ab剪枝算法：

class TicTacToe(object):  
 *'''  
 井字棋游戏:  
 player:当前落子玩家，-1代表AI，1代表人类  
 board:棋盘  
 '''* # 传入谁先手，不传入默认AI先手  
 def \_\_init\_\_(self, depth, first=-1):  
 self.depth = depth  
 self.restart(first)  
  
 # 重新开始游戏，player先手  
 def restart(self, player):  
 self.board = [0] \* 9  
 self.player = player  
 self.bestmove = -1  
  
 # 落子函数  
 def go(self, pos):  
 self.board[pos] = self.player  
 self.player = -self.player  
  
 # 撤销落子函数  
 def cancel\_go(self, pos):  
 self.board[pos] = 0  
 self.player = -self.player  
  
 def print\_board(self):  
 for i in range(3):  
 for j in range(3):  
 if self.board[i \* 3 + j] == -1:  
 print("", end="X ")  
 elif self.board[i \* 3 + j] == 1:  
 print("", end="O ")  
 else:  
 print("", end="\_ ")  
 print()  
 print("-" \* 20)

# 游戏未结束返回0，AI胜利返回-1,人类获胜返回1，平局返回2  
 def get\_winner(self):  
 if any([var == 3 or var == -3 for var in [sum(self.board[i:i + 3]) for i in range(0, 9, 3)]]):  
 return -self.player  
 if any([var == 3 or var == -3 for var in [sum(self.board[i:7 + i:3]) for i in range(0, 3)]]):  
 return -self.player  
 if any([var == 3 or var == -3 for var in [sum(self.board[:9:4]), sum(self.board[2:7:2])]]):  
 return -self.player  
 return 0 if self.board.count(0) > 0 else 2  
  
 def evaluate\_minmax(self):  
 winner = self.get\_winner()  
 if winner == 2:  
 return 0  
 return 1000000 \* winner  
  
 def evaluate\_nega\_max(self):  
 winner = self.get\_winner()  
 if winner == 2:  
 return 0  
 return 1000000 \* winner \* self.player  
  
 # alpha\_beta算法  
 def alpha\_beta(self, depth, alpha, beta):  
 # 搜索深度耗尽或者某一方获胜  
 winner = self.get\_winner()  
 if winner != 0 or depth == 0:  
 return self.evaluate\_nega\_max()  
 # 继续搜索  
 # 走步生成  
 moves = [x for x in range(9) if self.board[x] == 0]  
 for pos in moves:  
 self.go(pos)  
 value = -self.alpha\_beta(depth - 1, -beta, -alpha)  
 self.cancel\_go(pos)  
 if value >= beta:  
 if self.depth == depth:  
 self.bestmove = pos  
 return beta  
 if value > alpha:  
 if self.depth == depth:  
 self.bestmove = pos  
 alpha = value  
 return alpha  
# 8是深度，因为估值函数的问题，这里深度要大于等于8才能算正常落子  
game = TicTacToe(8)  
while game.get\_winner() == 0:  
 game.alpha\_beta(game.depth, -1000000, 1000000)  
 game.go(game.bestmove)  
 game.print\_board()  
  
print("result:", game.get\_winner())

运行结果：

