A1_search (3)

February 26, 2020

[1]:
```
import numpy as np
import random
from trueskill import Rating
from trueskill import rate_1vs1
```

[2]:
```
class HexBoard:
  BLUE = 1
  RED = 2
  EMPTY = 3
  def __init__(self, board_size):
    self.board = {}
    self.size = board_size
    self.game_over = False
    for x in range(board_size):
        for y in range (board_size):
            self.board[x,y] = HexBoard.EMPTY
  def is_game_over():
    return self.game_over
  def is_empty(self, coordinates):
    return self.board[coordinates] == HexBoard.EMPTY
  def is_color(self, coordinates, color):
    return self.board[coordinates] == color
  def get_color(self, coordinates):
    if coordinates == (-1,-1):
      return HexBoard.EMPTY
    return self.board[coordinates]
  def place(self, coordinates, color):
    if not self.game_over and self.board[coordinates] == HexBoard.EMPTY:
      self.board[coordinates] = color
      if self.check_win(HexBoard.RED) or self.check_win(HexBoard.BLUE):
        self.game_over = True
  def get_opposite_color(self, current_color):
    if current_color == HexBoard.BLUE:
      return HexBoard.RED
    return HexBoard.BLUE
  def get_neighbors(self, coordinates):
    (cx,cy) = coordinates
```

```
   neighbors = []
   if cx-1>=0:   neighbors.append((cx-1,cy))
   if cx+1<self.size: neighbors.append((cx+1,cy))
   if cx-1>=0    and cy+1<=self.size-1: neighbors.append((cx-1,cy+1))
   if cx+1<self.size  and cy-1>=0: neighbors.append((cx+1,cy-1))
   if cy+1<self.size: neighbors.append((cx,cy+1))
   if cy-1>=0:   neighbors.append((cx,cy-1))
   return neighbors
 def border(self, color, move):
   (nx, ny) = move
   return (color == HexBoard.BLUE and nx == self.size-1) or (color == HexBoard.
↪RED and ny == self.size-1)
 def traverse(self, color, move, visited):
   if not self.is_color(move, color) or (move in visited and visited[move]):␣
↪return False
   if self.border(color, move): return True
   visited[move] = True
   for n in self.get_neighbors(move):
     if self.traverse(color, n, visited): return True
   return False
 def check_win(self, color):
   for i in range(self.size):
     if color == HexBoard.BLUE: move = (0,i)
     else: move = (i,0)
     if self.traverse(color, move, {}):
       return True
   return False
 def print(self):
   print("   ",end="")
   for y in range(self.size):
       print(chr(y+ord('a')),"",end="")
   print("")
   print(" -----------------------")
   for y in range(self.size):
       print(y, "|",end="")
       for z in range(y):
           print(" ", end="")
       for x in range(self.size):
           piece = self.board[x,y]
           if piece == HexBoard.BLUE: print("b ",end="")
           elif piece == HexBoard.RED: print("r ",end="")
           else:
               if x==self.size:
                   print("-",end="")
               else:
                   print("- ",end="")
       print("|")
```

```
    print("   ------------------------")
```

[3]:
```
def getmoves(board):
    moves = []

    for i in range(board.size):
        for j in range(board.size):
            if board.is_empty((i,j)) and board.game_over == False:
                moves.append((i,j))

    return moves
```

[4]:
```
def ev(board):
    if board.game_over:
        return 0
    else:
        r = random.randint(1,10)
        return r
```

[25]:
```
board = HexBoard(3)
```

[26]:
```
def alpha(board, a=-99, b=99, depth=3, is_max=True, color = board.RED):
    if depth == 0 or board.check_win(HexBoard.BLUE) or board.check_win(HexBoard.
 ↪RED):
        g = ev(board)
        print(g)
        return g

    elif is_max == True:
        g = -99
        global v
        v = {}
        for c in getmoves(board):
            makeMove(c, color, board)
            board.print()
            n_g = alpha(board, a, b, depth=depth-1, is_max=False, color = color)
            g = max(g, n_g)
            v[n_g] = c
            print("max:",v)
            unmakeMove(c,board)
            a = max(a, g)
            if g>=b:
                break
        return g
    elif is_max == False:
        g = 99
        for c in getmoves(board):
            makeMove(c, board.get_opposite_color(color), board)
            board.print()
            g = min(g, alpha(board, a, b, depth=depth-1, is_max=True, color =␣
↪color))
            unmakeMove(c,board)
            b = min(b, g)
            if a>=g:
                break
        return g

def nextMove(board,d,c):
    g = alpha(board,depth=d,color=c)
    return v.get(g)
```

[27]:
```
def unmakeMove(move, board):
    board.board[move] = board.EMPTY

def makeMove(move, color, board):
    board.board[move] = color
```

[33]:
```
board = HexBoard(3)
board.print()
```

```
   a b c
 -----------------------
0 |- - - |
1 | - - - |
2 |  - - - |
   -----------------------
```

[34]:
```
board.place((2,0),board.BLUE)
board.print()
```

```
   a b c
 -----------------------
0 |- - b |
1 | - - - |
2 |  - - - |
   -----------------------
```

[36]:
```
alpha(board,depth = 2)
```

```
   a b c
 -----------------------
0 |r - b |
1 | - - - |
2 |  - - - |
   -----------------------
   a b c
 -----------------------
0 |r - b |
1 | b - - |
2 |  - - - |
   -----------------------
4
   a b c
 -----------------------
0 |r - b |
1 | - - - |
2 |  b - - |
   -----------------------
1
   a b c
 -----------------------
0 |r b b |
1 | - - - |
2 |  - - - |
   -----------------------
10
   a b c
 -----------------------
0 |r - b |
1 | - b - |
2 |  - - - |
   -----------------------
6
   a b c
 -----------------------
0 |r - b |
1 | - - - |
2 |  - b - |
   -----------------------
2
   a b c
 -----------------------
0 |r - b |
1 | - - b |
2 |  - - - |
   -----------------------
2
   a b c
 -----------------------
0 |r - b |
1 | - - - |
2 |  - - b |
   -----------------------
3
max: {1: (0, 0)}
   a b c
 -----------------------
0 |- - b |
1 | r - - |
2 |  - - - |
   -----------------------
   a b c
 -----------------------
0 |b - b |
1 | r - - |
2 |  - - - |
   -----------------------
3
   a b c
 -----------------------
0 |- - b |
1 | r - - |
2 |  b - - |
   -----------------------
2
   a b c
 -----------------------
0 |- b b |
1 | r - - |
2 |  - - - |
   -----------------------
1
max: {1: (0, 1)}
   a b c
 -----------------------
0 |- - b |
1 | - - - |
2 |  r - - |
   -----------------------
   a b c
 -----------------------
0 |b - b |
1 | - - - |
2 |  r - - |
   -----------------------
2
   a b c
 -----------------------
0 |- - b |
1 | b - - |
2 |  r - - |
   ----------------------
7
   a b c
 ----------------------
0 |- b b |
1 | - - - |
2 |  r - - |
   ----------------------
2
   a b c
 ----------------------
0 |- - b |
1 | - b - |
2 |  r - - |
   ----------------------
8
   a b c
 ----------------------
0 |- - b |
1 | - - - |
2 |  r b - |
   ----------------------
8
   a b c
 ----------------------
0 |- - b |
1 | - - b |
2 |  r - - |
   ----------------------
4
   a b c
 ----------------------
0 |- - b |
1 | - - - |
2 |  r - b |
   ----------------------
8
max: {1: (0, 1), 2: (0, 2)}
   a b c
 ----------------------
0 |- r b |
1 | - - - |
2 |  - - - |
   ----------------------
   a b c
 ----------------------
0 |b r b |
1 | - - - |
2 |  - - - |
   -----------------------
6
   a b c
 -----------------------
0 |- r b |
1 | b - - |
2 |  - - - |
   -----------------------
10
   a b c
 -----------------------
0 |- r b |
1 | - - - |
2 |  b - - |
   -----------------------
4
   a b c
 -----------------------
0 |- r b |
1 | - b - |
2 |  - - - |
   -----------------------
10
   a b c
 -----------------------
0 |- r b |
1 | - - - |
2 |  - b - |
   -----------------------
8
   a b c
 -----------------------
0 |- r b |
1 | - - b |
2 |  - - - |
   -----------------------
2
max: {1: (0, 1), 2: (1, 0)}
   a b c
 -----------------------
0 |- - b |
1 | - r - |
2 |  - - - |
   -----------------------
   a b c
 -----------------------
```

```
0 |b - b |
1 | - r - |
2 |  - - - |
   ----------------------
5
   a b c
 ----------------------
0 |- - b |
1 | b r - |
2 |  - - - |
   ----------------------
5
   a b c
 ----------------------
0 |- - b |
1 | - r - |
2 |  b - - |
   ----------------------
10
   a b c
 ----------------------
0 |- b b |
1 | - r - |
2 |  - - - |
   ----------------------
9
   a b c
 ----------------------
0 |- - b |
1 | - r - |
2 |  - b - |
   ----------------------
5
   a b c
 ----------------------
0 |- - b |
1 | - r b |
2 |  - - - |
   ----------------------
2
max: {1: (0, 1), 2: (1, 1)}
   a b c
 ----------------------
0 |- - b |
1 | - - - |
2 |  - r - |
   ----------------------
   a b c
 -----------------------
0 |b - b |
1 | - - - |
2 |  - r - |
   -----------------------
5
   a b c
 -----------------------
0 |- - b |
1 | b - - |
2 |  - r - |
   -----------------------
1
max: {1: (1, 2), 2: (1, 1)}
   a b c
 -----------------------
0 |- - b |
1 | - - r |
2 |  - - - |
   -----------------------
   a b c
 -----------------------
0 |b - b |
1 | - - r |
2 |  - - - |
   -----------------------
10
   a b c
 -----------------------
0 |- - b |
1 | b - r |
2 |  - - - |
   -----------------------
9
   a b c
 -----------------------
0 |- - b |
1 | - - r |
2 |  b - - |
   -----------------------
5
   a b c
 -----------------------
0 |- b b |
1 | - - r |
2 |  - - - |
   -----------------------
6
```

```
   a b c
 -----------------------
0 |- - b |
1 | - b r |
2 |  - - - |
   -----------------------
2
max: {1: (1, 2), 2: (2, 1)}
   a b c
 -----------------------
0 |- - b |
1 | - - - |
2 |  - - r |
   -----------------------
   a b c
 -----------------------
0 |b - b |
1 | - - - |
2 |  - - r |
   -----------------------
5
   a b c
 -----------------------
0 |- - b |
1 | b - - |
2 |  - - r |
   -----------------------
1
max: {1: (2, 2), 2: (2, 1)}
```

[36]: 2

[37]:
```
board = HexBoard(2)
board.print()
```

```
   a b
 -----------------------
0 |- - |
1 | - - |
   -----------------------
```

[39]:
```
board.place((0,0),board.BLUE)
board.print()
```

```
   a b
 -----------------------
0 |b - |
1 | - - |
   -----------------------
```

[40]:
```
alpha(board,depth=3)
```

```
   a b
 -----------------------
0 |b - |
1 | r - |
   -----------------------
   a b
 -----------------------
0 |b b |
1 | r - |
   -----------------------
6
   a b
 -----------------------
0 |b - |
1 | r b |
   -----------------------
   a b
 -----------------------
0 |b r |
1 | r b |
   -----------------------
9
max: {9: (1, 0)}
max: {9: (1, 0), 6: (0, 1)}
   a b
 -----------------------
0 |b r |
1 | - - |
   -----------------------
   a b
 -----------------------
0 |b r |
1 | b - |
   -----------------------
   a b
 -----------------------
0 |b r |
1 | b r |
   -----------------------
10
max: {10: (1, 1)}
   a b
 -----------------------
0 |b r |
1 | - b |
   -----------------------
   a b
 -----------------------
0 |b r |
1 | r b |
   -----------------------
6
max: {6: (0, 1)}
max: {6: (1, 0)}
   a b
 -----------------------
0 |b - |
1 | - r |
   -----------------------
   a b
 -----------------------
0 |b - |
1 | b r |
   -----------------------
   a b
 -----------------------
0 |b r |
1 | b r |
   -----------------------
9
max: {9: (1, 0)}
   a b
 -----------------------
0 |b b |
1 | - r |
   -----------------------
4
max: {9: (1, 0), 4: (1, 1)}
```

[40]: 6

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: