何解决轨迹不能相交
- 注意到相交的轨迹实际上给出了一条炮塔之间的路径



炮塔

- 主要考虑如何解决轨迹不能相交
- 注意到相交的轨迹实际上给出了一条炮塔之间的路径



- 我们规定这些路径都是从纵向炮塔走向横向炮塔的

炮塔

炮塔

- 轨迹不相交相当于不存在这样的一条路径，于是我们就可以考虑用最小割求解了

炮塔

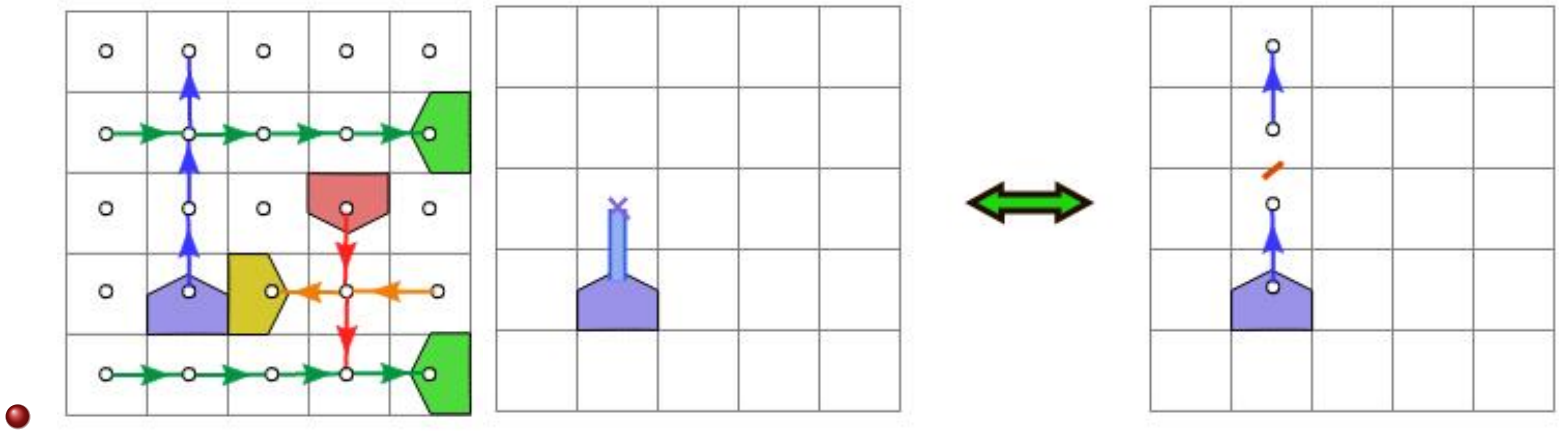
- 轨迹不相交相当于不存在这样的一条路径，于是我们就可以考虑用最小割求解了
- 现在重点是如何给每条边确定容量

炮塔

- 轨迹不相交相当于不存在这样的一条路径，于是我们就可以考虑用最小割求解了
- 现在重点是如何给每条边确定容量
- 首先我们先规定，一条 $(u \to v)$ 的割边说明我们选择攻击了 u 这个格子

炮塔

- 轨迹不相交相当于不存在这样的一条路径，于是我们就可以考虑用最小割求解了
- 现在重点是如何给每条边确定容量
- 首先我们先规定，一条 $(u \to v)$ 的割边说明我们选择攻击了 u 这个格子



炮塔

炮塔

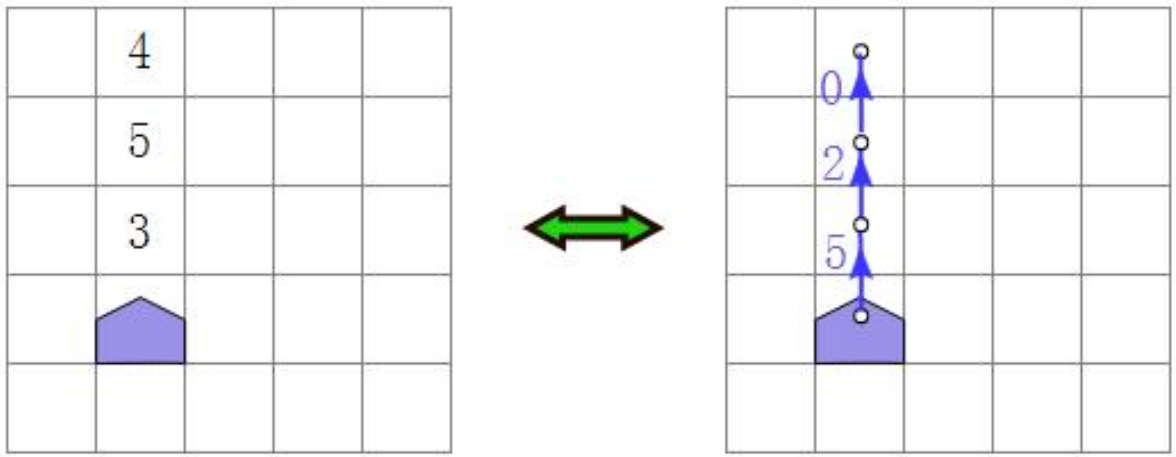
- 我们找出某个炮塔攻击方向上敌人的最大数量，假设为 max

炮塔

- 我们找出某个炮塔攻击方向上敌人的最大数量，假设为 max
- 首先答案 ans 加上 max，然后对于某个价值为 v 的格子，对应边的容量就为 $max - v$，含义就是选择它会损失这么多的价值

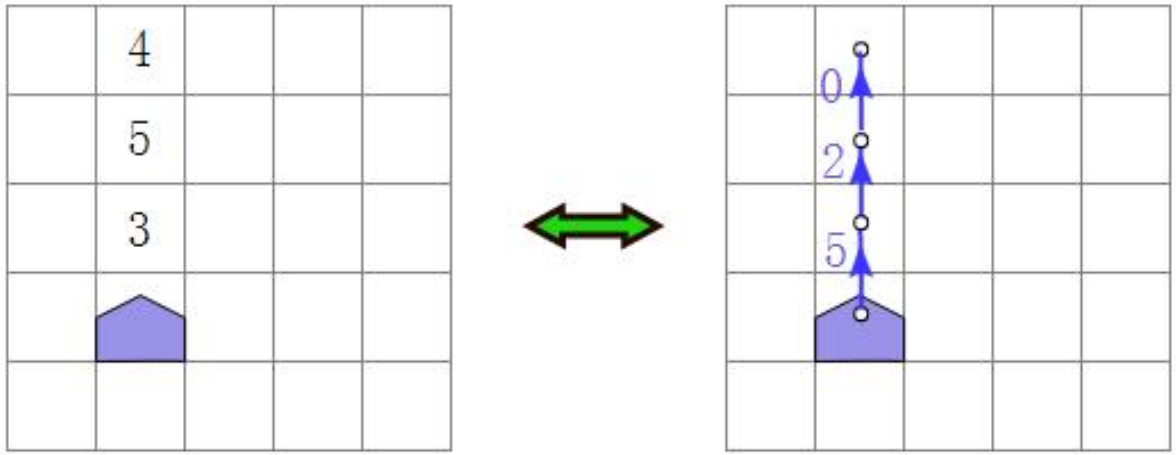
炮塔

- 我们找出某个炮塔攻击方向上敌人的最大数量，假设为 max
- 首先答案 ans 加上 max，然后对于某个价值为 v 的格子，对应边的容量就为 $max - v$，含义就是选择它会损失这么多的价值



炮塔

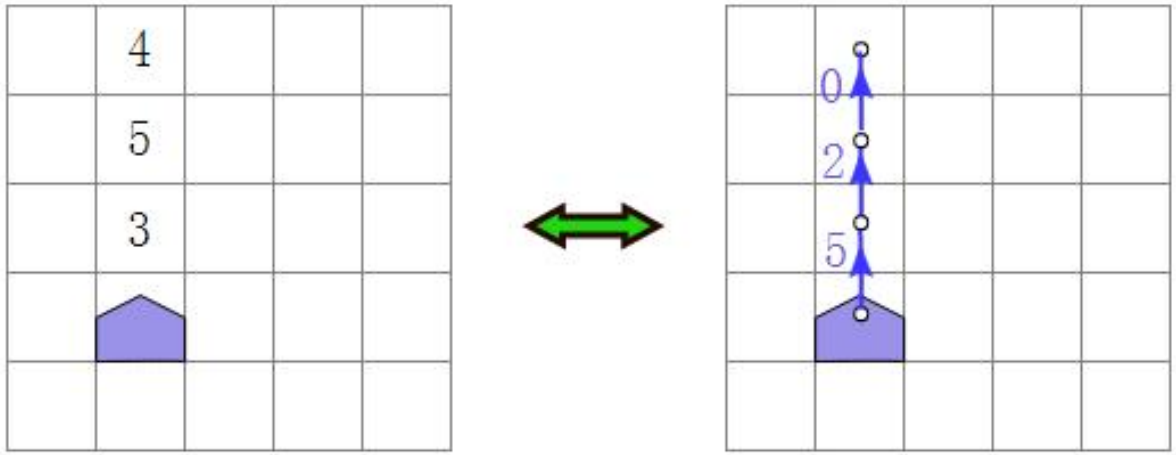
- 我们找出某个炮塔攻击方向上敌人的最大数量，假设为 max
- 首先答案 ans 加上 max，然后对于某个价值为 v 的格子，对应边的容量就为 $max - v$，含义就是选择它会损失这么多的价值



- 横向类似，这里就不给出具体图了

炮塔

- 我们找出某个炮塔攻击方向上敌人的最大数量，假设为 max
- 首先答案 ans 加上 max，然后对于某个价值为 v 的格子，对应边的容量就为 $max - v$，含义就是选择它会损失这么多的价值



- 横向类似，这里就不给出具体图了
- $ans - Mincnt$ 就是答案

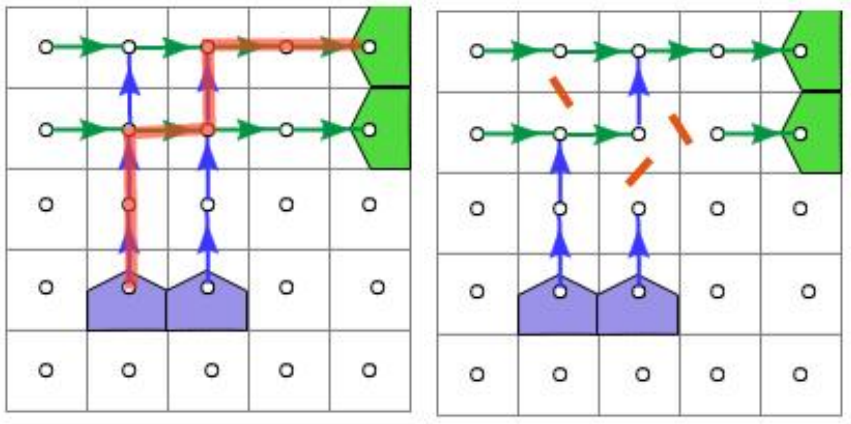
炮塔

炮塔

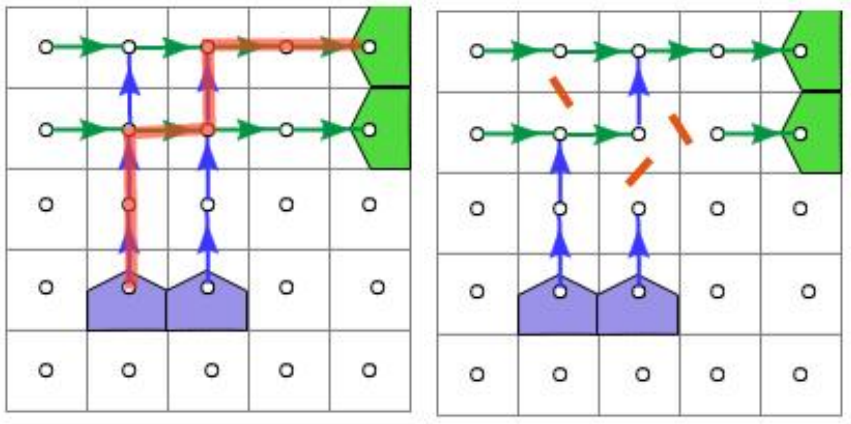
- 然而这个做法还存在一个问题

炮塔

- 然而这个做法还存在一个问题

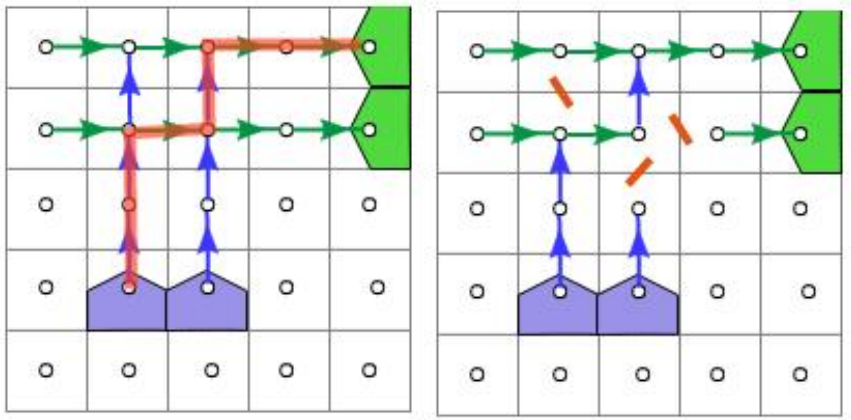


炮塔

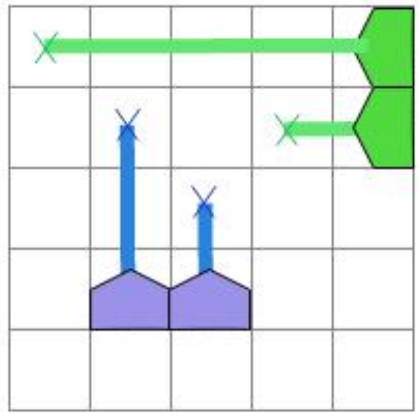
- 然而这个做法还存在一个问题
- 
- 上图虽然存在源到汇的路径，但是却有一个合法的方案所对应

炮塔

- 然而这个做法还存在一个问题



- 上图虽然存在源到汇的路径，但是却有一个合法的方案所对应



炮塔

炮塔

- 问题其实是这条路径并不合法

炮塔

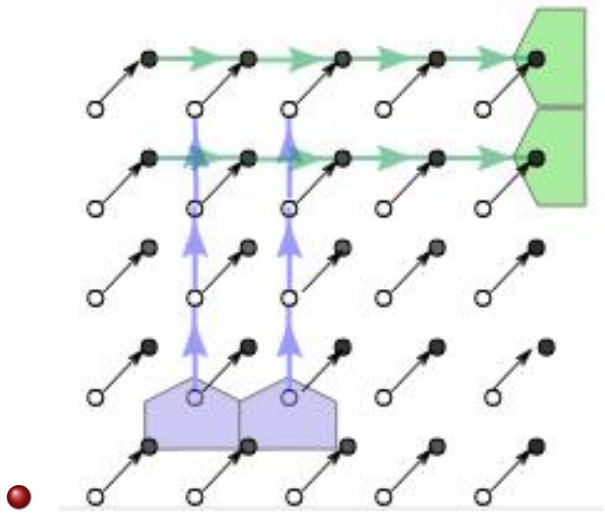
- 问题其实是这条路径并不合法
- 我们之前所规定的那条炮塔间的路径，应该只涉及一行一列，也就是我们不能让它转角多次

炮塔

- 问题其实是这条路径并不合法
- 我们之前所规定的那条炮塔间的路径，应该只涉及一行一列，也就是我们不能让它转角多次
- 解决方法也很简单，我们将每个格子拆为横向点与纵向点，横向点向纵向点连容量无穷大的边，纵向点没有边连向横向点即可

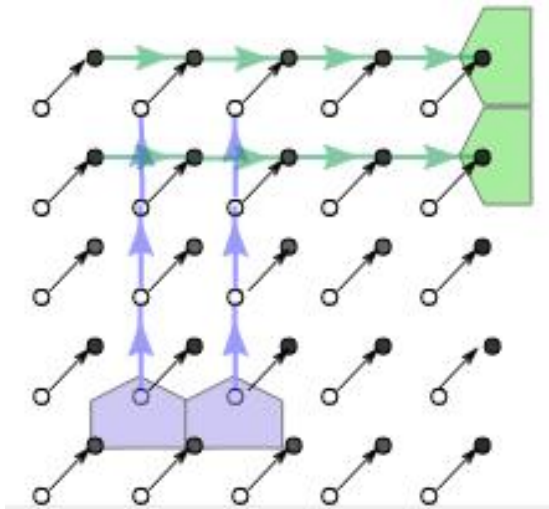
炮塔

- 问题其实是这条路径并不合法
- 我们之前所规定的那条炮塔间的路径，应该只涉及一行一列，也就是我们不能让它转角多次
- 解决方法也很简单，我们将每个格子拆为横向点与纵向点，横向点向纵向点连容量无穷大的边，纵向点没有边连向横向点即可



炮塔

- 问题其实是这条路径并不合法
- 我们之前所规定的那条炮塔间的路径，应该只涉及一行一列，也就是我们不能让它转角多次
- 解决方法也很简单，我们将每个格子拆为横向点与纵向点，横向点向纵向点连容量无穷大的边，纵向点没有边连向横向点即可



- 复杂度 $O(maxflow(n \times m, n \times m))$