四子棋实验指导书

关于本实验

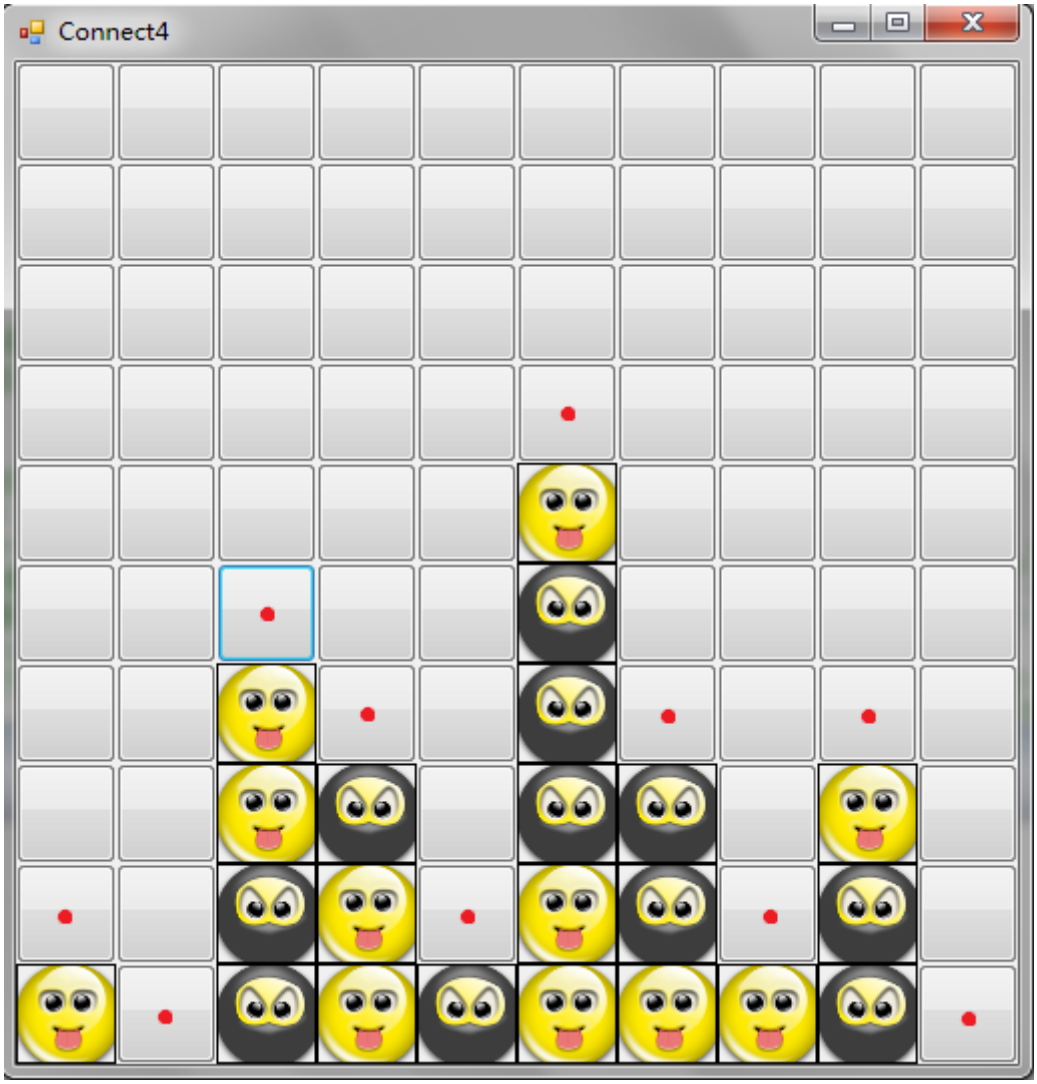
在人工智能领域，博弈问题一直以来都是一类具有代表性的研究课题，博弈问题中一些新课题的提出，也不断的推动着人工智能学科的发展。

本实验以四子棋博弈游戏为背景，主要包括了剪枝算法、棋局评价、威胁检测与排除以及同学们自己的创新性算法等方面，主要锻炼大家对人工智能算法的实际应用能力，从而使大家对人工智能有一个更为直观和真切的体会。

四子棋游戏介绍

基本游戏规则

游戏双方分别持不同颜色的棋子，设 A 持白子，B 持黑子，以某一方为先手依次落子。假设为 A 为先手，落子规则如下：在 M 行 N 列的棋盘中，棋手每次只能在每一列当前的最底部落子，如图中的红点处所示，如果某一列已经落满，则不能在该列中落子。在图形界面中，如果在某一列的任意一个按钮上点击，会自动在该列的最低端落子。



棋手的目标是在横向、纵向、两个斜向共四个方向中的任意一个方向上，使自己的棋子连成四个（或四个以上），并阻止对方达到同样的企图。先形成四连子的一方获胜，如果直到棋盘落满双方都没能达到目标，则为平局。

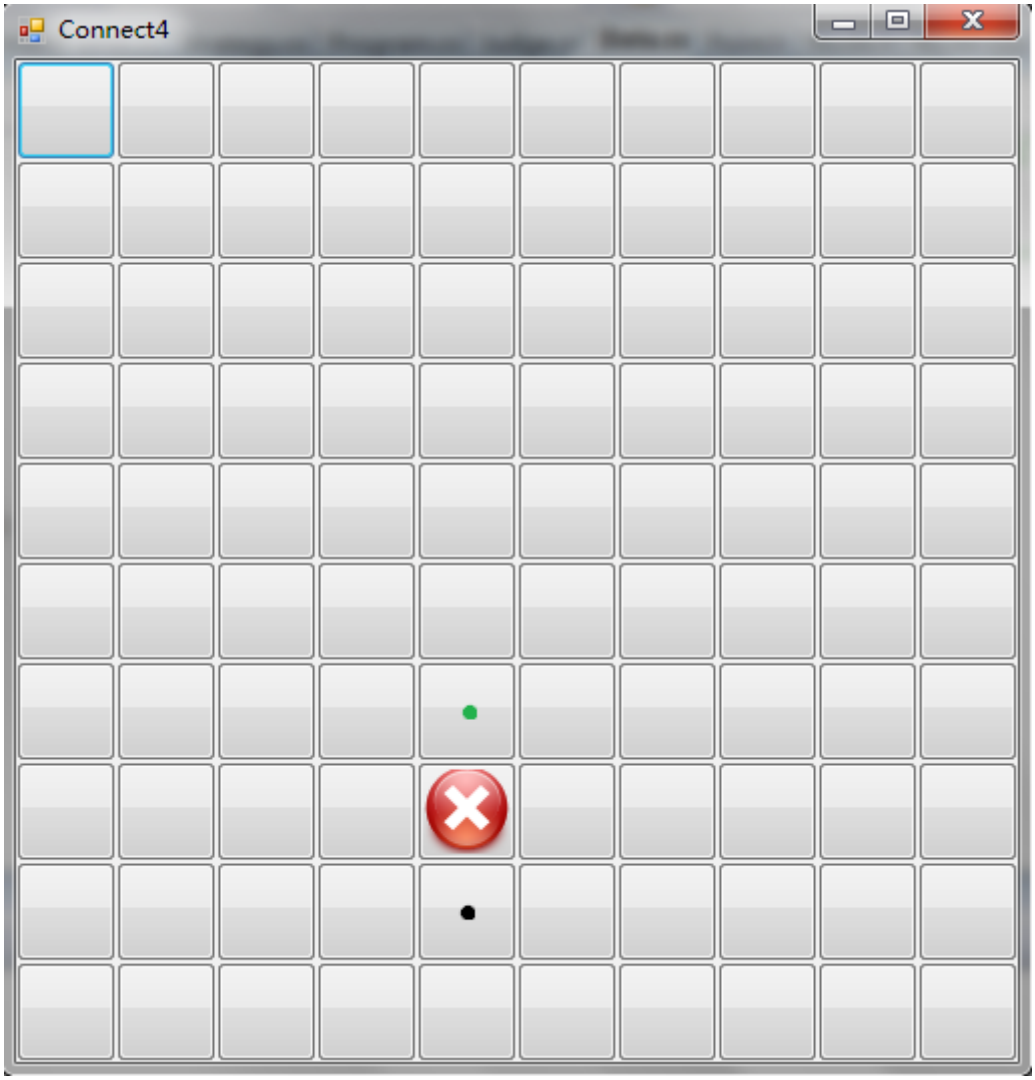
本实验对规则的扩展

如果你查阅一下相关文献和资料，就会发现某些棋盘规模下的四子棋游戏是具有必胜策略的，一个典型的例子是 6 行 7 列棋盘。

在 1987 年，美国计算机学家 James Dow Allen 首先提出了该棋盘下的算法策略，在该策略下先手一方可以保证不输，后手一方最多可以拿到平局。仅在几周之后，荷兰计算机科学家 Victor Allis 也宣布独立地完成了四子棋必胜策略的工作并发表了相关论文。

由于我们实验的评测方法是让大家的策略与己有的样例策略进行对弈，所以如果某一策略完全使用了必胜算法，将造成评测结果没有实际意义，因此我们对规则进行了如下改进：

- **棋盘的大小是随机的，而不是固定不变的，宽度和高度的范围均为** $[9, 12]$**。**由于并不是所有规模的棋盘都有必胜策略，也不是所有的必胜策略都是先手必胜，所以这里带来了不确定因素。你应该使自己的算法尽可能普适。
- **每次棋盘生成之后，会同时在棋盘上随机生成一个不可以落子的位置，如图中的红叉所示。**当某一次落子是在黑点处时，该列下一次可落子的位置就变为了绿点处，而不再是黑点上面的位置。这样防止同学们在自己的策略中针对每一种棋盘大小分别实现必胜算法，再根据棋盘的实际大小选择对应的算法。



当然，以上两个小修改的目的并不是完全阻止你研究和使用必胜策略，而是防止你不加思考地使用必胜算法。实际上，必胜算法的很多思想即使在这两条规则的限制下也是可以发挥作用的，如果你想使自己的策略更加智能、更加有力，就可以学习和研究相关算法，加以吸收和改进，并应用到你自己的策略中，这就需要你发挥自己的创造力了。

实验要求

本次实验的实验要求为，在我们提供的实验框架下完成四子棋游戏的人工智能决策部分，并将你的策略文件与我们提供的不同级别的样例策略文件进行对抗。如果你的策略文件能够打败某一级别的样例，就能获得该级别的分数。

该实验推荐的基本方法为 $\alpha-\beta$ 剪枝算法，当然这不是硬性的限制。本实验中大家可以使用任何可能的算法，只要该方法能使你的策略更强大即可。
在网络学堂作业附件中，我们对不同系统提供了对应的实验框架，见 AI 文件夹。

> 为了支持本地测试，我们对不同系统有不同的“祖传框架”以直接编译得到动态链接库，其中 Windows 配置了 Visual Studio 项目，Mac 配置了 XCode 项目，Linux 是简单的 Makefile。
>
> 但是由于线上平台（也是 Linux）的诞生，你完全可以放弃本地测试，直接使用 Liunx 框架，并用自己喜欢的编程工具配置项目、完成作业。
>
> 这里尤其不推荐使用 Windows 框架，Windows 框架下有一些 MSVC 编译器专用的关键字，每年都会有一部分同学因为没有看平台 提交要求 而编译失败。

无论何种选择，Strategy 项目是你唯一需要从中编写代码的项目，下面将对其进行介绍。

Judge.h、Judge.cpp

用来进行胜局检测的函数，你可能使用得到也可能使用不到，相关函数的具体说明请见代码中的详细注释。这里只强调一些细节：

- 由于在某一方落子得胜之前，一定不可能出现过任何一方的胜局，所以我们只需要在某次落子之后，以本次落子点为核心判断是否有多于四个的连子即可。`userWin` 函数和 `computerWin` 函数就是这样做的，并且我们只需要在 `user` 落子时候判断是否 `userWin`，在 `computer` 落子之后判断是否 `computerWin`。
- 由于 `isTie` 通过检测棋盘是否已满来判断是否为平局，这就意味着调用 `isTie` 必须在调用 `userWin` 或 `computerWin` 之后。假如某次为用户落子，那么之后当 `userWin` 为 `false` 时，调用 `isTie` 进行判断才有意义。

Point.h

`Point` 类，用来记录棋盘上的一个点。

注意框架中对坐标的规定：棋盘中左上角为坐标原点，纵向为 x 坐标轴，横向为 y 坐标轴，(x, y) 点对应于棋盘中第 x 行第 y 列的位置（从 0 开始计而不是从 1 开始计）。

Strategy.h、Strategy.cpp

定义了策略函数同外部调用程序之间的接口。

`getPoint` 函数：在对抗时，外部程序在每次需要落子时通过调用该函数获得你给出的落子点。你的策略最终应当封装到该函数中，给出你在本步中的落子。

`clearPoint` 函数：策略模块中动态定义的对象必须由策略模块中的函数来释放；`getPoint` 函数返回一个 `Point *`，该指针指向的对象应当通过在外面调用`clearPoint` 函数来释放。**你不需要修改该函数。**

注意点

1. 关于接口函数传入的参数
    1. `top` 数组给定了当前棋局各列的顶端位置，这些顶端的上面是你可以落子的地方。如果你的策略给出了非法的落子，那么程序会给出提示。
    2. `board` 二维数组则给出了当前棋局的所有情况。你可以假设自己策略正在同某一用户进行对弈，那么 `board` 中 `0` 为空位置，`1` 为有用户棋子的位置，`2` 为有计算机（自己的策略）的棋。不可落子点处的 `board` 值也为 `0`。
    3. **对于不可落子点，外部调用程序已经做出了处理。**比如某次落子点为 A 位置，而 A 上面的 B 位置为不可落子点，那么下次传递给你的 `top` 数组中该列列顶已被设为了 B，而不是 A，**所以你的策略中不需要考虑对不可落子点进行特殊处理。当然，如果你想充分分析不可落子点，以做出更好的策略也是可以的。**我们将不可落子点 (`noX`，`noY`) 直接传递给了你，你可以充分利用。
    4. 实际操作时，请注意看代码中详细的注释。这将对你理解整个项目有很大的帮助。
2. 最好用 Release 模式编译以保证你的落子速度。**打印操作会严重影响你的策略的速度，所以在最终版本中请务必去掉所有的打印调试信息。**
3. 不要随意修改工程的各种属性。
4. (`lastX`，`lastY`) 给出了上一次对方落子的位置，刚开局时传入的 `lastX` 和 `lastY` 值为 `-1`。当然，为了更好地组织你的策略，你可以不仅仅利用上一次落子这一步信息，而是在你的策略中将其保存下来，综合多步以便决策时使用。实际上，**传入的参数中 `top` 和 `board` 给定了当前的状态，`lastX`、`lastY` 给定了过去的过程，这样也就为你提供了全部可用信息。**

5. 进行思考时，可以认为自己的程序正与一个用户对抗。**对方的棋子用 1 标示，自己的程序的棋子用 2 标示。**
   在线平台网址：https://www.saiblo.net/

由于实验最终评测会在 Saiblo 上统一完成，**为保证练习环境与验收环境的一致性，强烈建议同学们使用 Saiblo 完成实验的日常开发和评测。**

重要说明

- 平台上初始用户名和密码都是 `<prefix_学号>`，其中 `prefix` 为年份和课程缩写（两者之间没有任何连字符号），课程缩写如下：
  - 《人工智能导论》：`IAI`
  - 《人工智能技术》：`AIT`
  - 《人工智能》：`AI`

  例如某同学学号为 `2021010000`，于 2022 年春季选修《人工智能导论》课程，则用户名密码均为 `<2022IAI_2021010000>`，注意不要遗漏了前后尖括号。
- 第一次登录后请修改密码，**如果忘记自己设置的密码请找助教重置，不要使用网站的「忘记密码」功能**，因为所分配帐号的邮箱并非真实邮箱。
- **请在 DDL 之前将最终版本的代码派遣到小组中。**最终成绩以网站评测为准，因此请务必保证提交的代码可以正常编译运行。

> 需要注意的是，平台搭建在 Linux 服务器。对于使用 Windows 和 Mac 系统的同学，请保证代码中不包含平台特定的库（例如 `Windows.h`）。

上传 AI

如果你选择手动上传 AI，首先请参考平台的 四子棋说明 内容以下载平台 SDK 并放置到正确位置，然后按要求打包成 zip 文件。

在 我的AI 页面点击「上传新AI」按钮以上传新 AI，你需要为你的 AI 设置名称、备注（可选），选择打包好的zip 文件，代码语言请选择 `C++ with Makefile [.zip]`。

在完成首次上传后一般不必再创建新 AI，只需要在「上传新AI」下方找到你的 AI 并点击它，然后再点击右边的「新增版本」按钮即可。建议你在完全更改 AI 思路时再考虑「上传新AI」。

创建清华 GitLab 仓库（可选）

除了上面的手动上传 AI 外，你也可以借用清华 GitLab 仓库进行进行自动上传，如果选择该方案，你将不必再使用作业附件的框架，而会得到一个可以直接用作最终提交的简练框架。建议你在有一定的 Git 基础后再考虑此选项。

在 我的AI 页面点击「创建cpp仓库」的按钮后，即可在你的清华 GitLab 中找到对应的仓库。仓库内的初始代码文件与实验框架的保持一致。

> 第一次点击时可能会先跳转到清华 GitLab 的登录与授权认证，请在完成该操作后再次点击「创建cpp仓库」。

> 请勿修改 `sdk` 目录下的代码，否则后果自负。

每当你往仓库中 push 一版代码时，仓库的 CI 会自动将你的代码提交至 Saiblo 平台，省去你手动打包上传代码的麻烦。

快速人机对局

在 我的AI 页面，找到某一版本的 AI 代码，点击右侧闪电状的「快速人机对局」按钮，即可与自己的 AI 进行人机对抗。

批量测试

注意：每人每天的批量测试额度有限，请留意自己的用量。

在 我的AI 页面，找到某一版本的 AI 代码，点击右侧齿轮状的「批量测试」按钮，即可与一系列 AI 快速进行批量测试。

`<<Connect4_100>>`、`<<Connect4_98>>`、……、`<<Connect4_2>>` 的棋艺依次递减。你可以只挑选一部分 AI 进行测试，也可以填入一些新的 AI token 来与其它 AI 进行测试。

可以访问 https://www.saiblo.net/batches 来查看个人历史批量测试。

最终提交

请务必于 DDL 前在**小组作业**内派遣自己最终版本的 AI。在「我的AI」页面派遣的 AI 不作为提交的依据。

为了避免最终评测中出现意外情况，请提前在平台上**充分测试**（例如用批量测试功能与 50 个样例 AI 对战）。

调试

与 `cout` 类似，你可以使用 `cerr` 将调试信息输出到 `stderr` 从而在网站的对局详情页面中查看程序运行时输出的调试信息。注意输出的信息不要超过 Linux 管道的缓冲区大小 64KB，否则可能会出现阻塞而导致运行超时。

建议在最终版本 AI 中去掉所有调试信息，为自己争取更多的计算时间。

请勿使用 `cout` 等向 `stdout` 输出调试信息，否则可能导致你的 AI 无法在平台上正确运行。

时空限制

- 最大内存 1GB
- 单回合最长时间 3s
- 上传代码大小不超过 64MB
- 输出调试信息总大小不超过 64KB

注意：为保证公平性，平台上的所有 AI 只能使用单线程执行。
随网络学堂附件提供给同学们的还有一个 Compete 项目，与 Strategy 在同级目录。该项目实际上是一个非界面、命令行式的对抗平台，你不需要关心这里面的代码。Compete 可执行程序允许你将两个策略文件进行指定轮数的对抗，并给出结果统计，避免你在图形界面上频繁的点鼠标。

具体使用方法可参见该项目下的 `readme.txt`。

基本用法

首先编译自己的 Strategy 项目，得到自己的策略文件。

- Windows 系统：经编译后生成的名为 `Strategy.dll` 的文件即为包含你的策略的策略文件（如果使用 Visual Studio 开发，该文件在 `Release` 或 `Debug` 目录内）。
- Mac 系统：经编译后生成的名为 `libStrategy.dylib` 的文件即为包含你的策略的策略文件（如果使用 Xcode 开发，该文件在 `Products` 下的 `Release` 或 `Debug` 目录内）。
- Linux 系统：经编译后生成的名为 `Strategy.so` 的文件即为包含你的策略的策略文件。

然后调用 Compete 项目下的可执行文件进行对抗测试。

批量测试

如果你希望进行本地批量测试，作业附件的 Batch 目录下提供了 Windows、Mac 和 Linux 系统各自的批量编译、对抗脚本，需要将脚本中的待编译文件及编译器路径修改为本地目录，同时指定对抗轮数等参数。具体使用方法见各自系统目录下 `readme.txt` 文件。

评分方法

在本实验中，我们共有 100 个样本策略文件，它们分别为 `100.dll/dylib/so~1.dll/dylib/so`。这些 `dll/dylib/so` 文件事先通过循环赛的方法确定了排名，其中`100.dll/dylib/so` 的棋艺最高超，`1.dll/dylib/so` 的棋艺最差。我们将偶数文件（`100.dll/dylib/so`、`98.dll/dylib/so`、……、`2.dll/dylib/so`）提供给大家用于训练和测试，保留奇数文件（`99.dll/dylib/so`、`97.dll/dylib/so`、……、`1.dll/dylib/so`）用于最终的评测。

1. 只要你的算法能够正常运行，不论其智能程度如何，都能得到基本分 40 分。正常运行主要包括生成的策略文件能够被测试平台正常载入和使用、程序不出bug、不给出非法落子（棋盘外/不在列顶/不可落子点处落子）、不超时等方面。
2. 实验报告占 20 分。实验报告应描述智能算法的基本思路、方法和评测结果，中英文均可，我们会仔细阅读实验报告并给出综合得分。
3. 在如上的基础上，我们会让你的策略文件同保留的 50 个评测文件的每一个进行比赛，采用积分制。与每个评测文件对抗 2 次，分别为先手和后手，共对抗 100 次。每次对抗中，胜利得 2 分，平局得 1 分，失败不得分。最终的积分被划归到 40 分满分（总积分*40/200），作为及格分之上的加分。

在同每一个对手进行对弈时，你的程序可能出现的情况如下表所示：

你的程序	比赛判定结果	积分
胜出	你赢	2
失败	对手赢	0
平局	平局	1
出 bug	对手赢	0
给出非法落子	对手赢	0
某步落子超时	对手赢	0
无法载入策略文件	对手赢	0

你的程序	比赛判定结果	积分
找不到需要的接口函数	对手赢	0

单步落子的时限定为 3s，这个时限仅仅是为了对死掉的程序进行处理，所以为了保险起见，建议你不要为了搜的更深而让自己的程序单步时间特别接近3s，考虑到测试服务器的性能在不同运行环境下有可能不同，所以特别接近于3s 将是危险的。

作业提交

- 在 Saiblo 平台 **小组作业** 内派遣自己最终版本的 AI。
- 在网络学堂上提交实验报告。

报告要求

- 实验报告格式为 PDF 或者 Word 文档，需要列出**对抗结果的统计数据，包括<胜数、负数、平数>**。
- 在实验报告中，你应当对你的策略进行充分的说明，根据你自己的实现充地阐释你所设计的方法，以突出你为什么觉得这样的算法将是聪明的。在将你的程序与样例进行对抗的同时，我们也会认真评价你所采用的方法的创新性，这将让你获得额外的加分。
- 列出统计数据时，建议你附上同50个样例AI对战的批量测试结果的网址以便助教核实，但不做强制要求。

迟交说明

> 迟交说明最后更新于2022年春季，该部分可能每学期不同，请以课程要求为准。

因第三次作业 DDL 靠后，原则上不再接受补交，请按时提交作业，尽量避免 DDL 当天 rush。