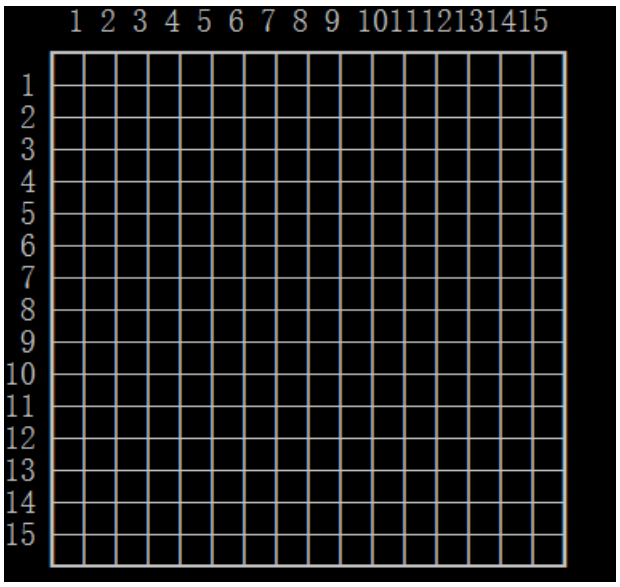
五子棋报告（第三周）

09016413　杨越森

【1】界面

因为实力有限，直接采用黑白框程序：



以上为一个 **15*15** 棋盘，若棋手想要下棋 则在想下的地方输入其坐标即可。

【2】AI 算法

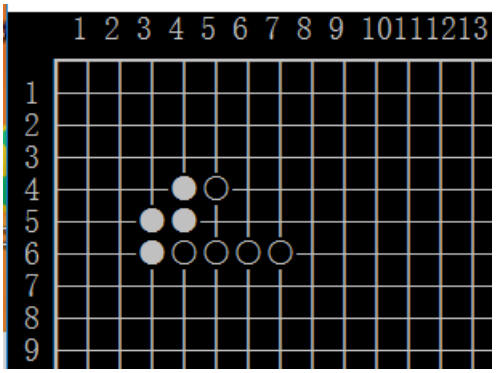
AI 的本质便是从所有选择中选择出最优解，所以设计 **AI** 的过程即使找出最优解的过程。

所以在设计 **AI** 的时候，思路为：

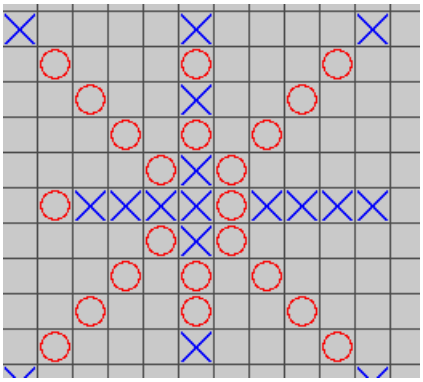
（1）先找出合理的所有选择：棋盘为 **15*15**，我们不能在每次计算的时候都把 **15*15=225** 这么多选择考究进去，如果这样的话，每次就要计算比较 **225** 个格数的值，浪费时间。

所以我们根据平时下五子棋的经验，我们落子只会在黑棋/白棋上一步落子的附近最抉择，起初设计算法的时候，

我只考虑了其附近一圈的情况（仅 8 个），这种情况刚开始下的时候还能做正当防守,但后来发现了如下漏洞：



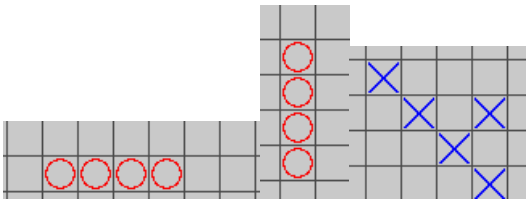
比如这种情况， 假如黑子最后落子不是（6，7）的话 白字就不会计算（6，8）的值，所以意识到 8 个不够， 因此考虑到五子连珠，所以干脆直接把目标扩大到 8*5.即以黑方/白方上次落子为中心展开



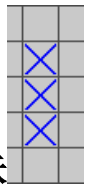
即考虑如下 40 个位置，计算这 40 个位置的值，找出最大的那个就是我们算法设计的思路。

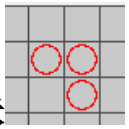
【3】 值的求法

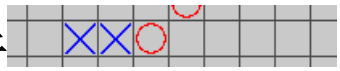
这个可以根据平时下棋的经验对状况进行层次分析，对几种情况进行排序：

1： 

分值最高的一定是 4 子连珠，这个不用说

2 三子连珠 

3 2 子连珠 

4 1 子 

5 分值最低的应该是黑白交织的状况，如下:

这样的分值应该是最低的。

以下代码是求每个分值的代码

```
void Gobang::value(int x, int y)//某一个点的40个值
{
    int s = 0;
    int t = 0;
    int a = x;
    int b = y;
    for (int j1 = 0; j1 < 5; j1++)
    {
        b += 1;
        for (int i1 = 1; i1 <=5; i1++)
        {
            if (M[a][b + i1] == 0) continue;
            if (M[a][b + i1] == 1)
            {
                if (t > 0) { s = -1; break; }
                s += 1;
            }
            if (M[a][b + i1] == -1)
```

```
            {
                if (s > 0) { t = -1; break; }
                t += 1;
            }
        }

        evaluate[0][j1] += Evaluate(s, t);

        }//右
a = y; b = x;
    for (int j1 = 0; j1 < 5; j1++)
    {
        b -= 1;
        for (int i1 = 1; i1 <= 5; i1++)
        {
            if (M[a][b -i1] == 0) continue;
            if (M[a][b - i1] == 1)
            {
                if (t > 0) { s = -1; break; }
                s += 1;
            }
            if (M[a][b -i1] == -1)
            {
                if (s > 0) { t = -1; break; }
                t += 1;
            }
        }
        evaluate[4][j1] += Evaluate(s, t);
    }
    a = y; b = x;
    // 左
    for (int j1 = 0; j1 < 5; j1++)
    {
        a += 1;
        for (int i1 = 1; i1 <= 5; i1++)
        {
            if (M[a+i1][b ] == 0) continue;
            if (M[a+i1][b ] == 1)
            {
                if (t > 0) { s = -1; break; }
                s += 1;
            }
            if (M[a+i1][b] == -1)
            {
```

```
                if (s > 0) { t = -1; break; }
                t += 1;
            }
        }
        evaluate[2][j1] += Evaluate(s, t);
    }
    a = y; b = x;
    // 下
    for (int j1 = 0; j1 < 5; j1++)
    {
        a -= 1;
        for (int i1 = 1; i1 <= 5; i1++)
        {
            if (M[a -i1][b] == 0) continue;
            if (M[a - i1][b] == 1)
            {
                if (t > 0) { s = -1; break; }
                s += 1;
            }
            if (M[a -i1][b] == -1)
            {
                if (s > 0) { t = -1; break; }
                t += 1;
            }
        }
        evaluate[7][j1] += Evaluate(s, t);
    }
    a = y; b = x;
    //上

    for (int j1 = 0; j1 < 5; j1++)
    {
        a += 1; b += 1;
        for (int i1 = 1; i1 <= 5; i1++)
        {
            if (M[a +i1][b+i1] == 0) continue;
            if (M[a+i1][b+i1] == 1)
            {
                if (t > 0) { s = -1; break; }
                s += 1;
            }
            if (M[a +i1][b+i1] == -1)
            {
                if (s > 0) { t = -1; break; }
```

```
                t += 1;
            }
        }
        evaluate[l][jl] += Evaluate(s, t);
    }
    a = y; b = x;//右下

    }
```

这个是求一个点右面 5 个点的 value 的代码，类似的还有左面的，上下，左下左上右上右下，一共 8 个方位，代码都是类似的。

【4】 分值确认

这个部分是这个算法的核心，前面的只能说是框架，而确定这些分值就是最重要的事情，这个事情我们之后再研究。

需要确定的分值：

1 对方四子

2 对方三子

3 对方二子

4 对方一子

5 己方一子

6 己方两子

7 己方三子

8 己方四子

9 无子

已经确定的分值：

10 黑白都有：0 分

小结：目前已经完成界面的设计以及算法的设计，接下来要做的就是根据实际情况对分值进行赋值，这个赋值便是下周要研究的内容。