电子科技大学

计算机专业类课程

实验报告

课程名称：程序设计基础

学院专业：计算机科学与工程学院

学生姓名：诸葛禹阳

学　　号：2020080602030

指导教师：俸志刚

日　　期：2021 年 06 月 18 日

电 子 科 技 大 学

实 验 报 告

实验一

一、实验室名称：

电子科技大学清水河校区基础实验大楼 502

二、实验项目名称：

五子棋 AI 实验

三、实验目的：

研究一个运算时间不会太长且棋力不会太差的五子棋 AI，好与玩家对下。

四、实验主要内容：

1. 能正常进行五子棋游戏，其中包含符合规则，能及时判断胜负
2. 电脑能正常地向人一样下棋，并且尽可能地赢
3. 调整改进算法使计算机思考的时间变短，并提升计算机的胜率

五、实验器材（设备、元器件）：

软件：DevC++

六、实验步骤：

1.算法分析与概要设计

（1）棋盘和游戏规则的制定和输赢的判断

（2）给棋盘当前局面赋值，以便后续判断形势好坏

（3）利用深度搜索，搜索接下来双方可能的棋路，选择对己方最有利的路线，其中假设我方和玩家方都会选择对自己最有利的方向

（4）其中由于默认我方和玩家最终都会选择对自己最有利的棋路，所以可以剪去那些不那么“划算”的路径，直接剪去后面的搜索，即为剪枝

2.核心算法的详细设计与实现

概念：活 n，即没有对手的棋子堵住的连着的 n 个棋子，如+OOO+

死 n，即有一方被对手的棋子堵住的连着的 n 个棋子，如□OOOO+

（其中 O 为棋子，+为还未下的空地方，□为对手的棋子）

（1）赋值：局势的评估值为电脑的棋子赋值之和减去玩家的棋子赋值之和，即 value_com-value_player。对棋子的赋值初步定为当棋盘上存在黑棋/白棋的活 n 时，其值为 10^n，若是死 n，则为 10^(n-1)，其中我们的活 n 死 n（尤其是 n>=3 中的）还包含如

O+OO，OO+OO，OOO+O 等

中间含且仅含一个空格的棋子分布。

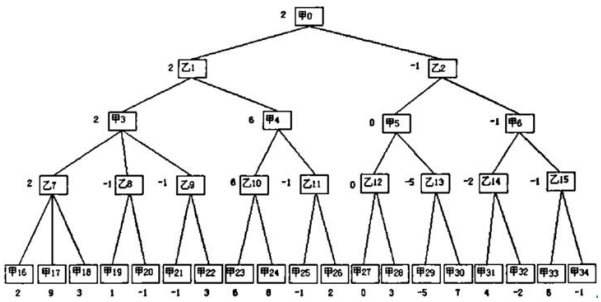
这种棋分布的类型与普通中间没空位对比，可易知前者若对手堵住空位，不仅死 n 都没有，可能连死 n-1 都没有，而后者无论被怎样堵，都必然是死 n 的形式，故前者对局面的贡献价值应当是小于后者的。

而又为了使整个棋盘对活 3，活 4，死 4 敏感，故我们给所有中间没空位的活 3，活 4，死 4 的赋值*2，既不改变它已有的赋值性质，又与中间有空格的活 4 活 3 死 4 区分开。

其判断也无需重新遍历判断，只需判断单次的赋值（value_i）是否大于等于 1000 且中间空位的数量（num）是否为 0。

（2）搜索：考虑一个树，每个节点为当前局势的评估值。

这里借用别人的一张图来说明搜索的过程：



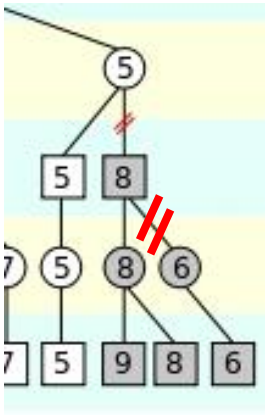
对于对应电脑的选择节点的那一层，电脑一定会选择其中能得到的最后的局势评估值尽可能大的选项；同样，对于玩家的那一层，玩家一定会选择其中能得到的最后的局势评估尽可能小（对自己有利的）的选项。这就形成了一种最大值-最小值搜索，相当于不断按照上述规则进行深搜。

我们初步让棋子的每一步决策都往后思考 6 层，即若现在在思考电脑这一步的走法，我们将搜索电脑-玩家的三个来回，以玩家的落子结束。之所以定为以对手的回合结束，是因为若不考虑对手的防守或进攻，己方刚落子后的局势评估值的大小毫无意义。而当思考层数变为 8 层后，计算次数将直接上升 2 个数量级，电脑每次思考时间大幅增加，就极大地破坏了玩家的游戏体验过程。而在试验中，我们发现 6 层的棋力还是能打败新手的，故我们定 6 层为我们的搜索深度。

又为了排除掉一些过于离谱的选项，我们用两种方式：一是编写了一个 nearby 的函数，筛掉那些离棋子分布太远的函数，减少分析的落棋点个数；二是我们只选择每一层落子后局势对自己较为有利的选项向下搜索，我们暂定为每一层选至多 5 个，增加选的个数也能在一定程度上增加棋力。

nearby 函数即判断新增的棋子的“米”字形范围内（步长范围为 2）是否有棋子在。

（3）剪枝：实际上，为了使整体思考时间更快，我们可以对整个树进行剪枝。其原理即，双方都会选择对自己最有利的选项。在这里，我们也借用网上的一张图来说明：



如图，黄层为玩家的抉择层，蓝层为电脑的抉择层，当已经搜索过蓝层前面的 5-5 节点后，我们记录 b=5，即记录当前的节点。现在我们记 5 8 那一层为 1 层，5 8 6 那一层为 2 层，而后我们将每一个 1 层的得到的数值和 b 作比较，若该数值比 b 小，则更新 b 为该数值（即保证 b 为 1 层的最小值）。因为我们知道玩家一定会选择最优的解（b）。而同时我们将每一个 1 层得到的数值和 b 作比较，若该数值比 b 大，则我们便可以不去搜索这个 1 层节点向下的之后的节点了。因为我们知道在 1 层，即电脑层，其一定会选择其下面分支中最大的值，而再上一层的玩家层只会选择电脑层中的最小值，故例如此时已经搜索到 8 了，电脑层肯定不会选比 8 小的数，故 1 层第二个节点的数值一定大于 8，即上面的玩家层一定不会选这个节点向下的路径了，所以就可以不去搜索 8 这个节点下面的其他 2 层节点。（简单来说，就是 5（1 层）<b，所以 b=5，8>b，所以 6 后面的路径被剪）

而对于电脑来讲也是一样的，所以我们还需一个记录最大值的 a。

实现也非常简单，只需让我们的搜索函数中带有记录用的 a，b 即可。其中在 max 的搜索函数中，不断地更新 a 的数值，而不断地利用 b 剪枝，而在 min 的搜索函数中，不断地更新 b 的数值，而利用 a 来剪枝。这大大提高了运行的效率，哪怕在棋局的后半段，进行 6 层的搜索，我们的电脑思考速度也会在 1s 以内，这大大提高了五子棋的游戏体验，且不影响原来的棋力。

六、实验数据及结果分析：

备注：

1. 该代码默认玩家先手，但中间各函数都是可以通用的，所以在对战网上别人做的五子棋 AI 的时候连输，但打败咱这种五子棋萌新绰绰有余。

2. 像 value_player, value_com 这种虽然可以写在一起的函数我分开写了，是为了方便我在 DevC++上的调试 Debug，以及 fuzhi[x][y]和 copy[x][y]的存在单纯是我为了防止 bug 以及后续更容易看出是哪里出问题了增加的，显得代码杂而长了很抱歉。

3. maxmax 和 minmax 分别为最大极大值和次大极大值，其区别在于前者是用来被别的最大最小值顶掉用的，后者是为了传递一种“情势非常好/非常不好”的讯息。

4. 这里认为棋子能下得中间的一点是更加有利的决定，因为中间不容易受边缘的束缚。

5. 这里尽量删去了调试时的产物，如有遗漏非常抱歉！

五子棋代码展示：

```
#include<iostream>
#include<cstdio>
#include<stdlib.h>
#include<windows.h>
#include<math.h>
using namespace std;
int qipan[15][15]={0};  //棋盘，0:空；1：Player 的；-1：电脑的
int fuzhi[15][15];  //复制，用来尽量避免可能出现的错误
```

```
int c_x,c_y;  //记录电脑下棋的位置
int maxmax=9999999;
int minmax=999999;

int value_player(int now[15][15]);  //玩家/电脑的局势赋值函数
int value_com(int now[15][15]);
int win(int now[15][15]);      //判断当前局势的胜负
int min_v(int now[15][15],int depth,int a,int b);     //深搜 min/max
int max_v(int now[15][15],int depth,int a,int b);
int search_l(int now[15][15],int k);  //通过四个方向的搜索来进行局势赋值
int search_r(int now[15][15],int k);
int search_xsright(int now[15][15],int p_c);
int search_xsleft(int now[15][15],int p_c);
void print_qipan();      //打印棋盘
void com_play(int now[15][15]);     //电脑的思考
int nearb(int i,int j);     //是否相邻

int main(){
    int a,b;
    print_qipan();
    while(scanf("%d %d/n",&a,&b)){
        qipan[a][b]=1;
        print_qipan();
        cout<<endl;
        c_x=0;c_y=0;
        if(win(qipan)==1){
            cout<<"You win!"<<endl;
            break;
        }
        for(int t=0;t<15;t++){
            for(int k=0;k<15;k++){
                fuzhi[t][k]=qipan[t][k];
            }
        }
```

```
            com_play(fuzhi);
            qipan[c_x][c_y]=-1;
             print_qipan();
             cout<<endl;
            if(win(qipan)==-1){
                cout<<"You lose!"<<endl;
                break;
            }
        }
}

int nearb(int i,int j){
       int n=0;
      for(int t=-2;t<=2;t++){
            if(qipan[i+t][j]!=0&&i+t>=0&&i+t<15) return 1;
            if(qipan[i][j+t]!=0&&j+t>=0&&j+t<15) return 1;
            if(qipan[i+t][j+t]!=0&&i+t>=0&&i+t<15&&j+t>=0&&j+t<15) return 1;
            if(qipan[i+t][j-t]!=0&&i+t>=0&&i+t<15&&j-t>=0&&j-t<15) return 1;
      }
      return 0;
}

int max_v(int now[15][15],int depth,int a,int b){
       int v;
       int vl[5],x[5],y[5];   //储存前 5 个最佳节点的 value，x,y 坐标
      int copy[15][15];
      for(int t=0;t<15;t++){
            for(int k=0;k<15;k++){
                copy[t][k]=now[t][k];
            }
      }
      if(depth<=0||win(copy)!=0){   //输了就直接返回次大最大值
            int v_f;
            if(win(copy)==1){
```

```
                v_f=-minmax;
            }
            else if(win(copy)==-1){
                v_f=minmax;
            }
            else{
                v_f=value_com(copy)-value_player(copy);
            }
            return v_f;
        }
        for(int i=0;i<5;i++){
            vl[i]=-maxmax;
            x[i]=15;
            y[i]=15;
        }
        int best=-maxmax;
        for(int i=0;i<15;i++){
            for(int j=0;j<15;j++){
                if(now[i][j]==0&&nearb(i,j)==1){
                    now[i][j]=-1;
                    for(int t=0;t<15;t++){
                        for(int k=0;k<15;k++){
                            copy[t][k]=now[t][k];
                        }
                     }
                    v=value_com(copy)-value_player(copy);
                    for(int t=0;t<5;t++){
                        if(vl[t]<v||(vl[t]==v&&abs(i-7)+abs(j-7)<abs(x[t]-7)+ab
s(y[t]-7))){
                            for(int k=4;k>t;k--){
                                vl[k]=vl[k-1];
                                x[k]=x[k-1];
                                y[k]=y[k-1];
                            }
```

```
                        vl[t]=v;
                        x[t]=i;
                        y[t]=j;
                    }
                }
                now[i][j]=0;
            }
        }
    }
    for(int i=0;i<5;i++){
        if(x[i]<15){
        now[x[i]][y[i]]=-1;
        for(int t=0;t<15;t++){
            for(int k=0;k<15;k++){
                copy[t][k]=now[t][k];
            }
        }
        vl[i]=min_v(copy,depth-1,a,b);
        if(vl[i]<a){   //剪枝
            a=vl[i];
        }
        now[x[i]][y[i]]=0;
        if(vl[i]>best){
            best=vl[i];
        }
        if(vl[i]<b){
            return best;
        }
    }
    }
    return best;
}

int min_v(int now[15][15],int depth,int a,int b){
```

```
  int v;
  int vl[5],x1[5],y1[5];
int copy[15][15];
for(int t=0;t<15;t++){
      for(int k=0;k<15;k++){
            copy[t][k]=now[t][k];
      }
}
if(depth<=0||win(copy)!=0){
      int v_f;
      if(win(copy)==1){
            v_f=-minmax;
      }
      else if(win(copy)==-1){
            v_f=minmax;
      }
      else{
            v_f=value_com(copy)-value_player(copy);
      }
      return v_f;
}
for(int i=0;i<5;i++){
      vl[i]=maxmax;
      x1[i]=15;
      y1[i]=15;
}
int best=maxmax;
for(int i=0;i<15;i++){
      for(int j=0;j<15;j++){
            if(now[i][j]==0&&nearb(i,j)==1){
                  now[i][j]=1;
                  for(int t=0;t<15;t++){
                        for(int k=0;k<15;k++){
                              copy[t][k]=now[t][k];
```

```
                    }
                }
                v=value_com(copy)-value_player(copy);
                for(int t=0;t<5;t++){
                    if(vl[t]>v||(vl[t]==v&&abs(i-7)+abs(j-7)<abs(x1[t]-7)+a
                      bs(y1[t]-7))){
                     for(int k=4;k>t;k--){
                         vl[k]=vl[k-1];
                         x1[k]=x1[k-1];
                         y1[k]=y1[k-1];
                     }
                     vl[t]=v;
                     x1[t]=i;
                     y1[t]=j;
                    }
                }
                now[i][j]=0;
            }
        }
    }
    for(int i=0;i<5;i++){
        if(x1[i]<15){
        now[x1[i]][y1[i]]=1;
        for(int t=0;t<15;t++){
            for(int k=0;k<15;k++){
                copy[t][k]=now[t][k];
            }
        }
        vl[i]=max_v(copy,depth-1,a,b);
        if(vl[i]>b){
            b=vl[i];
        }
        now[x1[i]][y1[i]]=0;
        if(vl[i]<best){
```

```
                best=vl[i];
            }
            if(vl[i]<a){
                return best;
            }
        }
        }
        return best;
}

void com_play(int now[15][15]){
        int player,comp;
        int computer,max;
        int x[5],y[5],vl[5];
        int v;
        int copy[15][15];
        max=-maxmax;
        for(int i=0;i<5;i++){
            vl[i]=max;
            x[i]=15;
            y[i]=15;
        }
        for(int i=0;i<15;i++){
            for(int j=0;j<15;j++){
                if(now[i][j]==0&&nearb(i,j)==1){
                    now[i][j]=-1;
                    for(int t=0;t<15;t++){
                        for(int k=0;k<15;k++){
                            copy[t][k]=now[t][k];
                        }
                      }
                    v=value_com(copy)-value_player(copy);
                    for(int t=0;t<5;t++){
```

```
if(vl[t]<v||(vl[t]==v&&abs(i-7)+abs(j-7)<abs(x[t]-7)+abs(y[t]-7))){
                                for(int k=4;k>t;k--){
                                    vl[k]=vl[k-1];
                                    x[k]=x[k-1];
                                    y[k]=y[k-1];
                                }
                                vl[t]=v;
                                x[t]=i;
                                y[t]=j;
                            }
                        }
                        now[i][j]=0;
                    }
                }
            }
            for(int i=0;i<5;i++){
                if(x[i]<15){
                now[x[i]][y[i]]=-1;
                for(int t=0;t<15;t++){
                    for(int k=0;k<15;k++){
                        copy[t][k]=now[t][k];
                    }
                }
                vl[i]=min_v(copy,6,maxmax,-maxmax);
                now[x[i]][y[i]]=0;
                if(vl[i]>max){
                    max=vl[i];
                    c_x=x[i];
                    c_y=y[i];
                }
            }
            }
        }
```

```
int win(int now[15][15]){
    int w=0;
    int i,j;
    for(int i=0;i<15;i++){
        for(int j=0;j<15;j++){
            if(now[i][j]==1){
                if(w<0){
                    w=0;
                    w++;
                }
                else{
                    w++;
                }
                if(w>4){
                    return 1;
                }
            }
            else if(now[i][j]==-1){
                if(w>0){
                    w=0;
                    w--;
                }
                else{
                    w--;
                }
                if(w<-4){
                    return -1;
                }
            }
        }
        w=0;
    }
    for(int j=0;j<15;j++){
        for(int i=0;i<15;i++){
```

```
            if(now[i][j]==1){
                if(w<0){
                    w=0;
                    w++;
                }
                else{
                    w++;
                }
                if(w>4){
                    return 1;
                }
            }
            else if(now[i][j]==-1){
                if(w>0){
                    w=0;
                    w--;
                }
                else{
                    w--;
                }
                if(w<-4){
                    return -1;
                }
            }
        }
        w=0;
    }
    for(int t=-14;t<15;t++){
        if(t<0){
            i=0;j=-t;
        }
        else{
            i=t;j=0;
        }
```

```
        for(int k=0;k<15-abs(t);k++){
            if(now[i+k][j+k]==1){
                if(w<0){
                    w=0;
                    w++;
                }
                else{
                    w++;
                }
                if(w>4){
                    return 1;
                }
            }
            else if(now[i+k][j+k]==-1){
                if(w>0){
                    w=0;
                    w--;
                }
                else{
                    w--;
                }
                if(w<-4){
                    return -1;
                }
            }
        }
        w=0;
    }
    for(int t=0;t<29;t++){
        if(t<15){
            i=t;j=0;
        }
        else{
            i=14;j=t-14;
```

```
            }
            for(int k=0;k<15-abs(14-t);k++){
                if(now[i-k][j+k]==1){
                    if(w<0){
                        w=0;
                        w++;
                    }
                    else{
                        w++;
                    }
                    if(w>4){
                        return 1;
                    }
                }
                else if(now[i-k][j+k]==-1){
                    if(w>0){
                        w=0;
                        w--;
                    }
                    else{
                        w--;
                    }
                    if(w<-4){
                        return -1;
                    }
                }
            }
            w=0;
        }
        return 0;
}

int value_player(int now[15][15]){
    int valuep=0;
```

```
    valuep=valuep+search_l(now,1);
    valuep=valuep+search_r(now,1);
    valuep=valuep+search_xsright(now,1);
    valuep=valuep+search_xsleft(now,1);
    return valuep;
}

int value_com(int now[15][15]){
    int valuep=0;
    valuep=valuep+search_l(now,-1);
    valuep=valuep+search_r(now,-1);
    valuep=valuep+search_xsright(now,-1);
    valuep=valuep+search_xsleft(now,-1);
    return valuep;
}

int search_l(int now[15][15],int k){
    int value=0;
    int value_i,num;
    int cl;
    for(int i=0;i<=14;i++){
        int j=0;
        while(j<=14){
            value_i=0;
            if(now[i][j]==k){
                value_i=10;
                cl=0;
                If((j>0&&now[i][j-1]==-k)||j==0){
                    value_i=value_i/10;
                    cl=j;
                }
                num=0;
                j++;
```

```
while((now[i][j]==k||(now[i][j]==0&&now[i][j+1]==k&&num==0))&&j<15){
                                if(now[i][j]==k){
                                        value_i=value_i*10;
                                }
                                else{
                                  num++;
                                }
                                j++;
                        }
                        if(cl!=0||now[i][j]==-k)value_i=value_i-1;
                        if(now[i][j]==-k){
                                if(j-cl<5){
                                        value_i=0;
                                }
                                else{
                                  value_i=value_i/10;
                                }
                        }
                        if(j==15){
                           break;
                        }
                }
                if(value_i>=1000&&num==0)value_i=value_i*2;
                value+=value_i;
                j++;
        }
    }
    return value;
}

int search_r(int now[15][15],int k){
     int value=0;
    int value_i,num;
    int cl;
```

```
for(int j=0;j<=14;j++){
    int i=0;
    while(i<=14){
        value_i=0;
        if(now[i][j]==k){
            value_i=10;
            cl=0;
            If((i>0&&now[i-1][j]==-k)||i==0){
                value_i=value_i/10;
                cl=i;
            }
            num=0;
            i++;

while((now[i][j]==k||(now[i][j]==0&&now[i+1][j]==k&&num==0))&&i<15){
                if(now[i][j]==k){
                    value_i=value_i*10;
                }
                else{
                  num++;
                }
                i++;
            }
            if(cl!=0||now[i][j]==-k)value_i=value_i-1;
            if(now[i][j]==-k){
                if(i-cl<5){
                    value_i=0;
                }
                else{
                  value_i=value_i/10;
                }
            }
            if(i==15){
               break;
```

```
                    }
                }
                if(value_i>=1000&&num==0)value_i=value_i*2;
                value+=value_i;
                i++;
            }
        }
        return value;
}

int search_xsright(int now[15][15],int p_c){
        int value=0;
        int value_i,num,i,j;
        int cl;
        for(int t=-14;t<15;t++){
            if(t<0){
                    i=0;j=-t;
              }
              else{
                    i=t;j=0;
              }
            int k=0;
            while(k<15-abs(t)){
                    value_i=0;
                    if(now[i+k][j+k]==p_c){
                            value_i=10;
                            cl=0;
                            If((k>0&&now[i+k-1][j+k-1]==-p_c)||k==0){
                                    value_i=value_i/10;
                                    cl=k;
                            }
                            num=0;
                            k++;
```

```
while((now[i+k][j+k]==p_c||(now[i+k][j+k]==0&&now[i+k+1][j+k+1]==p_c&&nu
m==0))&&k<15-abs(t)){
                                if(now[i+k][j+k]==p_c){
                                      value_i=value_i*10;
                                }
                                else{
                                  num++;
                                }
                                k++;
                            }
                            if(cl!=0||now[i+k][j+k]==-p_c)value_i=value_i-1;
                            if(now[i+k][j+k]==-p_c){
                                if(k-cl<5){
                                      value_i=0;
                                }
                                else{
                                  value_i=value_i/10;
                                }
                            }
                            if(k==15-abs(t)){
                               break;
                            }
                        }
                        if(value_i>=1000&&num==0)value_i=value_i*2;
                        value+=value_i;
                        k++;
                    }
                }
                return value;
            }

            int search_xsleft(int now[15][15],int p_c){
                int value=0;
                int value_i,num,i,j;
```

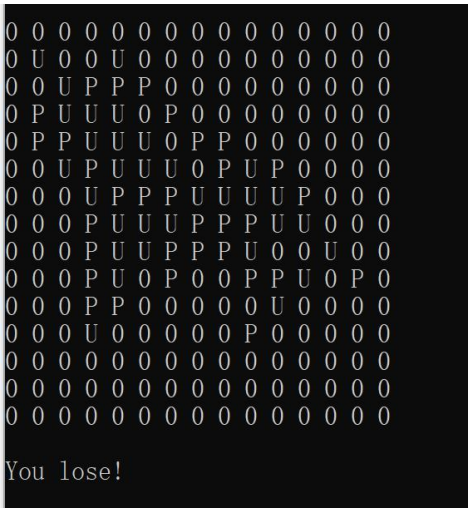
```
int cl;
for(int t=0;t<29;t++){
    if(t<15){
        i=t;j=0;
    }
    else{
        i=14;j=t-14;
    }
    int k=0;
    while(k<15-abs(14-t)){
        value_i=0;
        if(now[i-k][j+k]==p_c){
            value_i=10;
            cl=0;
            If((k>0&&now[i-k+1][j+k-1]==-p_c)||k==0){
                value_i=value_i/10;
                cl=k;
            }
            num=0;
            k++;

while((now[i-k][j+k]==p_c||(now[i-k][j+k]==0&&now[i-k-1][j+k+1]==p_c&&num==0))&&k<15-abs(14-t)){
                if(now[i-k][j+k]==p_c){
                    value_i=value_i*10;
                }
                else{
                    num++;
                }
                k++;
            }
            if(cl!=0||now[i-k][j+k]==-p_c)value_i=value_i-1;
            if(now[i-k][j+k]==-p_c){
                if(k-cl<5){
```

```
                    value_i=0;
                }
                else{
                    value_i=value_i/10;
                }
            }
            if(k==15-abs(14-t)){
                break;
            }
        }
        if(value_i>=1000&&num==0)value_i=value_i*2;
        value+=value_i;
        k++;
    }
  }
  return value;
}

void print_qipan(){
  for(int i=0;i<15;i++){
    for(int j=0;j<15;j++){
      if(qipan[i][j]==0){
        cout<<"0 ";
      }
      if(qipan[i][j]==1){
        cout<<"P ";
      }
      if(qipan[i][j]==-1){
        cout<<"U ";
      }
    }
    cout<<endl;
  }
}
```

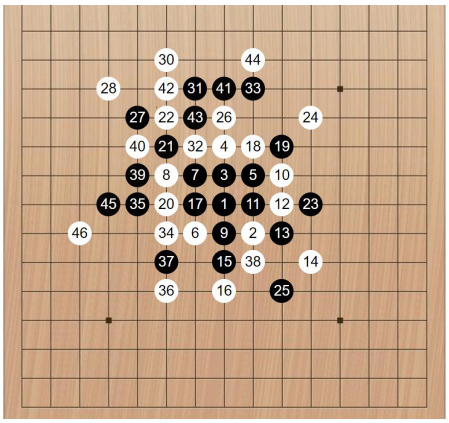
运行结果：



这是和室友下棋的结果。

但我的五子棋棋力也只能打打新手。

小插曲：



自己和网上的五子棋下棋
意外地赢了一次，
还挺高兴的。

八、总结及心得体会：

总体来讲，做五子棋这个程序的过程还蛮有意思的。整体都是自己想加上大佬的启发，本来想靠自己的方法做，后来发现深搜+剪枝真的非常好用，这种方法应该在以后很多其他问题的解决上也派上很大的用处吧。

以及及时编译很重要，吃了大亏！

中间在不断思考自己写的电脑究竟是如何思考的，算是受益匪浅吧。最后能有一个基本过得去的成品还是很自豪的。

以后有时间希望能给它做个 UI，以及改进改进。现在只能先当作作业交了。

九、对本实验过程及方法、手段的改进建议及展望：

1. 算杀没完成，这点有点可惜。少了更直观的符合人的思维的下棋方式。但或许能通过调整一定程度上的赋值达到相同棋力的效果，但会比较抽象。另一方面主要的提升可能是时间方面。留给后续去思考了。

改进方式：基本判断方式会和 win 的差不多，提交后我也可以去参考一下别人算杀的写法。

2. 没有做 UI，因为实在还是没习惯 CLion，这个暑假努力转到那个上面去，DevC++还是不知道该怎么做 UI，所以整体看着很不舒服。

改进方式：熟悉起别的编译器，DevC++有时候还是不太好用，接下来要多去尝试一下自己不熟悉的东西

3. 是我在写实验报告时发现的，X+OOO 这样的类型 X 会被忽视，一是赋值不够精确，二是每层 5 个实在太少，实际上增加到 6-7 时间上完全没问题。

4. 没有使用多文件，实在是没时间做改变了。

展望：希望老师多捞捞，想利用多文件，把 UI 也搞起来，把指针运用进去，自己的代码充斥着幼稚的语气，也想慢慢能向自己理想的代码靠近。

报告评分：

指导教师签字：