表示棋局的方法-当前的状态

走法规则 从所有走法选择最佳 对局面评估：当前局面，推演局面

棋盘表示 二维数组 用数字代表棋子，0表示没棋

走法产生（）

搜索技术：向下判断若干步 假定我们想到的对手也考虑到了

时间复杂度 alpha\_beta剪枝 迭代深化 置换表 历史启发 优化手段

估值：不太可能一直搜索到结局状态，这需要判断所有双方可能性 有限深度的当前局面评估 估值函数不一定是可靠的

博弈缺陷 和补救 水平效应 开局库 残局库 循环探测

{1，2，3，4，5，6}

{7，8，9，10，11，12}

早期甚至用4位表示一个交点 一个字节是两个交点（需要移位操作） 45个字节

比特棋盘 64位数（8\*8棋盘）表示棋子位置，不设棋盘数组，全部用棋子位置表示

3、走法产生，下一步可以走哪？ 五子，扫描空位 象棋，每个棋子可以走哪？

列举所有合法的走法 把走法输出

优化 直接保存每个棋子在每个位置的可走坐标 ，阻碍坐标，然后取出来，使用方便查找的数据结构 比如结构表

用已经算出的数据，来避免即时计算 建立小型数据库

取内存数据会快吗？在mov []时 跟高速缓存有关系吗？ 内存分页？ 虚拟内存会加载全局数据吗？什么时候加入？

搜索：从一种走法搜索起，列出所有的走法搜索

剪枝算法 不需要全部走一遍，这时候可能需要对走法进行排序，然后只走最好的

走法队列 放在预先的内存，需要多大？ 一个局面有多少走法？

走法要算几层？ 每层同样的空间？

四、基本搜索技术

扫雷 搜索，逐一翻开 盲目搜索 两人扫雷：交替搜索，对抗性搜索

博亦树 根是初始局面 子节点A的每种走法局面 子节点的子节点是对应的B每种走法的局面

最后的叶子节点是胜利、失败、平局

倒推出胜利的局面下法

包含所有可能的是完全搜索树 由于越向下，可能越多，应该是不会这么做的

很多情形是无法到达叶的，比如来回移动

所以应该给当前局面找一步好棋，评估棋局的优劣 建固定深度的树 估值函数肯定是不精确的，否则就能直接算出胜负了 利用知识构成估值函数-启发函数-启发式搜索

深度优先搜索： 是先生成树 还是只产生搜索的节点 有必要整棵树吗？ 搜过删除

只生成树的一部分，搜过后删除

返回值给B 极大极小树 甲方取极大，乙方取极小 在所有步骤中取极大、极小

每个节点都返回估值

1. 局面的极大极小判断

选择的是最终为胜的节点分支

我们需要选择的只有下一步，中间的局面只是最终状态的过渡

2、负极大值 双方都取极大值

五、估值 1、棋子估值 按照经验估值？ 边价值=存活棋子\*值

棋子灵活性：活动范围 走法数量 走法价值

2、棋子关系评估 走法 棋子被对方威胁，价值降低 没有被威胁，价值不变

判断当前是谁走棋，如果是对方的话，威胁较高，减价值高 如果是自己的话，威胁较低

3、两边价值差

极大极小 最终价值都是 红-黑

负极大

搜索过程，判断这条分支是否分出胜负，已分出胜负就停止向下搜索，返回估值

没分出胜负在搜索树最后节点进行局面估值（并不需要每一步都估啊，这没意义），把最终分支估值返回， 下出最优的一步

六、 1、界面 接受输入（下棋） 显示输出（AI的下棋）

2、走法产生 向搜索程序提供所有走法 3、向下搜索n层， 每条分支估值，然后选出最好的一条

象棋 二维数组 宏名称 按顺序编号 黑帅 黑车 ，黑马 …………

判断函数 是否黑（） 是否红（） 两棋是否同色（）

走法结构体 { chess from to value} 棋子位置 (x,y)

走法产生： 每一层都要放走法队列（包含所有棋的走法）

检查走法是否合法（） 走法列表 movelist[层数][走法数] 插入新走法add()

产生某棋在某位置的所有走法（） 产生所有走法（）

注意 有的棋是跨界的，所以红黑的走法可能性和棋子值没区别？

是否非法：首先该走法需要判断目的地的棋子 走在原位 吃自己的棋 老将见面

出九宫格？ 马脚 象眼

理论上应该判断基本错误（位置错误），不过理论上生成走法时就不该出错，除非是很粗糙的走法判断，比如将可以向任何位置走 士可以出九宫

搜索 最大深度 走法产生器 估值引擎 走下一步

产生走过的棋盘 恢复为上一个棋盘 判断是否胜负？ 最佳走法

走法队列每层是静态的还是动态的？ 理论上需要一个固定的多层二维数组，开空间

，然后每层搜索的过程中都 产生走法队列。 如果返回上层，切换分支，则需要重算下层可能走法

深度优先是一个后序吗？ 先子后父

：如何推出一个方向？ 从下往上推

首先直接到某分支叶估值，返回，然后到另一个叶，估值，返回 从这些节点选最大值作为父节点的值 ，然后其他父节点同样如此估值 随后父节点们取最小值给爷爷节点，爷爷们继续比较 最大值给曾祖父

值只在叶节点计算 剩下的是比较和筛选的过程， 任何搜索方法都必须算出所有叶节点的值

走法数组， 分支切换的时候是要重新算

当前局面->产生走法->选择走法之一->

产生局面->产生走法->选择走法之一->

产生局面->产生走法->选择走法之一->

产生局面->产生走法->选择走法之一->

产生最终局面->估值

->选择走法之二->产生最终局面2->估值……产生最终n 估值

回退一层 选责走法之二->产生局面->产生走法->->选择走法之一->产生最终局面n+1

需要根据不同局面，产生不同走法路径，到最终走法

局面估值

估值 基本价值 灵活性分数 被威胁 被保护 总价值 附加值

当发现可以将死时（此时节点估值应该为最大）

**根的枝等于叶的值，多选一**

七、搜索改进

Alpha\_beta 数据冗余

alpha剪枝 A B C DE A取BC最大 C取DE最小 B=18 D=16 C=min(DE)<=16

A=max(BC)>=18 此时说明C以下的所有分支都不需要了

beta剪枝 当A为小节点时 B=8 D=18 同样的道理 A=min(BC)<=8 C=max（DE）>=18

C和以下的分支同样得剪掉

在奇数层alpha 偶数层beta 这是一个子节点返回值判断 需要三层

用负极大值的话只用每层进行beta

需要分支排序，加快效率

正负无穷 alpha,beta搜索边界（初始是正负无穷大，向下搜索的时候范围缩小）

Alpha=min() beta=max()

，递归子节点

2、返回当前估值？ Fail\_soft alpha beta

3、渴望搜索

4、极小窗口搜索PVS算法

5、置换表 不同分分支有完全相同的子节点（树）：完全相同的局面和下层分支

已经搜索过的节点结果用表记录

如果已经有记录的节点，就直接从表中取出

保存节点需要多少空间？ 不同节点，不同子树完全是不同情况，叶值不同，组合非常多 ，

而且有一层子树 二层子树 多层子树

如何从空间查找到？顺序查找非常慢的 ，如果排序的话也很耗时间

6、哈希表 查找快， 加入快，不排序移位

每个局面都放在表中是不可能的 只能把部分局面放相同位置

节点hash地址 校验值如何生成？ 表中数据是否想要的项？

Hashtable[index].checksum==Check(key) 每个key有一个地址，一个校验值 我们用key

哈希表每次搜索不用清空，直接覆盖和查找 ，冲突不用解决，直接覆盖已有数据

如何快速求局面哈希（增量法） 把棋盘数组和棋类型作为3维数组

填满随机数 然后每个局面把所有棋子[type][x][y] 相加得到哈希

8、迭代深化

控制搜索时间，超时则退出？

9、历史启发

历史得分

历史记录 搜索过程对走法进行得分 然后搜索时按照得分顺序走

所有走法的from to xy [xy][xy] 00-8181 二维数组保存[from][to] 这个数组太大，建议用二进制表示

权值： 好走法得 2^depth分

根据得分，对产生的走法队列排序 alpha 大-小 beta 小-大

10、杀手启发 引发剪枝最多的走法 每一层都记录（最多剪枝） 同样层的时候优先使用杀手走法

11、SSS算法 状态 路径 状态图表

复杂难理解，内存需求大，执行速度慢，要维护有序队列

13、MTD（f）算法 多次alpha beta 空窗 极小窗口

空窗检测，内存记录

14、组合使用 根据层数选择不同情况

八、估值优化

1、提高估值速度 扫描棋盘棋子关系

终点评估 列举所有状况,综合算 互动关系

棋子价值表精度低 T[type][x][y] 一个三维数组， 每个棋子不同位置的值 局面值是所有值的和 没有棋子关系 威胁 保护

1. 估值 准确和性能

性能=速度\*知识

知识量应该多大？才能估值准确 深度重要还是知识重要

高层知识重要 低层深度重要

1. 估值和比重

测试是动态输入（比如循环，随机，或者从表中取，不同类型，不同内容），检查返回值（或者其他输出，比如修改的外部值），自动校验返回的范围，语义，类型 输入N个，输出N个 也可以用肉眼看

单元测试，不要调用整个程序，编译运行，会很卡

一层层测试，从底层函数，到上层封装的函数，测试行为是否正常。

输入的形式：连续值测试 行为测试（默认应该进行的输入行为）

设计测试函数要获取输出的值，并把该值进行检测，并以某种形式显示出来（图像显示，数字显示，运动模拟）

棋价值 灵活性（移动力） 控制的空间（） 受威胁减分 保护升高 形 安全

定势

不同的参数有不同的比重

调试方法：请自动调试

爬山法，每次修改一点参数，然后电脑自动检测胜率是否提高，提高就保留，要下降就停止改变（但是存在连续起伏的可能性，可能那山比这山高）

蒙特卡罗算法 不同的起点多次爬山，可能会出现更优解，范围覆盖越广越好

模拟退火 遇到表现不好的参数仍然有可能继续，MetroPolis重要性采样

遗传算法 复制交叉变异 去掉较差种群

九章、 其他话题

水平效应 由于搜索层次固定，可能之后会有很糟糕的局面，没被找到 落入陷阱

层数浅就很危险

如果是很多步的陷阱，仍然会上当

1. 用估值函数算危险定势 给予一定知识，降低危险局面的估值
2. 静止期搜索 当叶节点局面不稳定时 继续向下搜索到局面相对平静

吃子 或者是将军？

把估值函数换成静止期搜索，最终的下层估值作为最终估值

3、扩展搜索 大部分走法是愚蠢的，所有局面都搜索其实不好

早期剪枝 在搜索的开始就抛弃 检查是否值得展开 （这种判断是不准确的），虽然提高了深度却可能错过好走法

部分展开是用于防水平效应 ，需要猜测是否需要继续向下

引发扩展搜索的条件：将军，吃子 异常 威胁 巨变 （估值剧烈变化）

4、开局库

电脑经常不会开局 ，开局信息太少了，不好判断哪种走法好，优势难以判断，所有走法估值差不多，选了也白选

制作棋谱程序 随机选择一组走法，开始几步

1. 残局库

残局请况太多， 最好是限定5子以下，直接察觉胜利失败

穷举所有五子残局 可以搜索很多层（几十） 必胜走最短路径 必败走最长路径 50层

6、如何防止循环 犯规？

九、 其他

轻视因子 让AI放松一点 比如降低层数之类的

机器学习

并行搜索

围棋：胜负局面计算难 ，估值复杂（串，块， 优劣需要模糊判断），分支复杂、需要专家系统（非常复杂的知识） 可能需要早期剪枝（准确性下降），机器学习比较难

十、五子棋

2种 棋 15\*15 分支多，约200 胜负连线 棋子递增

棋盘表示 走法产生 搜索技术 估值

走法：所有空白位置都可以走

X,y 棋子位置 走法分数

位置价值表？

生成走之后的棋盘 恢复棋盘 判断是否胜负

棋盘估值

应该使用历史启发的ab剪枝法 走法队列肯定要排序 置换表

估值 棋型 眠二 活二 眠三 活三 冲四 活四

估值 棋盘打分 对所有棋子进行四方向（水平垂直斜向，其实是八方向）分析找出周围棋型

直线分析， 分析每行

算各棋型数量

估值规则相当复杂 不过理论上来说 棋盘越活，型号越多越有可能活

直线分析

返回极值。已经赢了

状态机初始状态到最终状态