五子棋

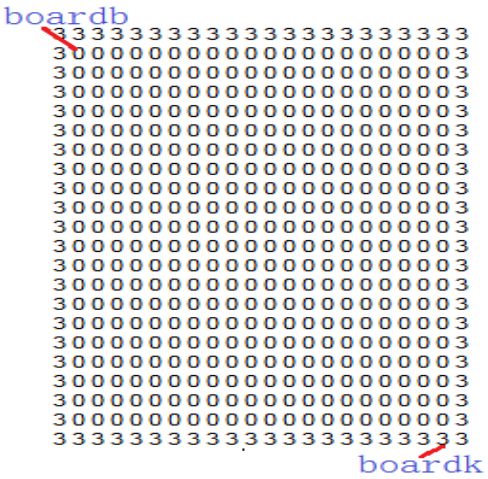
----几个关键数据结构的说明

一、 棋盘存储

棋盘以一维数组的形式存在 board 中，并在棋盘边界各留一个棋位设为边界。棋盘上棋子类型：(board.h 中定义)

```
#define EMPTY_MOVE          0    //空
#define TURN_MOVE           1    //引擎自身
#define OPPONENT_MOVE       2    //对手
#define OUTSIDE_MOVE        3    //棋盘边界
```

所以初始化棋盘棋盘如下（20 X 20）：



棋盘需要记录几个重要变量：

1. boardb 指针表示的是棋盘开始位置，boardk 表示棋盘结束位置；
2. width2 表示棋盘宽度（包括两个边界位），height2 表示棋盘高度（包括两个边界位）。
3. int diroff[9],各个方向上在棋盘的一维数组中位移差值。

```
/*
*当前位在 8
*543
*682
*701
*/
diroff[0] = sizeof(Tsquare);                    //向下一步
diroff[4] = -diroff[0];                         //向上一步
diroff[1] = sizeof(Tsquare)*(1+height2);    //向右下一步
diroff[5] = -diroff[1];                         //向左上一步
diroff[2] = sizeof(Tsquare)* height2;           //向左一步
diroff[6] = -diroff[2];                         //向右一步
diroff[3] = sizeof(Tsquare)*(-1+height2);   //向右上一步
diroff[7] = -diroff[3];                        //向左下一步
```

```
        diroff[8] = 0;     //本身
    }
```

两个重要函数：（里面的 s 传的正是 diroff 里面的值）

```
//在方向上向前 i 个位置，返回找到的位置指针
#define nxtP(p,i) (p=(Psquare)(((char*)p)+(i*s)))
//在 direction 方向上向后 i 个位置，返回找到的位置指针
#define prvP(p,i) (p=(Psquare)(((char*)p)-(i*s)))
```

4. 棋步生成函数 GenerateMoves (search.cpp中定义)

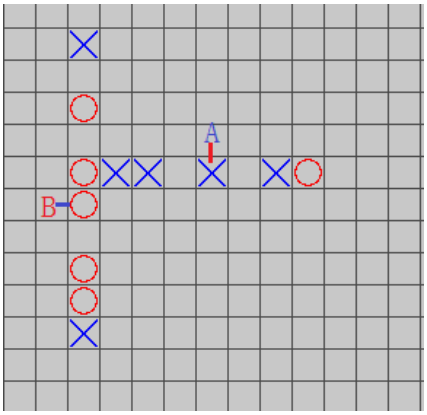
   通过查找棋盘上有子的位置，把距离为3子内的空棋位加入棋步列表中（通过做标记避免重复），如此可以减少可行棋步数量，达到剪枝作用。

二、 分析棋盘信息的数据结构

```
//统计棋盘信息，用于分析棋型(board.h 中定义)
  struct ChessAnalyzeData{
    int adjsameNxt;          //记录与(x, y)Next 相邻的连续同色棋子数
    int adjemptyNxt;         //记录 adjsame 后连续空位数
    int jumpsameNxt;         //记录 adjempty 后连续同色棋子数
    int jumpemptyNxt;        //记录 jumpsame 后的空位数
    int jumpjumpsameNxt;     //记录 jumpempty 后的连续同色棋子数

    int adjsamePre;          //记录与(x, y)pre 相邻的连续同色棋子数
    int adjemptyPre;         //记录 adjsame 前连续空位数
    int jumpsamePre;         //记录 adjempty 前连续同色棋子数
    int jumpemptyPre;        //记录 jumpsame 前的空位数
    int jumpjumpsamePre;     //记录 jumpempty 前的连续同色棋子数
  }
```

如下图所示，在 A 点统计的黑棋在水平方向(diroff[2])上的信息（按照上面结构顺序）是 1,1,1,0,0, 1,1,2,0,0 。B 点统计的红棋在竖直方向(diroff[0])上的信息是 1,1,2,0,0, 2,1,1,1,0。



通过统计这些信息，然后通过(evaluation.cpp中定义的)AnalysisBoardType(…)这个函数就能分

析对应的棋型。

三、 修改建议

1. 使用更加智能的博弈搜索方法（alpha-beta 等）来实现引擎，具体就是在 search.cpp 中增加相应方法，并在 search.cpp 的 brain_turn 函数中进行调用。
2. 改进走法生成方法，例如采用一些策略限制实际搜索的走法分支数量，来换取相同时间内搜索更深的空间。
3. 使用历史启发，可在提供代码基础上改进历史表的实现，寻找更好的定义历史启发评估的标准。
4. 修改评估函数，具体在 evaluation.cpp 的 Evaluation(…)函数，可定义更好的评估标准。
5. 使用转置表，可在提供代码基础上改进转置表的实现，寻找更好的定义标准。