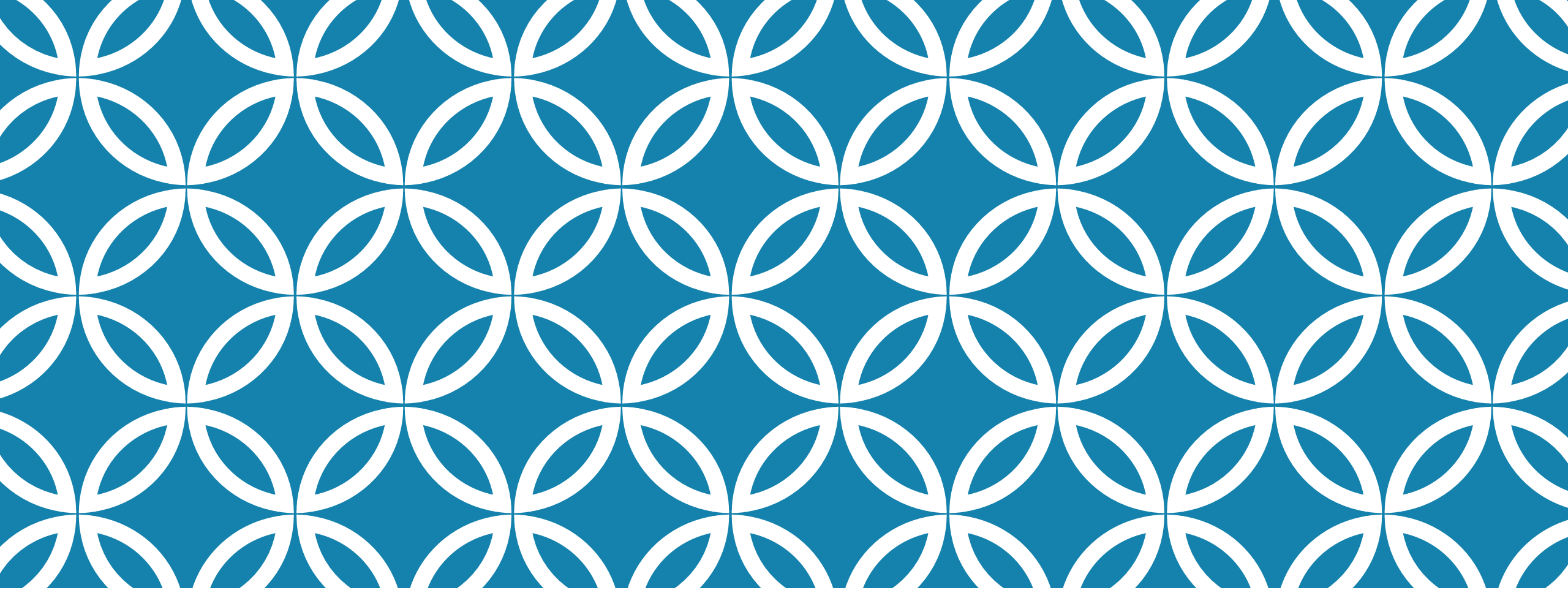
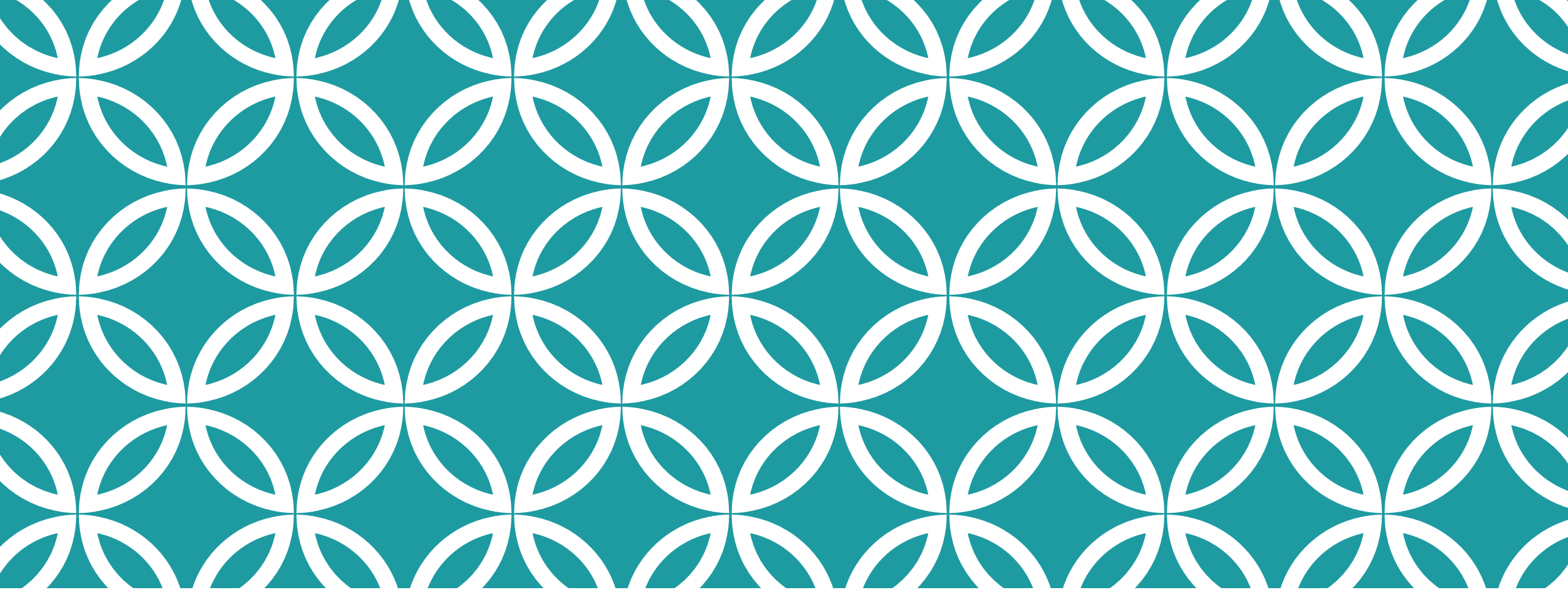


算法设计与分析课设汇报

2304班 算迷小组
汇报人：张佳栋

目录



项目概述

项目概述

本项目旨在开发一款融合经典算法与图形可视化的迷宫探险游戏。通过将分治、动态规划、贪心、回溯、分支限界等算法设计策略贯穿于迷宫生成、资源收集、谜题解锁、战斗系统等多个关键模块，实现算法理论与工程实践的深度融合。

项目主要有以下模块：

1. 分治法生成随机迷宫
2. 动态规划求解最优资源收集路径
3. 贪心算法实时策略模拟
4. 启发式算法实时策略模拟
5. 回溯法解锁机关密码
6. 分支限界法优化BOSS战斗策略
7. 基于 OpenGL 的可视化演示

迷宫生成算法

1. 初始化迷宫
创建二维数组表示迷宫，尺寸为奇数（如 $height \times width$）。
2. 递归分割区域
输入参数：子问题迷宫的左上方坐标和右下方坐标
终止条件：区域宽度或高度 < 2。
3. 选择分割方向
随机选择一个纵向墙体（非外围）和一个横向墙体（非外围），视其把迷宫分割为四个部分作为其子问题并求解，同时在四片墙体中随机选取三个墙体将其打通，由于 2×2 迷宫的特殊性，这样生成的迷宫路径唯一且无环路，且无需合并子问题。
4. 设置入口和出口
入口：在迷宫最外围墙体（非四个角）上随机选择一个作为入口
出口：在迷宫最外围墙体（非四个角）上随机选择一个作为出口，若与出口重合，则重新生成直到不同。

时间复杂度分析

下面用归纳法证明对于处理一个规模为 $a \times b$ 的问题的时间复杂度是 $O(a \times b)$。

由于最小子问题是一个 $1 \times n$ 或 $\mathrm{n} \times 1$ 大小的块，处理这样一个子问题的复杂度是 $O(n)$ 的，满足归纳假设。

假设对于所有 $a < A, \mathrm{b} < \mathrm{B}$ 的问题的复杂度 $\mathrm{T(a,b)} = O(a \times b)$。

对于一个 $\mathrm{A} \times B$ 的问题，假设划分大小为 p, q。划分的复杂度为 $O(1)$。

那么复杂度 $\mathrm{T(A,B)} = \mathrm{O(1)} + \mathrm{T(p,q)} + \mathrm{T(p,B-q)} + \mathrm{T(A-p,q)} + \mathrm{T(A-p,B-q)} = \mathrm{O(1)} + \mathrm{O(p \times q)} + O\big(\mathrm{p \times (B-q)}\big) + O\big(\mathrm{(A-p) \times q}\big) + O\big(\mathrm{(A-p) \times (B-q)}\big) = O(A \times B)$。

所以，规模为 $a \times b$ 的问题的时间复杂度是 $O(a \times b)$。

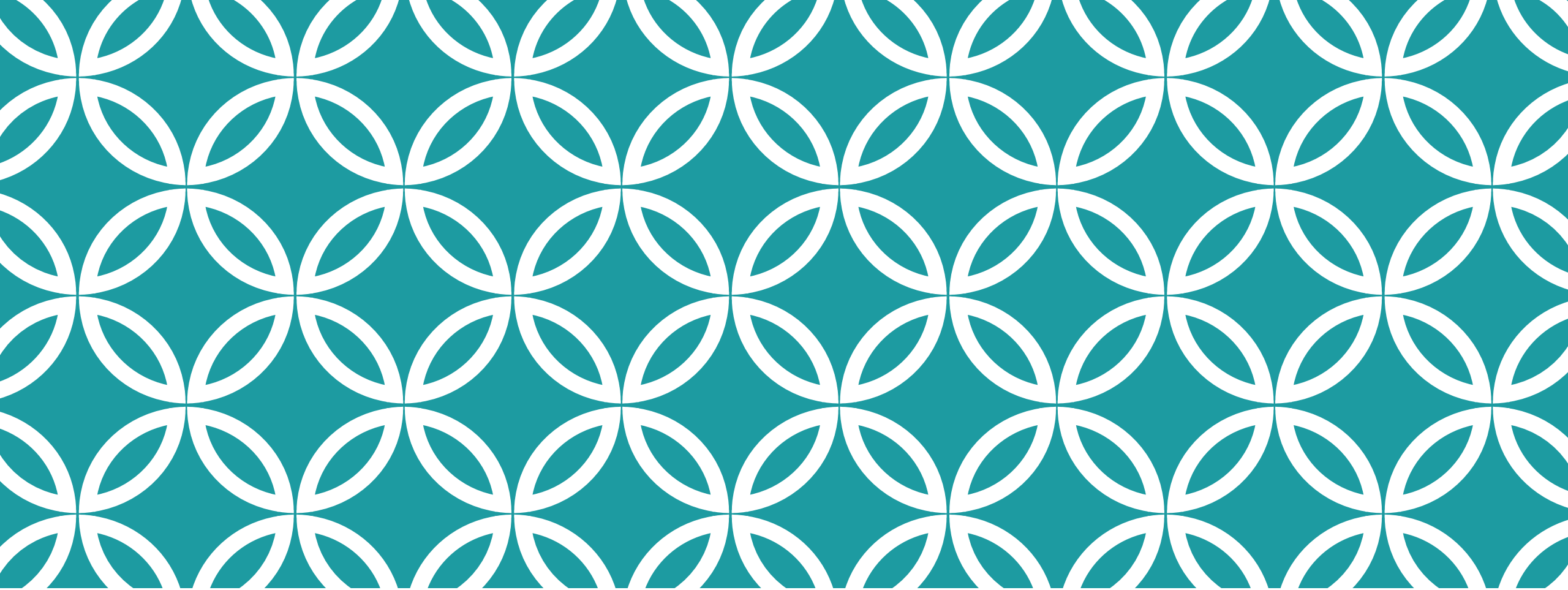
特别的，当 $a = b = n$ 时，复杂度为 $O(n^2)$。

回溯法解密

- ✓ 算法内容：根据线索还原三位密码，支持素数、奇偶性、位置限定等多种线索，密码验证基于 SHA-256 + salt 哈希匹配。
- ✓ 算法思想：枚举所有000~999的三位数字，根据线索进行剪枝，满足线索后再哈希与目标比对。
- ✓ 支持的线索类型：指定位是奇数/偶数，某一位是固定值，所有数字为素数且互不相同。

分支限界法 BOSS 战

- ✓ 算法框架：分支限界法 + 估价函数
- ✓ 状态：BOSS HP数组、技能冷却状态、当前回合、路径
- ✓ 限界函数：f(n) = 当前回合数 + 估算剩余回合
- ✓ 剪枝：若估值 ≥ 当前最优解则丢弃



DP 算法讲解

问题分析

目标：从迷宫的起点出发，到达终点，经过必须经过的机关和 BOSS，收集最多的资源。

分析：地图保证了路径唯一，可知地图实际是一个树形结构，可以采用树形 DP。我们以起点为根，构建地图对应的树。问题就转化为了有一颗带权树，要求找一条从根到终点的通路，使得通路上的权值和最大。

状态设计及状态转移方程

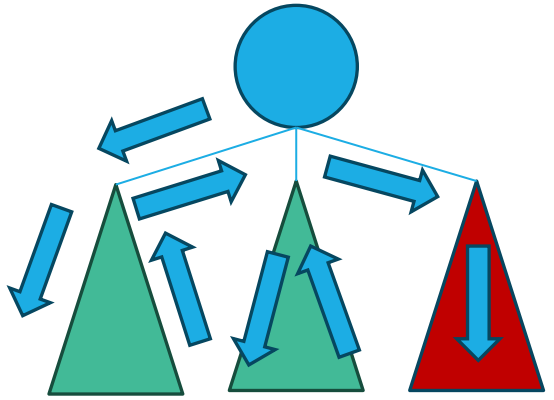
定义 $dp_{i,j}$ 表示从点 (i,j) 出发能获得的最大价值， $must_{i,j}$ 表示点 (i,j) 是否为从起点到终点必须经过的点。那么答案即为 dp_s，其中 s 为起点的坐标。

那么有转移

$$\mathrm{must}_p = \begin{cases} \bigvee\limits_{x \in \mathrm{son}_p} \mathrm{must}_x, & \text{if } x \text{ is not E} \\ 1, & \text{if } x = \text{is E} \end{cases}$$

$$\mathrm{dp}_p = w_p + \sum_{x \in \mathrm{son}_p} \begin{cases} \mathrm{dp}_x, & \text{if must}_x \\ \max(\mathrm{dp}_x, 0), & \text{else} \end{cases}$$

最优路径构建



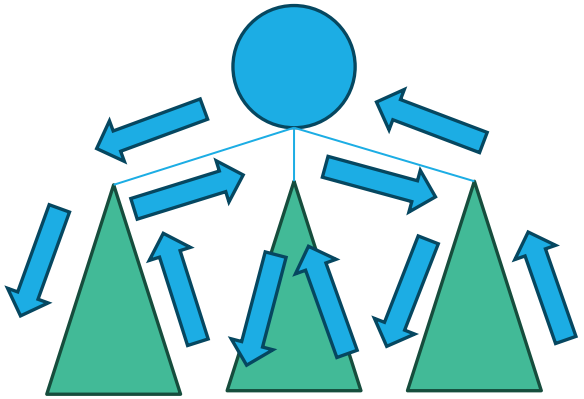
对于不同 $must$ 的点的构建方法不同。

对于 $must$ 为真的点，需要先输出自己的坐标，然后依次构建 $must$ 为假并且采用其中权值的子树的路径，中间穿插自己的坐标，最后构建 $must$ 为真的子树的路径。

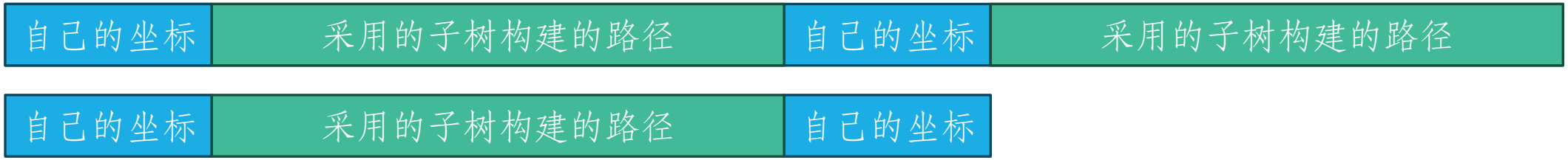
自己的坐标	must为假且采用的子树构建的路径	自己的坐标	must为假且采用的子树构建的路径

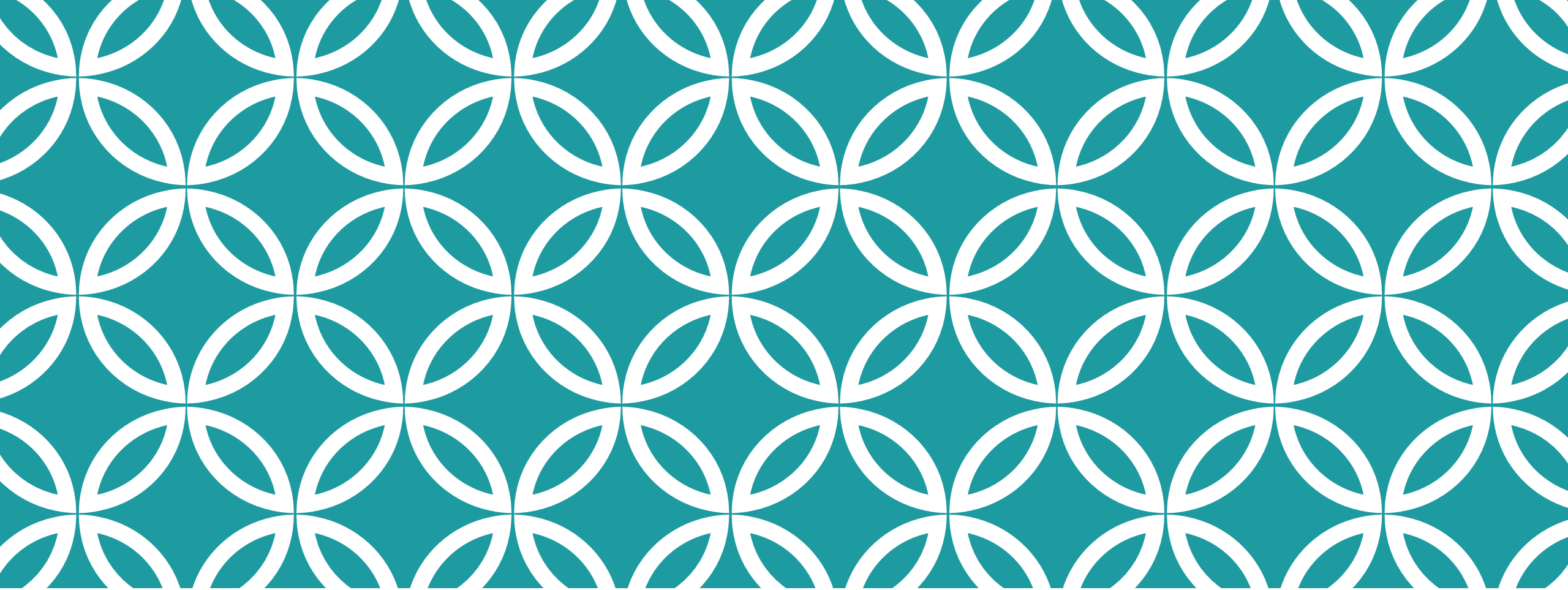
自己的坐标	must为真的子树构建的路径

最优路径构建



对于对于 $must_{i,j}$ 为假的点，也需要先输出自己的坐标，然后依次构建采用其中权值的子树的路径，中间穿插自己的坐标，并且最后也要输出自己的坐标。





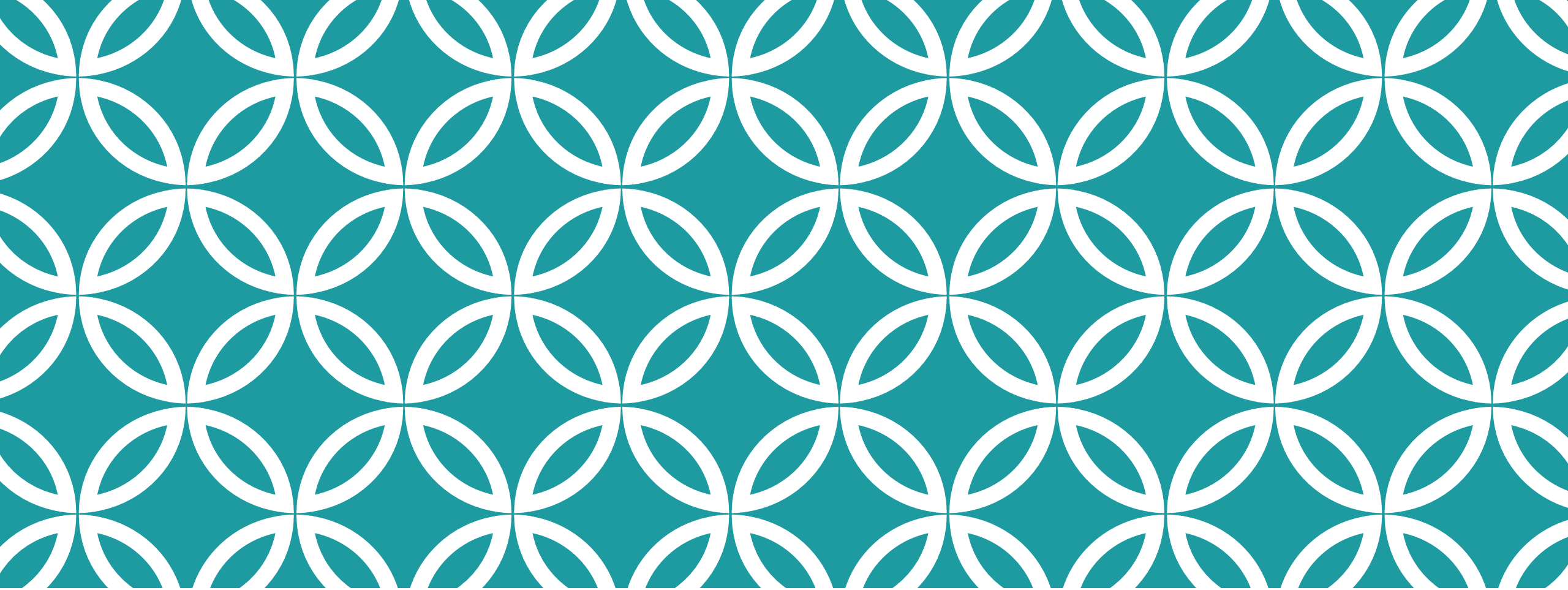
贪心算法讲解

贪心策略

1.优先级驱动的贪心算法
分四个阶段搜索目标(BOSS/机关 > 金币 > 通路 > 出口)
定义目标优先级
每个阶段只考虑当前最高优先级的可行目标

2.记忆已探索地图
维持视野为3*3范围，但记录已探索的区域
利于 收集暂时被搁置的资源/探索待探索的区域
动态更新环境认知
优先利用已知信息，逐步扩展未知区域

3.最优→次优
优先尝试非陷阱路径
失败后才降级允许陷阱路径
当高优先级目标不可达时，会降级搜索低优先级目标



启发式算法讲解

限制描述

在一个迷宫中，主角只能看到以自己为中心的 3×3 区域，目标是在有限视野下逐步探索迷宫，收集金币、避开陷阱、激活机关（B、L），最终到达终点，获取尽可能多的资源值。

原有算法的不足和改进

原本的贪心算法虽然可以找到一个尽可能最大化资源且符合要求的路径，但是有些情况下，它会被陷阱“吓到”而损失了金币，毕竟谁也不知道一个陷阱的背后是另一块陷阱还是一大片金币。换而言之，原本的贪心算法十分“保守”。

对此，我们提出了一种启发式的算法，该算法在原有贪心的基础上，引入了估价函数，对于一片被陷阱挡住的未知区域，我们可以计算其期望的价值来判断是否“冒险”进入。

算法描述

为了实现以上改进，我们优化后的算法有以下几个部分组成：

- ✓视野更新机制
- ✓优先级策略设计（策略核心）
- ✓预期资源收益估计

视野更新机制

1. 只更新当前位置周围 3×3 的格子；
2. 未探索区域标记为未知；
3. 每当角色移动到新位置，就拓展一圈视野，更新地图。

优先级策略设计（策略核心）

核心目标：在已知有限信息下，选择最“有希望”的方向前进。

在没有找全 BOSS 和机关的情况下：金币=通路=BOSS=机关>陷阱>终点

在找全 BOSS 和机关的情况下：金币=通路=BOSS=机关>陷阱=终点

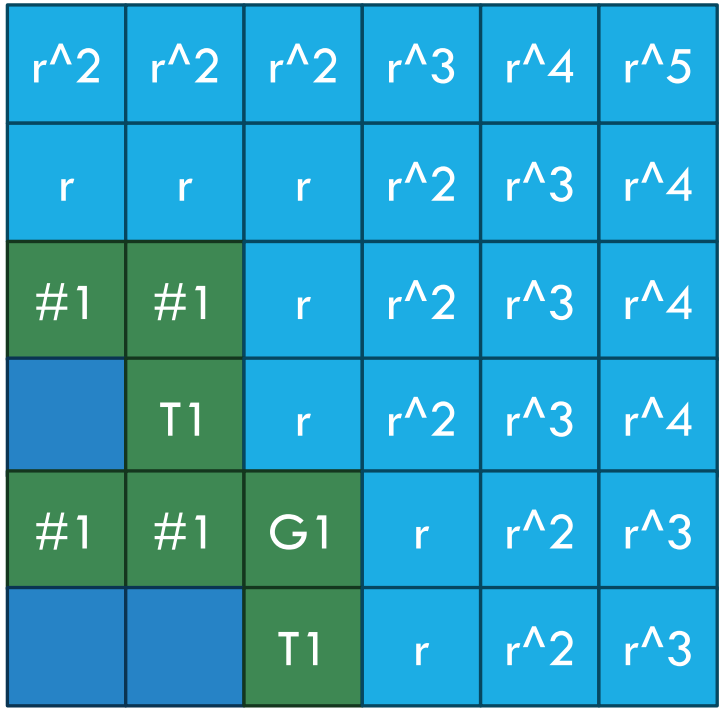
对于找全 BOSS 和机关的情况下如果可见且未到达方块中只有陷阱和终点，我们需要计算陷阱和终点的预期资源收益估计，如果一个陷阱的预期资源收益估计为正，我们就会选择去“探索”。

预期资源收益估计

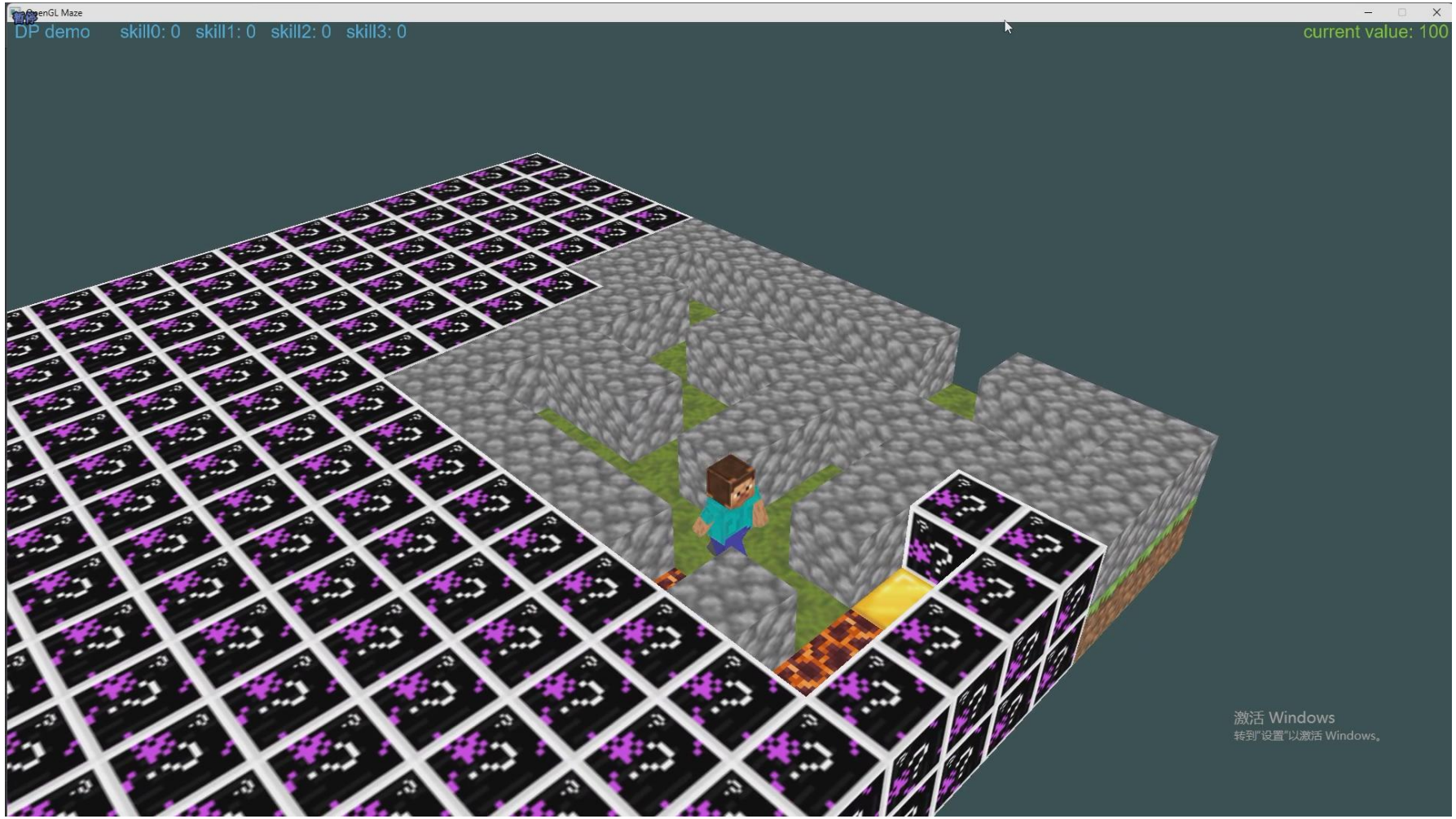
使用类似启发式扩展的方法，预测从某个点出发能收集多少资源：

可见格子收益为 1 倍；

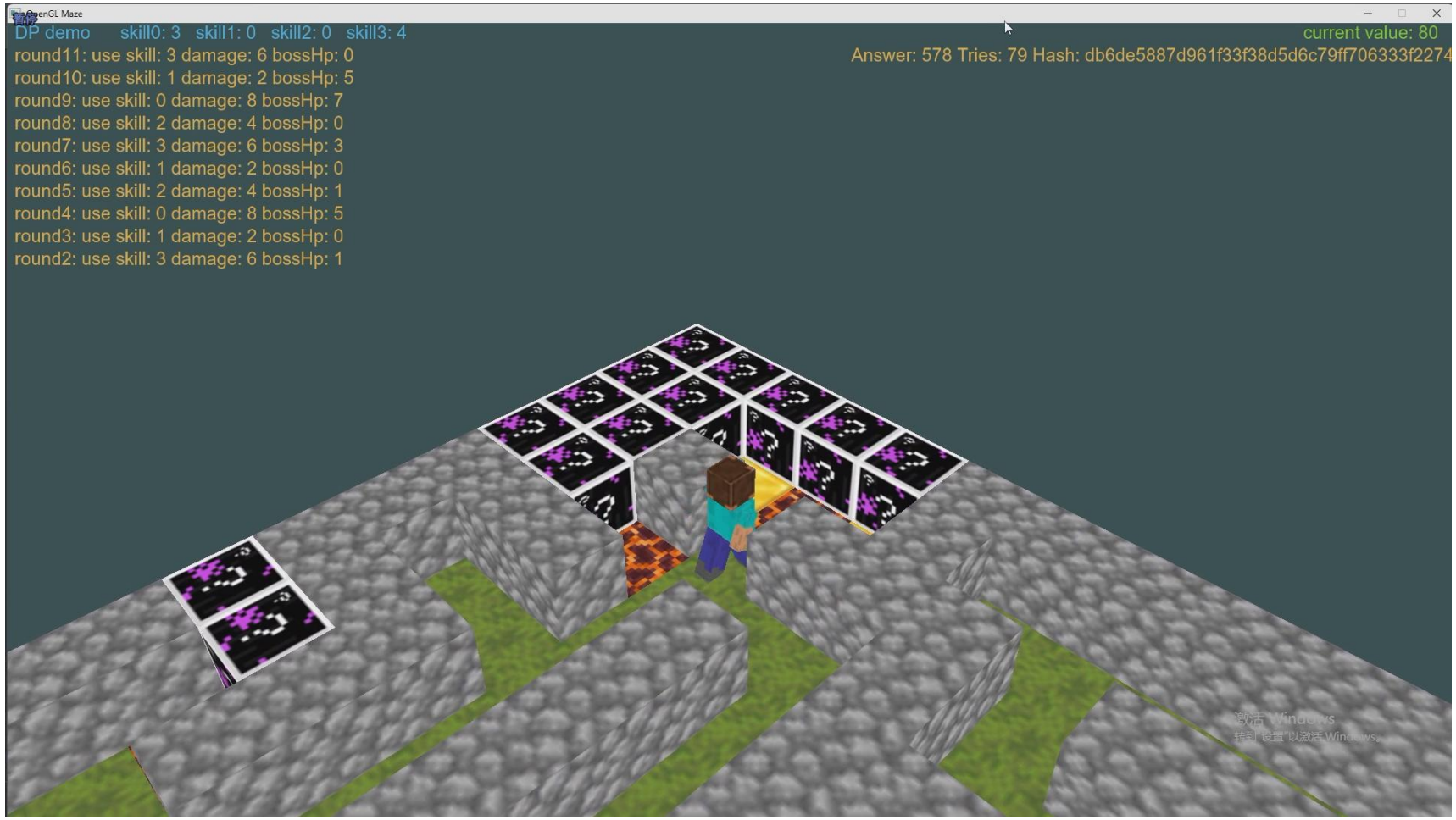
未知格子按经验折扣估值，乘一个系数 r；



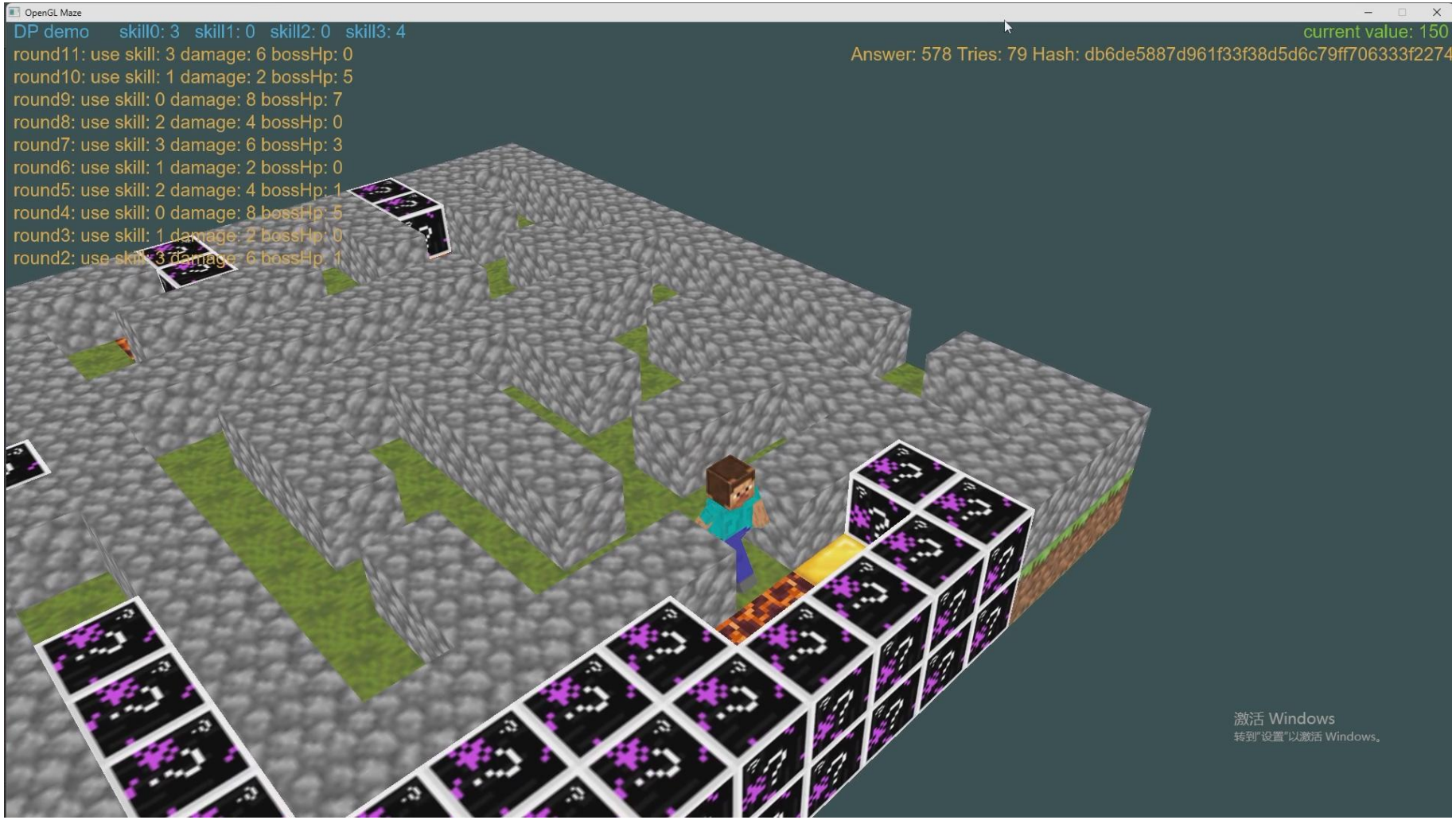
例子


OpenGL Maze
DP demo
skill0: 0
skill1: 0
skill2: 0
skill3: 0
current value: 100
激活 Windows
转到"设置"以激活 Windows。

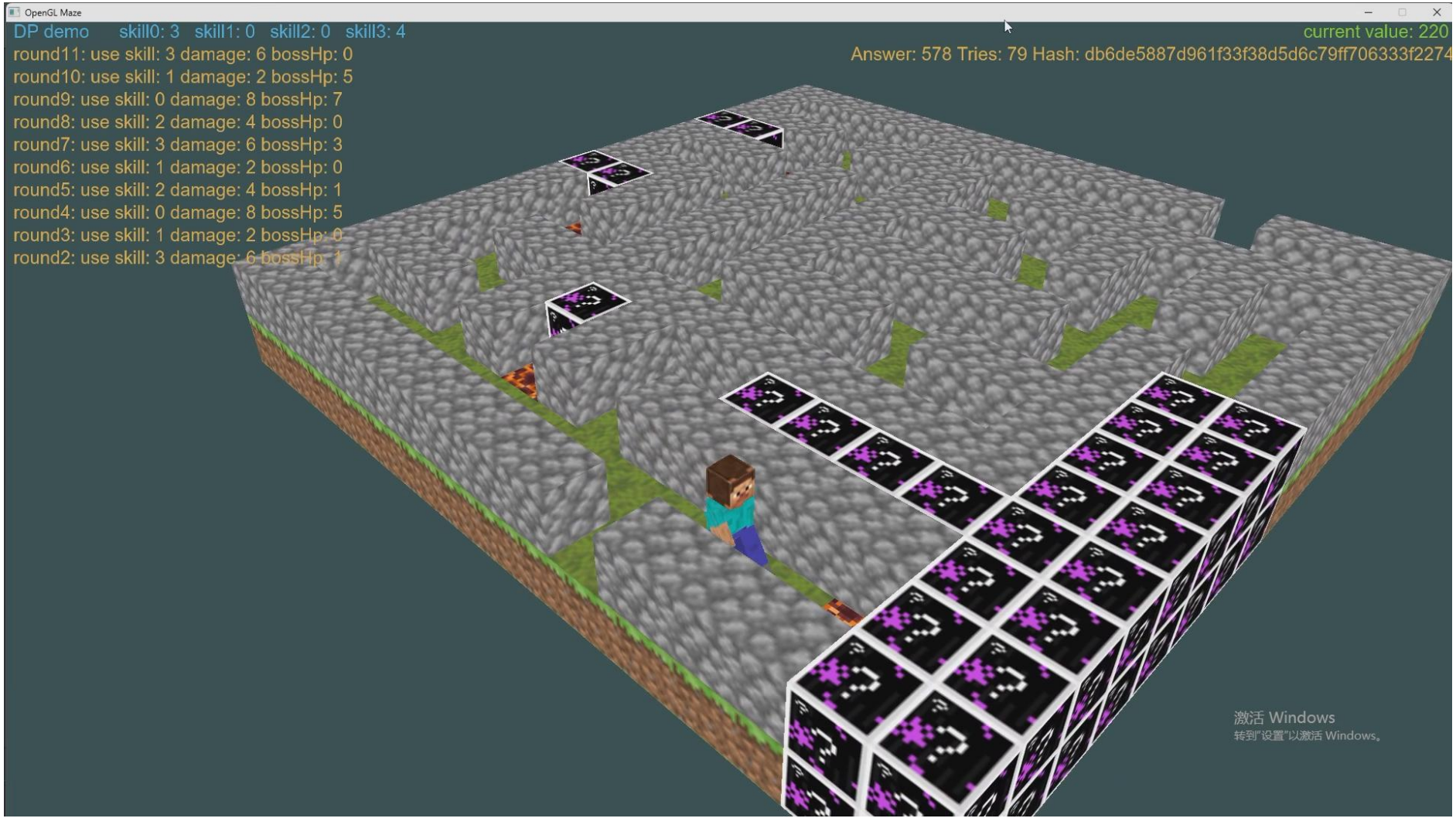
例子


OpenGL Maze
DP demo skill0: 3 skill1: 0 skill2: 0 skill3: 4
round11: use skill: 3 damage: 6 bossHp: 0
round10: use skill: 1 damage: 2 bossHp: 5
round9: use skill: 0 damage: 8 bossHp: 7
round8: use skill: 2 damage: 4 bossHp: 0
round7: use skill: 3 damage: 6 bossHp: 3
round6: use skill: 1 damage: 2 bossHp: 0
round5: use skill: 2 damage: 4 bossHp: 1
round4: use skill: 0 damage: 8 bossHp: 5
round3: use skill: 1 damage: 2 bossHp: 0
round2: use skill: 3 damage: 6 bossHp: 1
current value: 80
Answer: 578 Tries: 79 Hash: db6de5887d961f33f38d5d6c79ff706333f2274

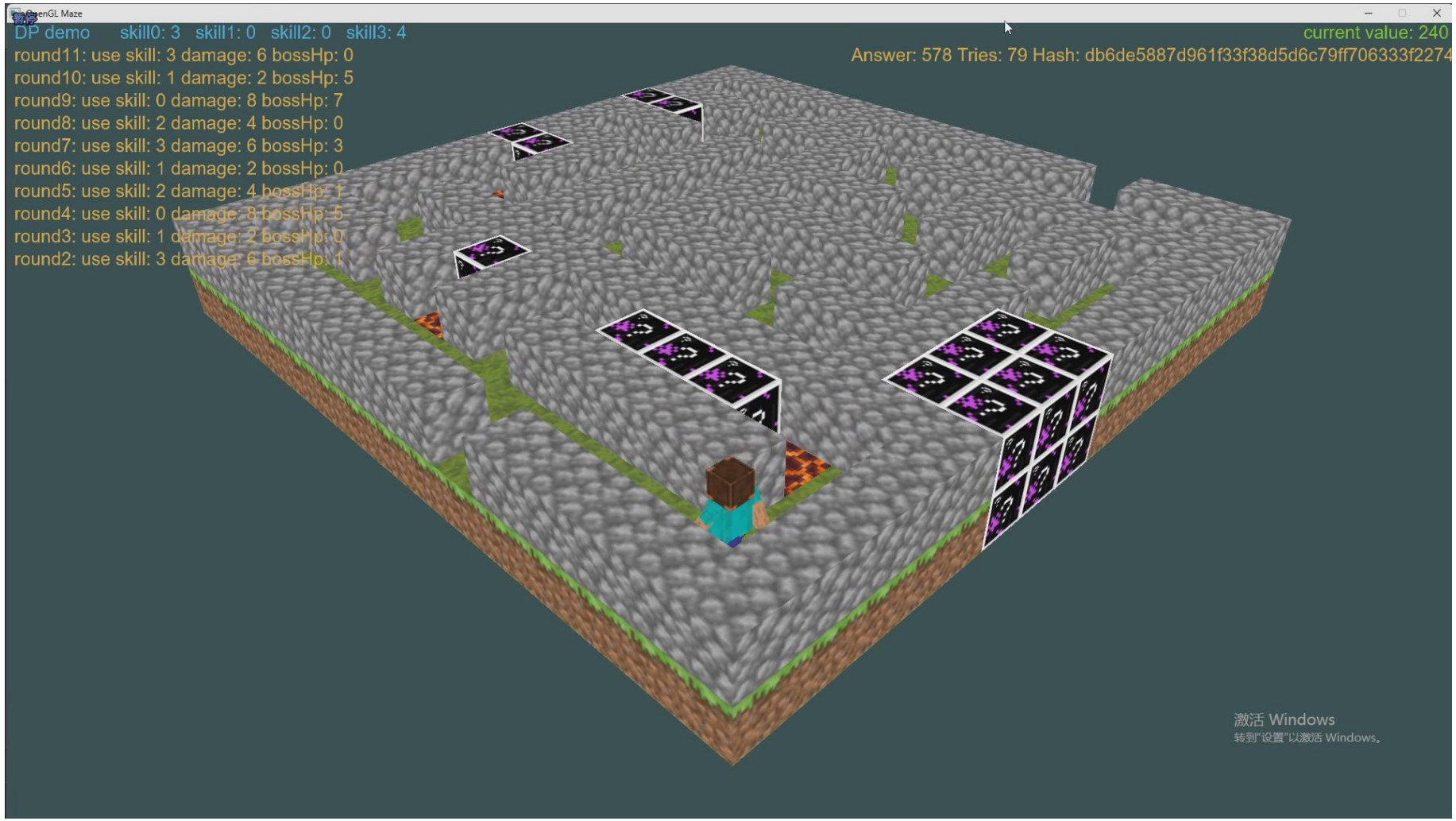
例子


OpenGL Maze
DP demo skill0: 3 skill1: 0 skill2: 0 skill3: 4
round11: use skill: 3 damage: 6 bossHp: 0
round10: use skill: 1 damage: 2 bossHp: 5
round9: use skill: 0 damage: 8 bossHp: 7
round8: use skill: 2 damage: 4 bossHp: 0
round7: use skill: 3 damage: 6 bossHp: 3
round6: use skill: 1 damage: 2 bossHp: 0
round5: use skill: 2 damage: 4 bossHp: 1
round4: use skill: 0 damage: 8 bossHp: 5
round3: use skill: 1 damage: 2 bossHp: 0
round2: use skill: 3 damage: 6 bossHp: 1
current value: 150
Answer: 578 Tries: 79 Hash: db6de5887d961f33f38d5d6c79ff706333f2274
激活 Windows
转到"设置"以激活 Windows。

例子


OpenGL Maze
DP demo skill0: 3 skill1: 0 skill2: 0 skill3: 4
round11: use skill: 3 damage: 6 bossHp: 0
round10: use skill: 1 damage: 2 bossHp: 5
round9: use skill: 0 damage: 8 bossHp: 7
round8: use skill: 2 damage: 4 bossHp: 0
round7: use skill: 3 damage: 6 bossHp: 3
round6: use skill: 1 damage: 2 bossHp: 0
round5: use skill: 2 damage: 4 bossHp: 1
round4: use skill: 0 damage: 8 bossHp: 5
round3: use skill: 1 damage: 2 bossHp: 0
round2: use skill: 3 damage: 6 bossHp: 1
current value: 220
Answer: 578 Tries: 79 Hash: db6de5887d961f33f38d5d6c79ff706333f2274
激活 Windows
转到"设置"以激活 Windows。

例子


OpenGL Maze
DP demo skill0: 3 skill1: 0 skill2: 0 skill3: 4
round11: use skill: 3 damage: 6 bossHp: 0
round10: use skill: 1 damage: 2 bossHp: 5
round9: use skill: 0 damage: 8 bossHp: 7
round8: use skill: 2 damage: 4 bossHp: 0
round7: use skill: 3 damage: 6 bossHp: 3
round6: use skill: 1 damage: 2 bossHp: 0
round5: use skill: 2 damage: 4 bossHp: 1
round4: use skill: 0 damage: 8 bossHp: 5
round3: use skill: 1 damage: 2 bossHp: 0
round2: use skill: 3 damage: 6 bossHp: 1
current value: 240
Answer: 578 Tries: 79 Hash: db6de5887d961f33f38d5d6c79ff706333f2274
激活 Windows
转到"设置"以激活 Windows。

实验结果

测试方法

对于 DP，贪心，启发式算法在随机生成的 100 个地图下测试，计算平均权值。

结果

7×7

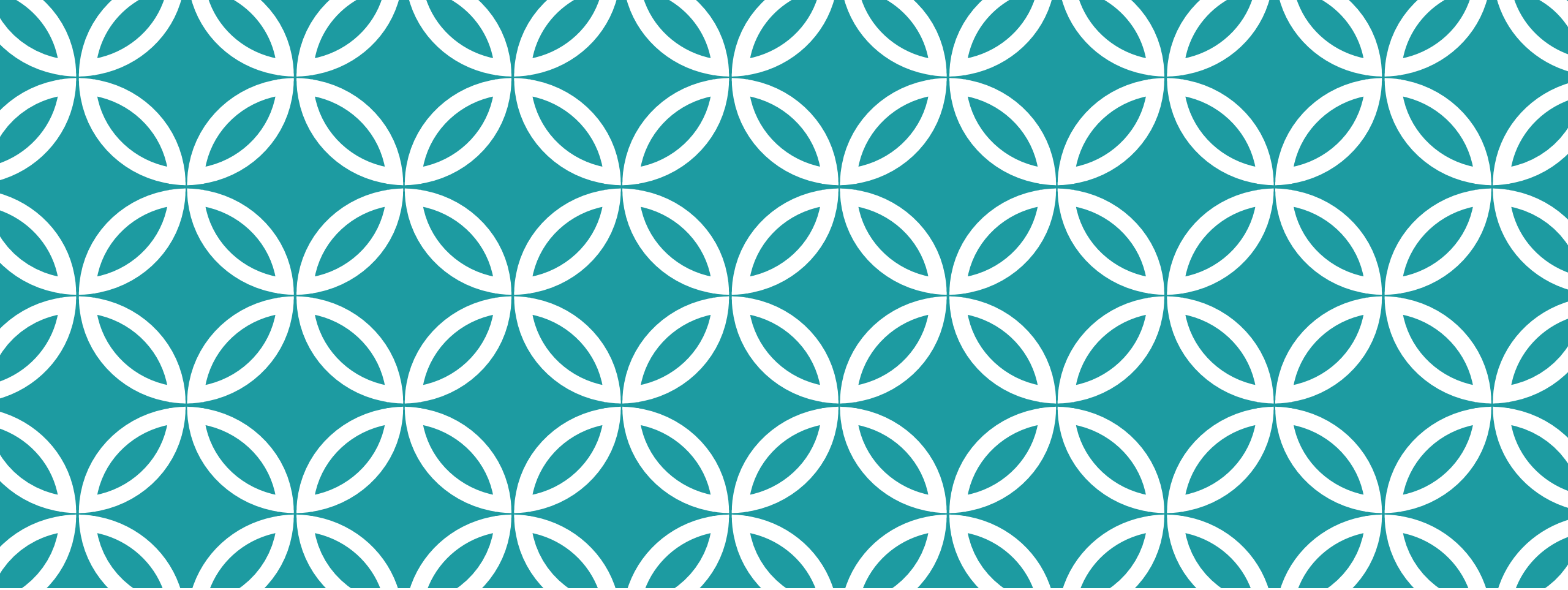
```
sum DP value: 559  average: 5.59
sum greedy value: 510 average: 5.1
sum smart value: 511 average: 5.11
```

11×11

```
sum DP value: 2878  average: 28.78
sum greedy value: 2592 average: 25.92
sum smart value: 2705 average: 27.05
```

15×15

```
sum DP value: 6824  average: 68.24
sum greedy value: 6019 average: 60.19
sum smart value: 6527 average: 65.27
```



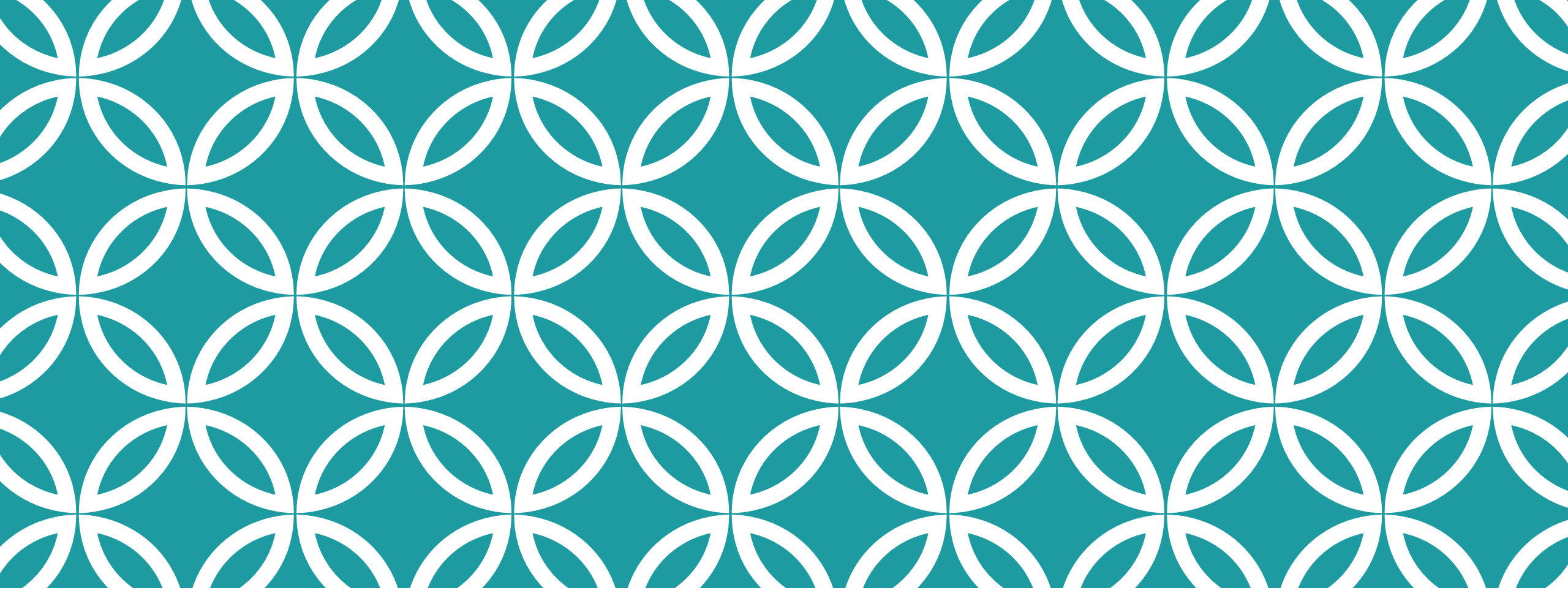
总结

总结

DP 得分最高：由于拥有全图信息，它能保证路径最优；

贪心策略表现最差：虽快速但容易落入局部最优；

启发式方法综合表现优秀：仅凭有限视野，得分接近全图最优解，展现出“预期收益评估 + 策略优先级”的强大决策能力



感谢聆听