Table evaluations

July 1, 2015

In [1]:
```
cd /data/convchess/src/play/
```

```
/data/convchess/src/play
```

In [2]:
```
import keras
import cPickle as pkl
import sys
import numpy as np
sys.setrecursionlimit(40000)
class CNN_evaluator:
        """docstring for CNN_evaluation"""
        def __init__(self, model_file):
                self.model = pkl.load(open(model_file, 'r'))

        def evaluate(self, im):
                return self.model.predict(np.asarray([im]), verbose=0)[0][0]

        def evaluate_batch(self, batch):
                #batch is much faster than one individually
                #print batch
                return self.model.predict(np.asarray(batch), verbose=0, batch_size=2048)[:,0]
```

In [3]:
```
evaluator = CNN_evaluator('regression_models/model_g07_bp_largedeep.pkl')
```

```
Using gpu device 0: GeForce GTX 760
```

In [5]:
```
from play4 import *
import sunfish
pos = sunfish.Position(sunfish.initial, 0, (True,True), (True,True), 0, 0)
bb = pos_board_to_bitboard(pos.board)
print bb
im = convert_bitboard_to_image(bb)
im = np.rollaxis(im, 2, 0)
print im
```

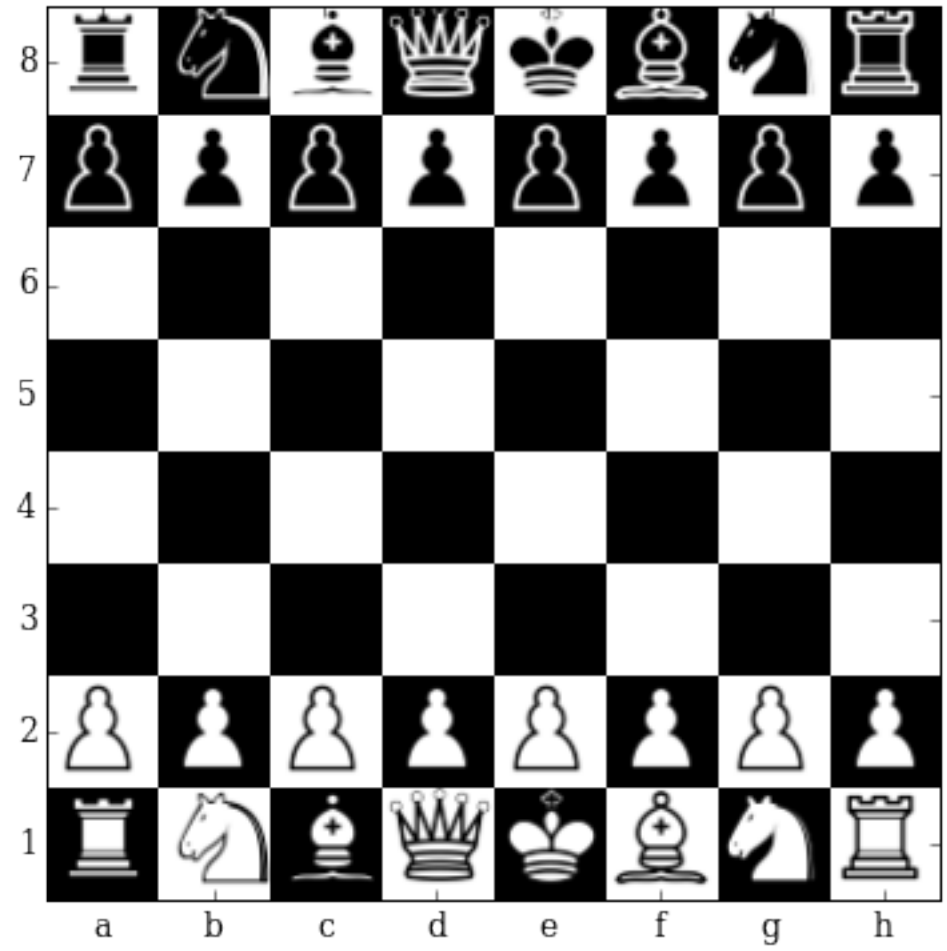
```
r n b q k b n r
p p p p p p p p
. . . . . . . .
. . . . . . . .
. . . . . . . .
. . . . . . . .
P P P P P P P P
R N B Q K B N R
[[[ 0.  0.  0.  0.  0.  0.  0.  0.]
  [-1. -1. -1. -1. -1. -1. -1. -1.]
```

```
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 1.  1.  1.  1.  1.  1.  1.  1.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]]

 [[-1.  0.  0.  0.  0.  0.  0. -1.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 1.  0.  0.  0.  0.  0.  0.  1.]]

 [[ 0. -1.  0.  0.  0.  0. -1.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  1.  0.  0.  0.  0.  1.  0.]]

 [[ 0.  0. -1.  0.  0. -1.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  1.  0.  0.  1.  0.  0.]]

 [[ 0.  0.  0. -1.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  1.  0.  0.  0.  0.]]

 [[ 0.  0.  0.  0. -1.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  0.  0.  0.  0.]
  [ 0.  0.  0.  0.  1.  0.  0.  0.]]]
```

In [6]:
```
print "Evaluation of the starting board %f"%evaluator.evaluate(im)
```

Out[6]:
```
0.0015796410152688622
```

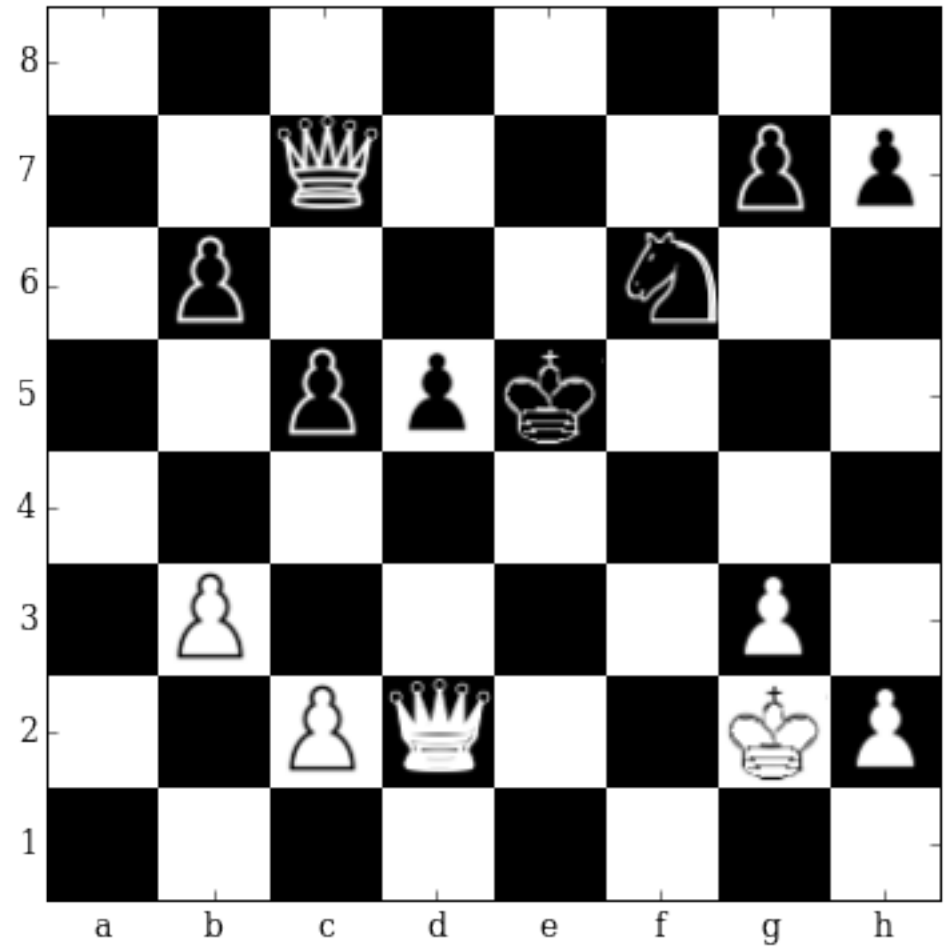
```
In [7]: from example_moves.draw_board.draw import *

In [8]: cd example_moves/draw_board/

/data/convchess/src/util/draw_board

In [9]: def parseFEN(fen):
            """ Parses a string in Forsyth-Edwards Notation into a Position """
            board, color, castling, enpas, hclock, fclock = fen.split()
            board = re.sub('\d', (lambda m: '.'*int(m.group(0))), board)
            board = ' '*19+'\n ' + '\n '.join(board.split('/')) + ' \n'+' '*19
            wc = ('Q' in castling, 'K' in castling)
            bc = ('k' in castling, 'q' in castling)
            ep = sunfish.parse(enpas) if enpas != '-' else 0
            score = sum(sunfish.pst[p][i] for i,p in enumerate(board) if p.isupper())
            score -= sum(sunfish.pst[p.upper()][i] for i,p in enumerate(board) if p.islower())
            pos = sunfish.Position(board, score, wc, bc, ep, 0)
            return pos if color == 'w' else pos.rotate()
    def posboard_to_im(pos):
        bb = pos_board_to_bitboard(pos)
        im = convert_bitboard_to_image(bb)
        im = np.rollaxis(im,2,0)
        return im
    def fen_to_im(fen):
        pos = parseFEN(fen)
        return posboard_to_im(pos.board)
    def image_board(pos):
        plt.xticks(range(8), string.lowercase)
        ax = plt.imshow(draw_board(posboard_to_im(pos)), cmap=cm.Greys_r)
        f = plt.gcf()
        f.set_size_inches(6,6)
        ax.set_extent([-0.5,7.5,0.5,8.5])
        plt.show()

In [10]: %matplotlib inline
         import matplotlib.pyplot as plt
         import matplotlib.cm as cm
         plt.rc('text', usetex=False, antialiased=True, hinting='auto', dvipnghack=True)
         plt.rc('font', family='serif', size=12)
         image_board(pos.board)
```




```
In [11]: def top_bottom_moves(fen, top=5, bottom=5):
             pos = parseFEN(fen)
             bb = pos_board_to_bitboard(pos.board)
             image_board(pos.board)
             im = convert_bitboard_to_image(bb)
             im = np.rollaxis(im, 2, 0)
             print "\tCurrent evaluation: %f"%evaluator.evaluate(im)
             future_boards = []
             pos_children = []
             moves = list(pos.genMoves())
             print "Total %d moves possible"%len(moves)
             for move in moves:
                 pos_child = pos.move(move).rotate()
                 bb = pos_board_to_bitboard(pos_child.board)
                 im = convert_bitboard_to_image(bb)
                 im = np.rollaxis(im, 2, 0)
                 future_boards.append(im)
                 pos_children.append(pos_child.board)
             #future_boards = np.asarray(future_boards)
             values = evaluator.evaluate_batch(future_boards)
```

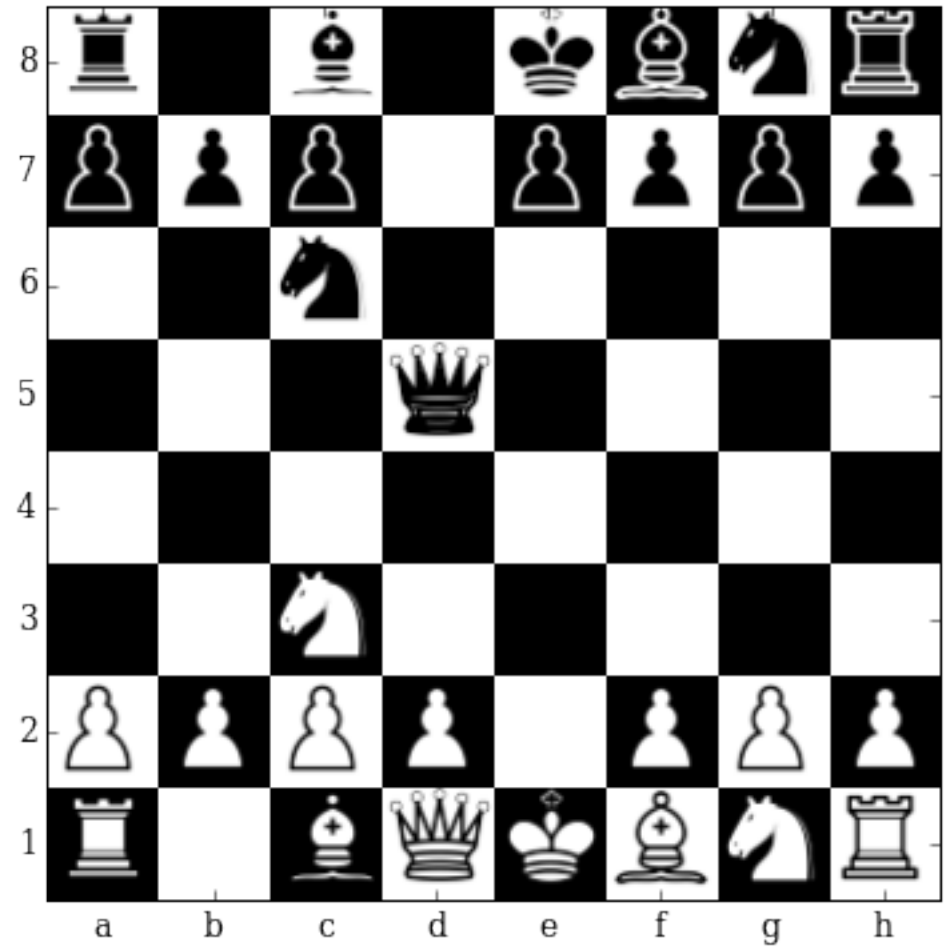
```
        poschildren_vals = sorted(zip(pos_children , moves, values), key=operator.itemgetter(2), r
        print "TOP 5 Moves"
        for pos_child, move, value in poschildren_vals[0:top]:
            image_board(pos_child)
            print '\t',sunfish.render(move[0]) + sunfish.render(move[1]), value
        print "WORST 5 Moves"
        for pos_child, move, value in poschildren_vals[-bottom:]:
            image_board(pos_child)
            print '\t',sunfish.render(move[0]) + sunfish.render(move[1]), value
```

In [12]:
```
'''
Piece        pawn        knight        bishop        rook        queen
Value        1        3        3        5        9
'''
fen='8/2q3pp/1p3n2/2ppk3/8/1P4P1/2PQ2KP/8 w - - 0 0'
im = fen_to_im(fen)
pos = parseFEN(fen)
image_board(pos.board)
layer_scores=np.array([1,5,3,3,9,100])
def fen_to_tableval(fen):
    im = fen_to_im(fen)
    return im_to_tableval(im)
def im_to_tableval(im):
    #for a 6 layer representation
    return np.sum(np.sum(im,(1,2))*layer_scores)
print "Value of the table: %d"%im_to_tableval(im)
```

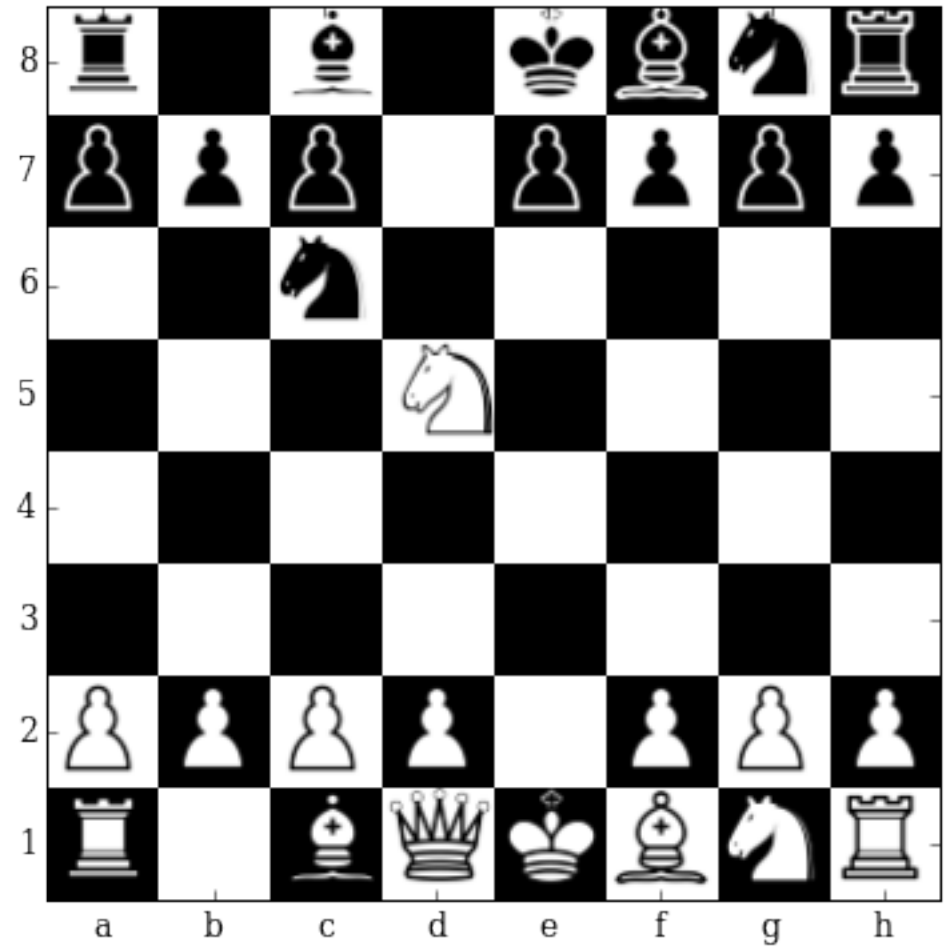



```
Value of the table: -4
```

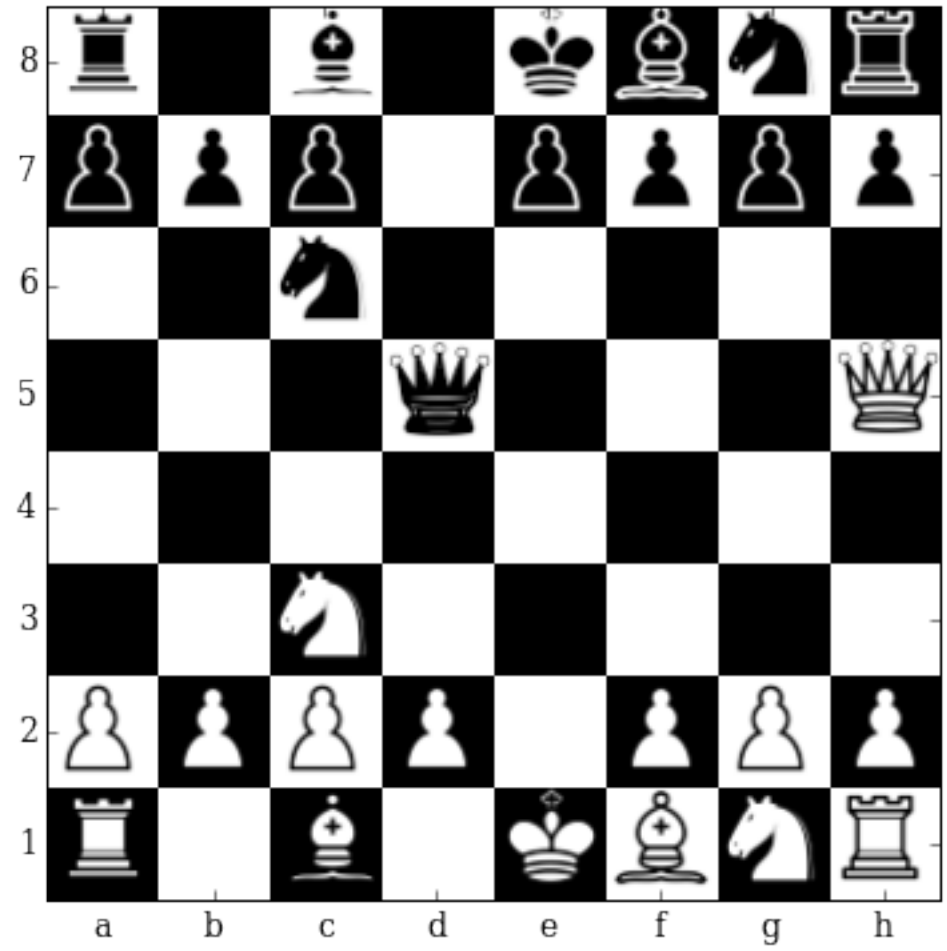
In [13]:
```
'''
Killing Queen with a knight
(Works)
'''
top_bottom_moves('r1b1kbnr/ppp1pppp/2n5/3q4/8/2N5/PPPP1PPP/R1BQKBNR w KQkq - 0 4')
```



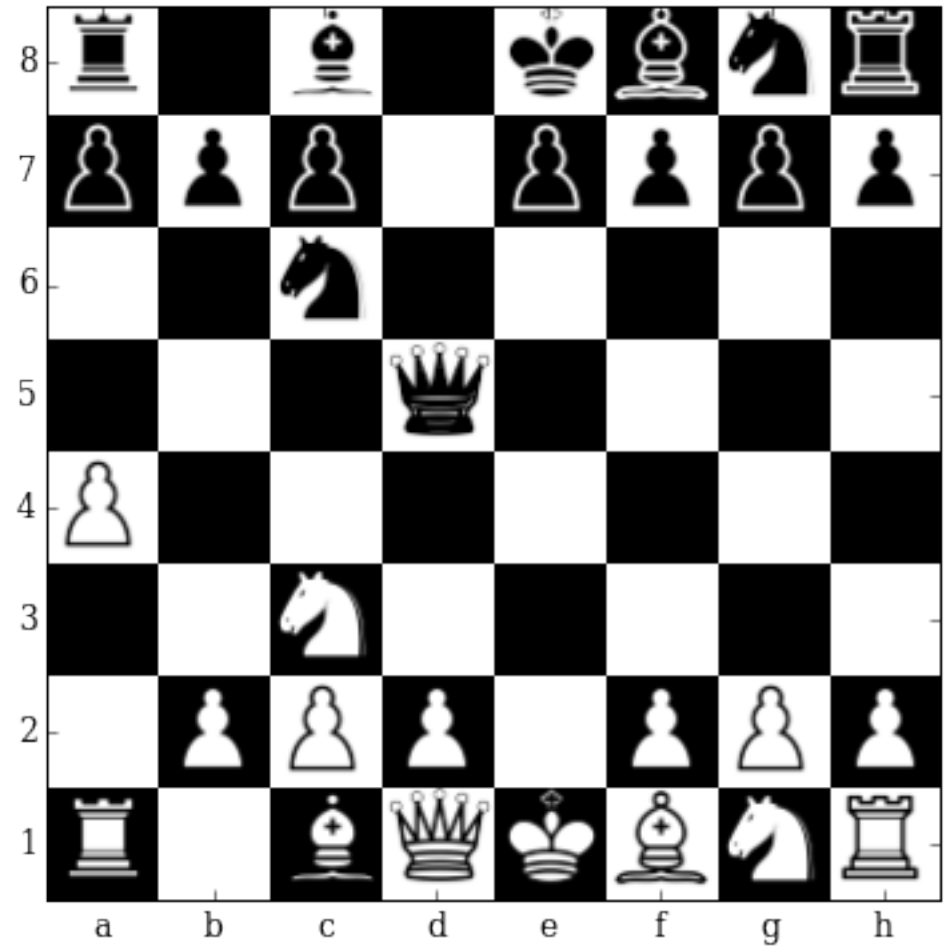

```
Current evaluation: 0.003406
Total 32 moves possible
TOP 5 Moves
```



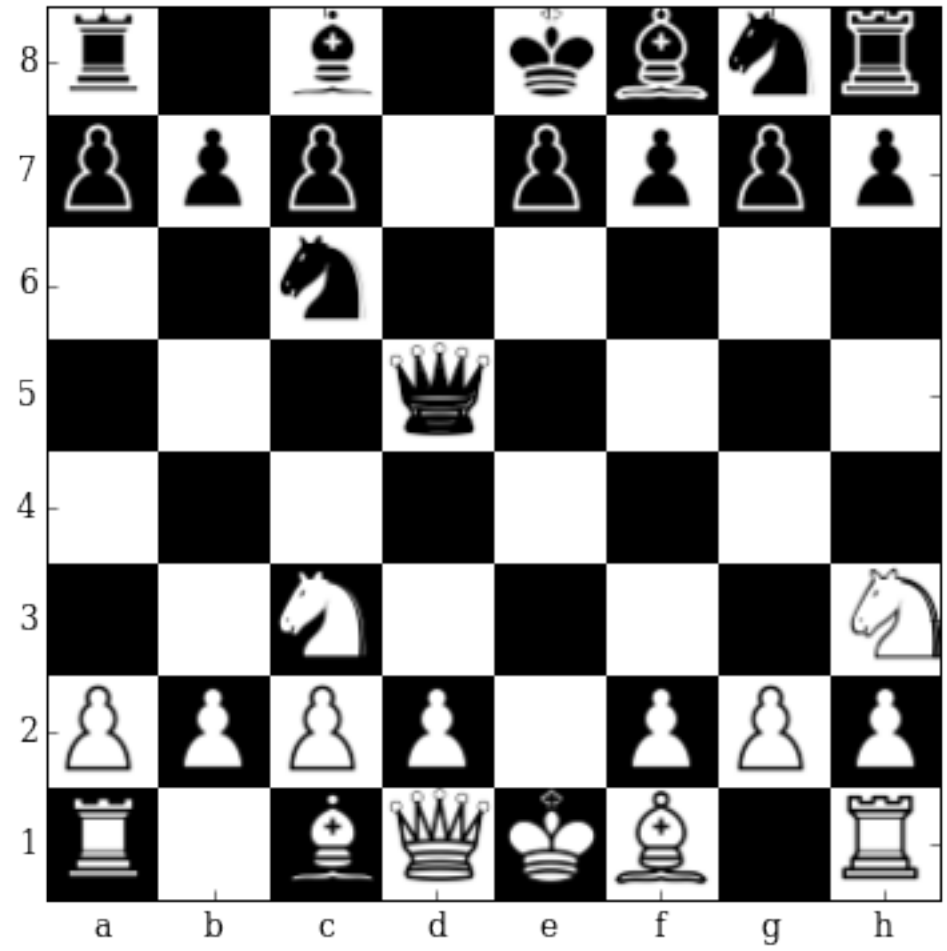

```
c3d5 0.0545512884855
```



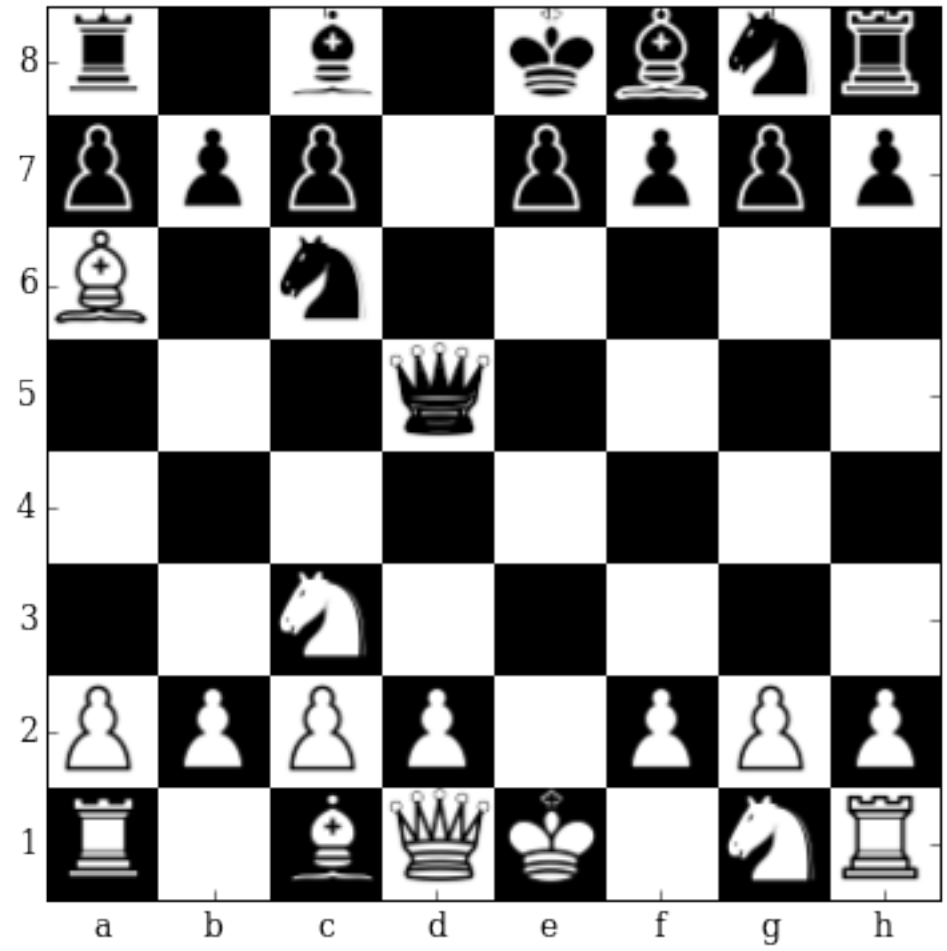
```
d1h5 0.0144034344703
```



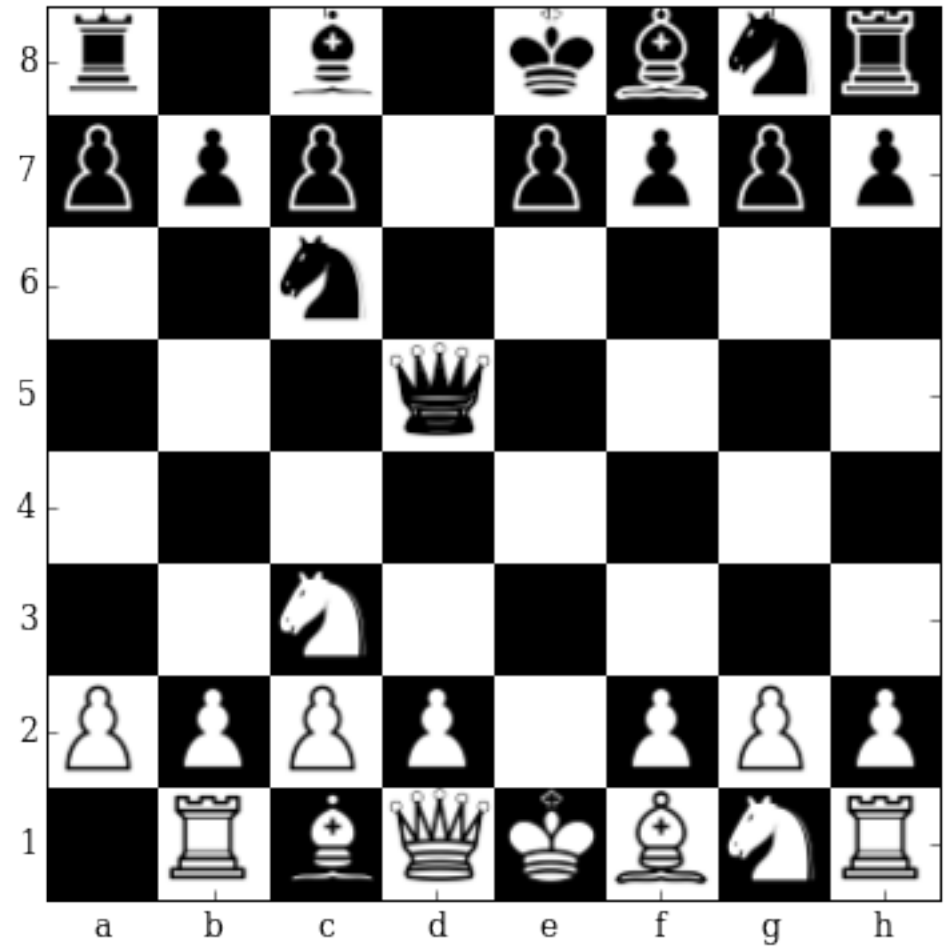

```
a2a4 0.0119769489393
```



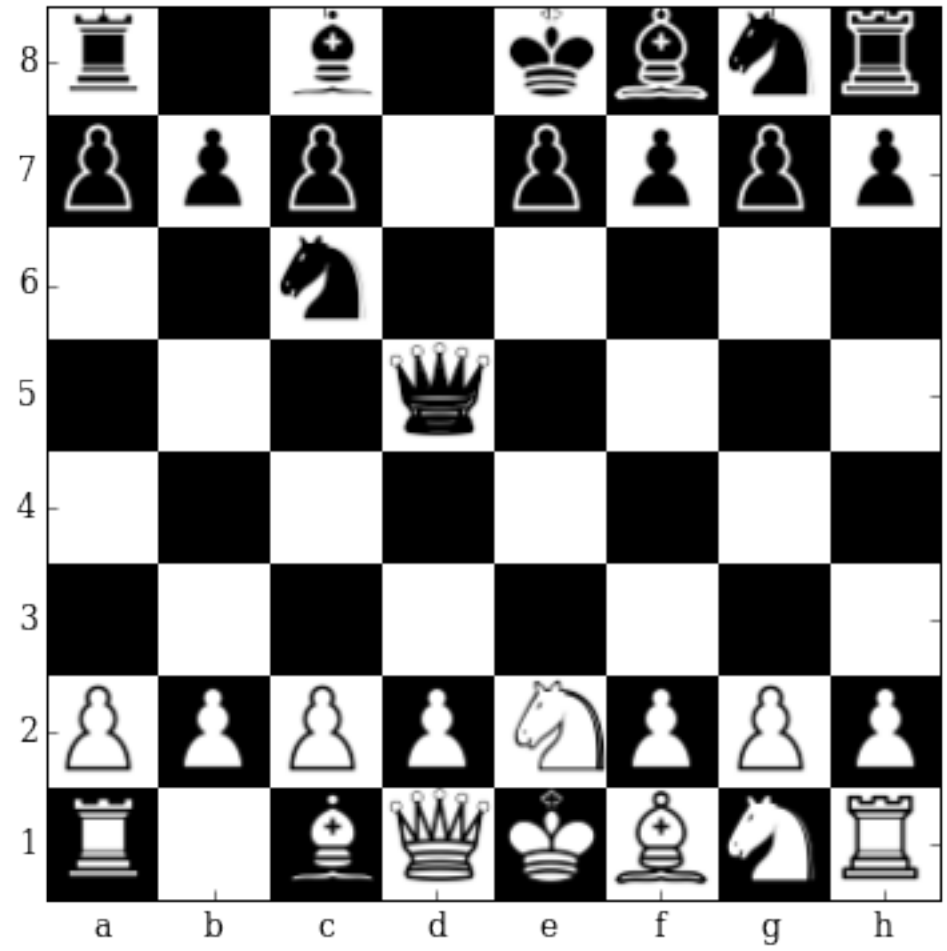
```
g1h3 0.0115632014349
```



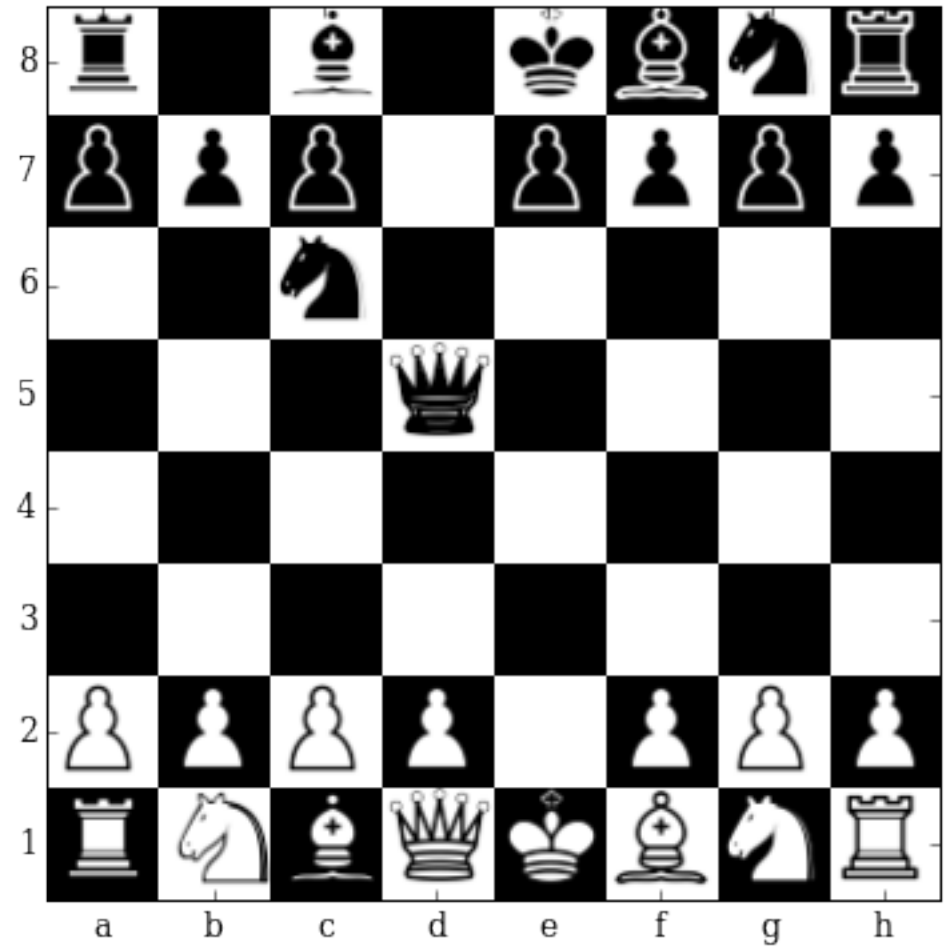

```
f1a6 0.0110727213323
WORST 5 Moves
```



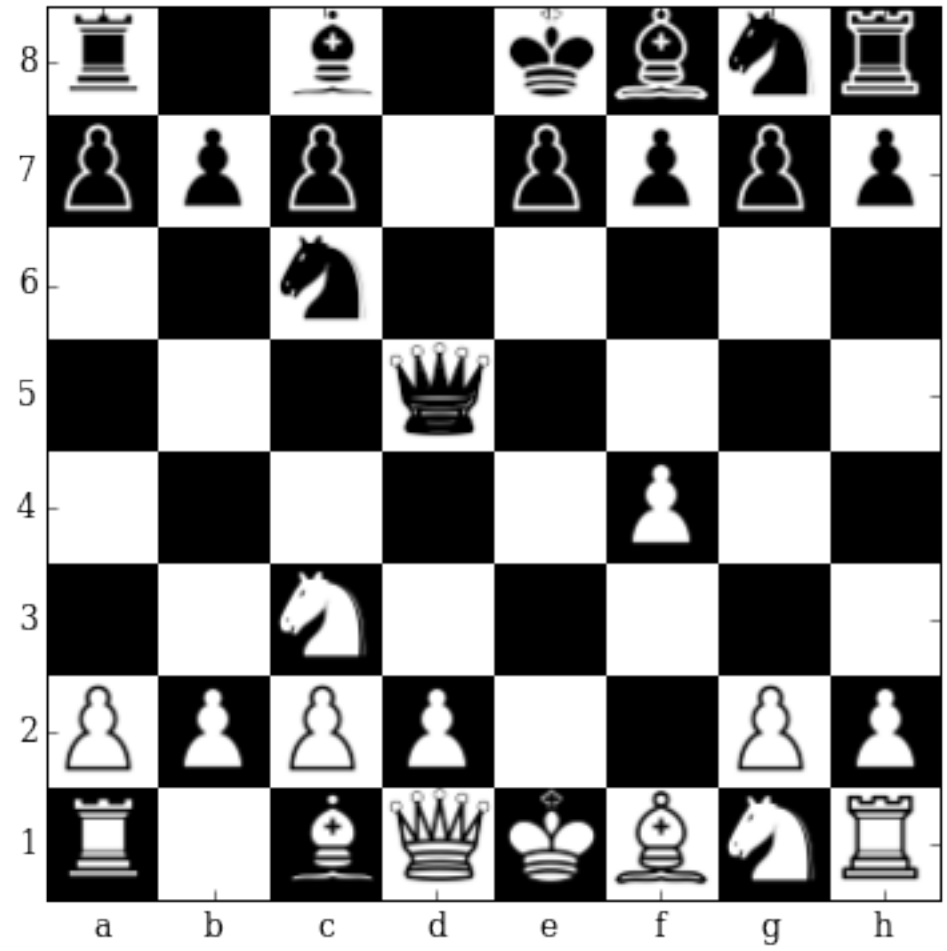

```
a1b1 -0.000657647731714
```



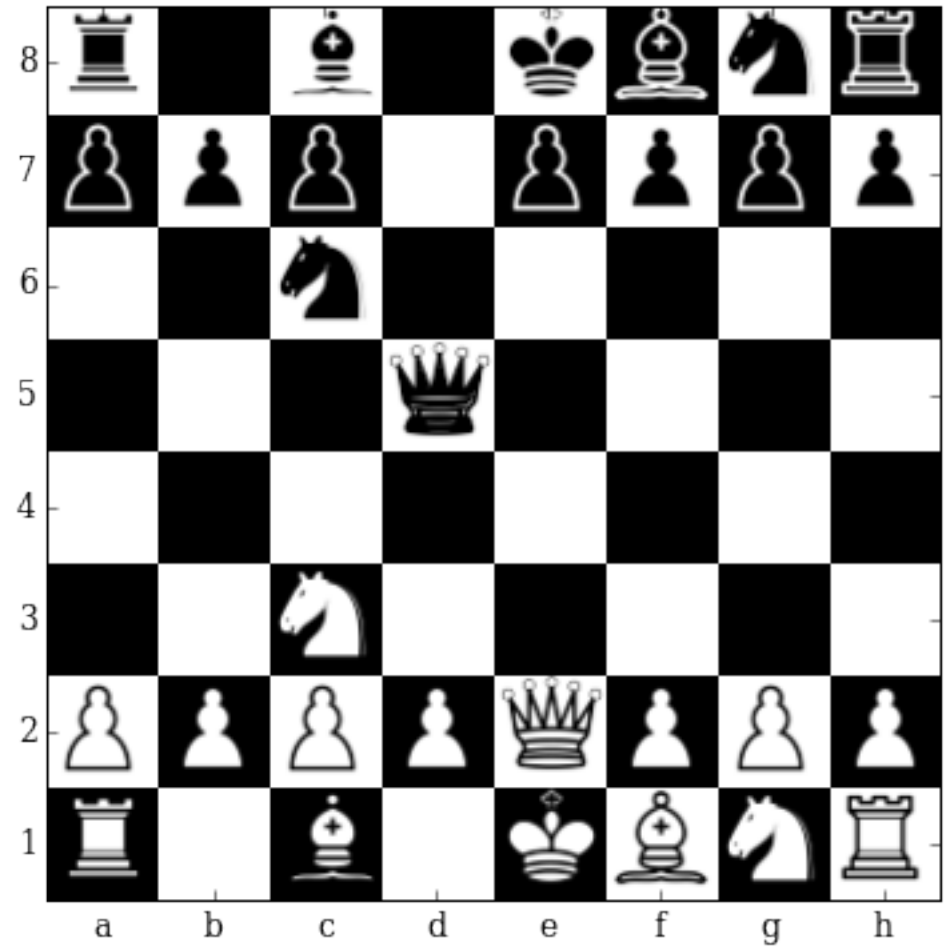

```
c3e2 -0.00339175574481
```




```
c3b1 -0.00497450307012
```

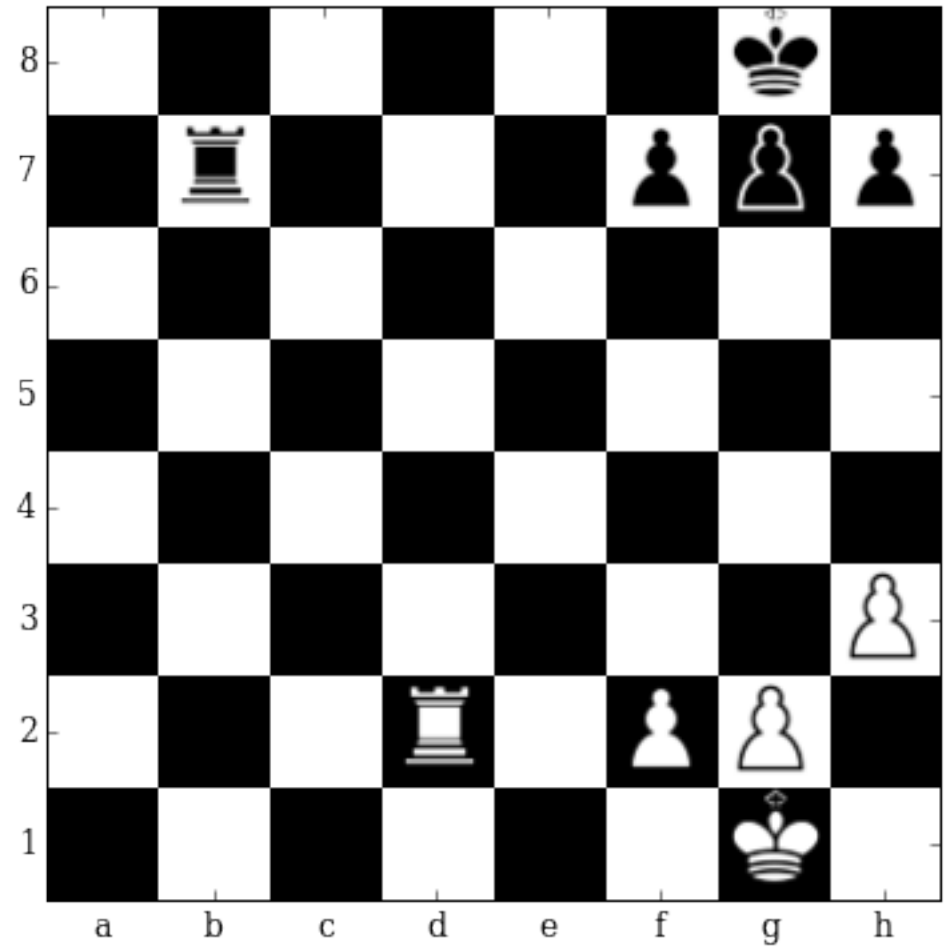



```
f2f4 -0.00507036363706
```

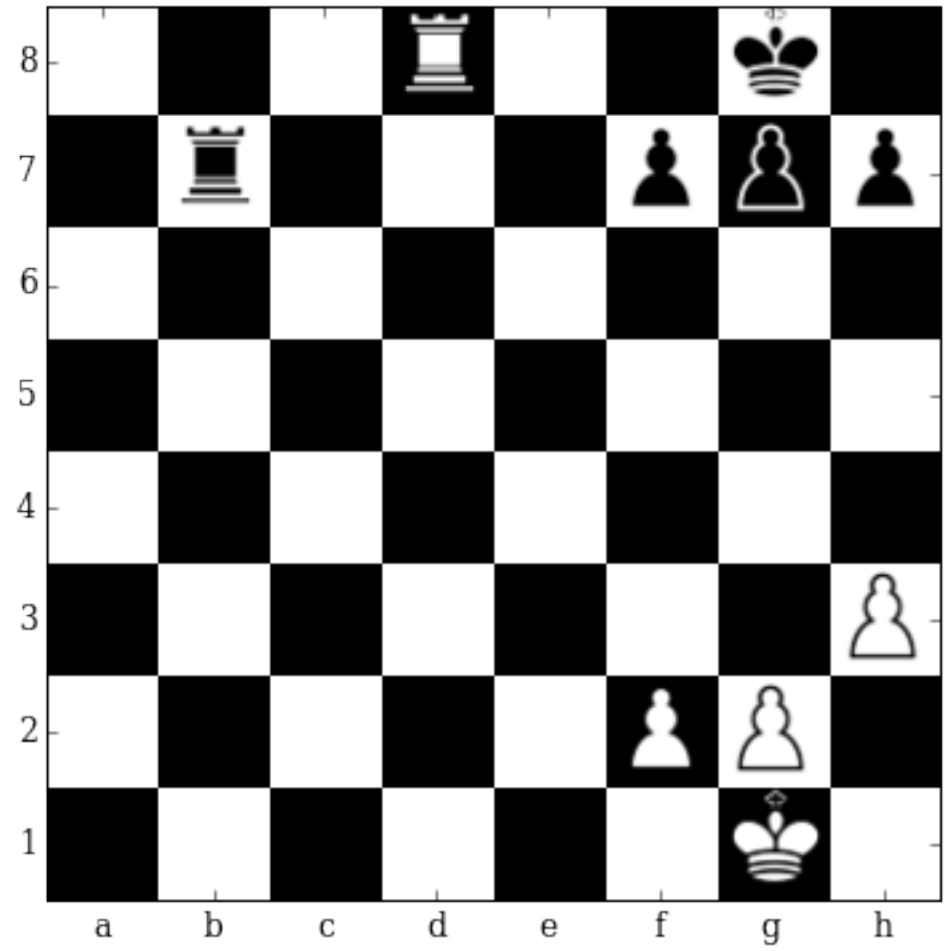



```
d1e2 -0.00937258545309
```

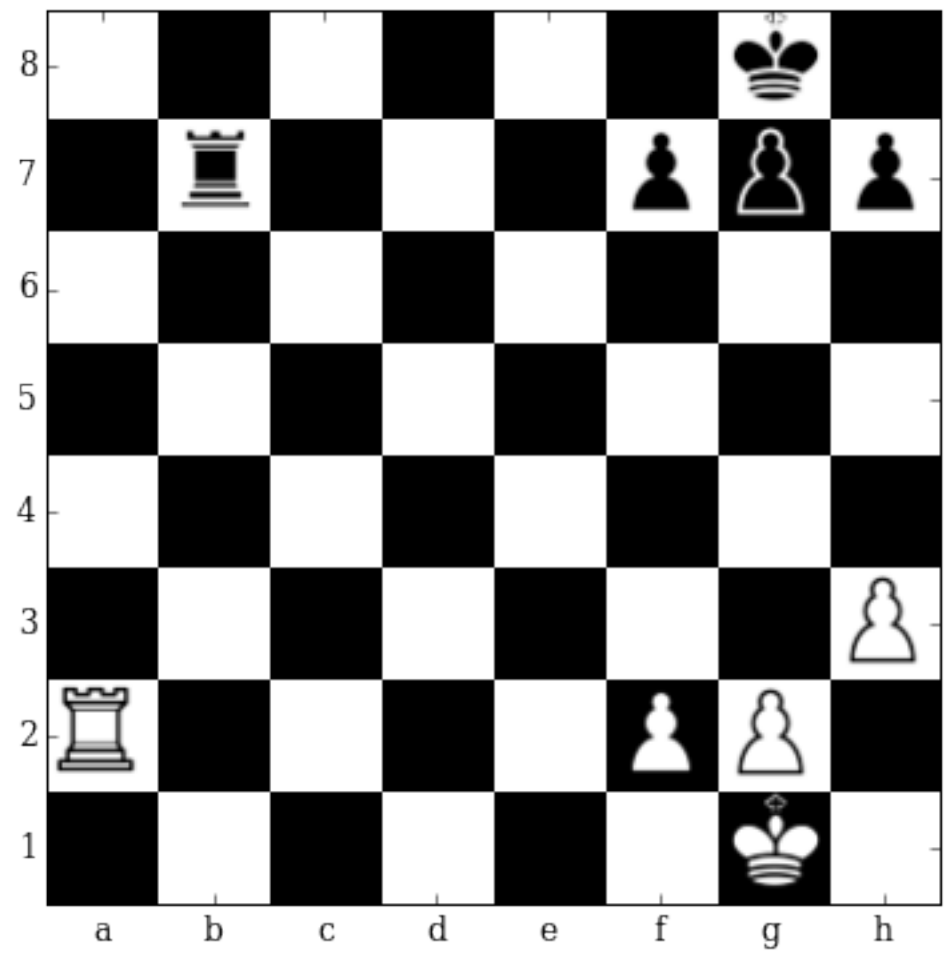
```
In [14]: '''
         Checkmating with a rook
         (Works)
         '''
         top_bottom_moves('6k1/1r3ppp/8/8/8/7P/3R1PP1/6K1 w - - 0 0')
```



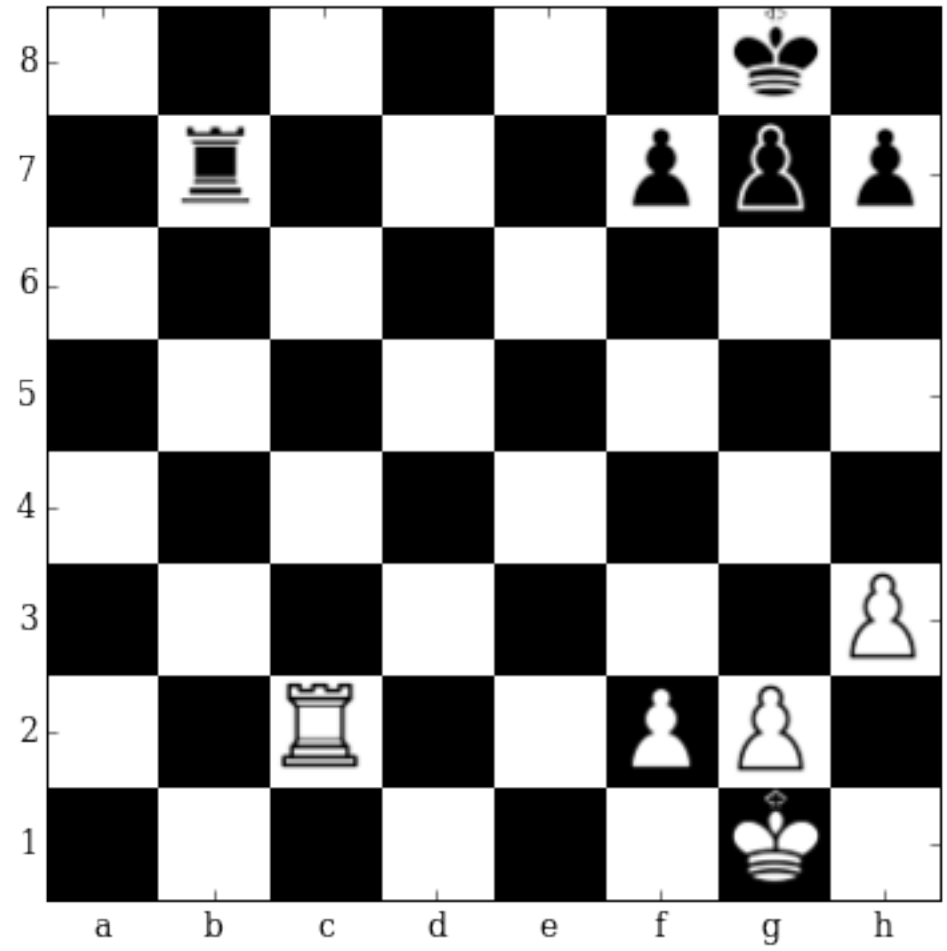

```
Current evaluation: -0.006740
Total 19 moves possible
TOP 5 Moves
```



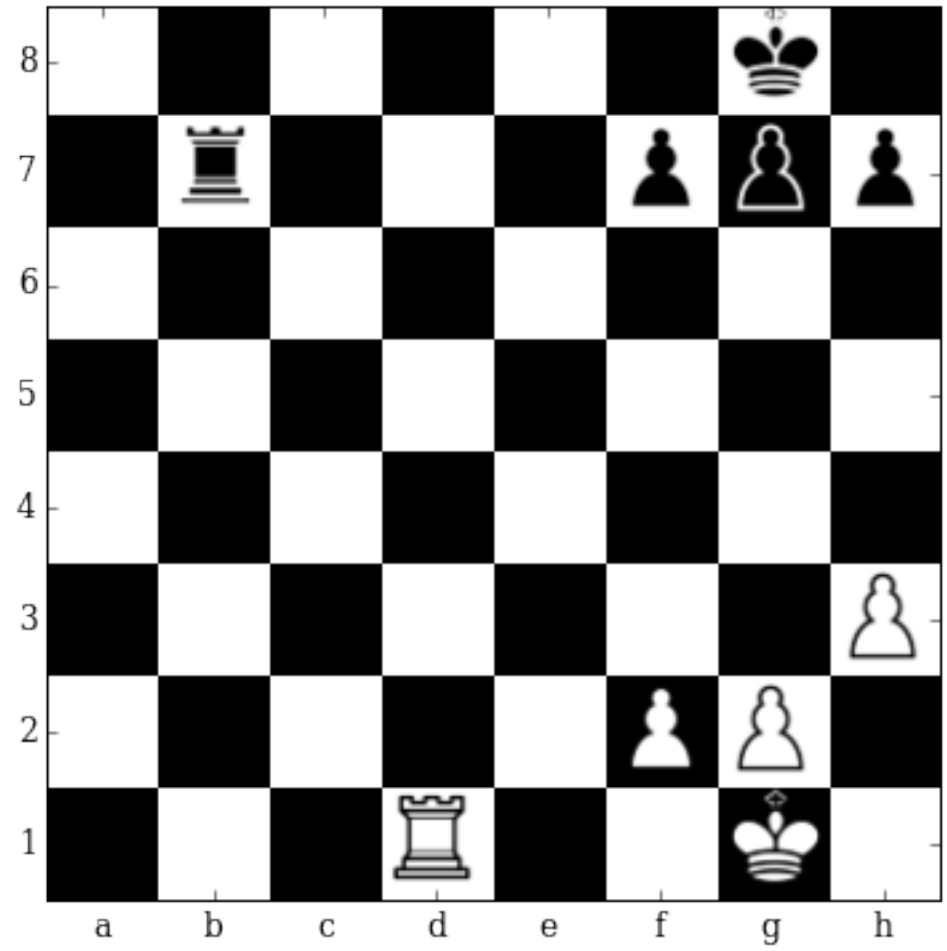

```
d2d8 0.0452536605299
```



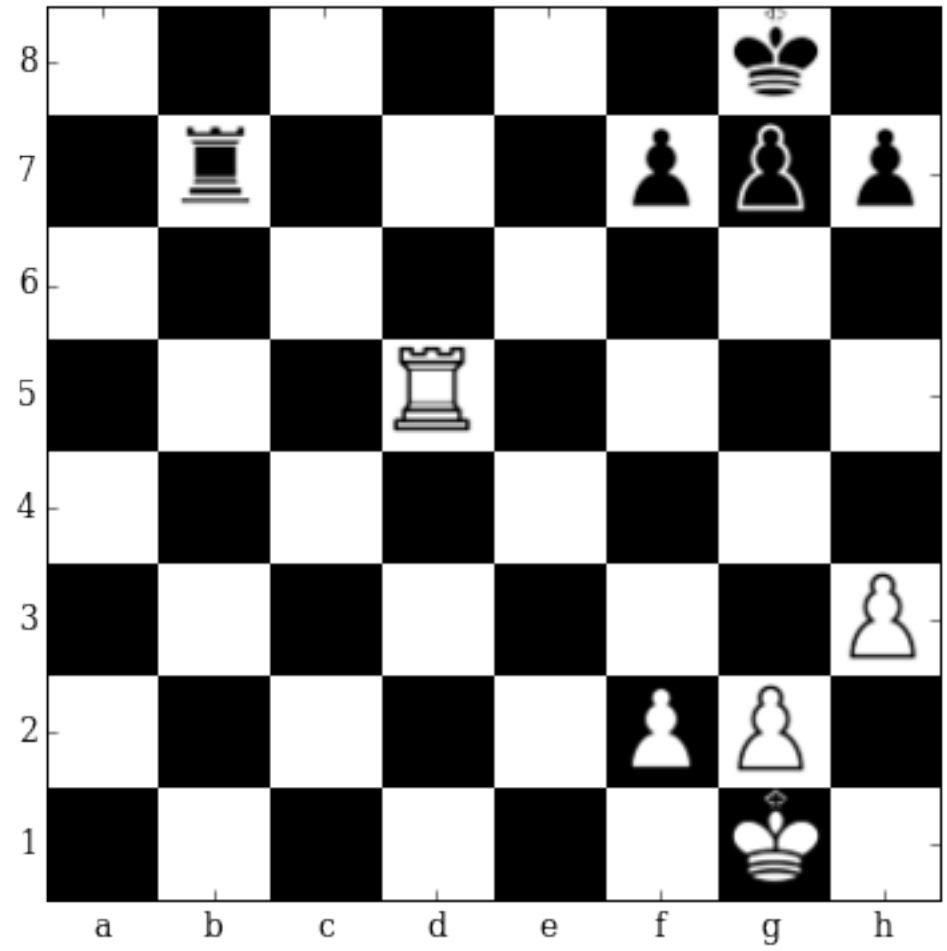
```
d2a2 0.0176544804126
```



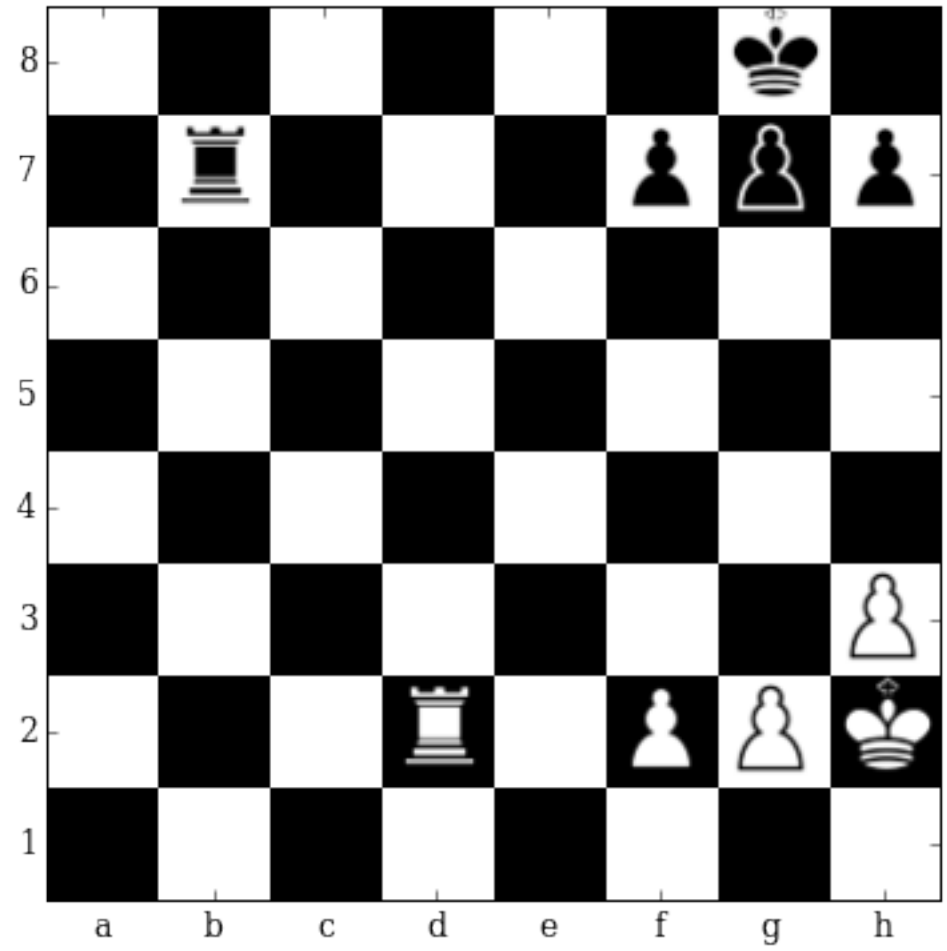
```
d2c2 0.00976785086095
```



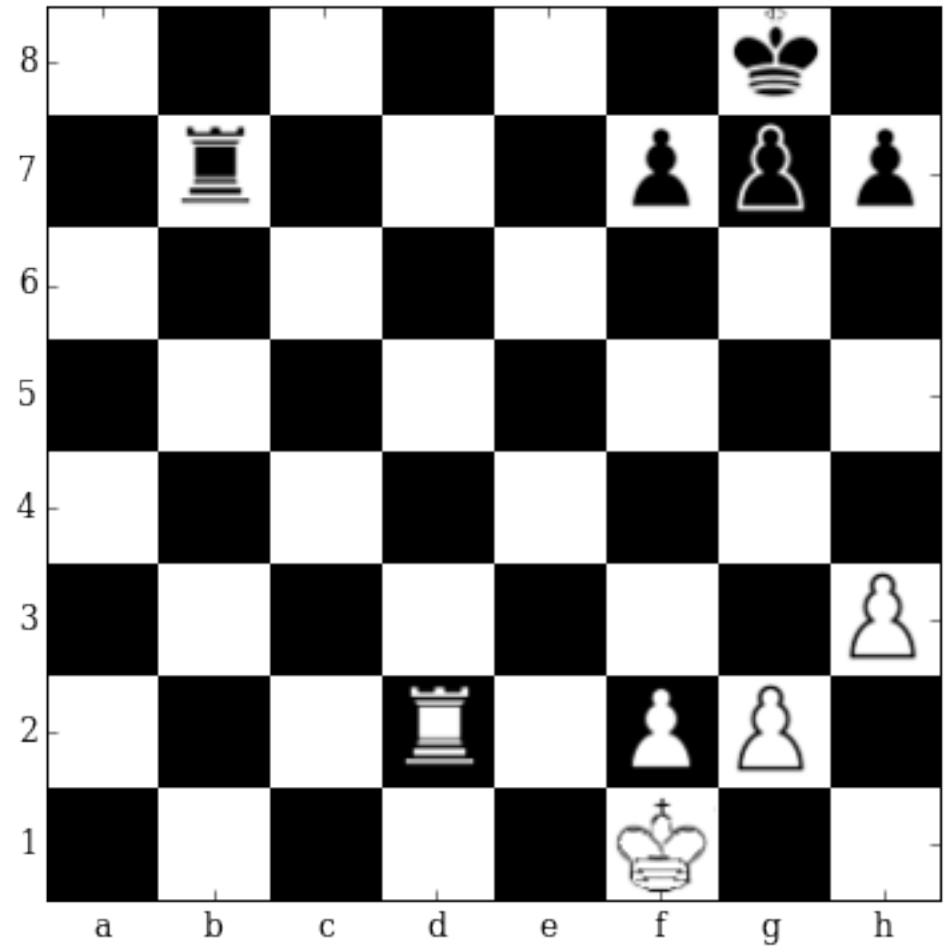
```
d2d1 0.00954846478999
```



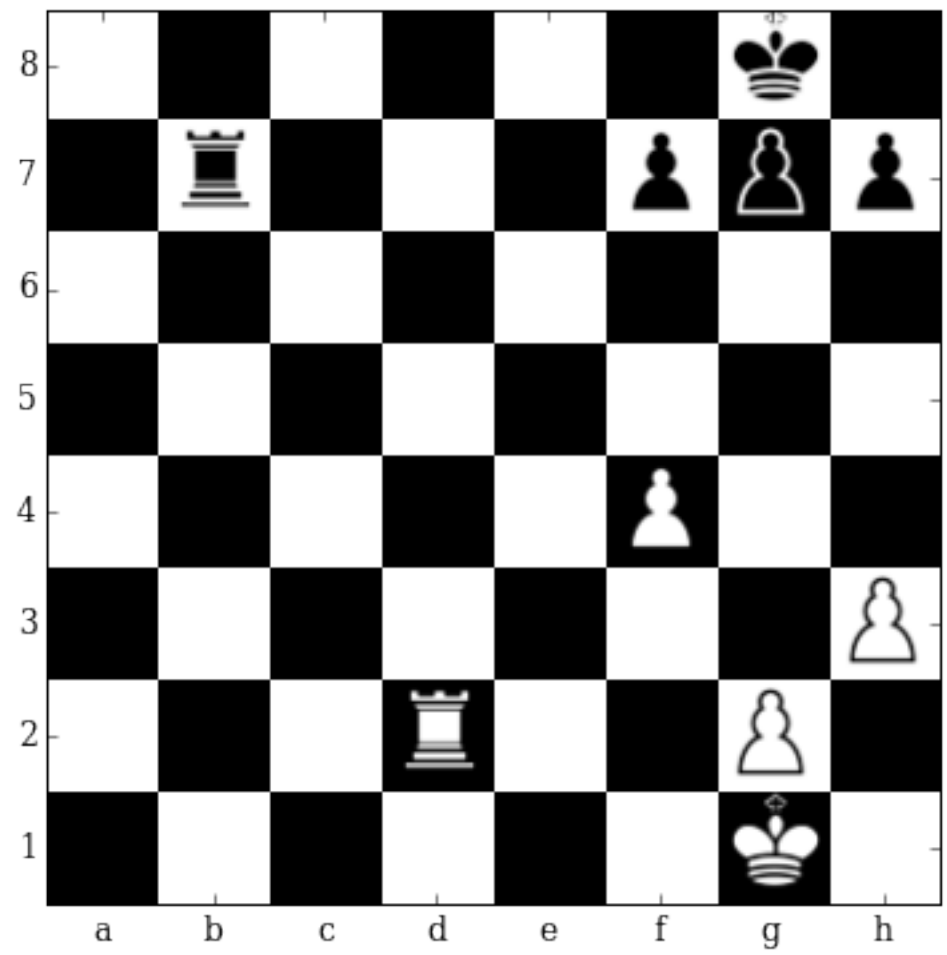

```
d2d5 0.00700665730983
WORST 5 Moves
```



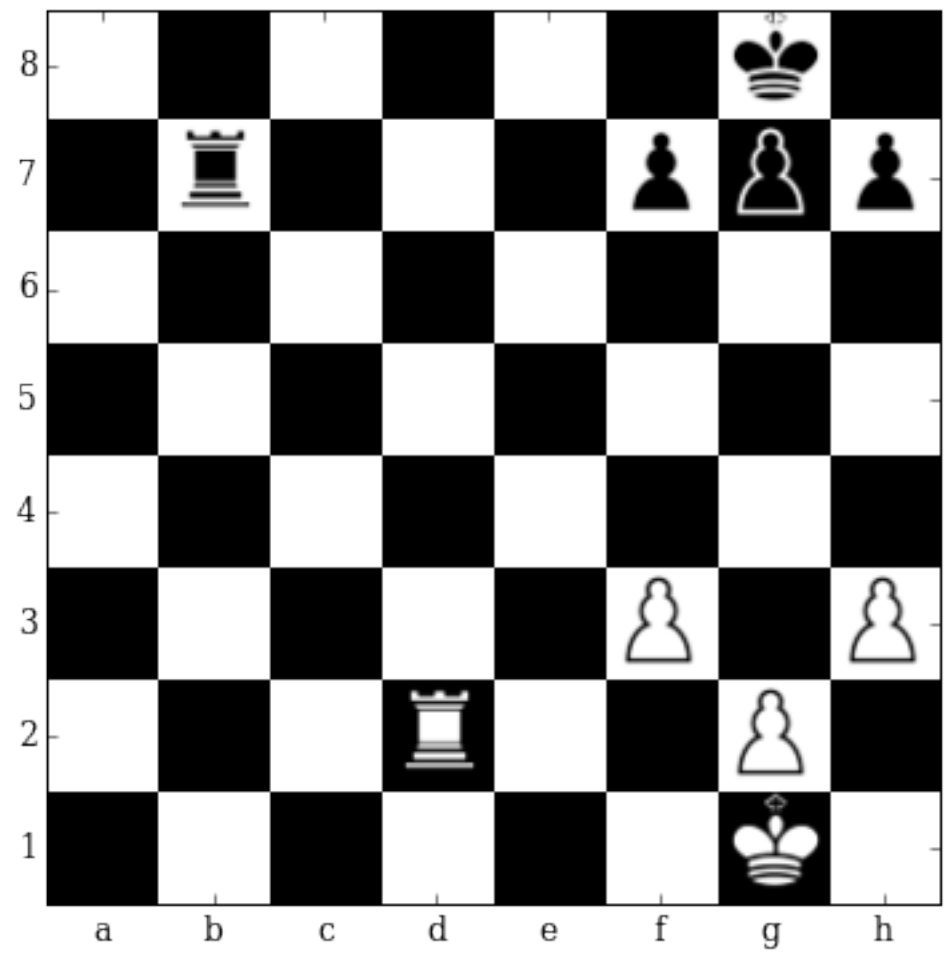
```
g1h2 -0.0147265279666
```



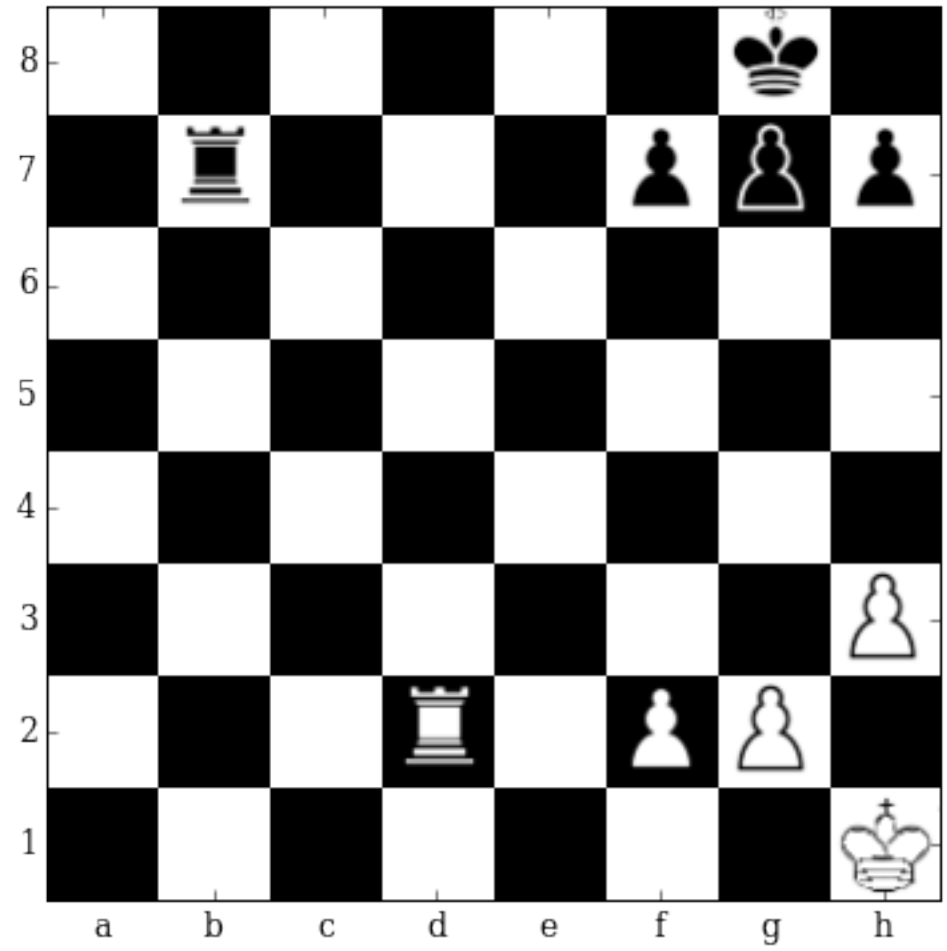

```
g1f1 -0.0185413267463
```



```
f2f4 -0.0185624714941
```

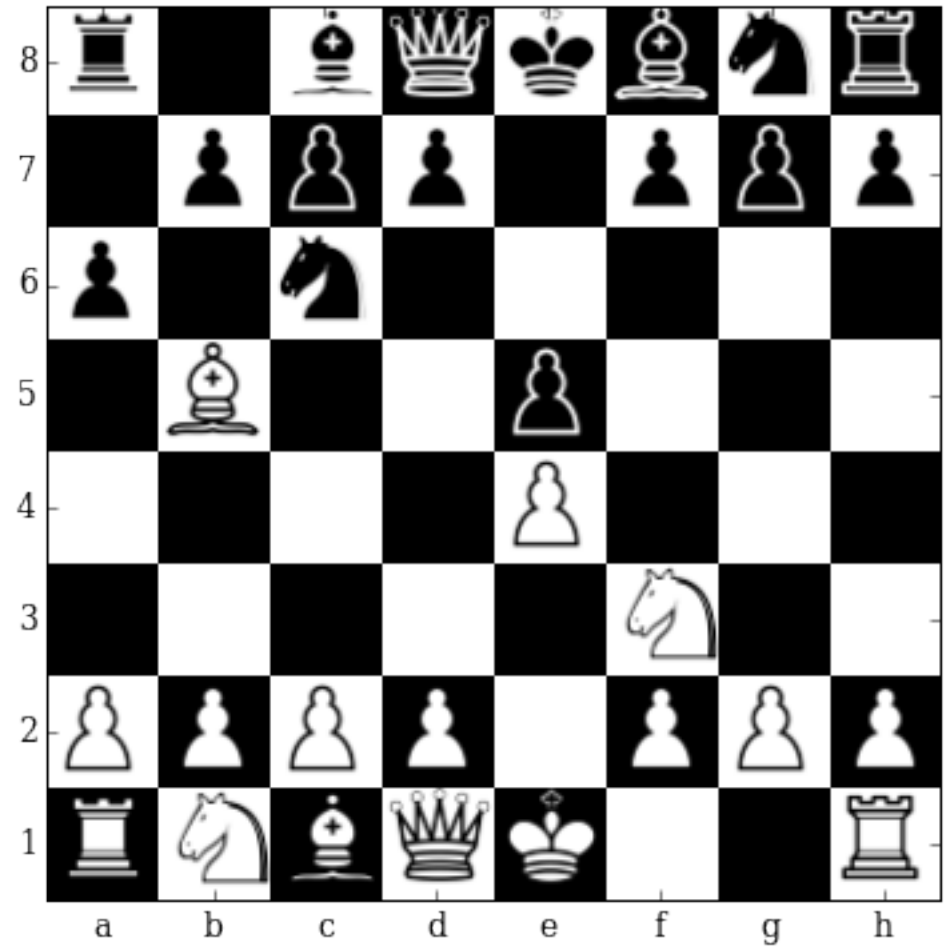



```
f2f3 -0.0199430584908
```

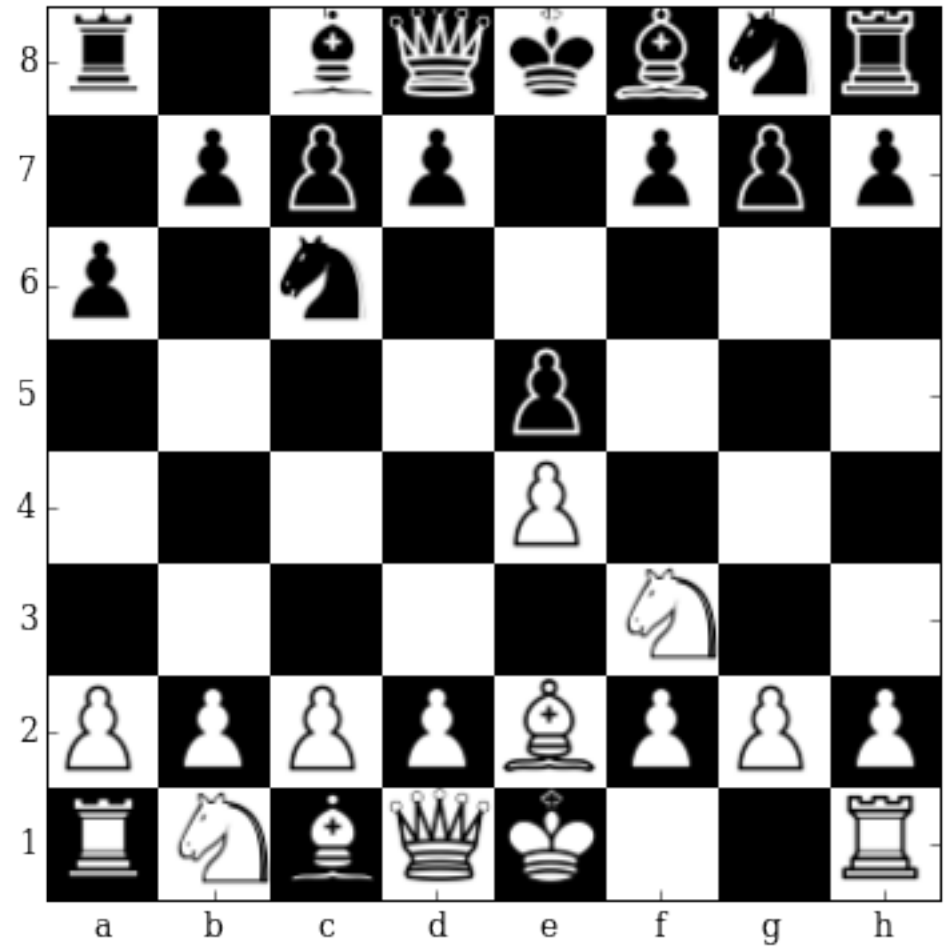



```
g1h1 -0.0221134386957
```

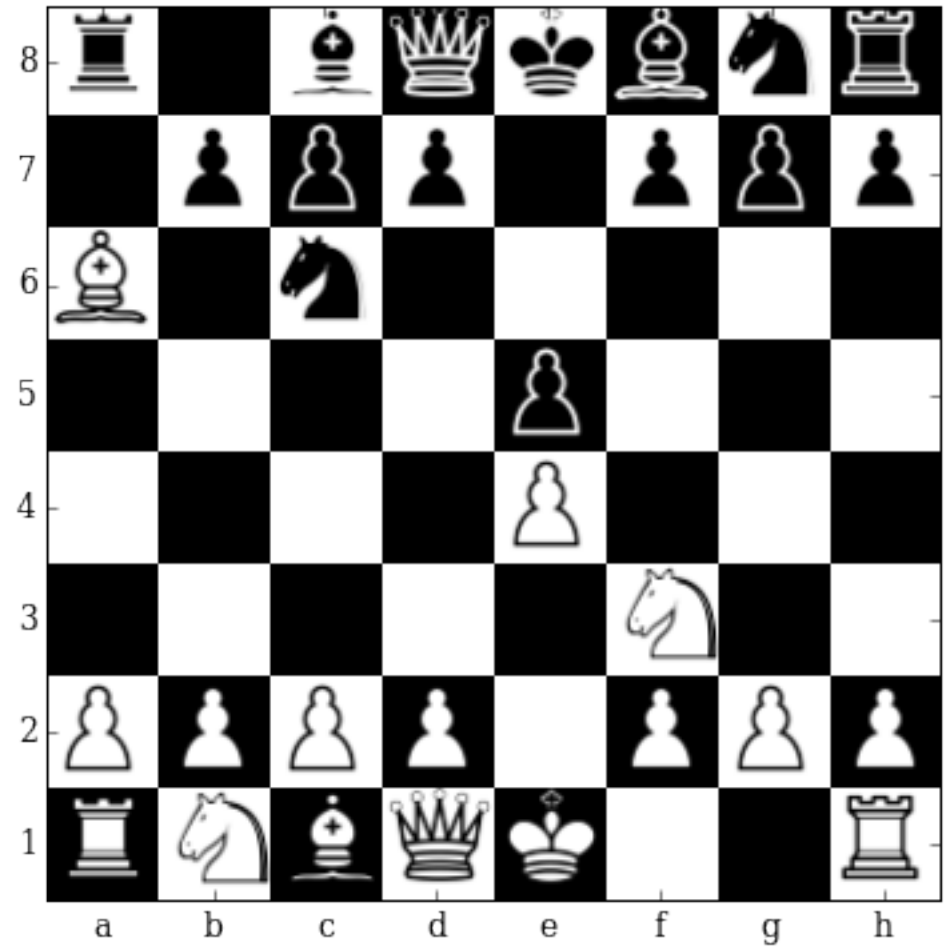
```
In [15]: '''
          Trading the knight with a bishop
         '''
         top_bottom_moves('r1bqkbnr/1ppp1ppp/p1n5/1B2p3/4P3/5N2/PPPP1PPP/RNBQK2R w KQkq - 0 0')
```




```
Current evaluation: -0.003171
Total 32 moves possible
TOP 5 Moves
```

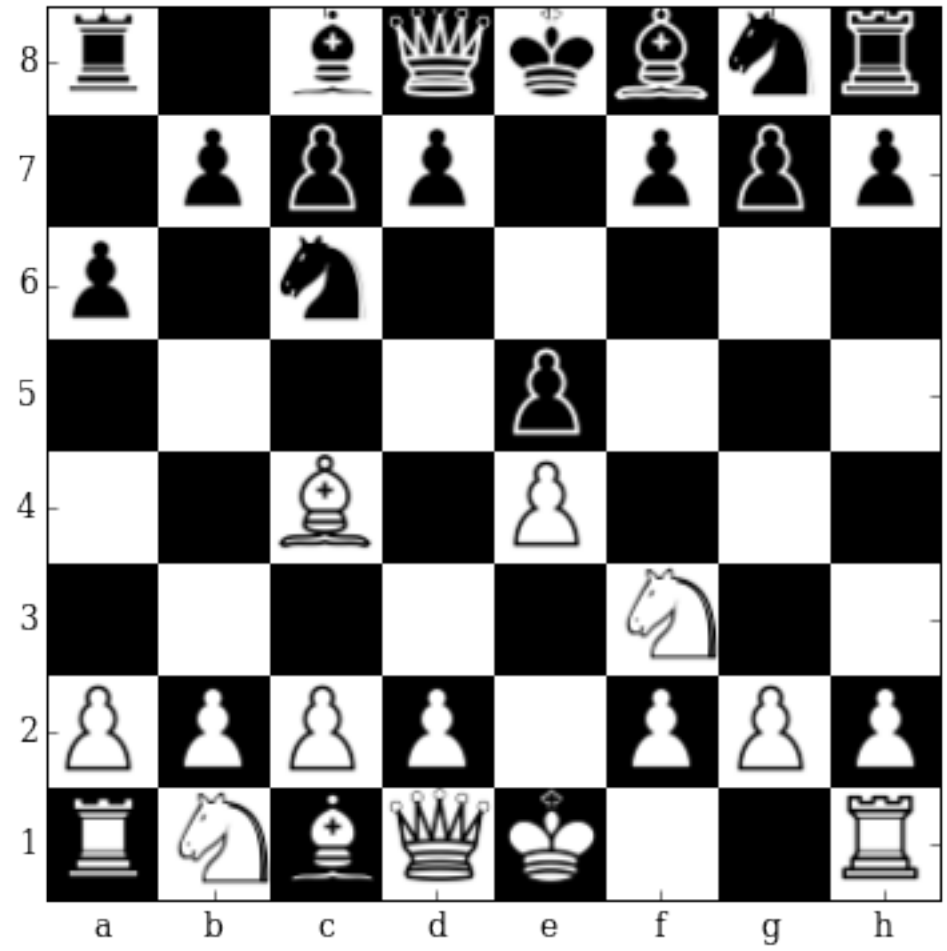



```
b5e2 0.00623769778758
```

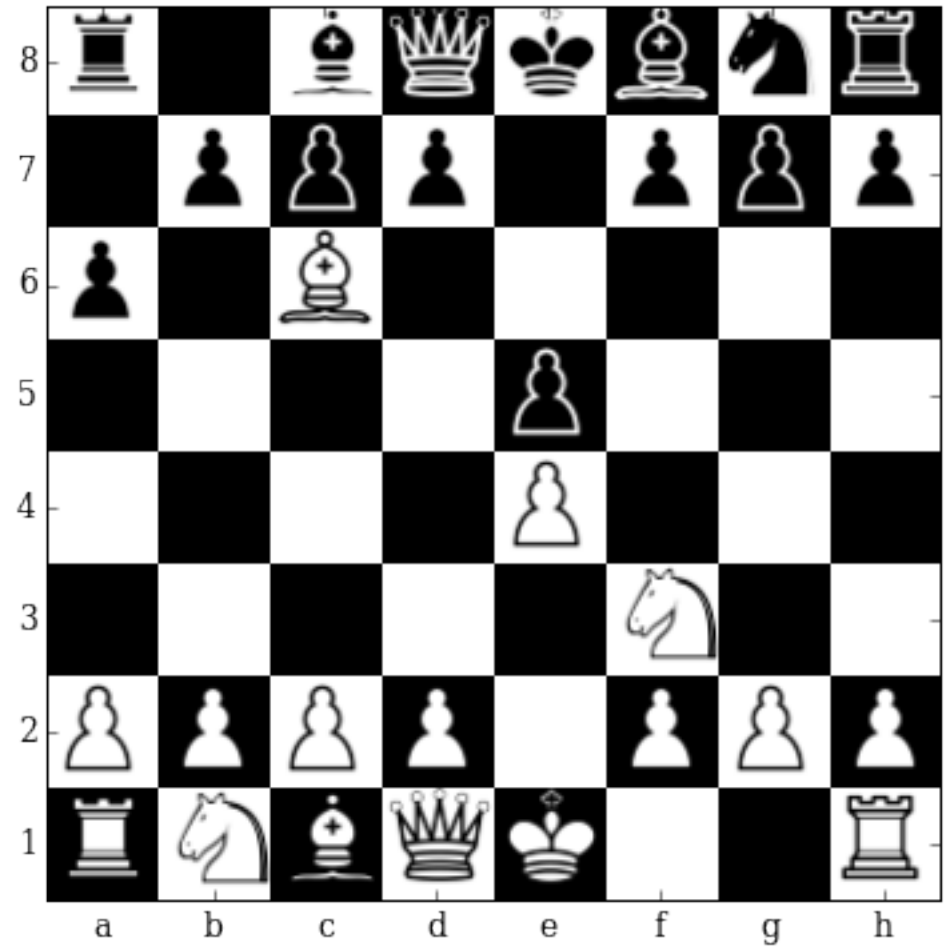



```
b5a6 0.00556590827182
```

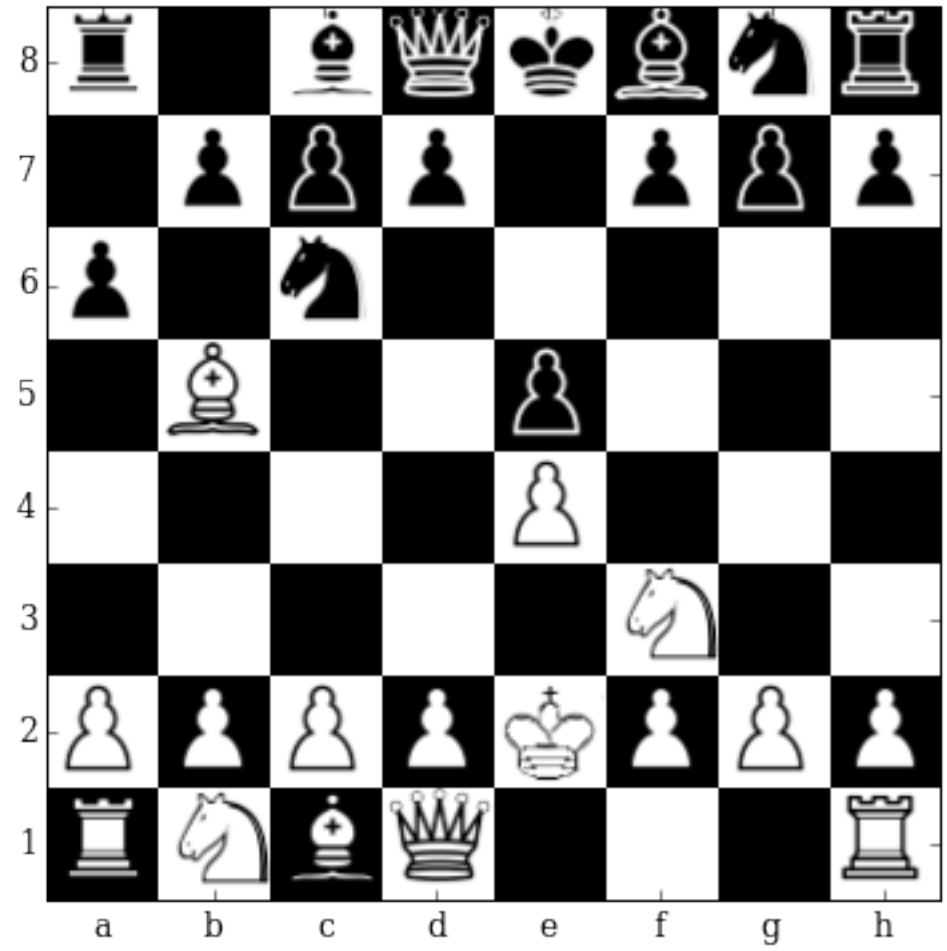

```
b5a4 0.00405460782349
```



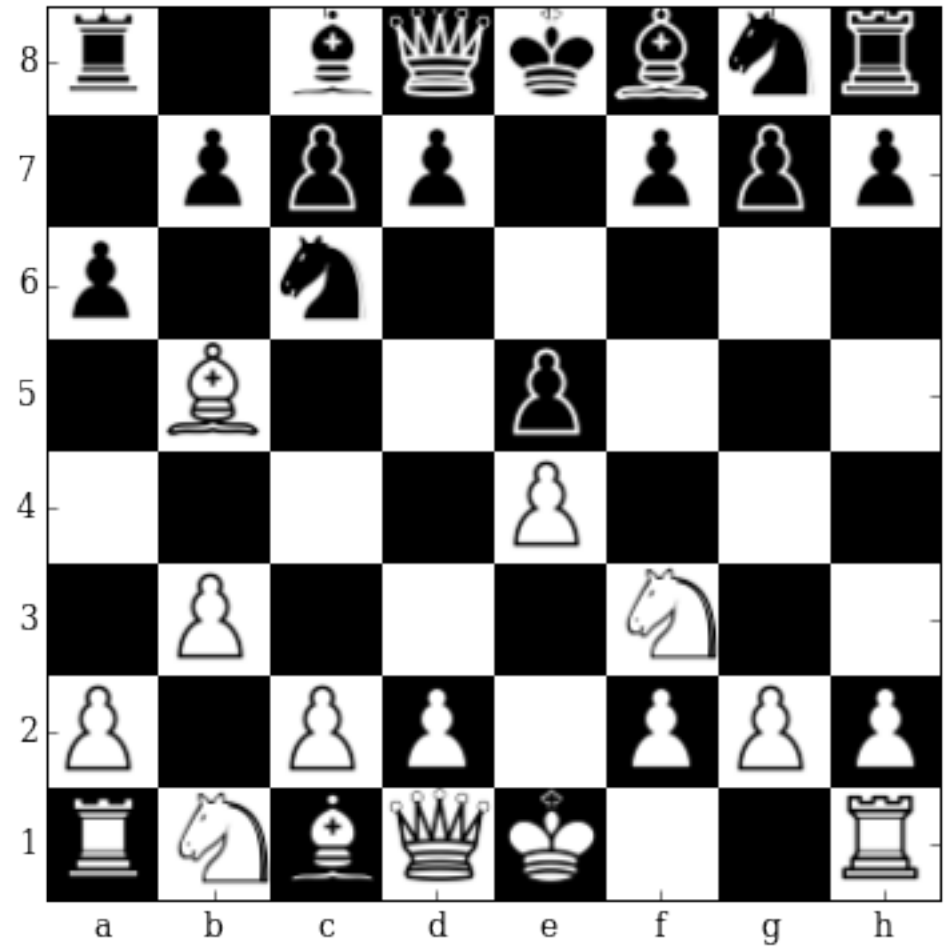

```
b5c4 0.00325289717875
```



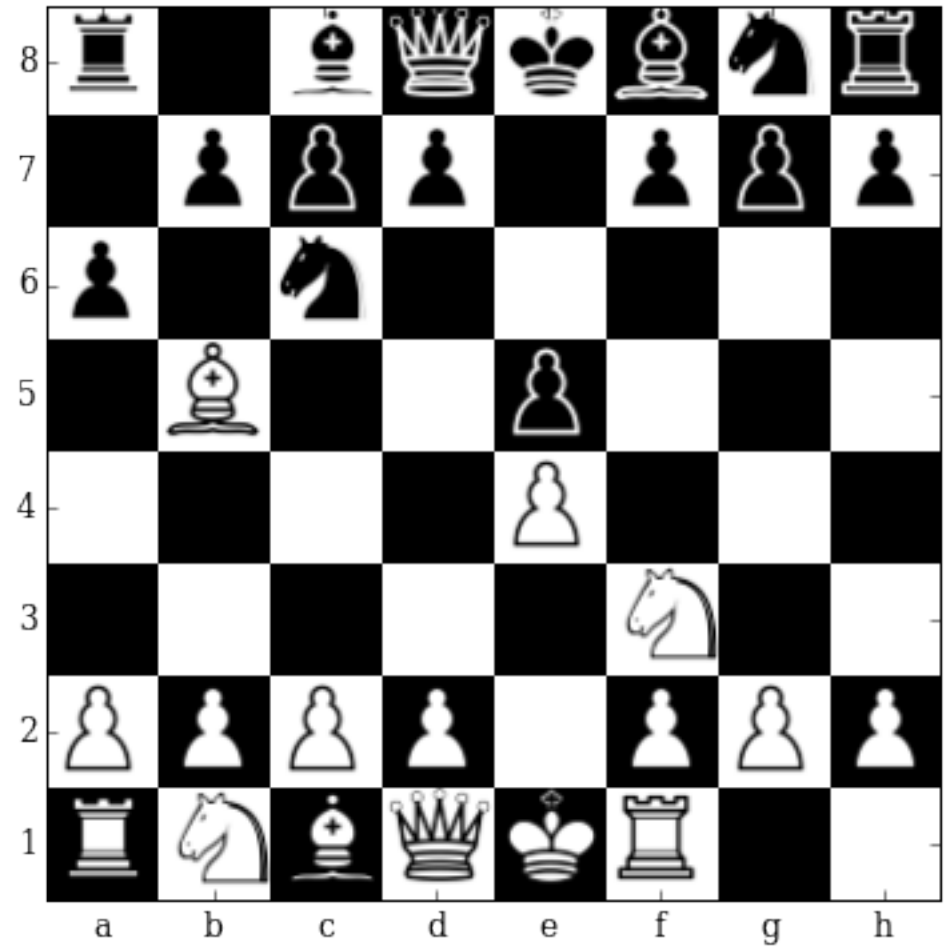

```
b5c6 0.00300058745779
WORST 5 Moves
```



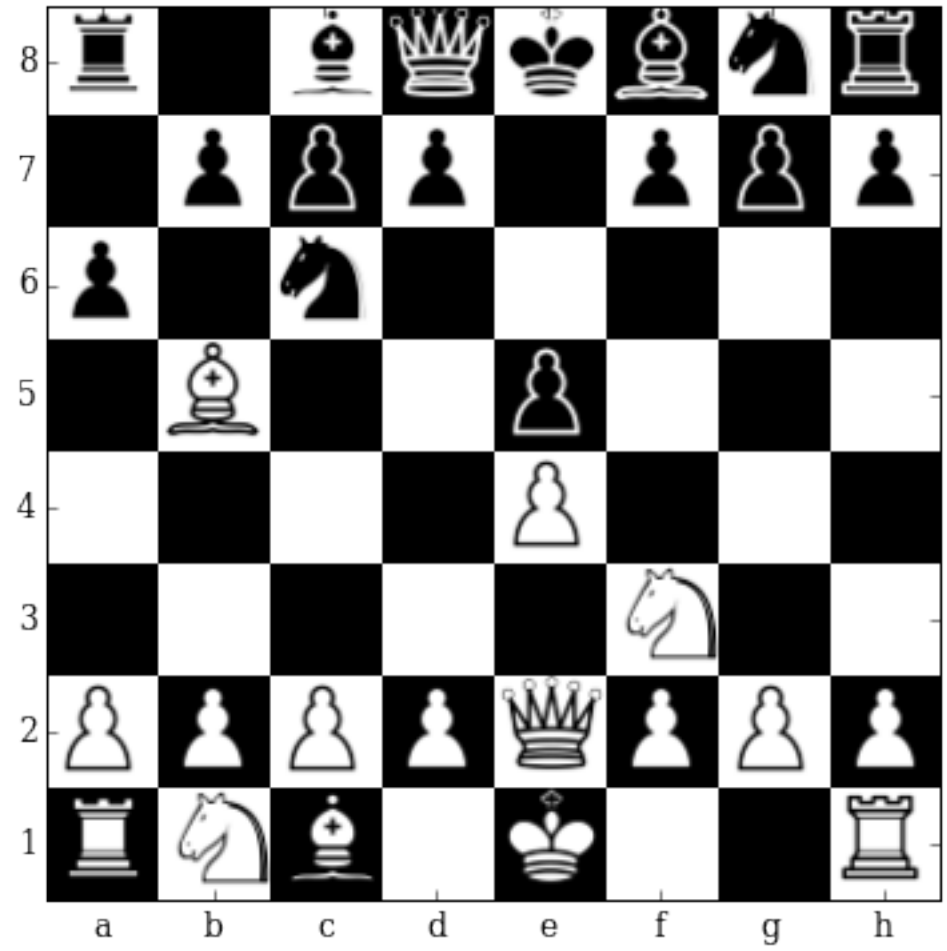

```
e1e2 -0.00794988125563
```



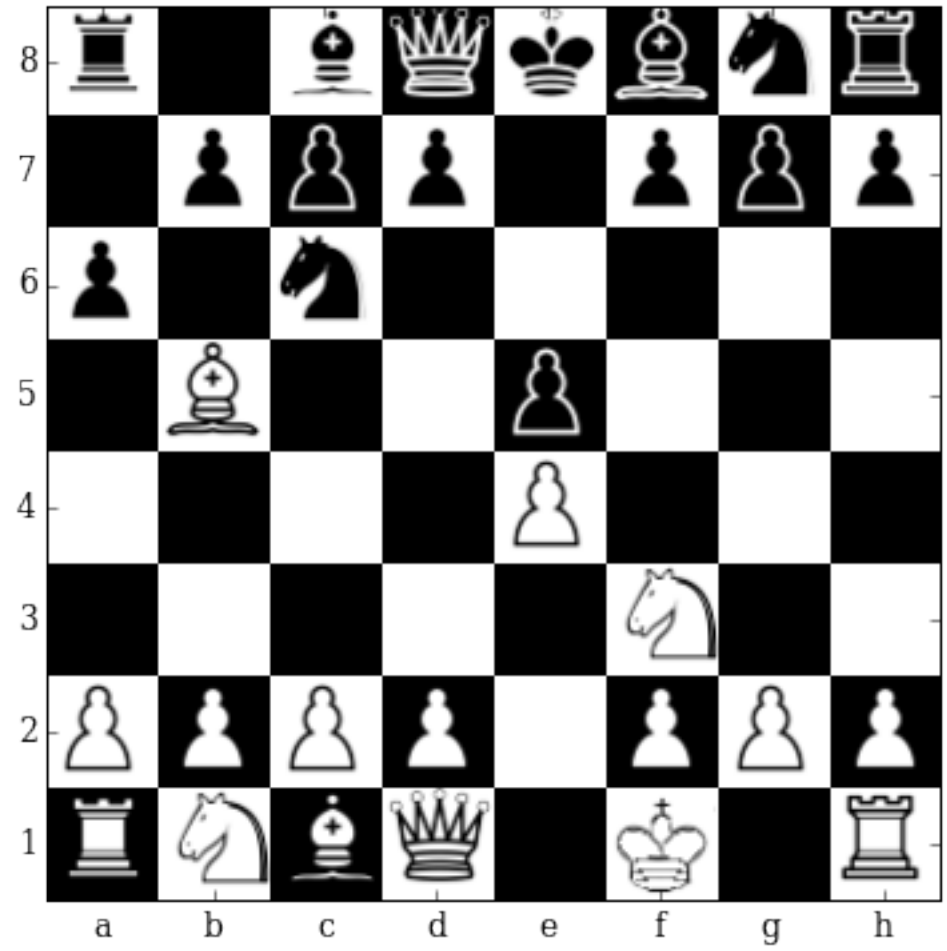

```
b2b3 -0.00797471962869
```




```
h1f1 -0.00896198488772
```

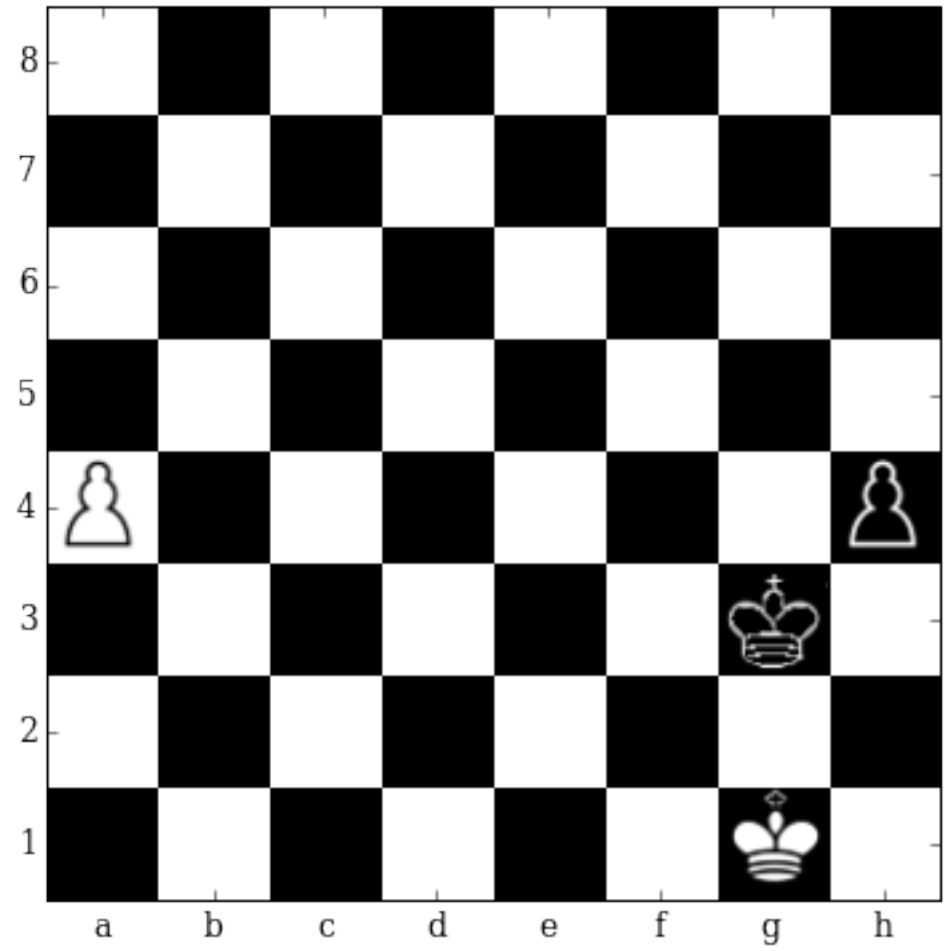



```
d1e2 -0.0102248834446
```

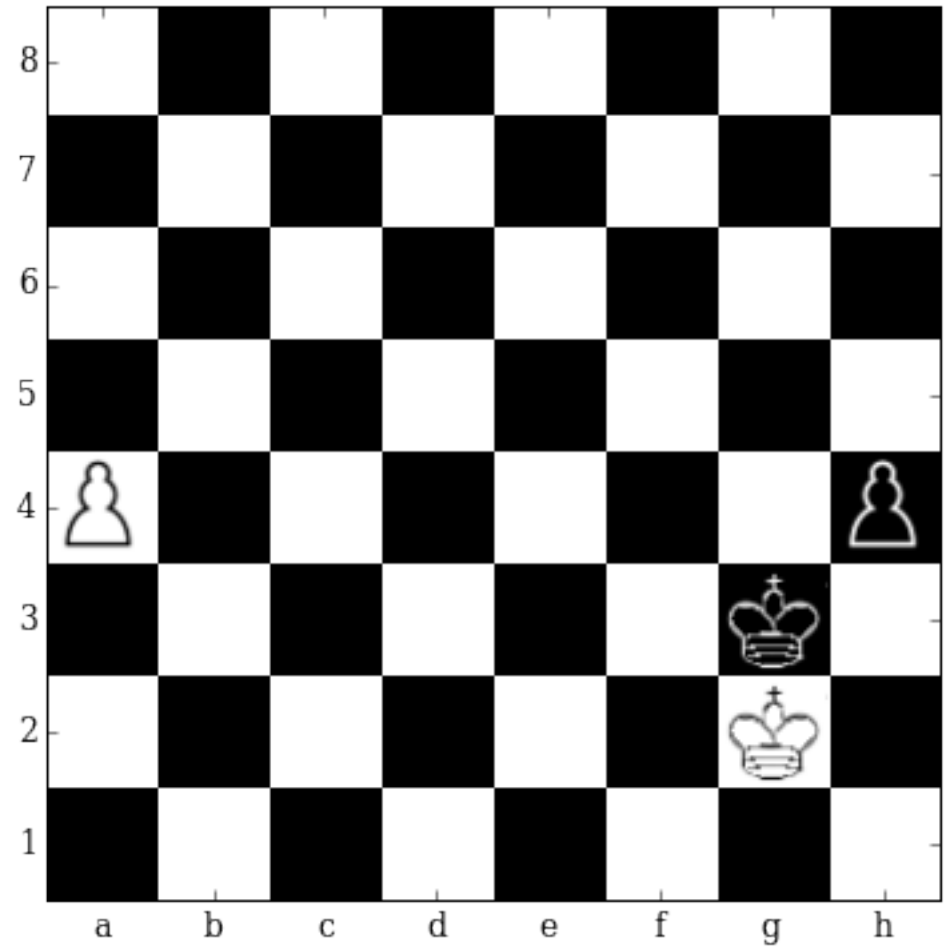



```
e1f1 -0.0129785845056
```

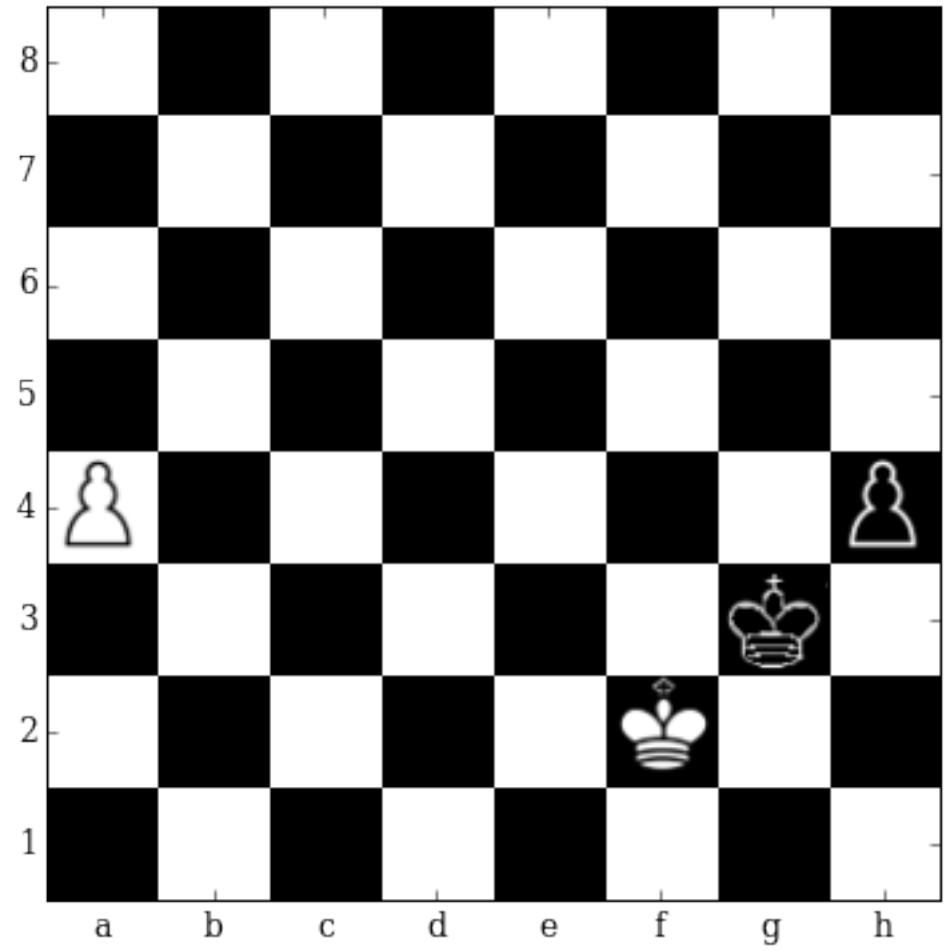
```
In [16]: '''
         Runaway Pawn
         '''
         top_bottom_moves('8/8/8/8/P6p/6k1/8/6K1 w - - 0 0')
```



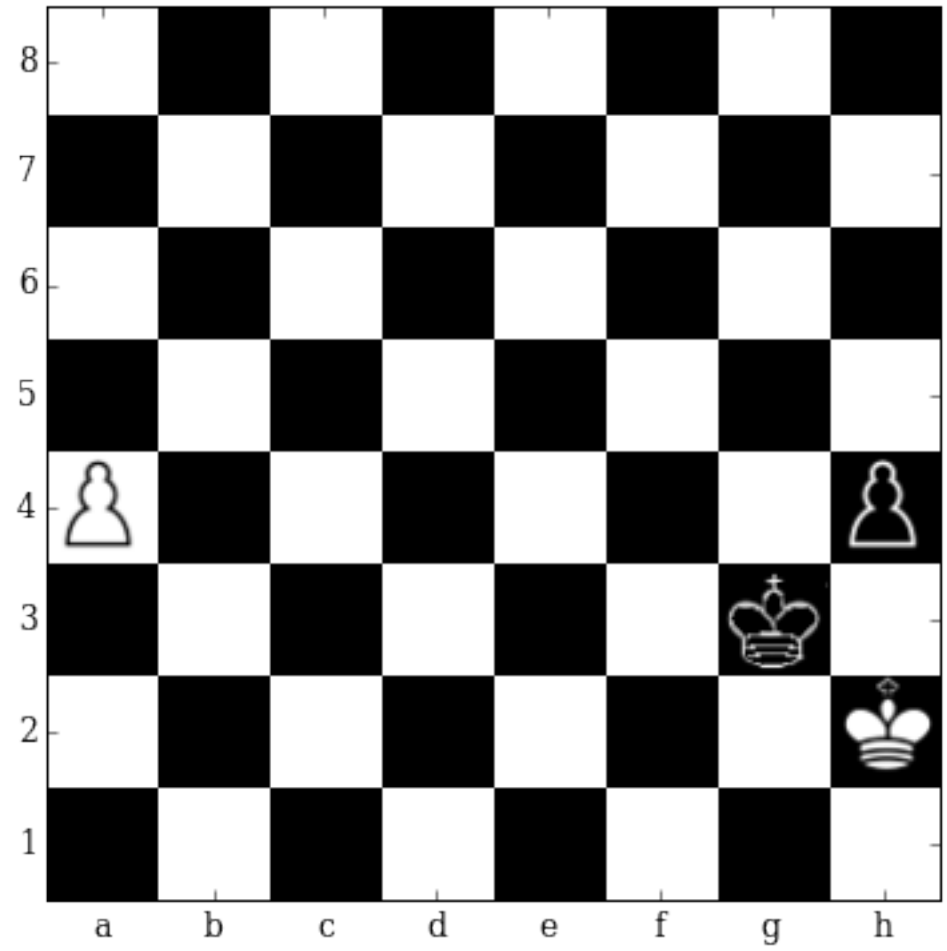

```
Current evaluation: -0.049161
Total 6 moves possible
TOP 5 Moves
```



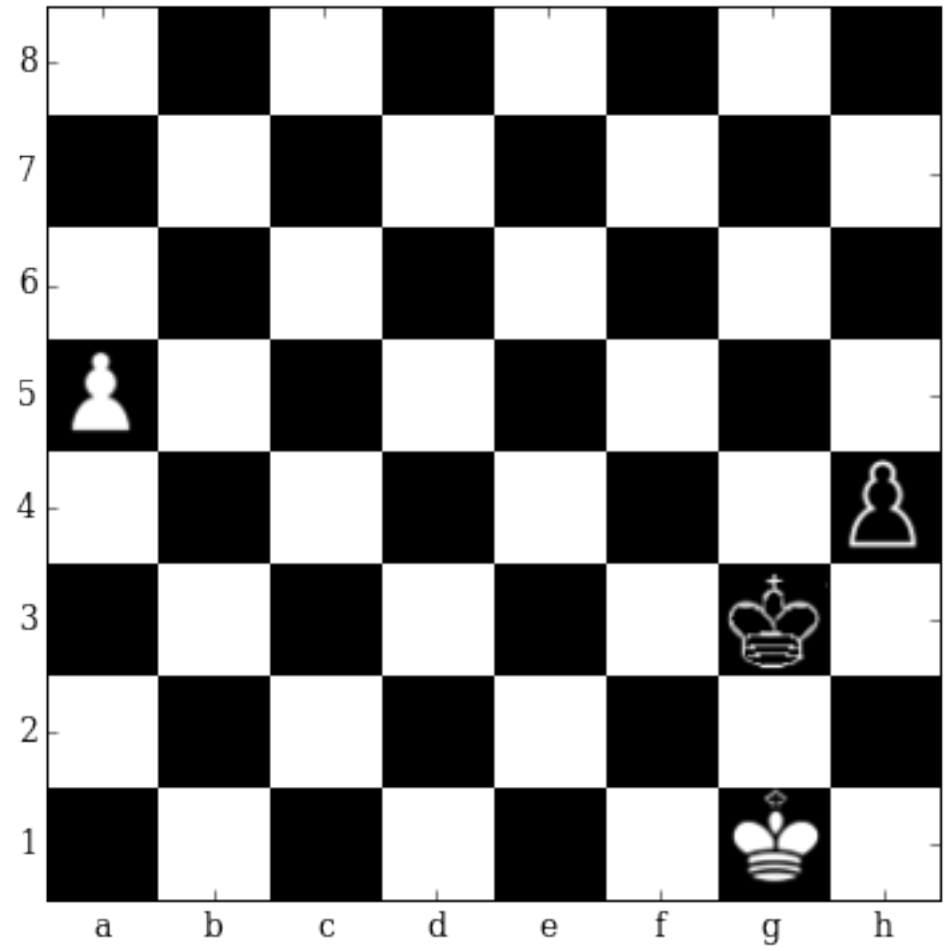

```
g1g2 0.00462216185406
```



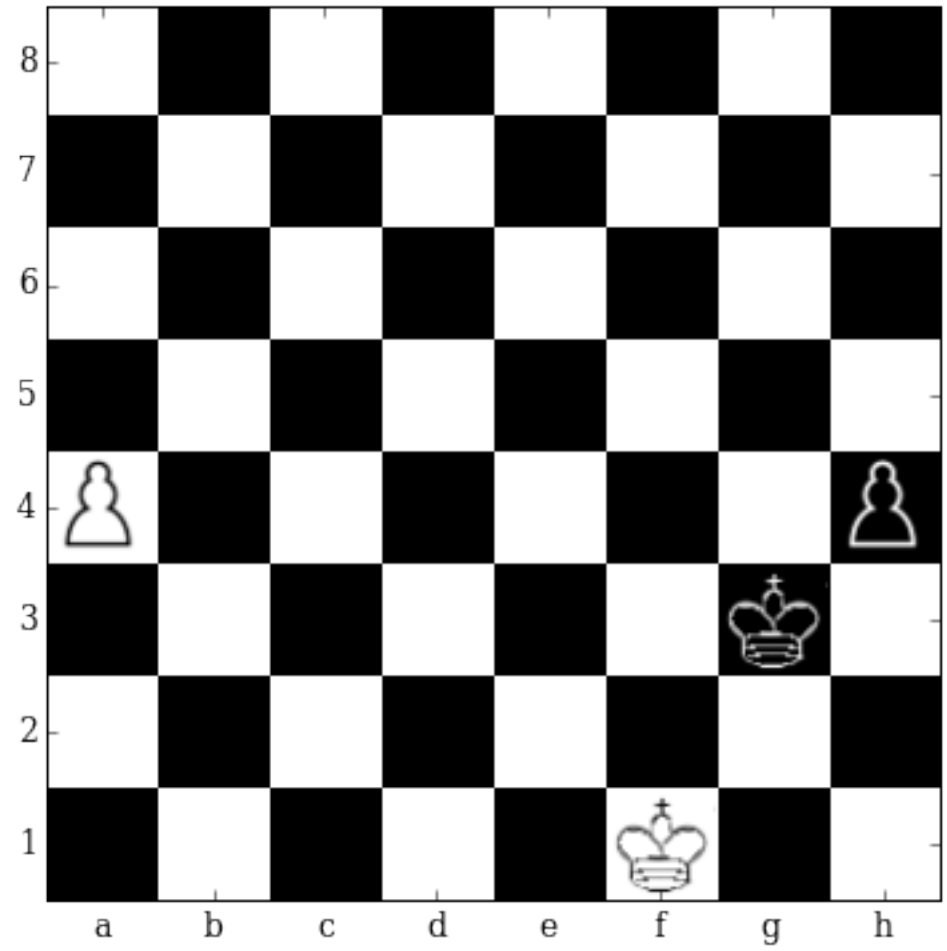

```
g1f2 -0.0209280643612
```



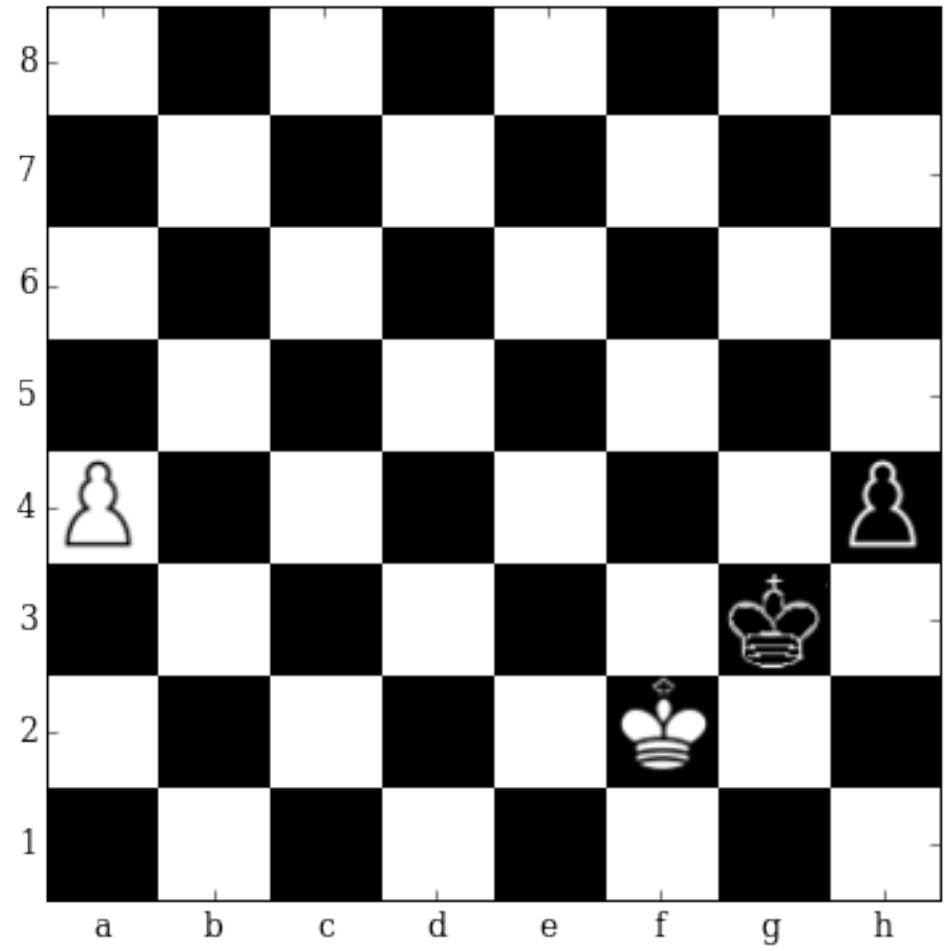

```
g1h2 -0.0317683890462
```



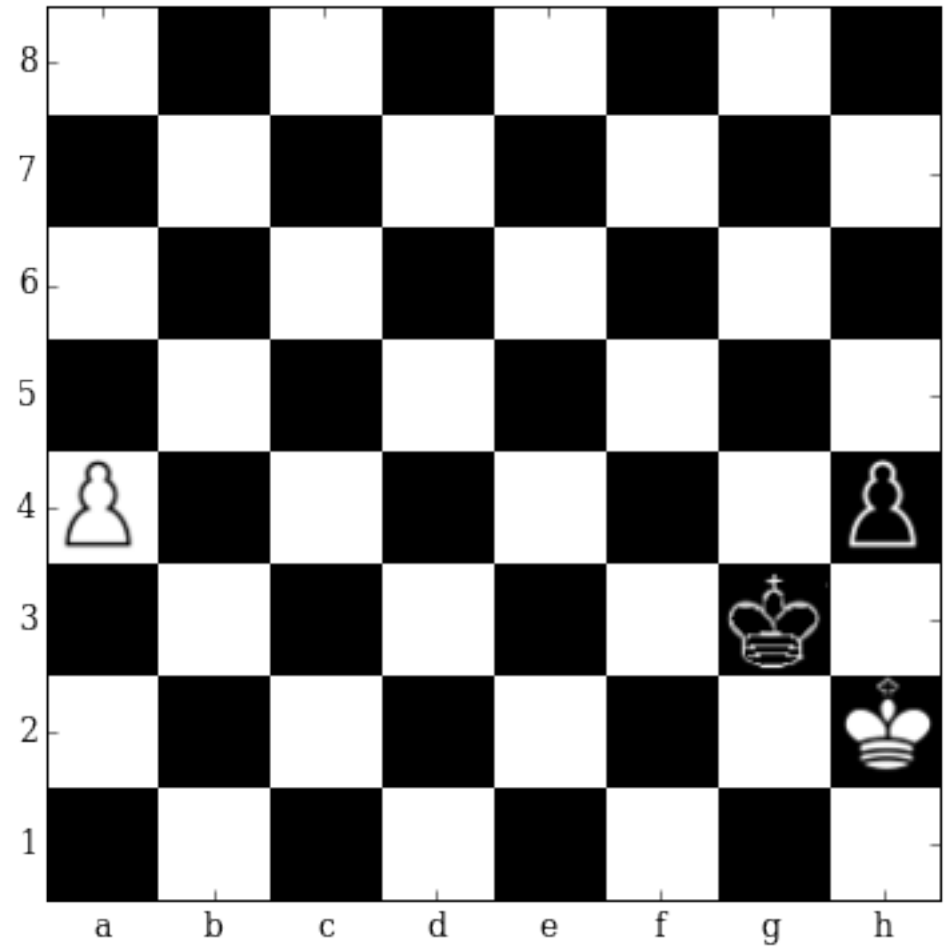

```
a4a5 -0.0482270270586
```



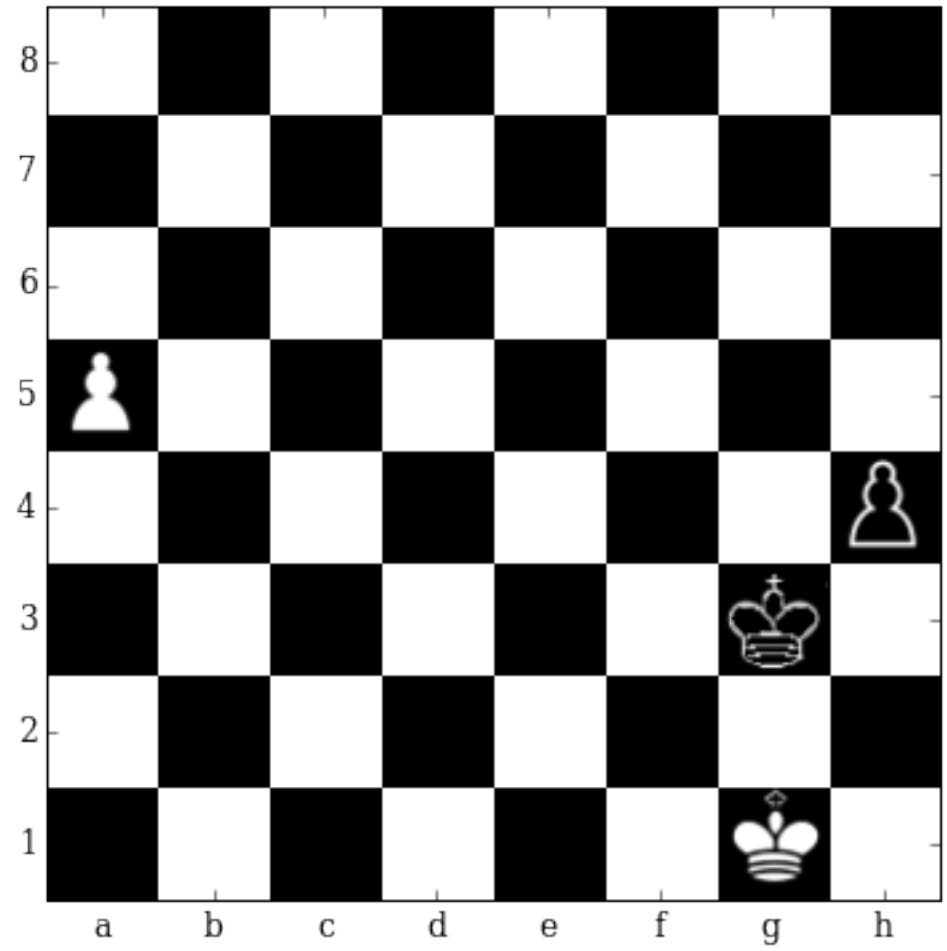

```
g1f1 -0.0665239691734
WORST 5 Moves
```



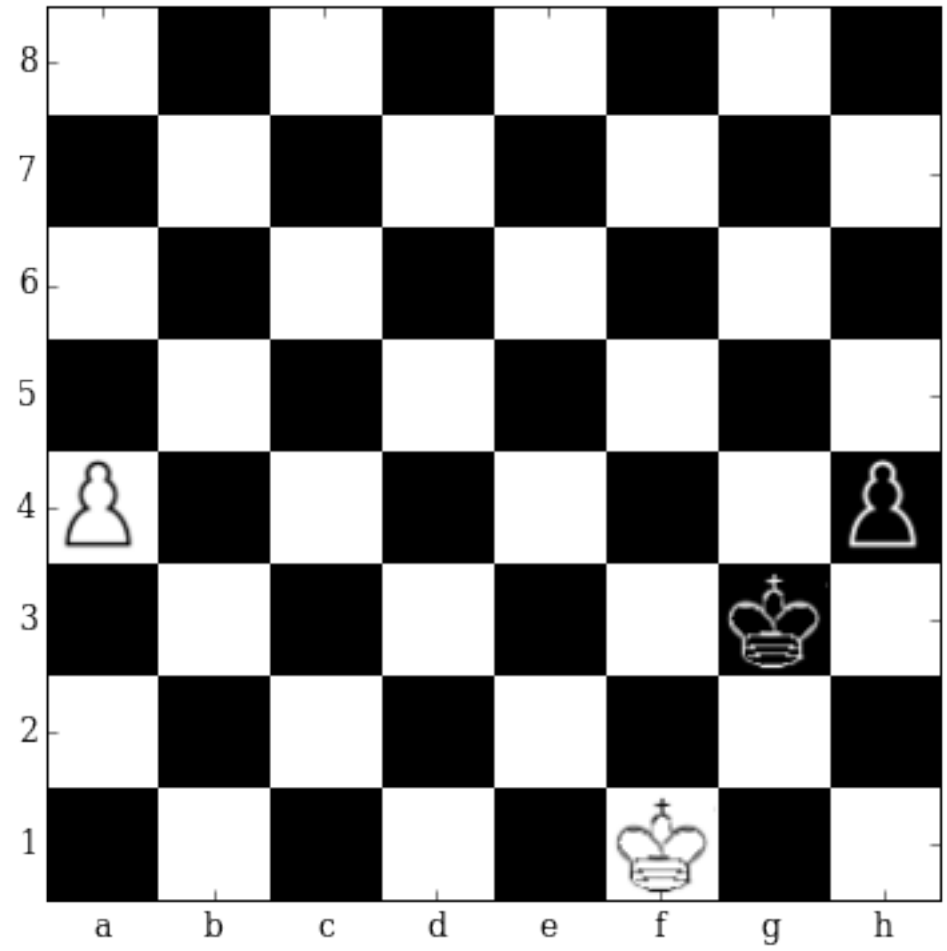

```
g1f2 -0.0209280643612
```



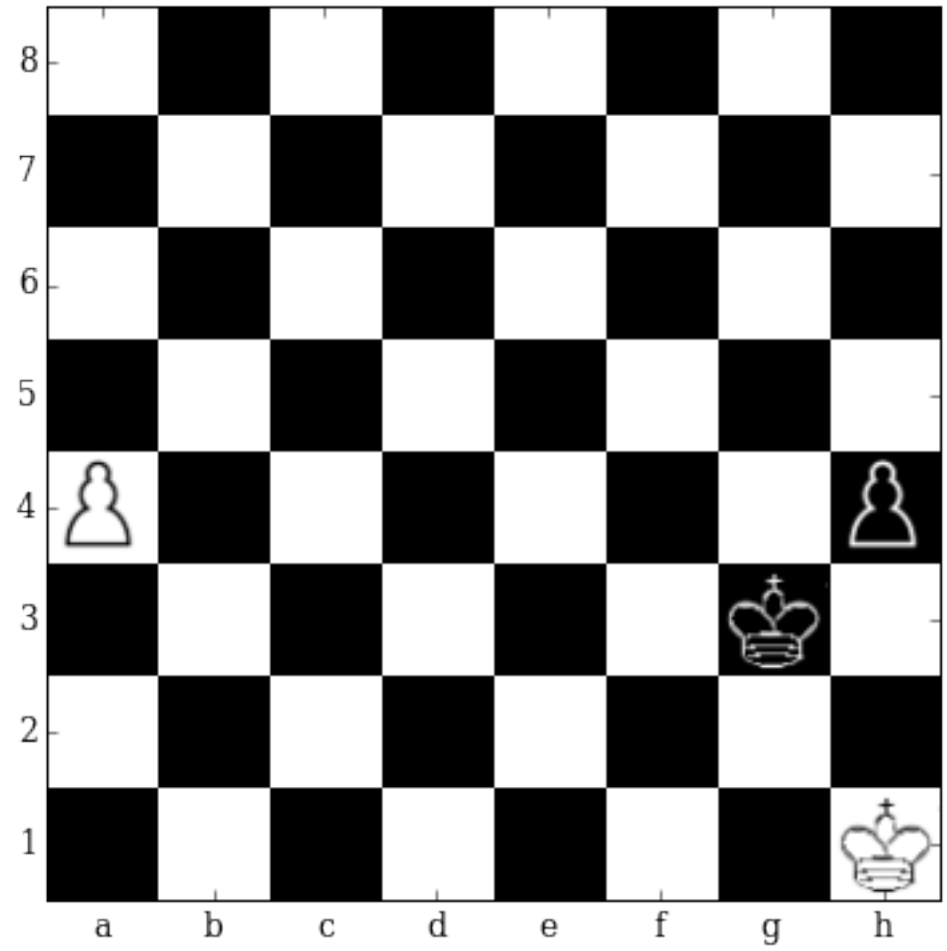

```
g1h2 -0.0317683890462
```




```
a4a5 -0.0482270270586
```

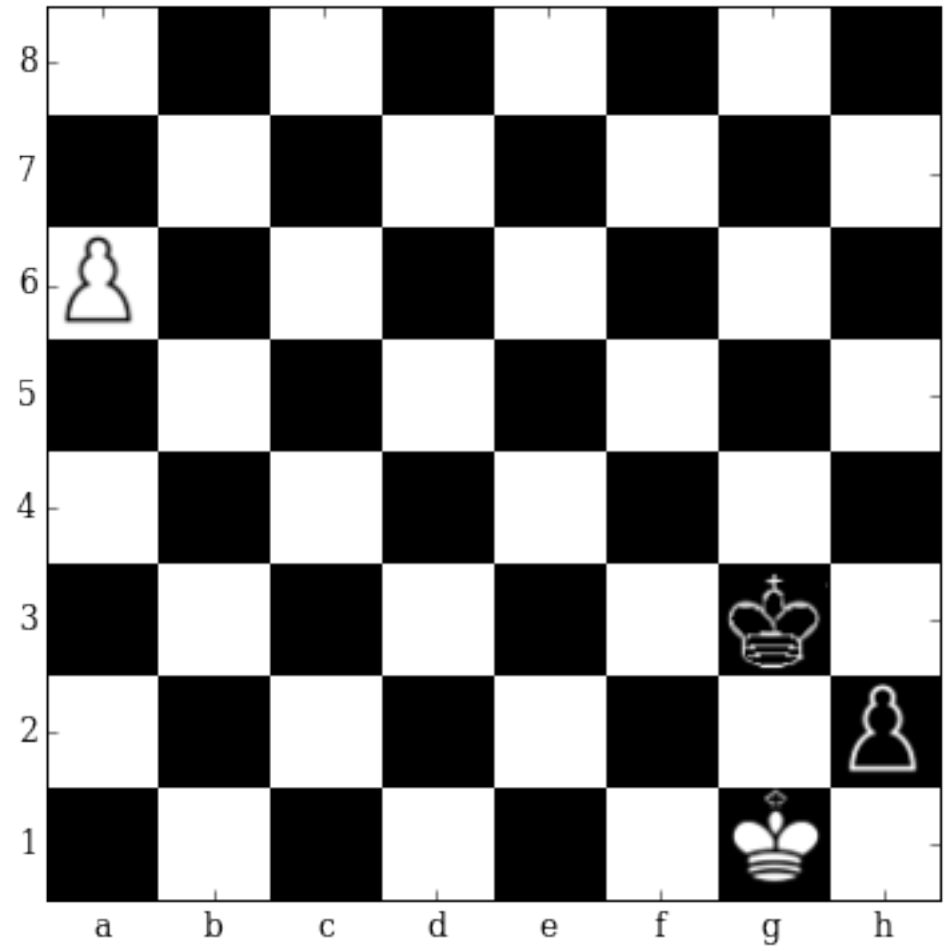



```
g1f1 -0.0665239691734
```

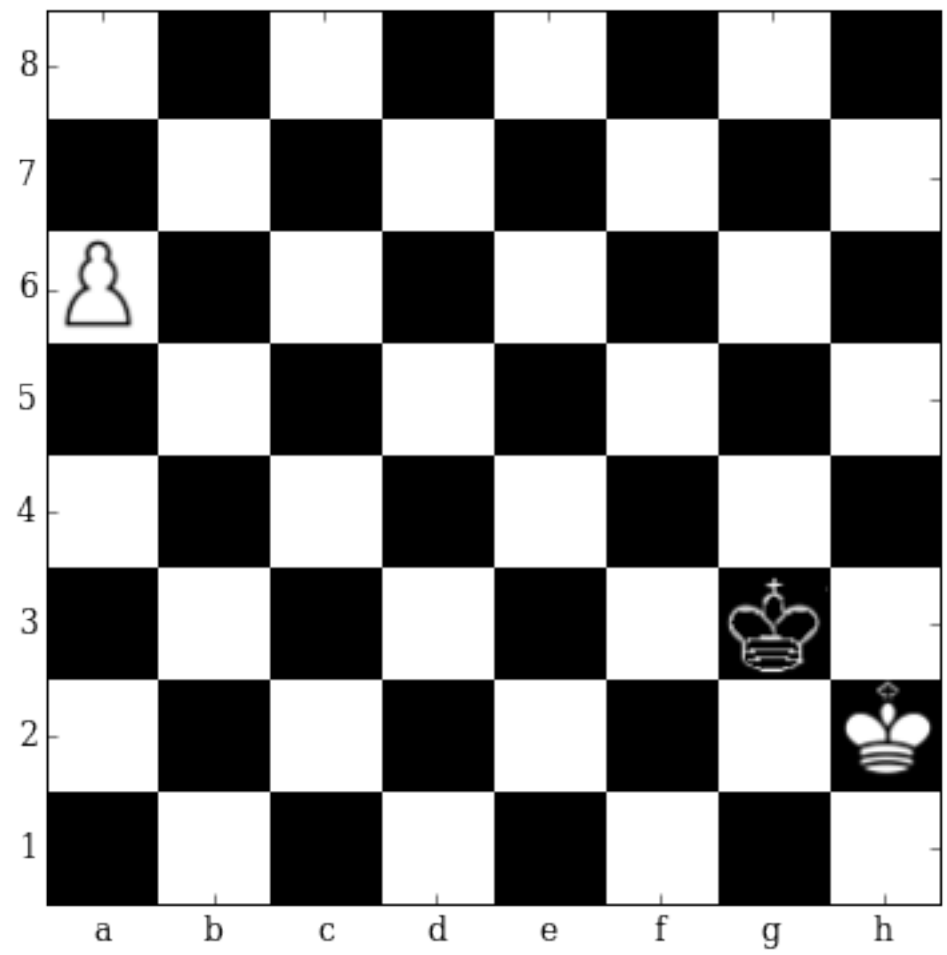



```
g1h1 -0.0737288221717
```

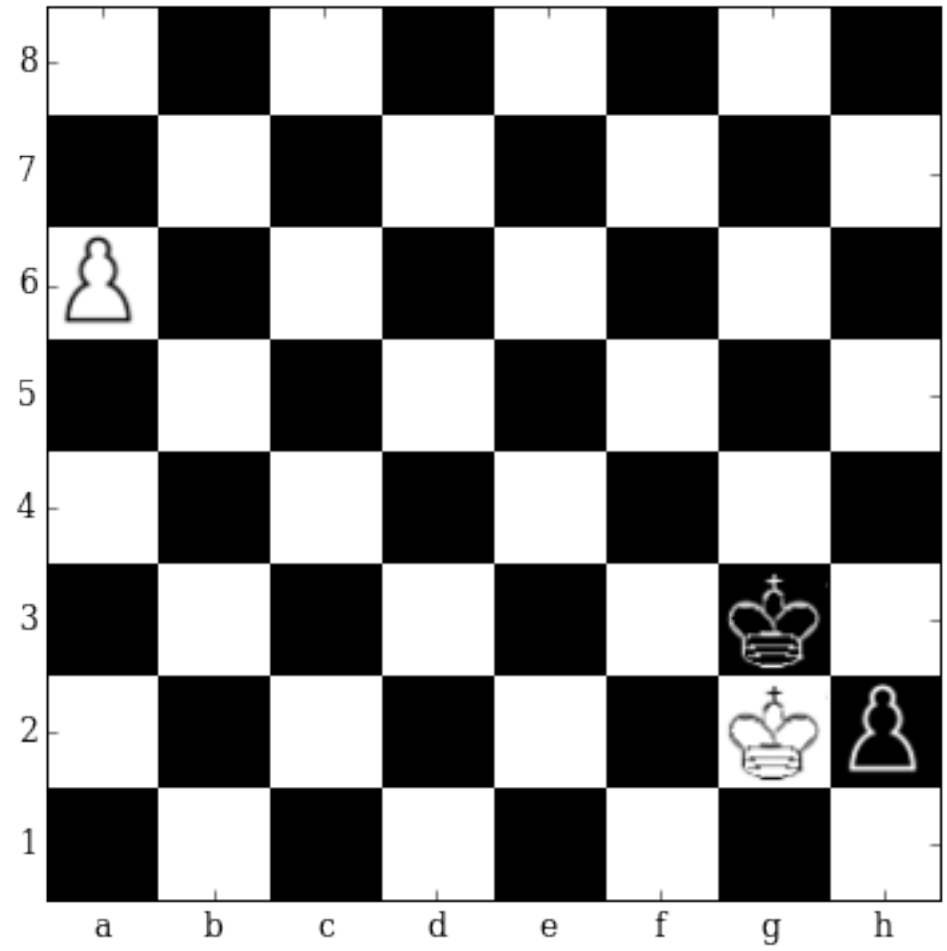
```
In [17]: '''
         Check and Blocking a promotion
         '''
         fen = '8/8/P7/8/8/6k1/7p/6K1 w - - 0 0'
         top_bottom_moves(fen)
```



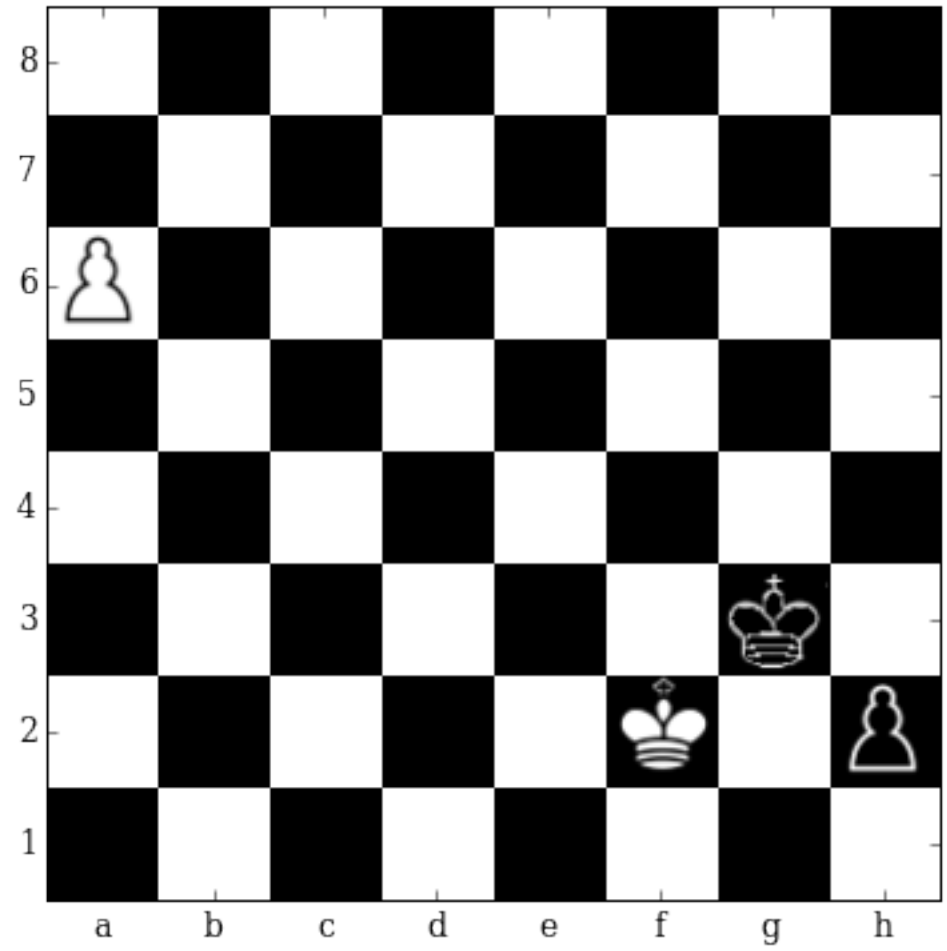

```
Current evaluation: -0.049778
Total 6 moves possible
TOP 5 Moves
```



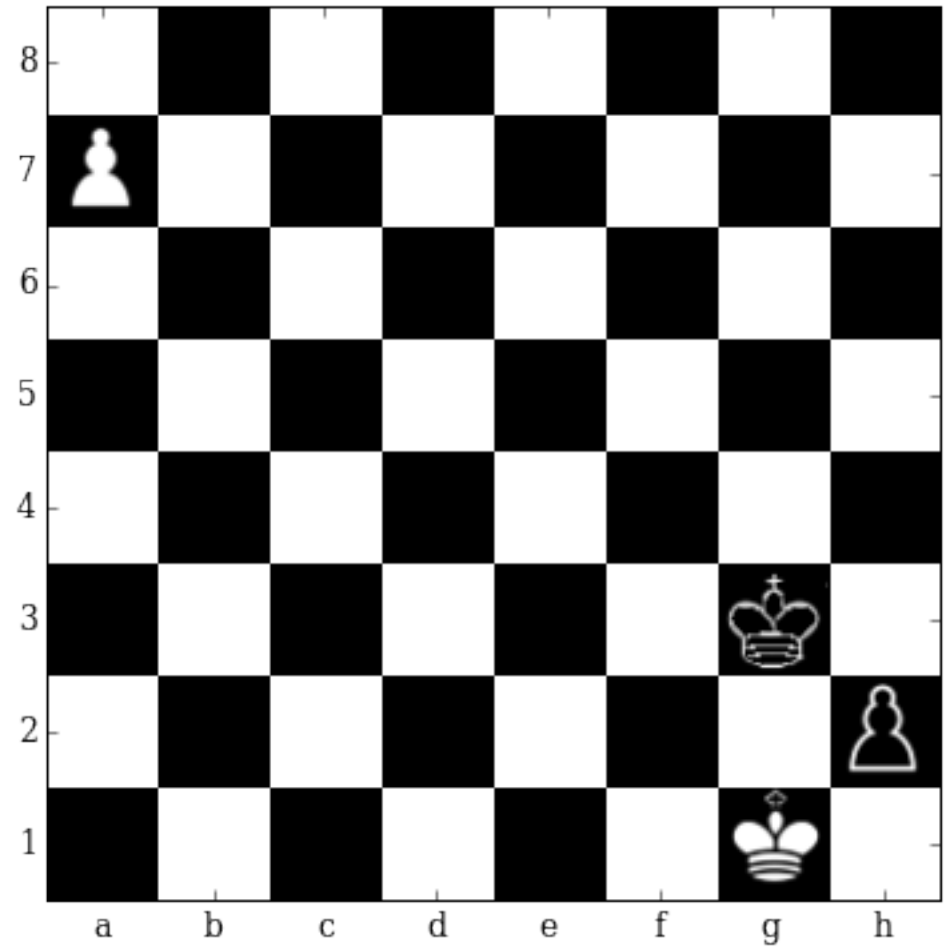

```
g1h2 0.0215898770839
```



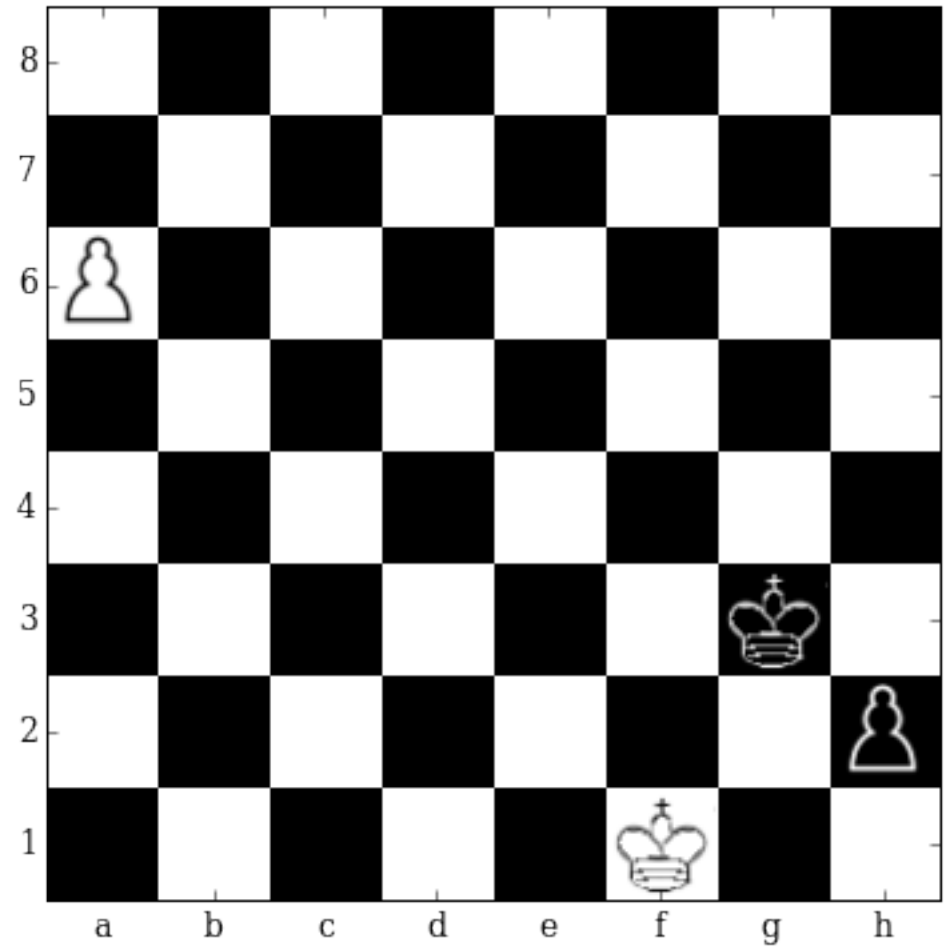

```
g1g2 0.00836602784693
```



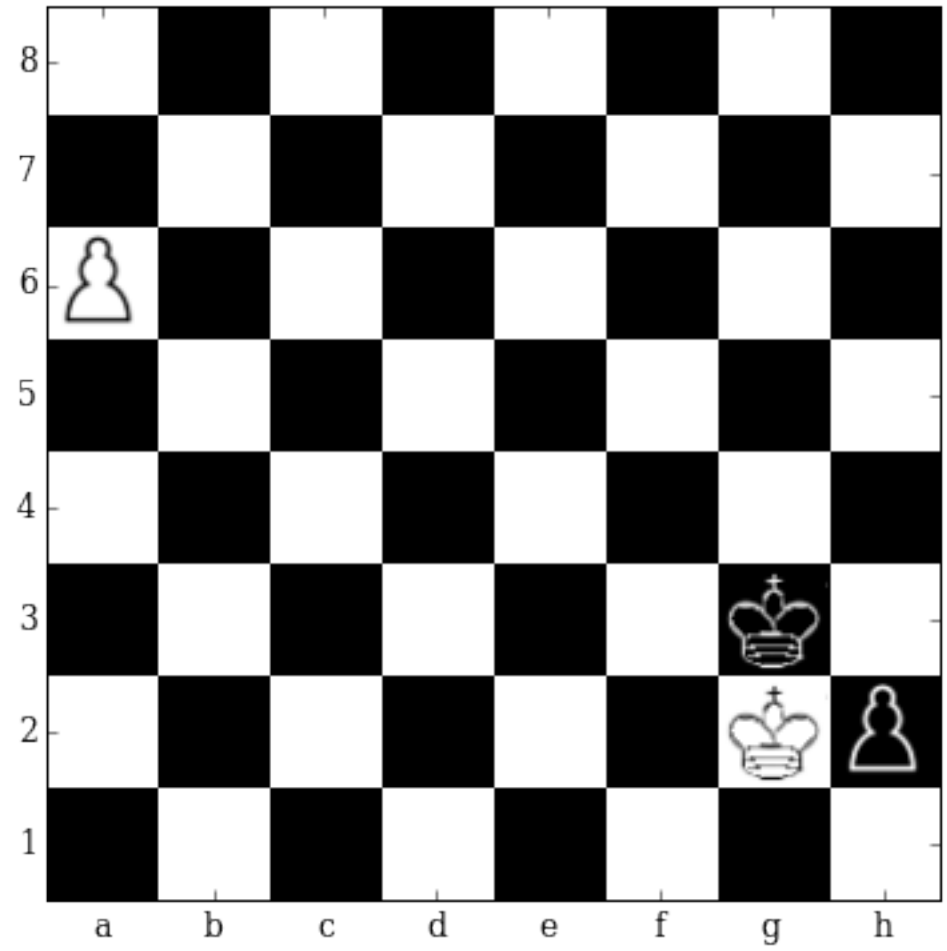

```
g1f2 -0.0188560783863
```



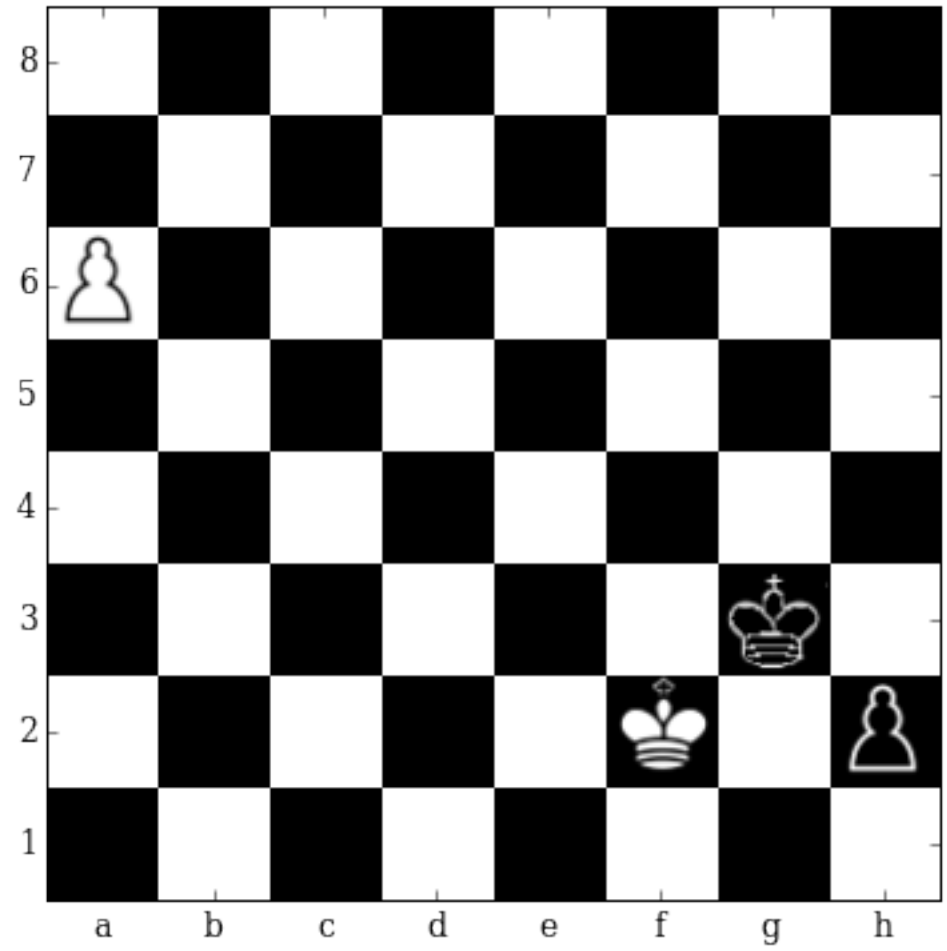

```
a6a7 -0.054150197655
```



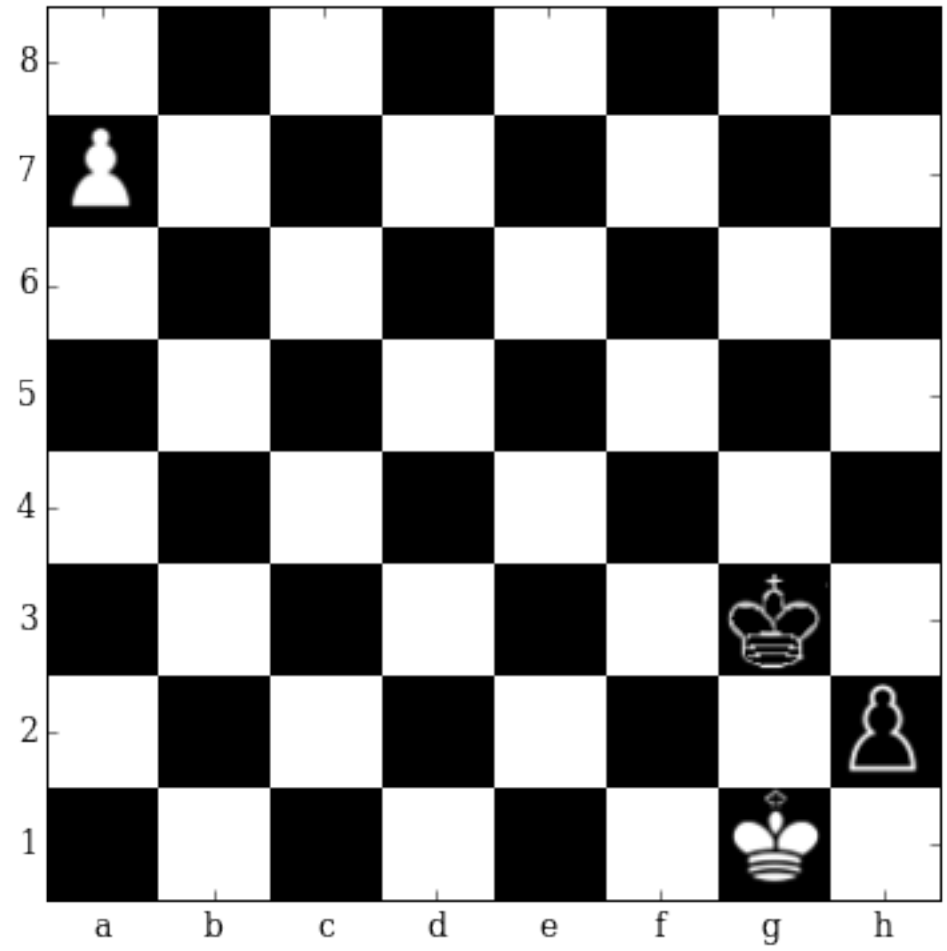

```
g1f1 -0.065742097795
WORST 5 Moves
```



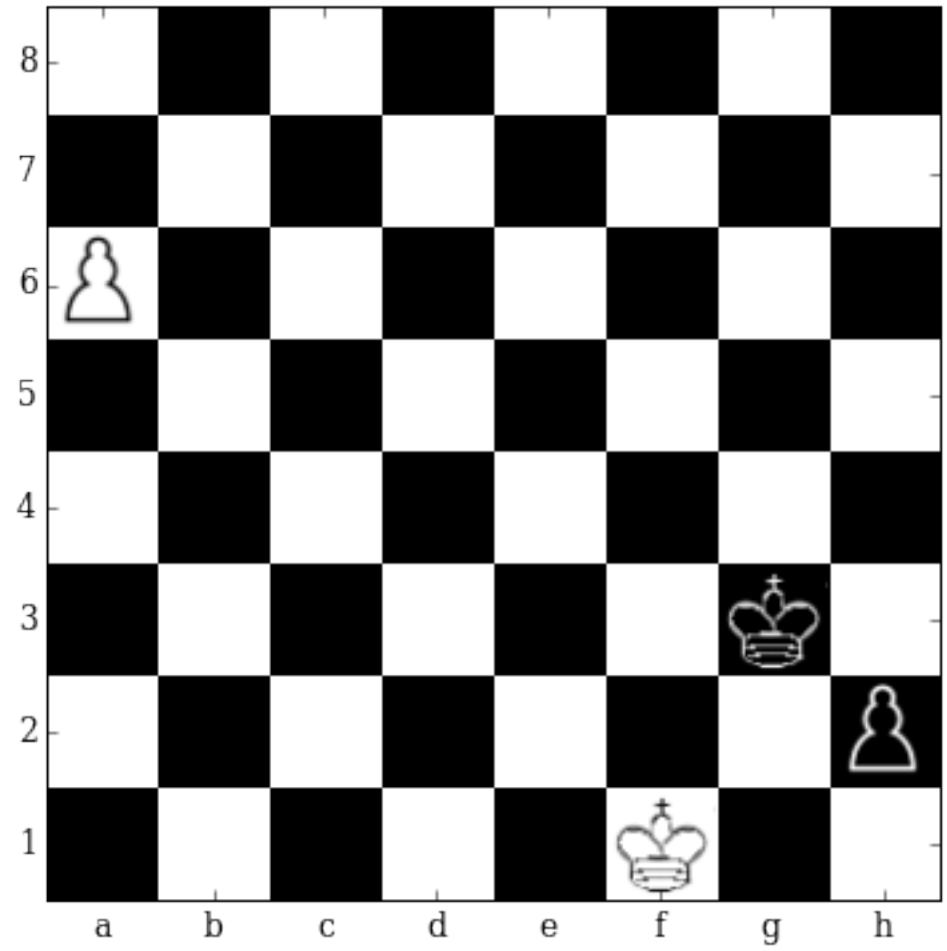

```
g1g2 0.00836602784693
```



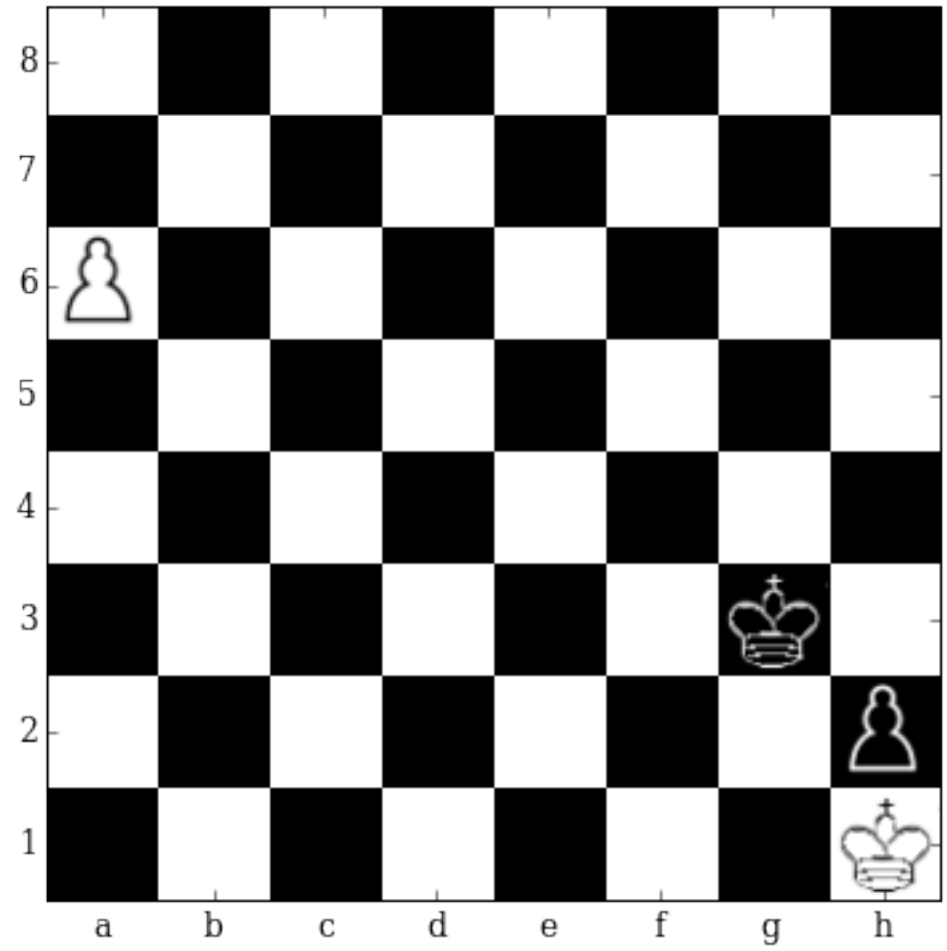

```
g1f2 -0.0188560783863
```




```
a6a7 -0.054150197655
```

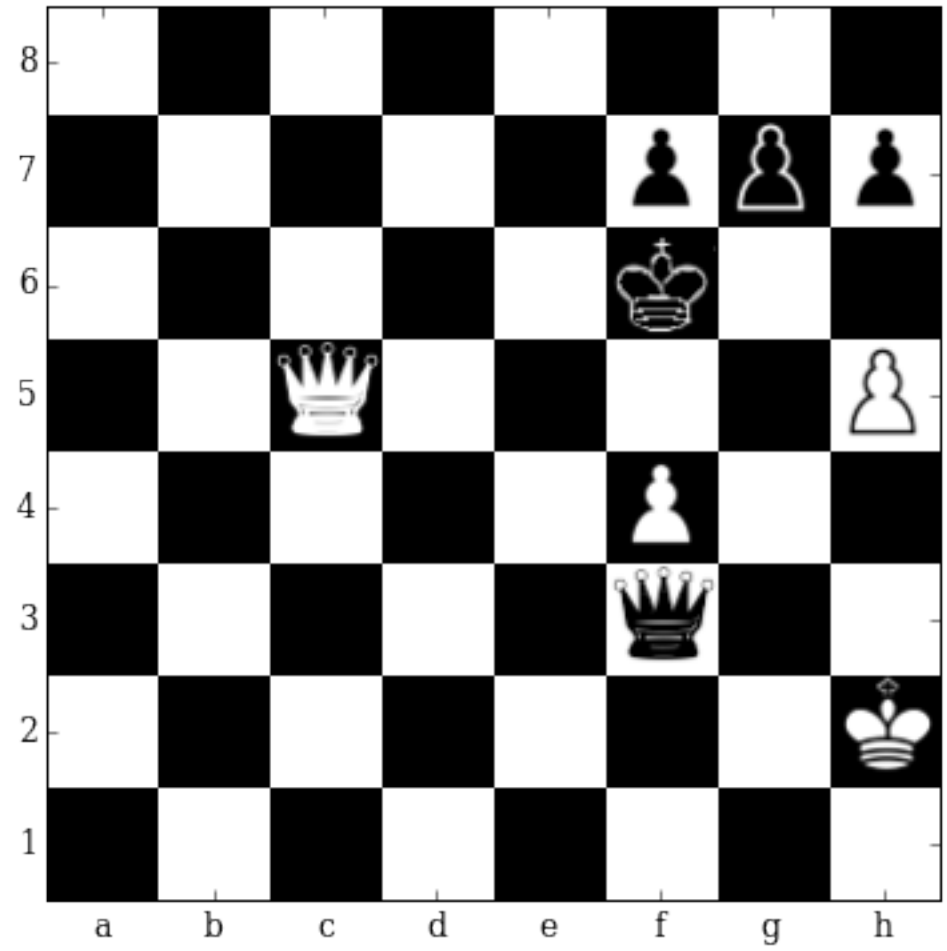



```
g1f1 -0.065742097795
```

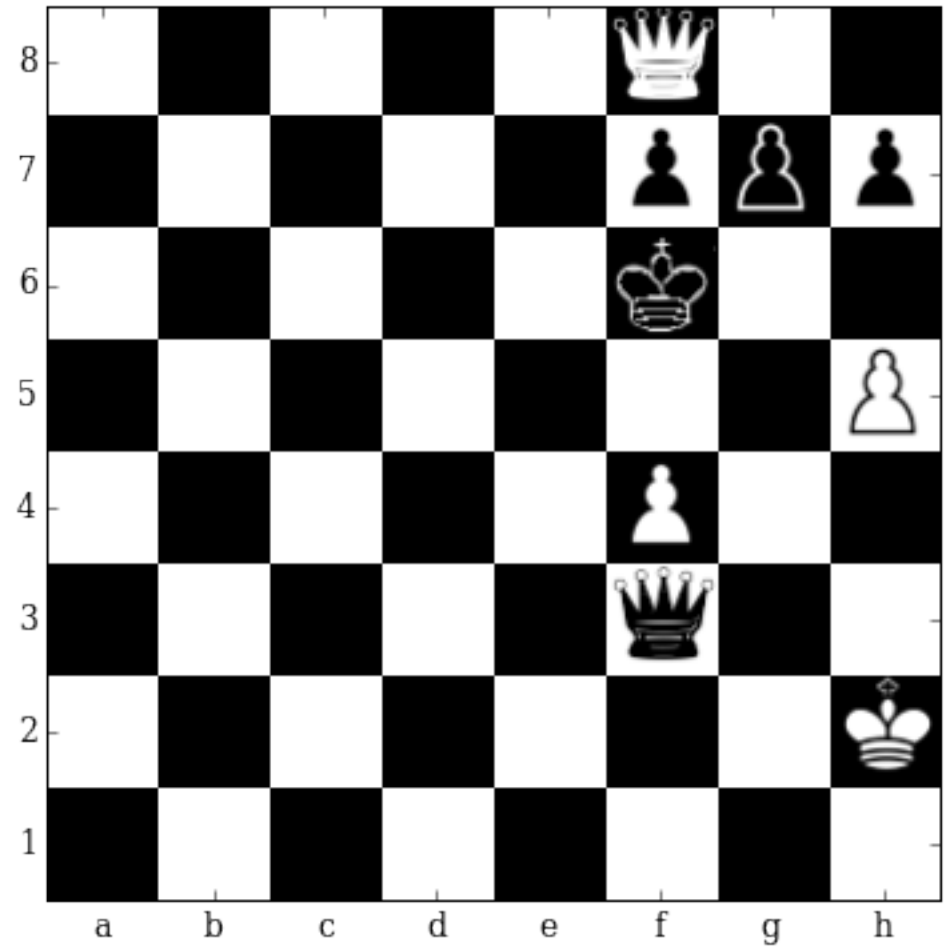



```
g1h1 -0.0684899687767
```

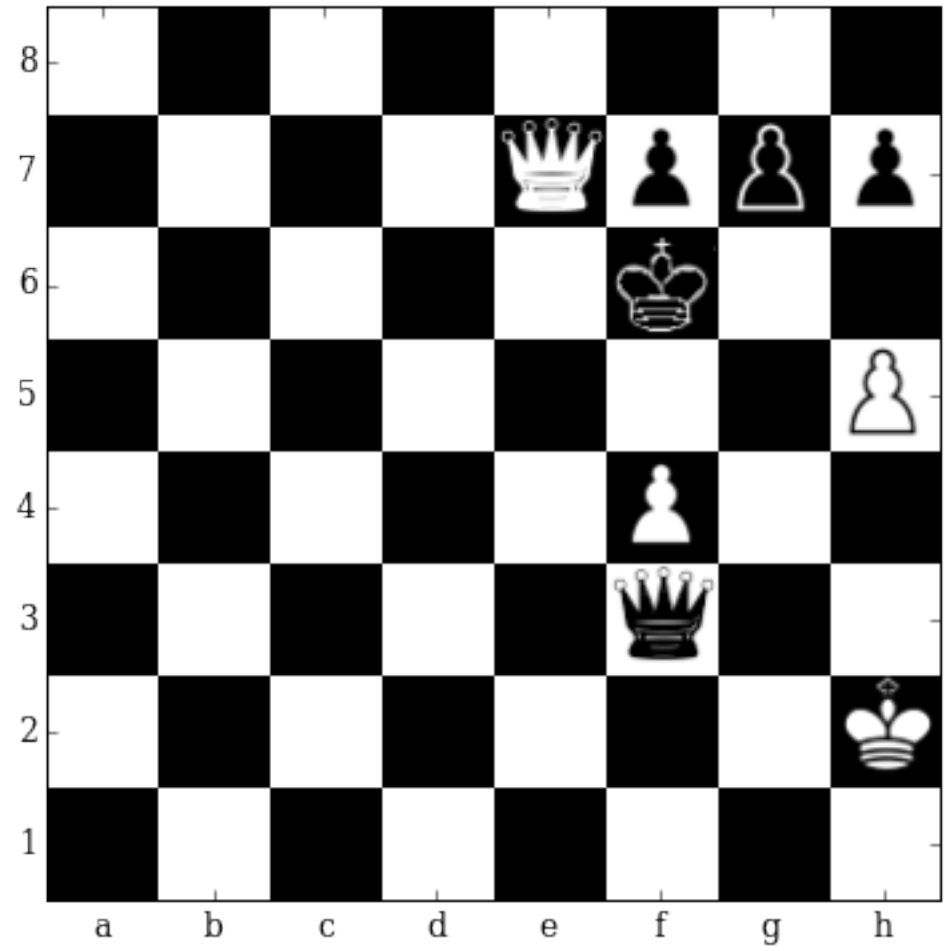
```
In [18]: '''
         Mate in 1
         '''
         fen='8/5ppp/5k2/2Q4P/5P2/5q2/7K/8 w - - 0 0'
         top_bottom_moves(fen)
```



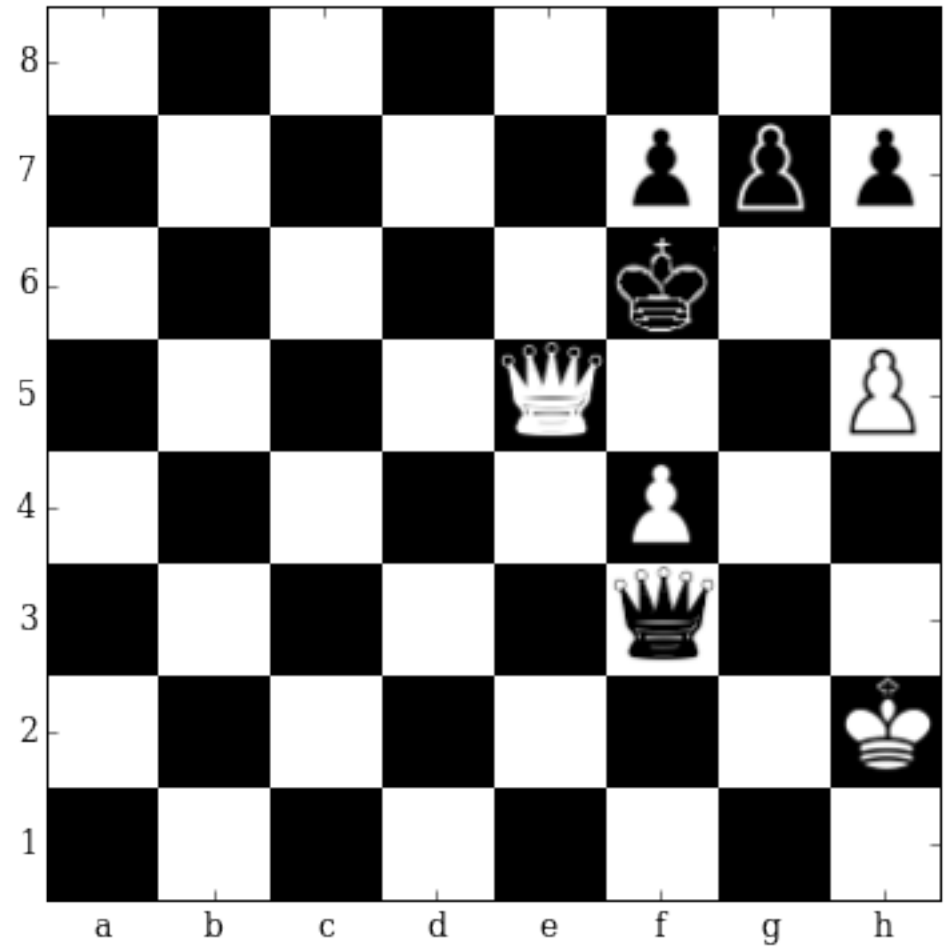

```
Current evaluation: -0.067688
Total 31 moves possible
TOP 5 Moves
```



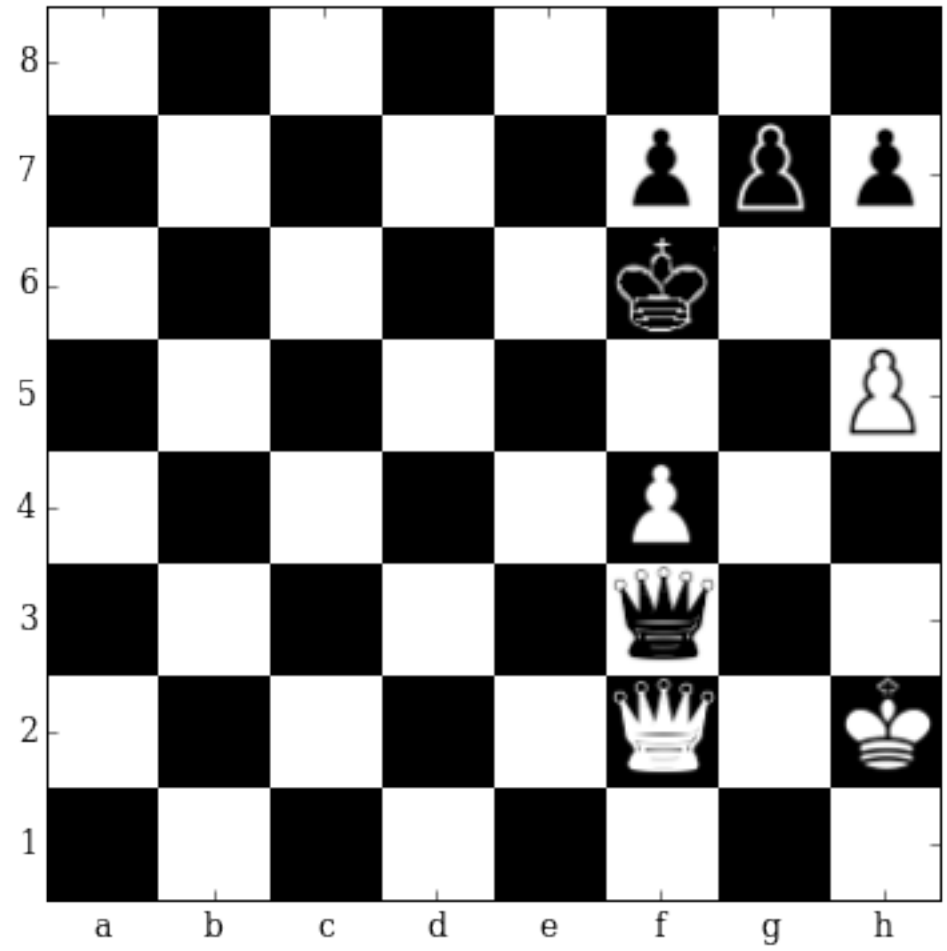

```
c5f8 0.0379827916622
```



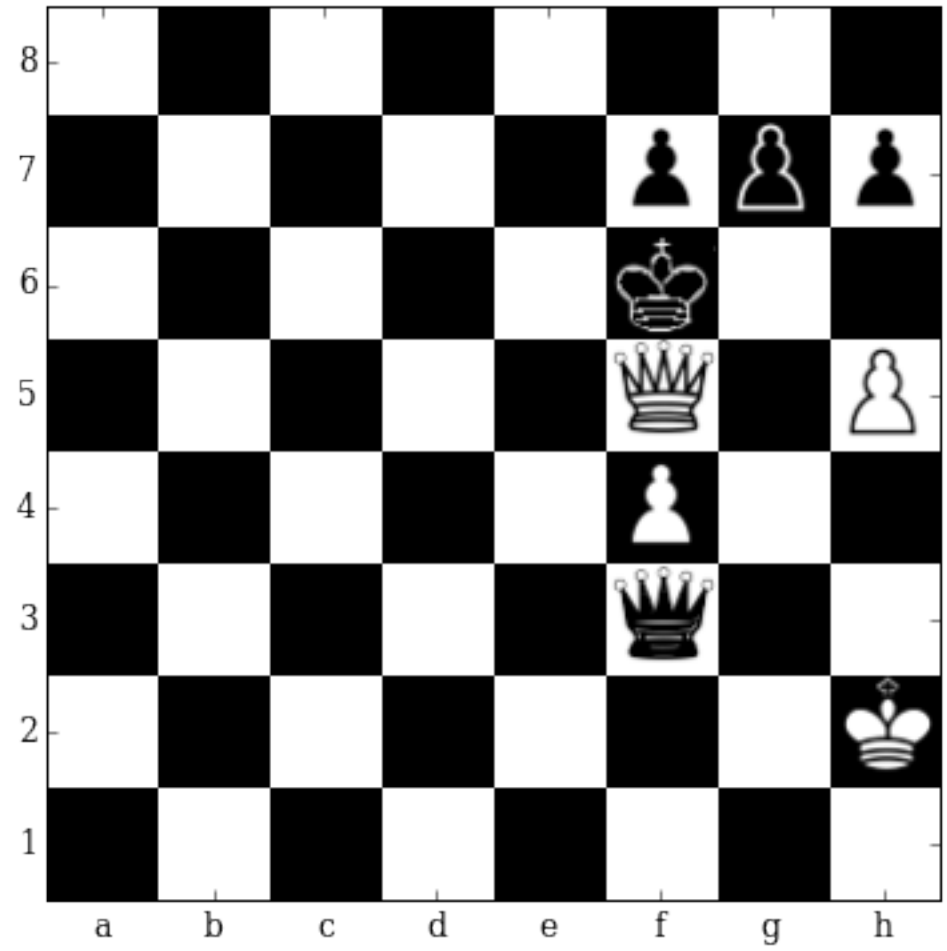

```
c5e7 0.028200475499
```



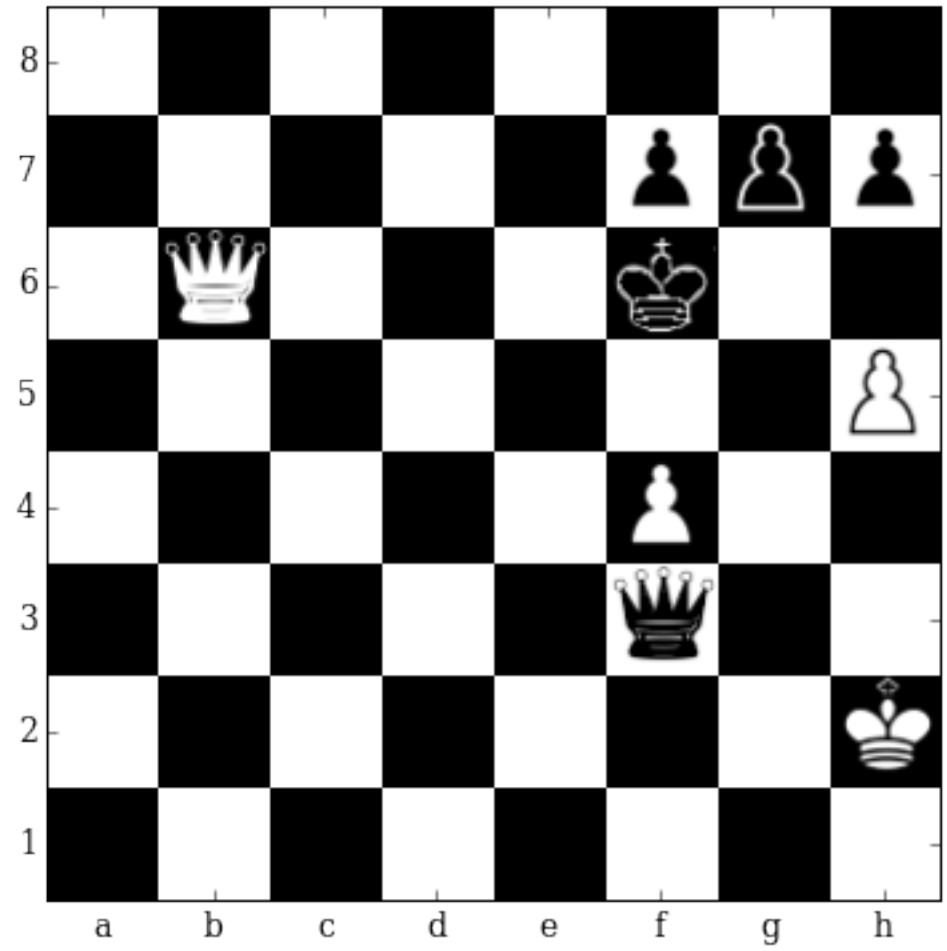

```
c5e5 -0.0355458073318
```



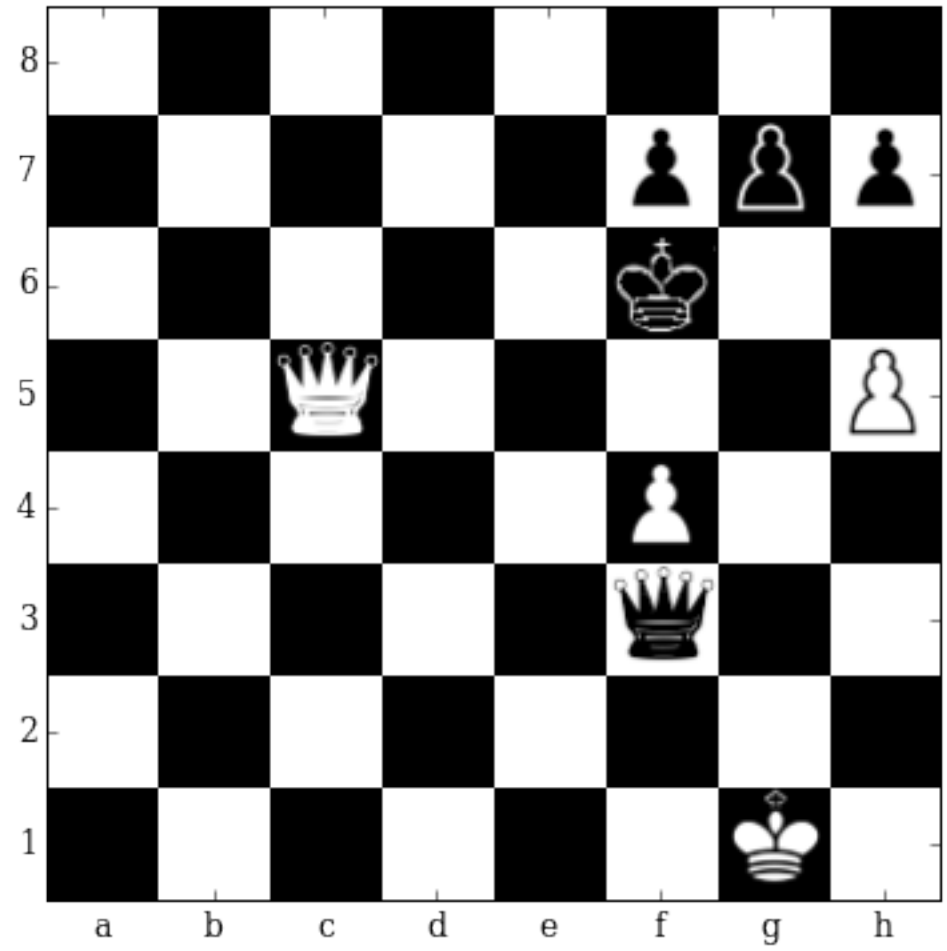

```
c5f2 -0.0388837084174
```



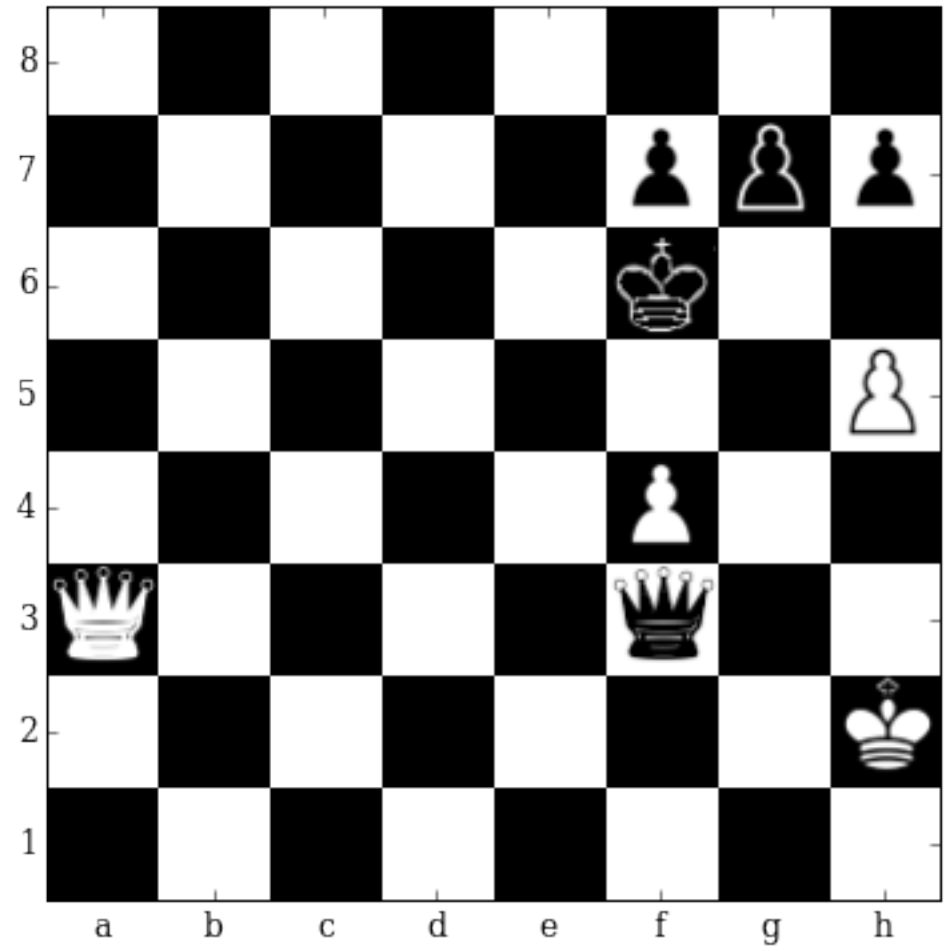

```
c5f5 -0.0413541197777
WORST 5 Moves
```



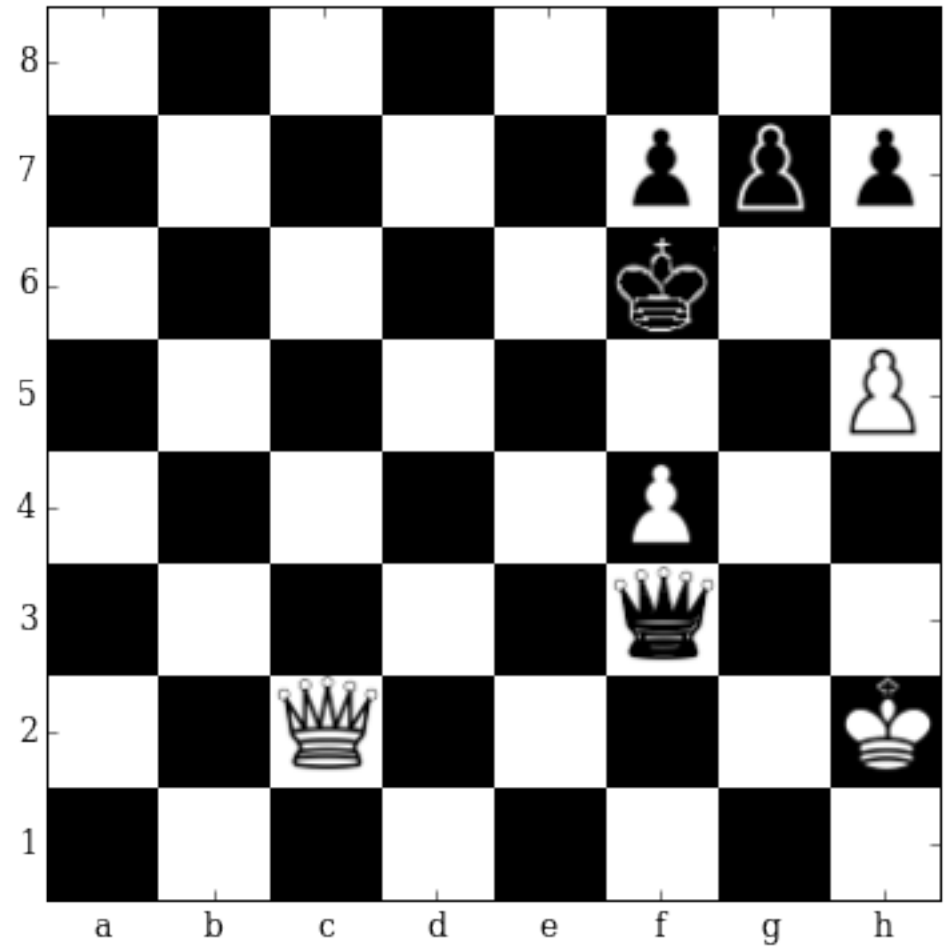

```
c5b6 -0.0826028212905
```



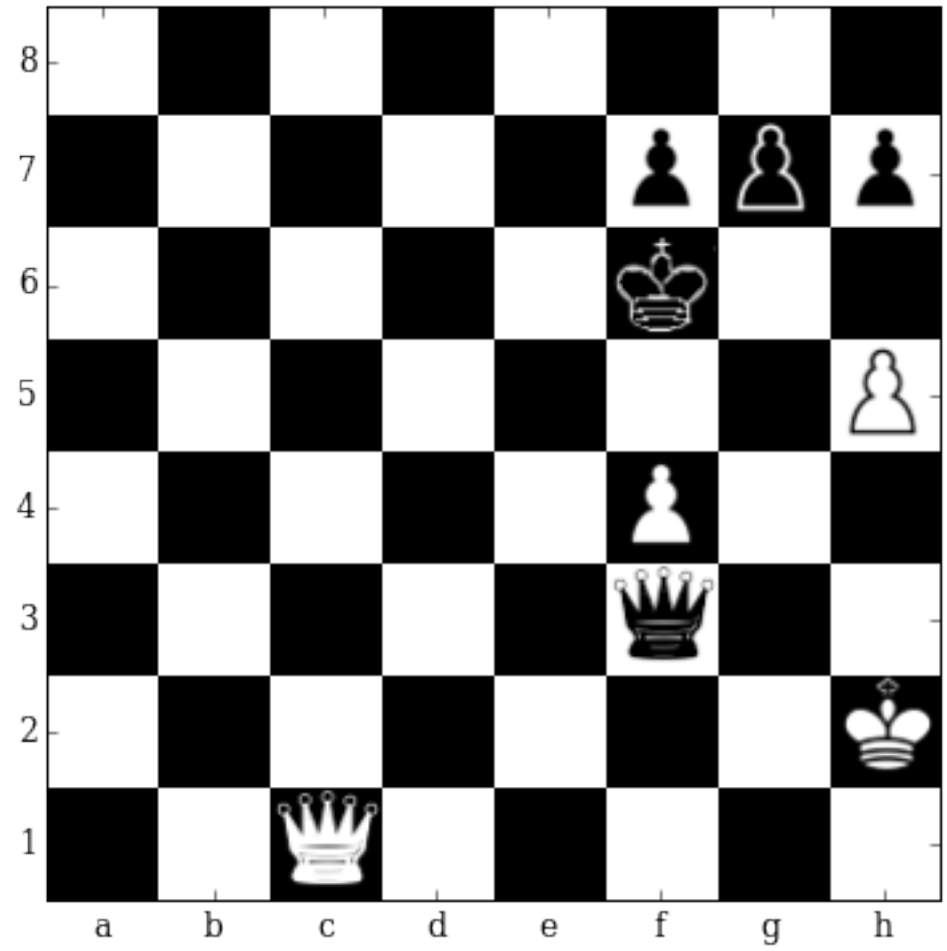

```
h2g1 -0.0835127830505
```




```
c5a3 -0.0964041724801
```

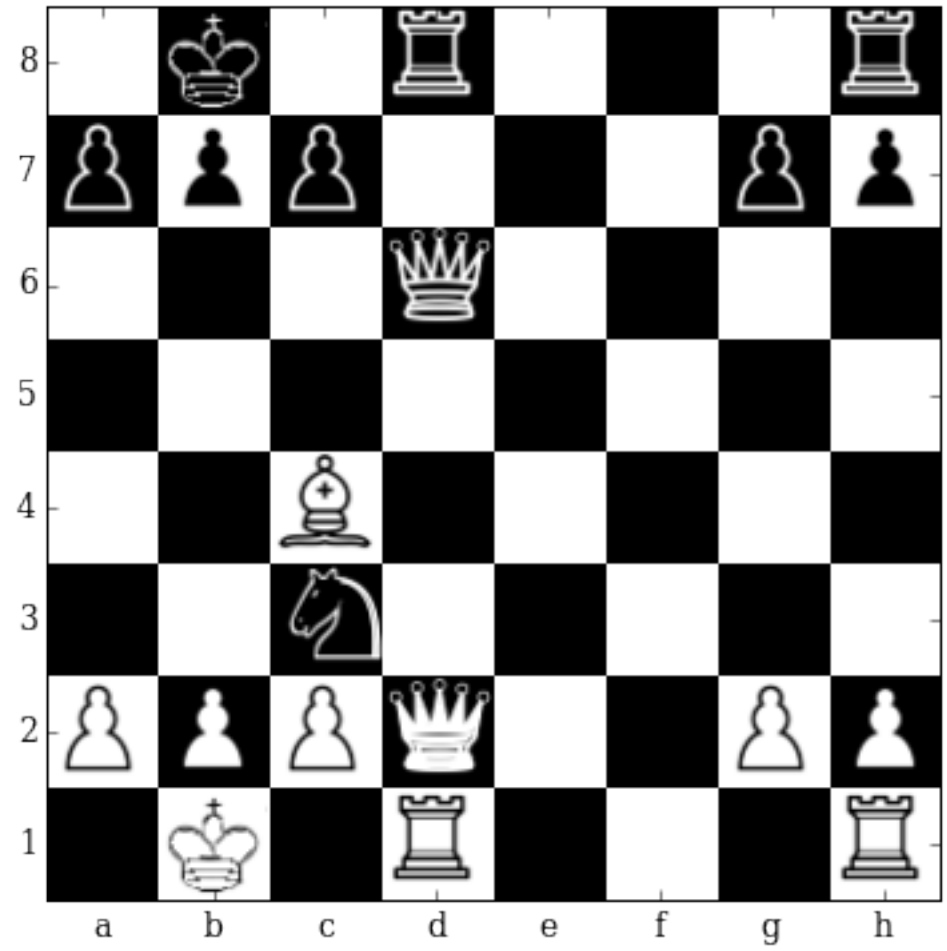



```
c5c2 -0.111014157534
```

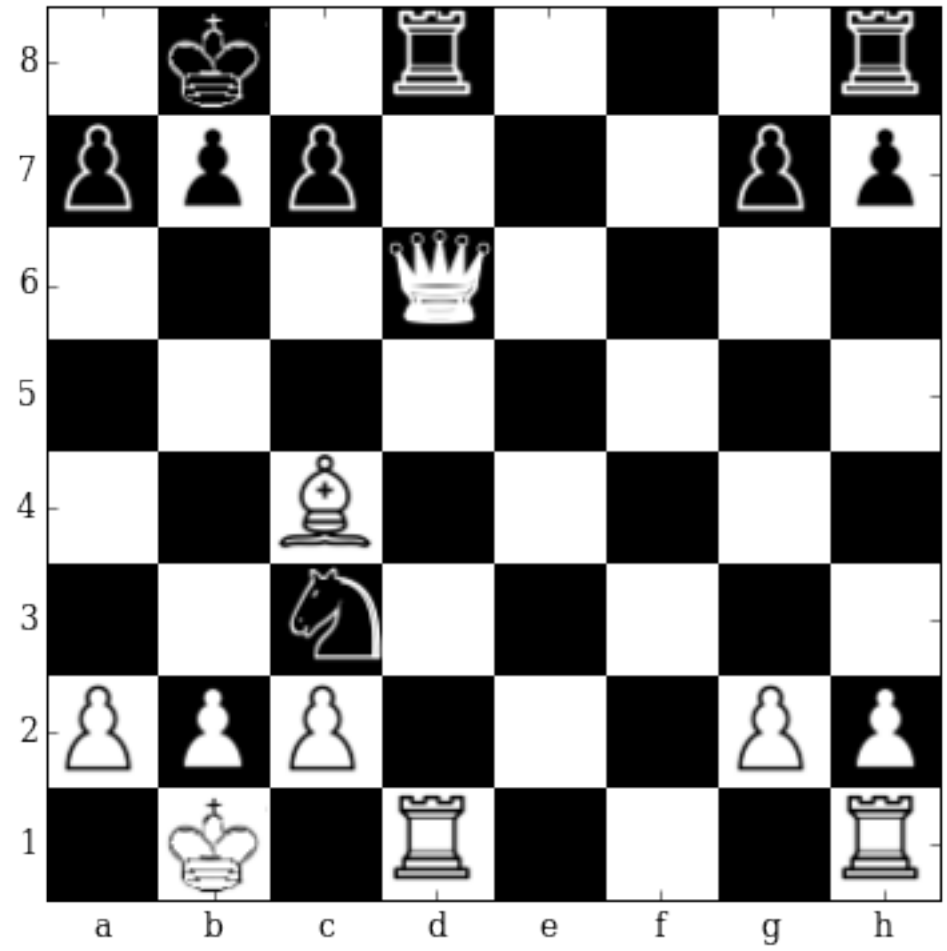



```
c5c1 -0.129943683743
```

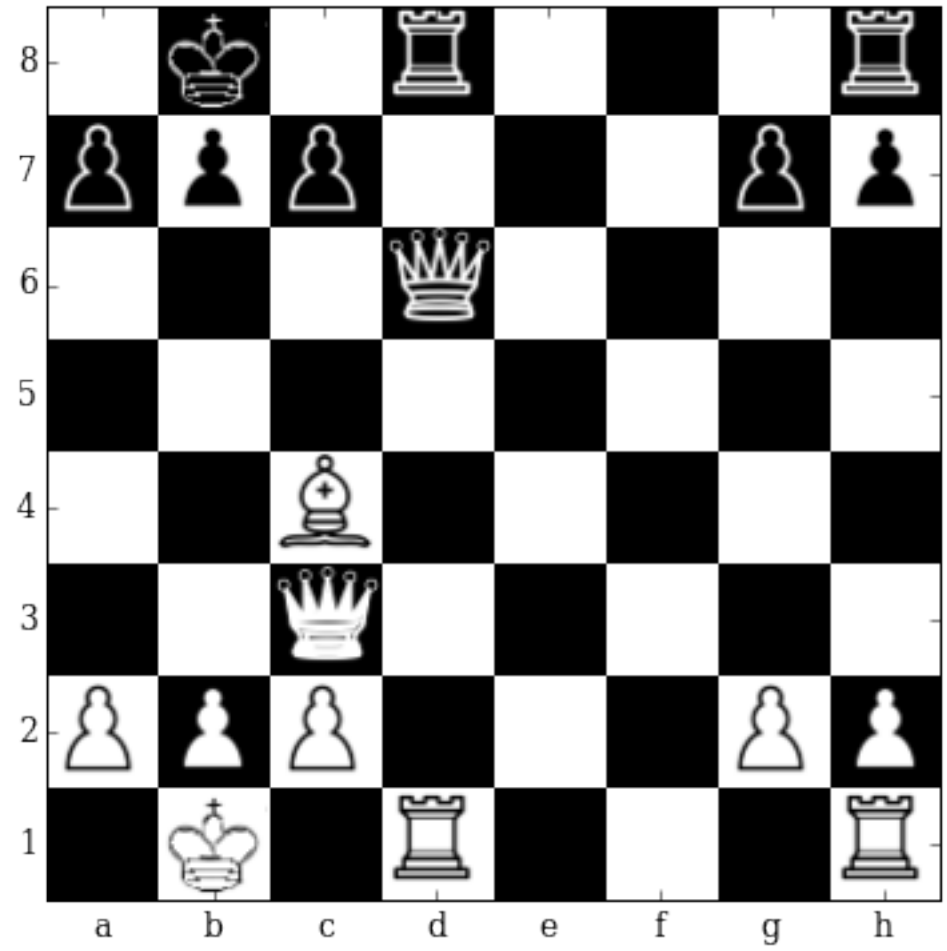
```
In [19]: '''
         Check and fork by the knight
         '''
         fen='1k1r3r/ppp3pp/3q4/8/2B5/2n5/PPPQ2PP/1K1R3R w - - 0 0'
         top_bottom_moves(fen)
```



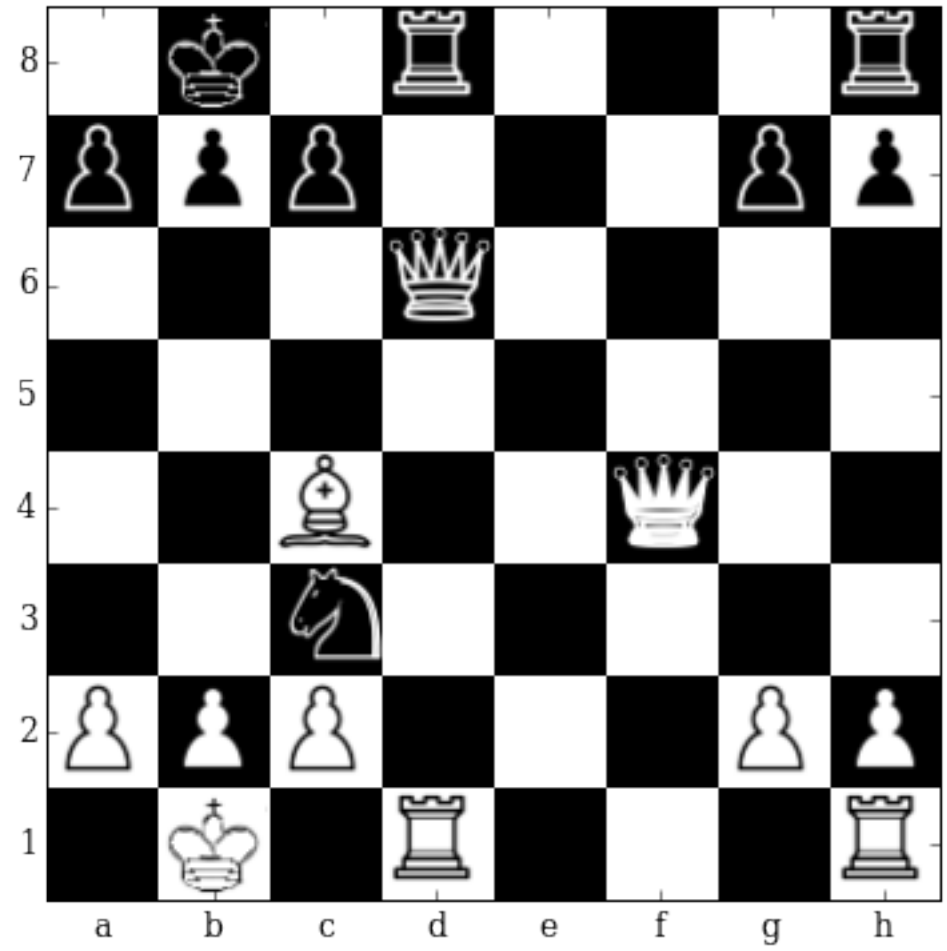

```
Current evaluation: -0.007698
Total 41 moves possible
TOP 5 Moves
```



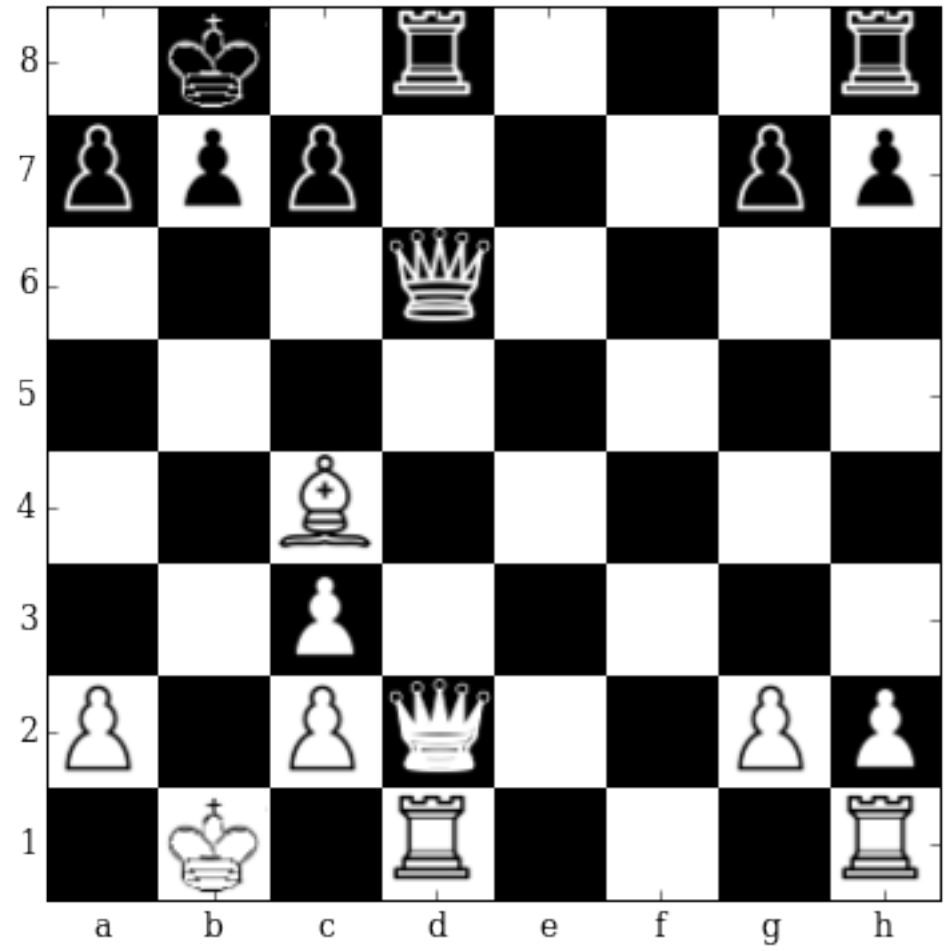

```
d2d6 0.139244303107
```



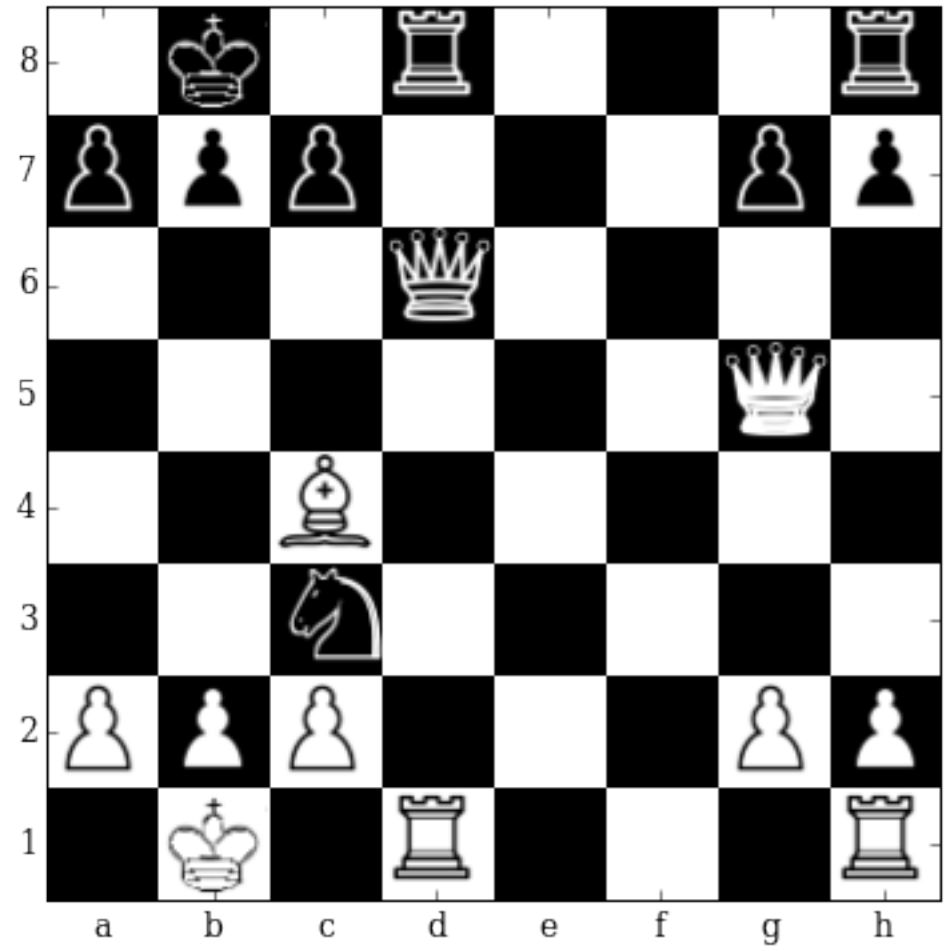
```
d2c3 0.0400848388672
```



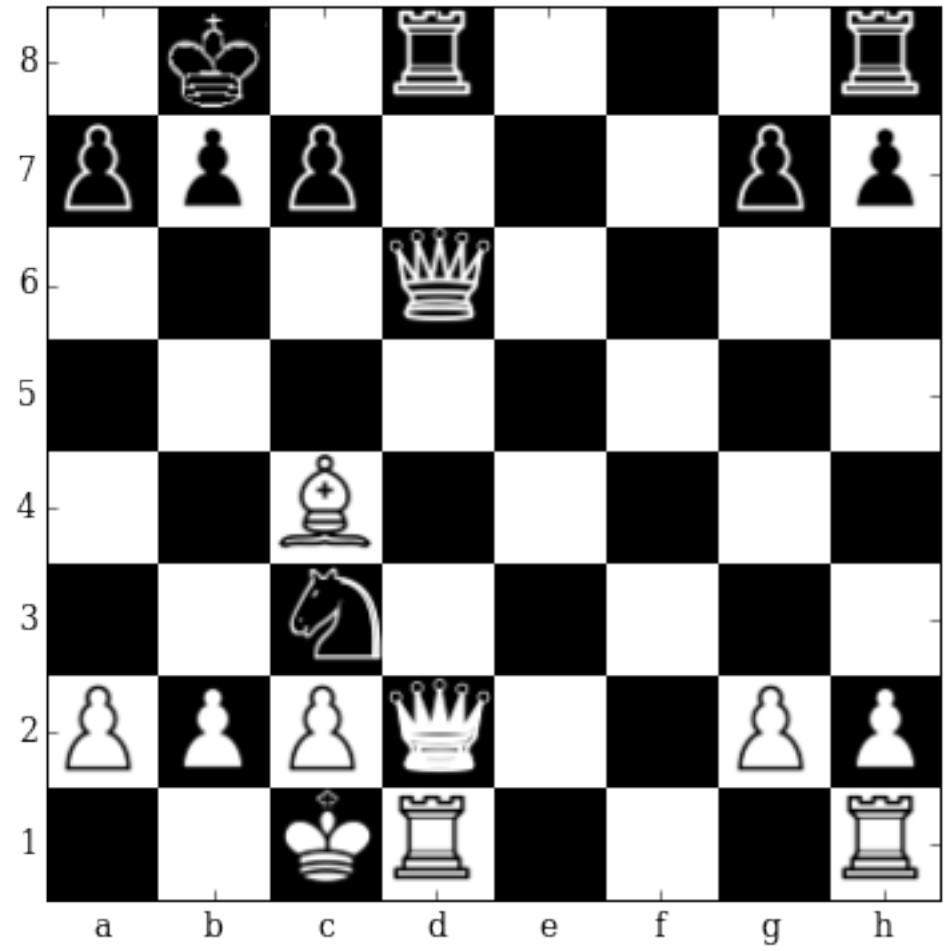

```
d2f4 0.0280776247382
```



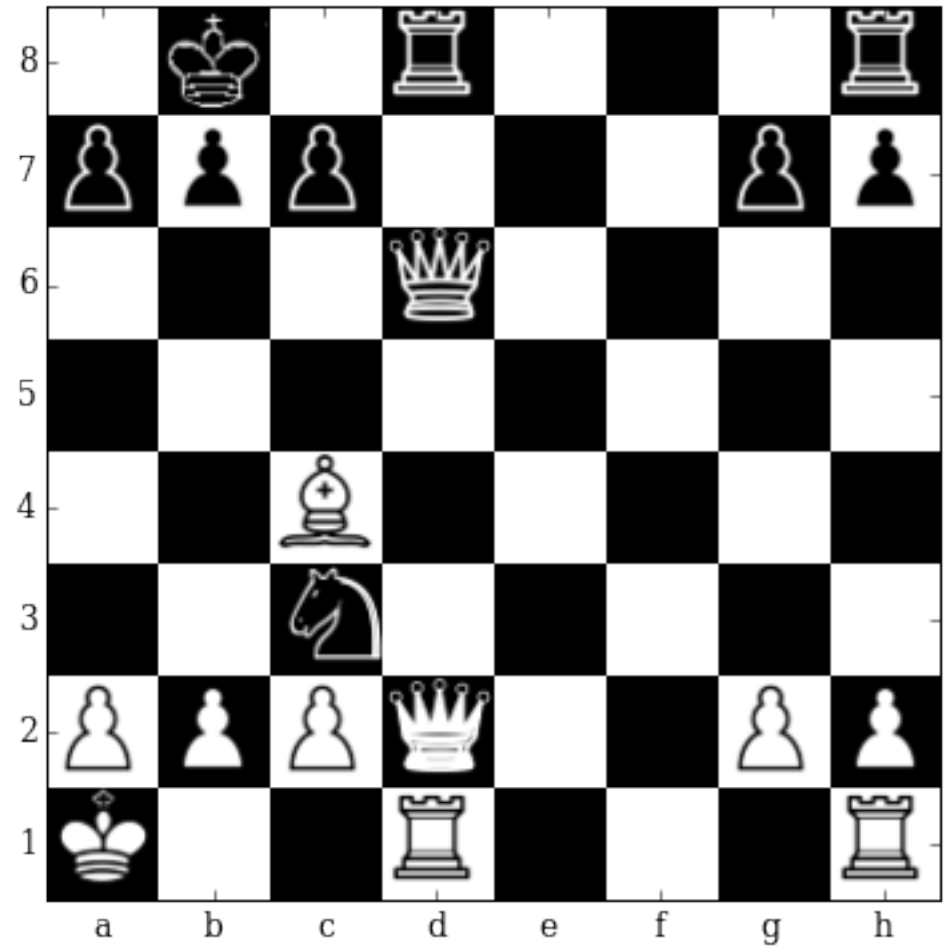
```
b2c3 0.0228184554726
```



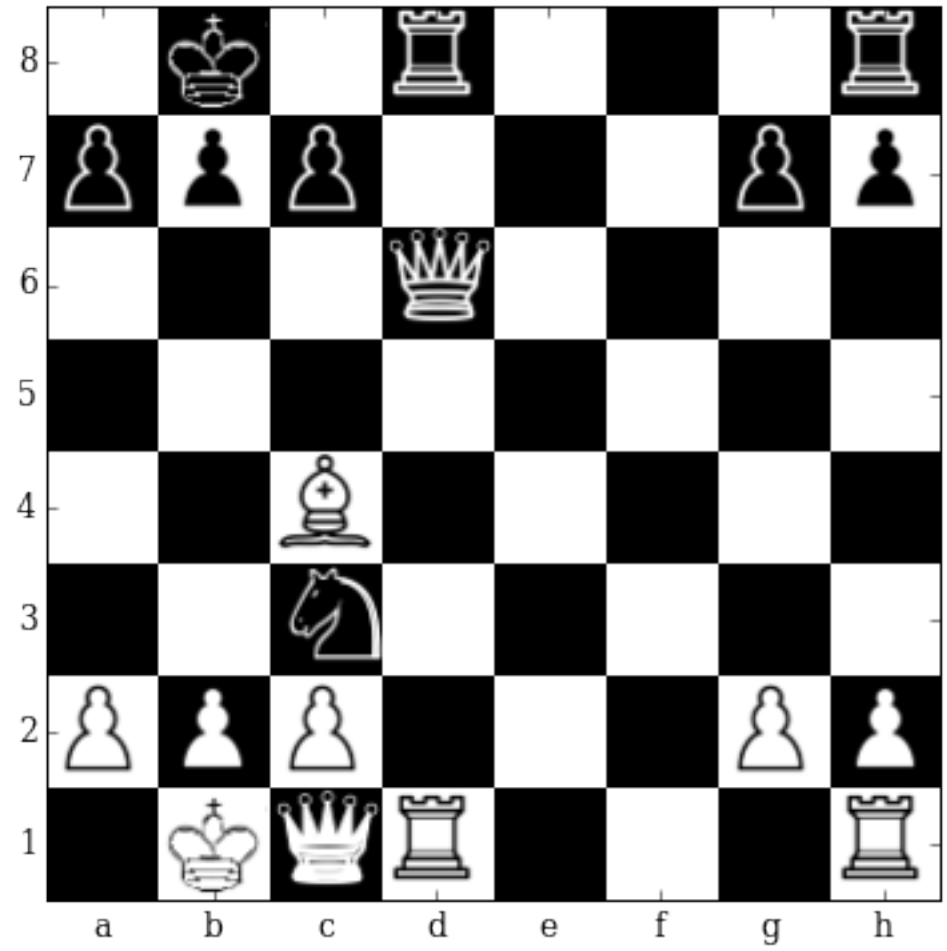

```
d2g5 0.0207833554596
WORST 5 Moves
```



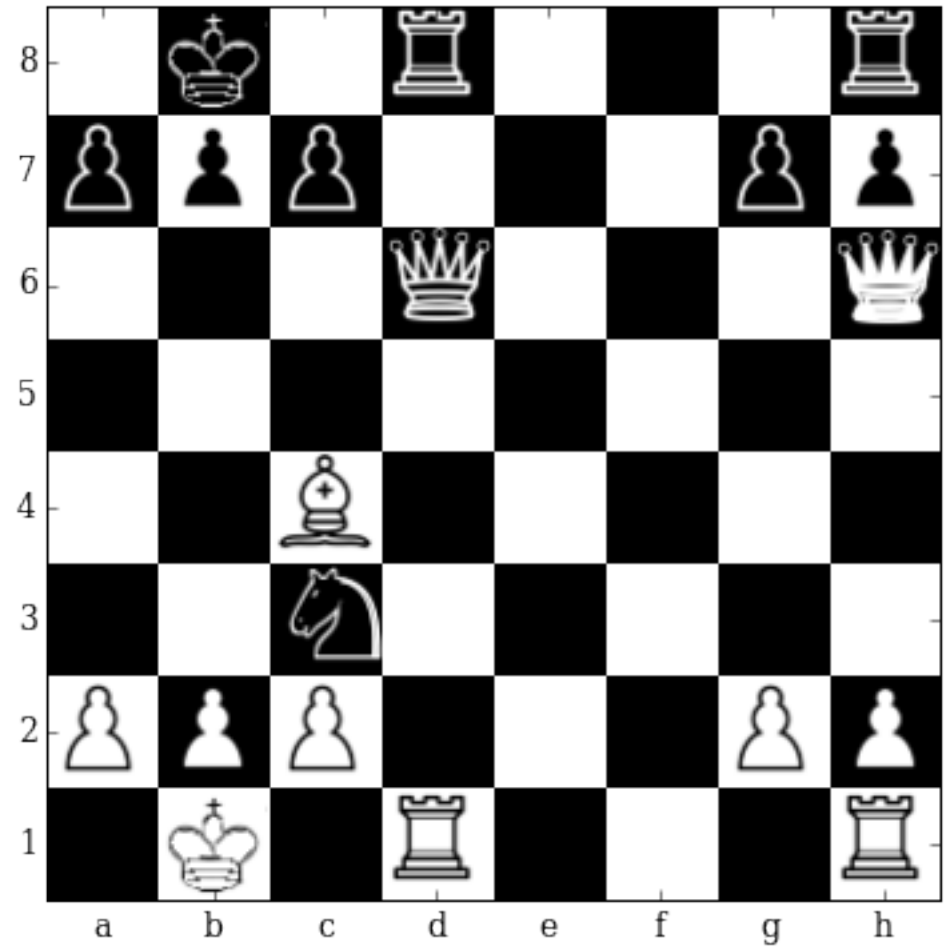

```
b1c1 -0.0201652310789
```



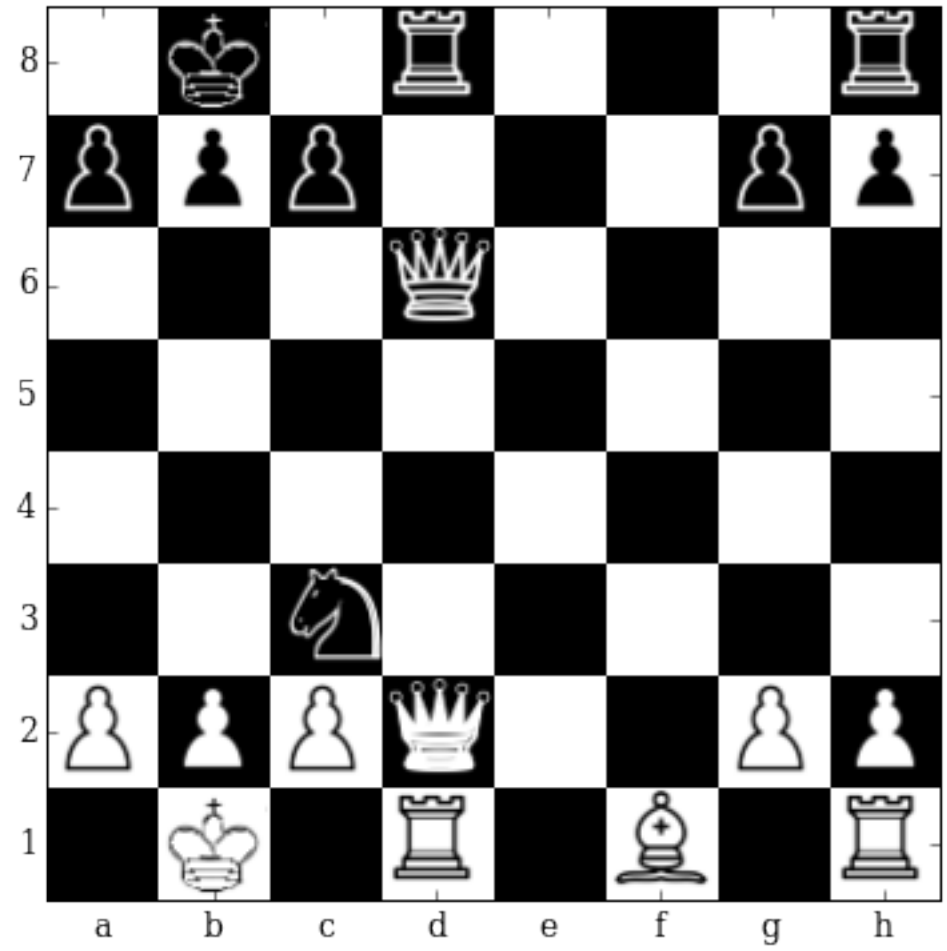

```
b1a1 -0.0206535439938
```




```
d2c1 -0.0241448543966
```

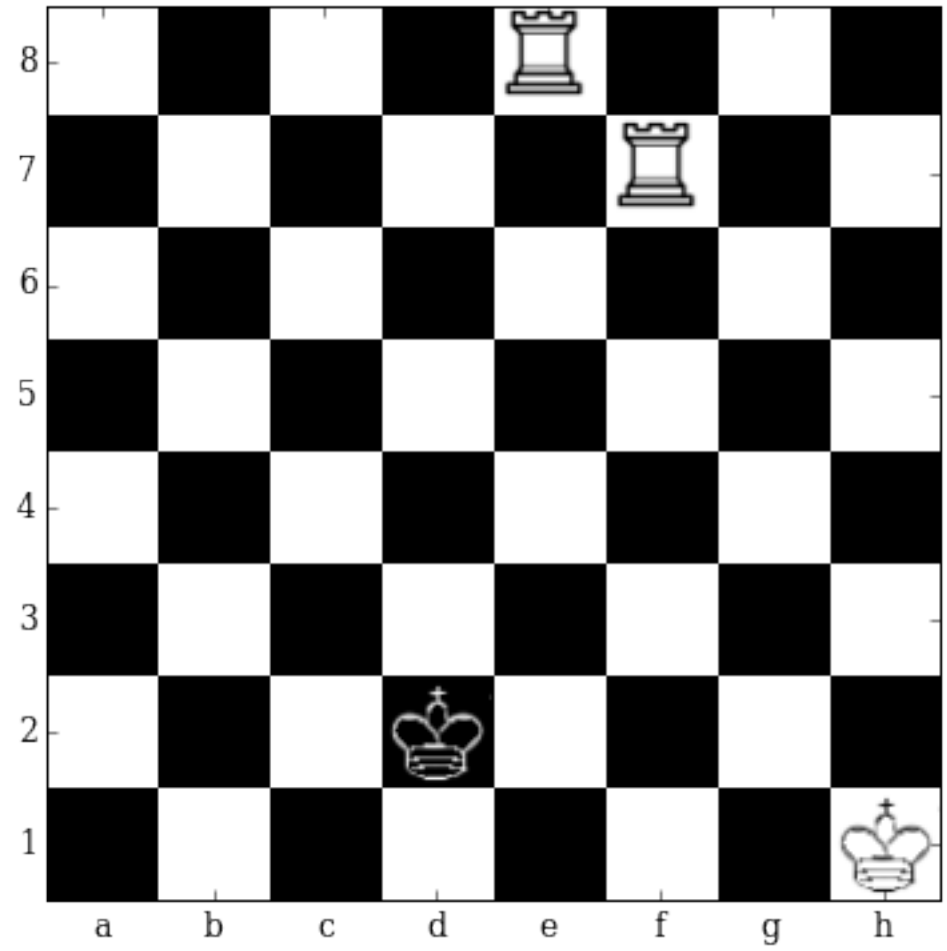


```
d2h6 -0.0248840861022
```

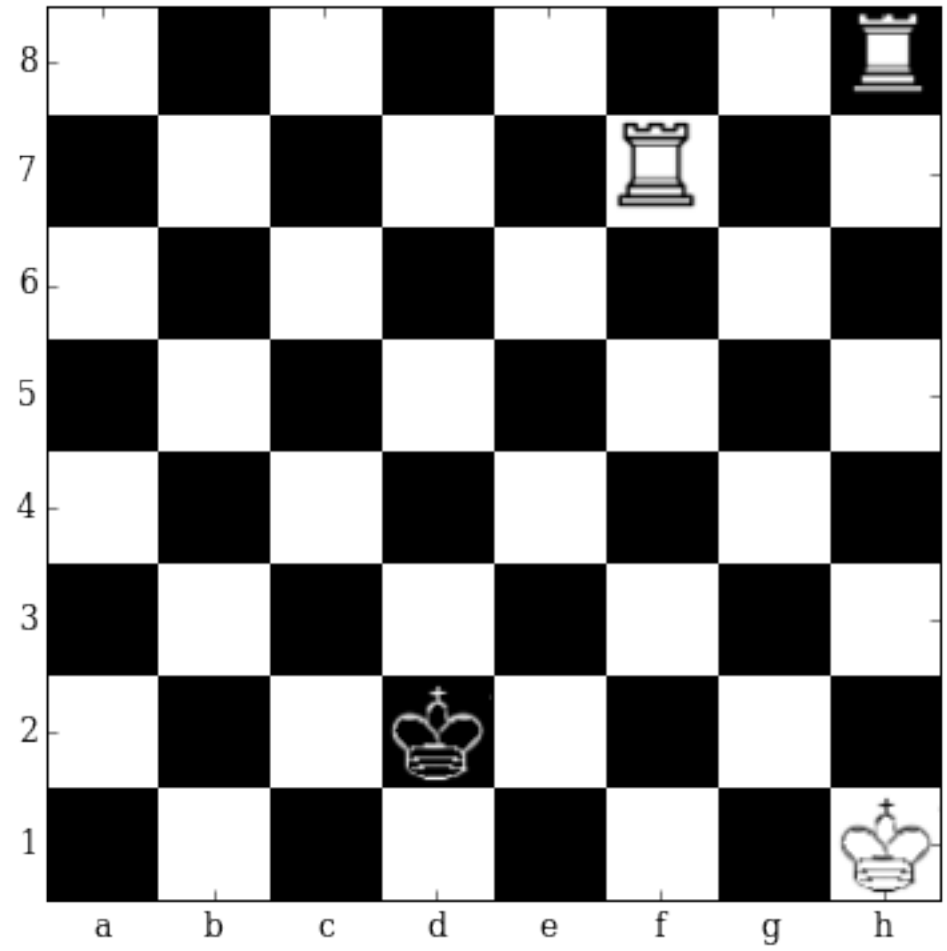



```
c4f1 -0.0262713693082
```

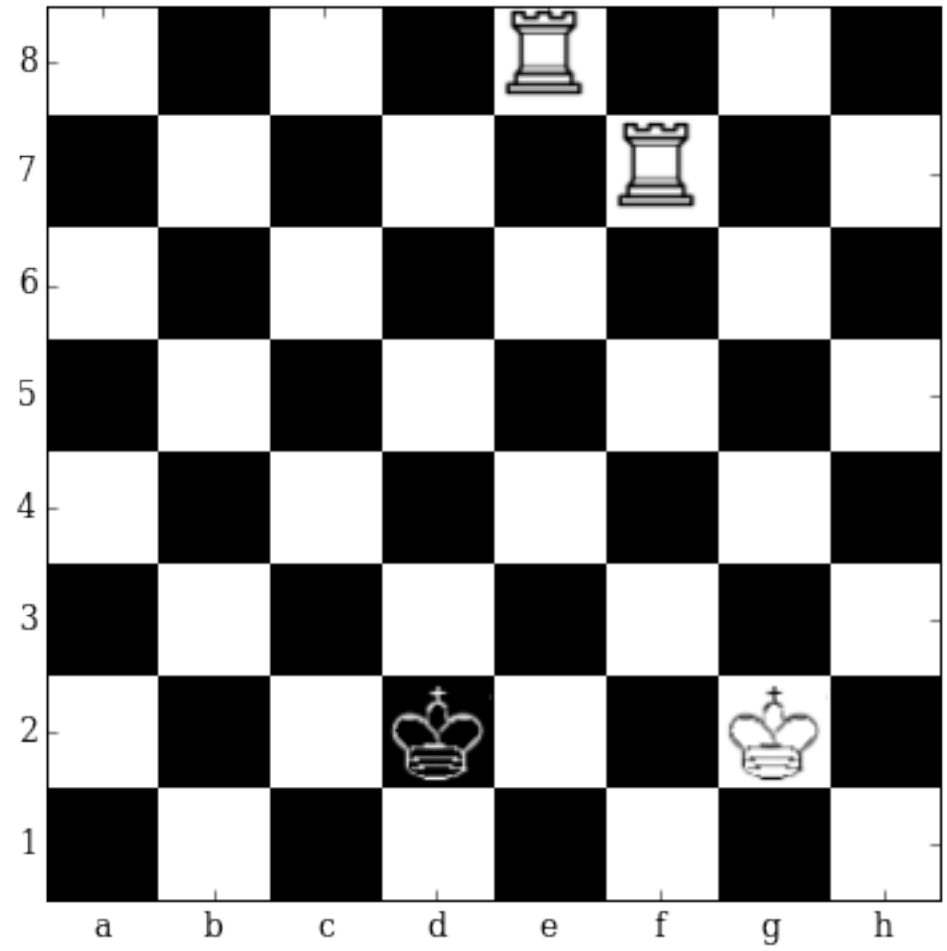
```
In [20]: '''
         Pushing the broom
         '''
         fen='4R3/5R2/8/8/8/8/3k4/7K w - - 0 0'
         top_bottom_moves(fen)
```



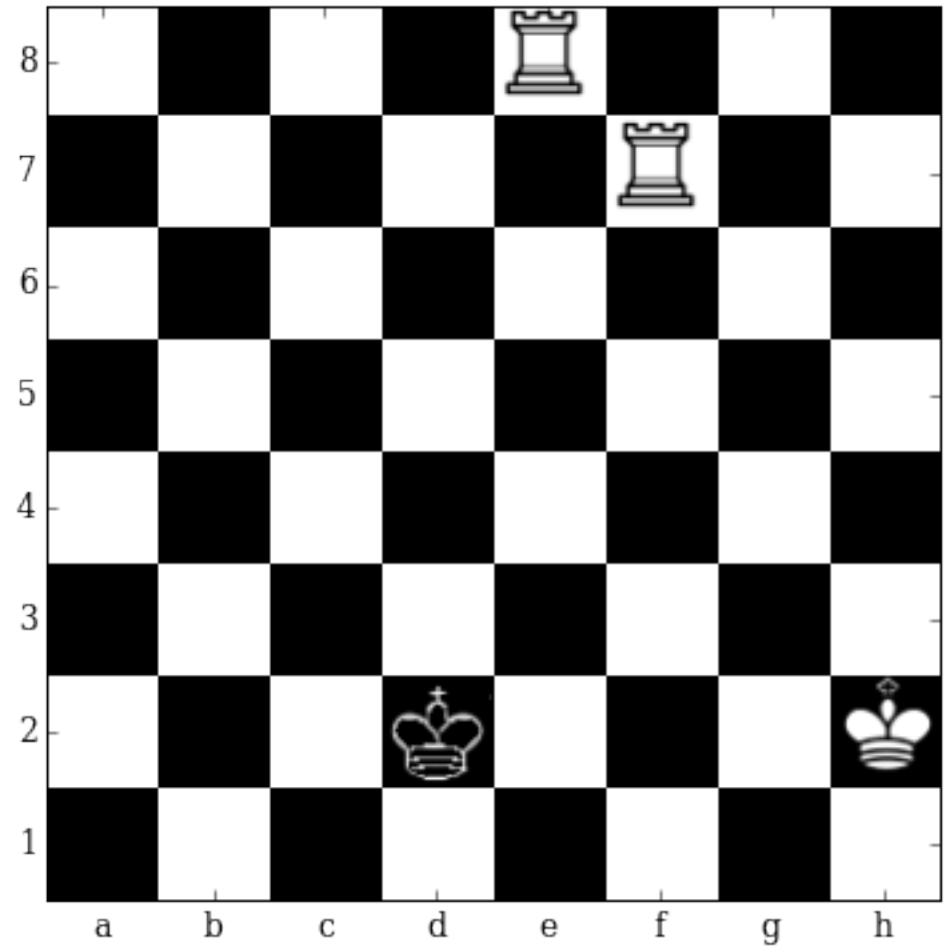

```
Current evaluation: 0.227653
Total 31 moves possible
TOP 5 Moves
```



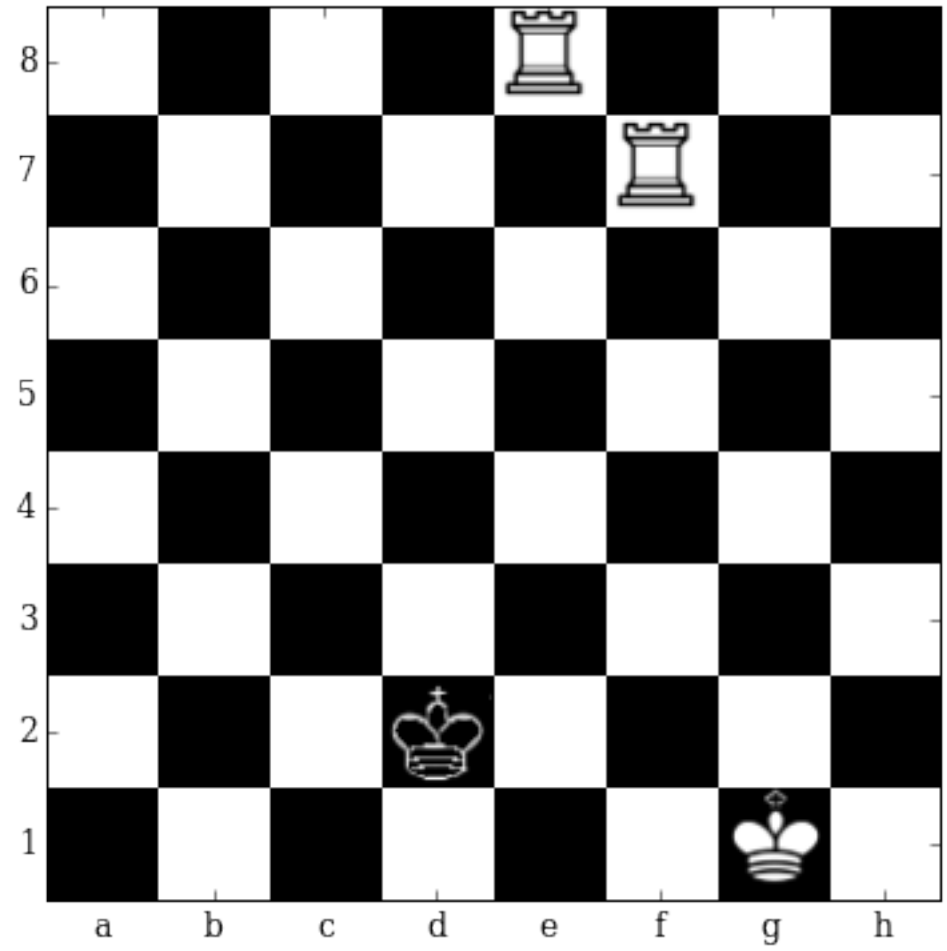

```
e8h8 0.307574003935
```



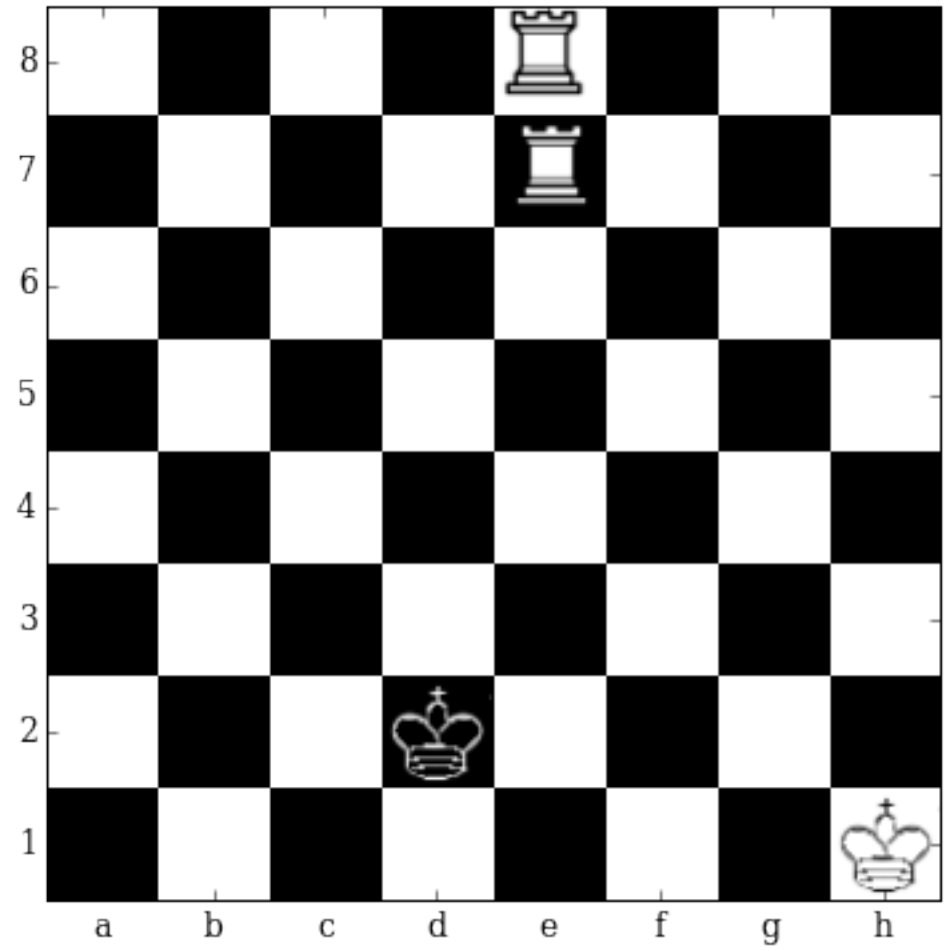

```
h1g2 0.279015421867
```



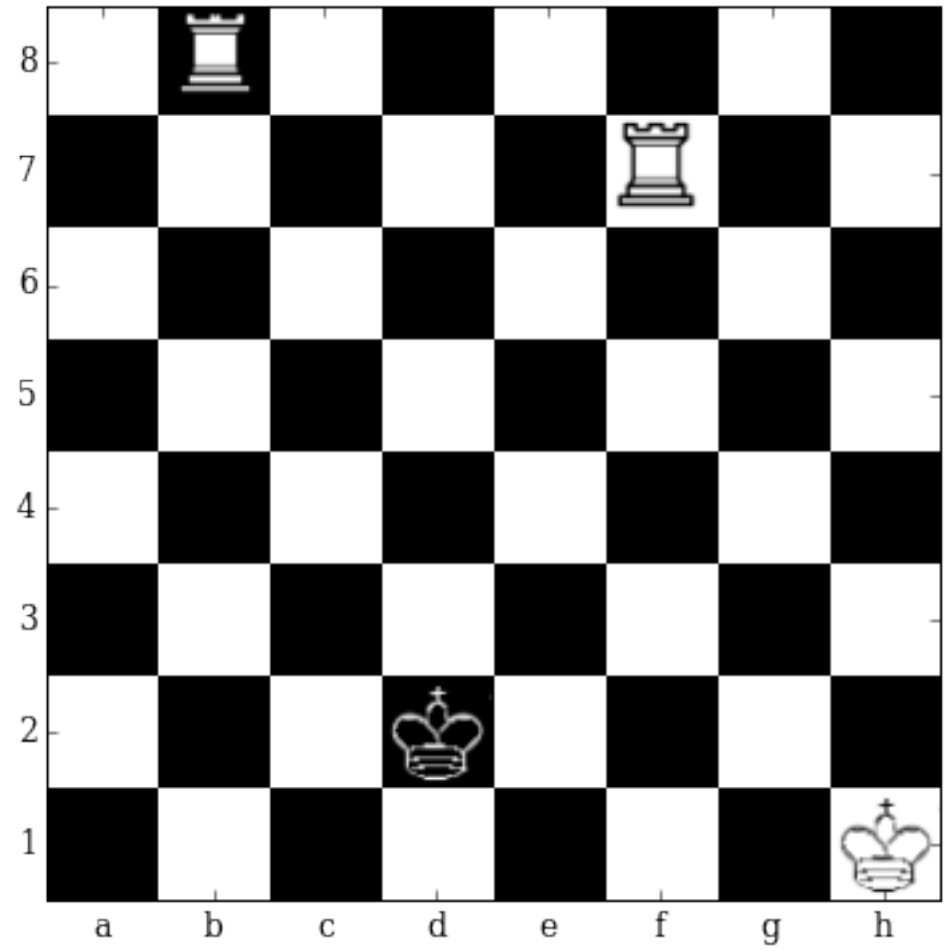

```
h1h2 0.25475075841
```



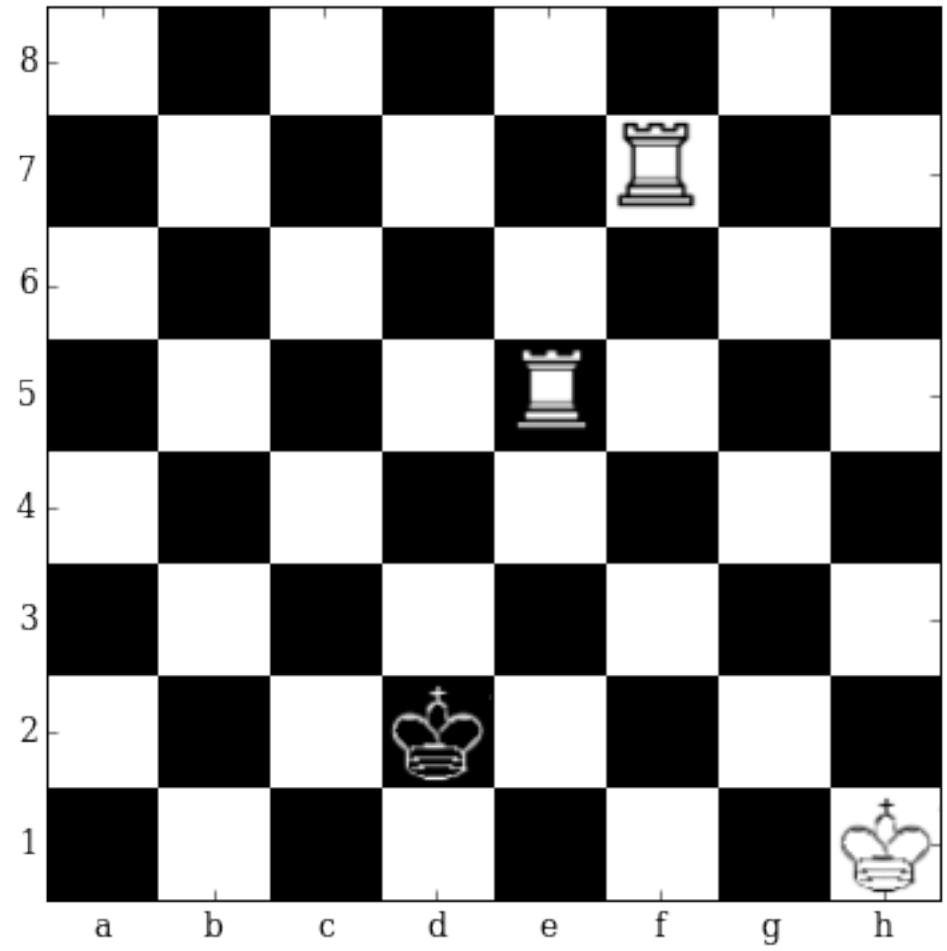

```
h1g1 0.244157657027
```



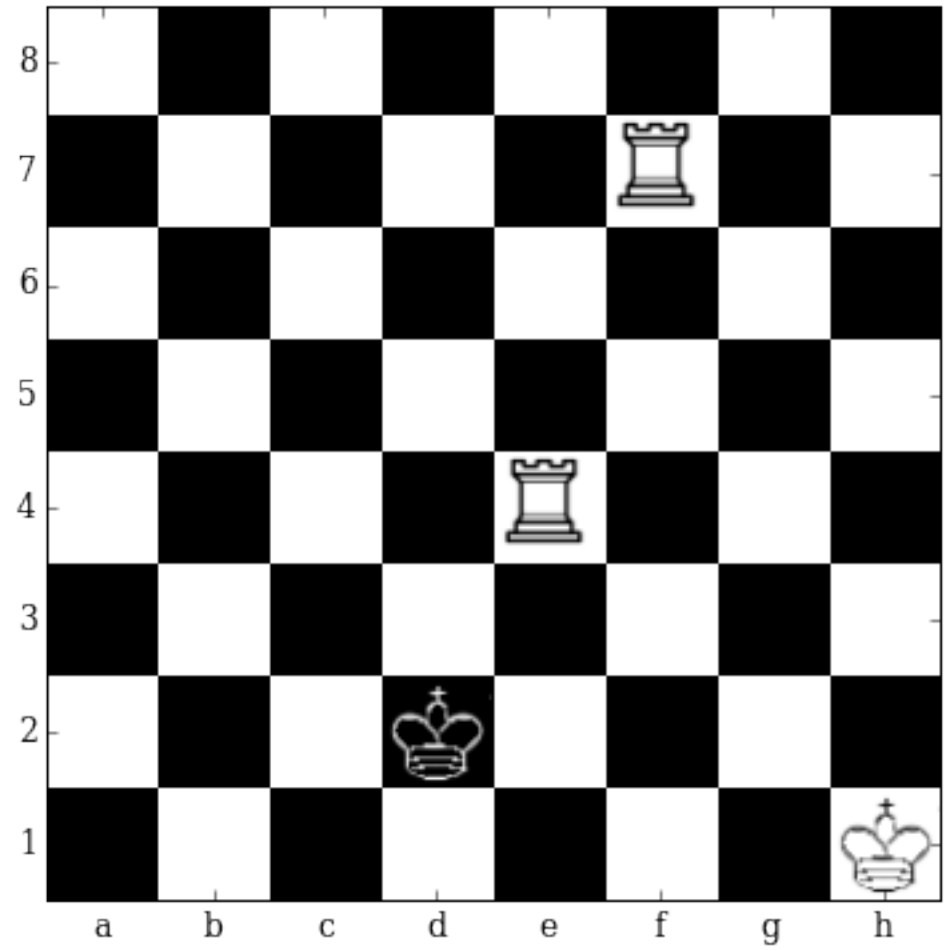

```
f7e7 0.242229655385
WORST 5 Moves
```



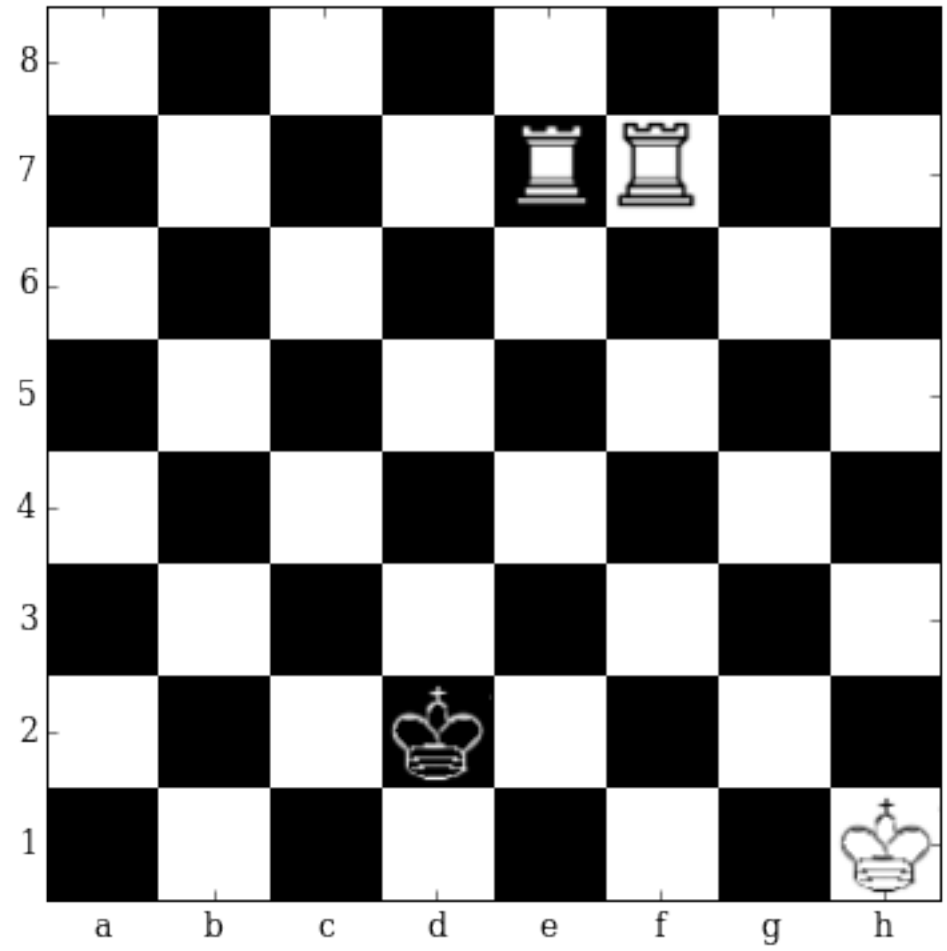

```
e8b8 0.180342525244
```



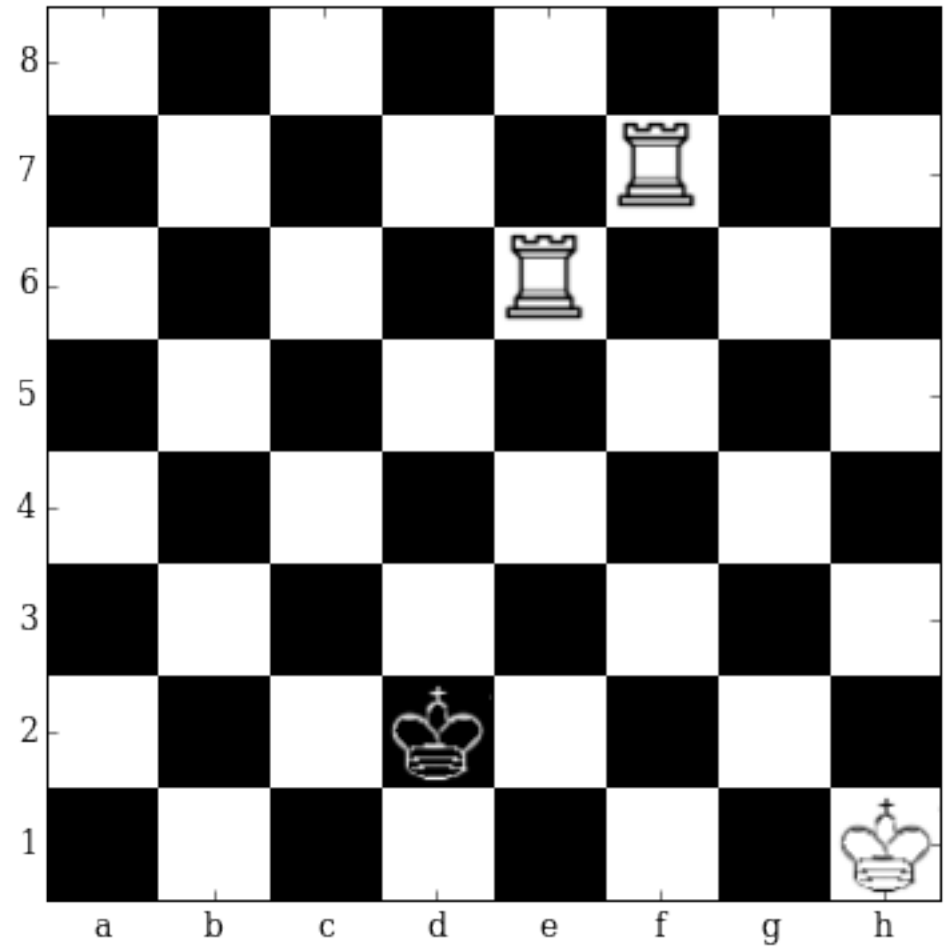

```
e8e5 0.179203212261
```




```
e8e4 0.179086685181
```

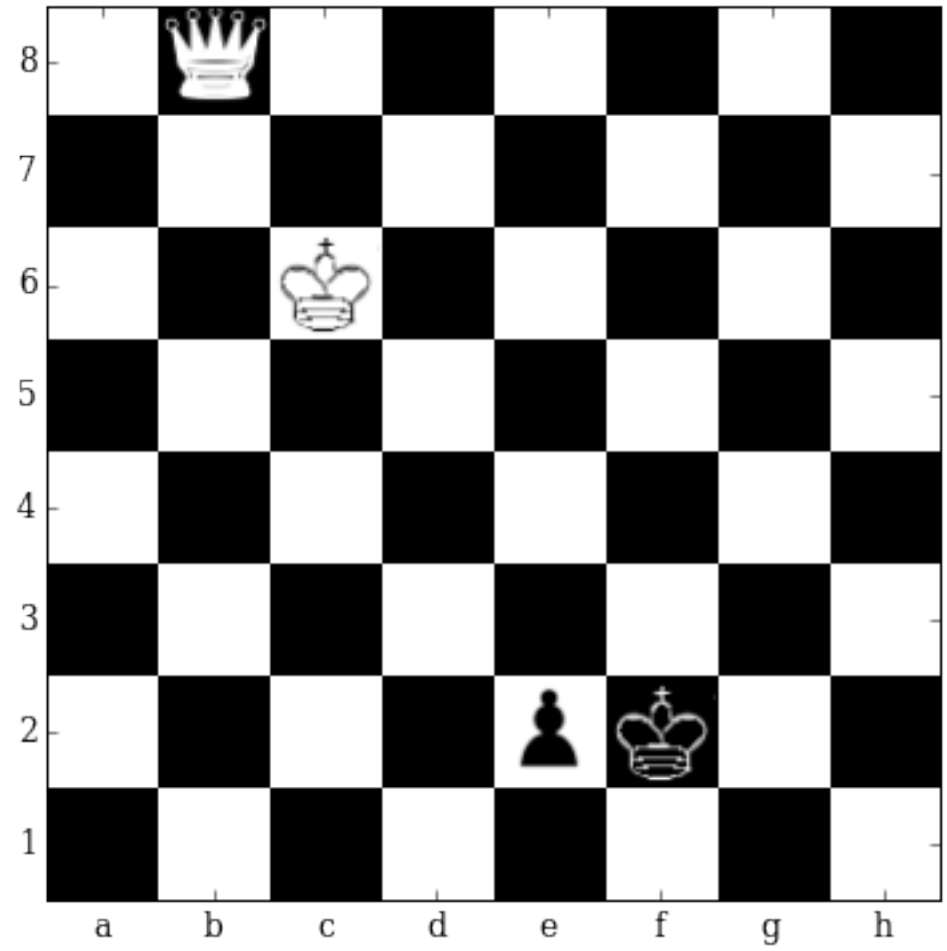



```
e8e7 0.178259670734
```

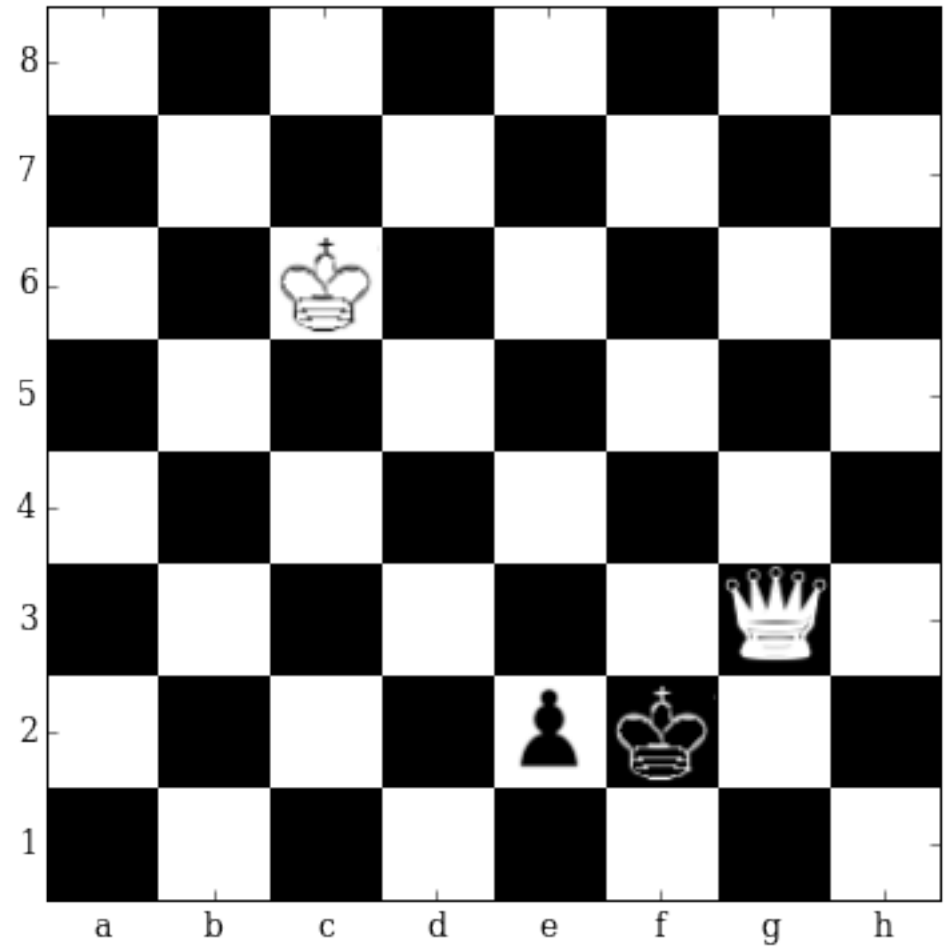



```
e8e6 0.162149891257
```

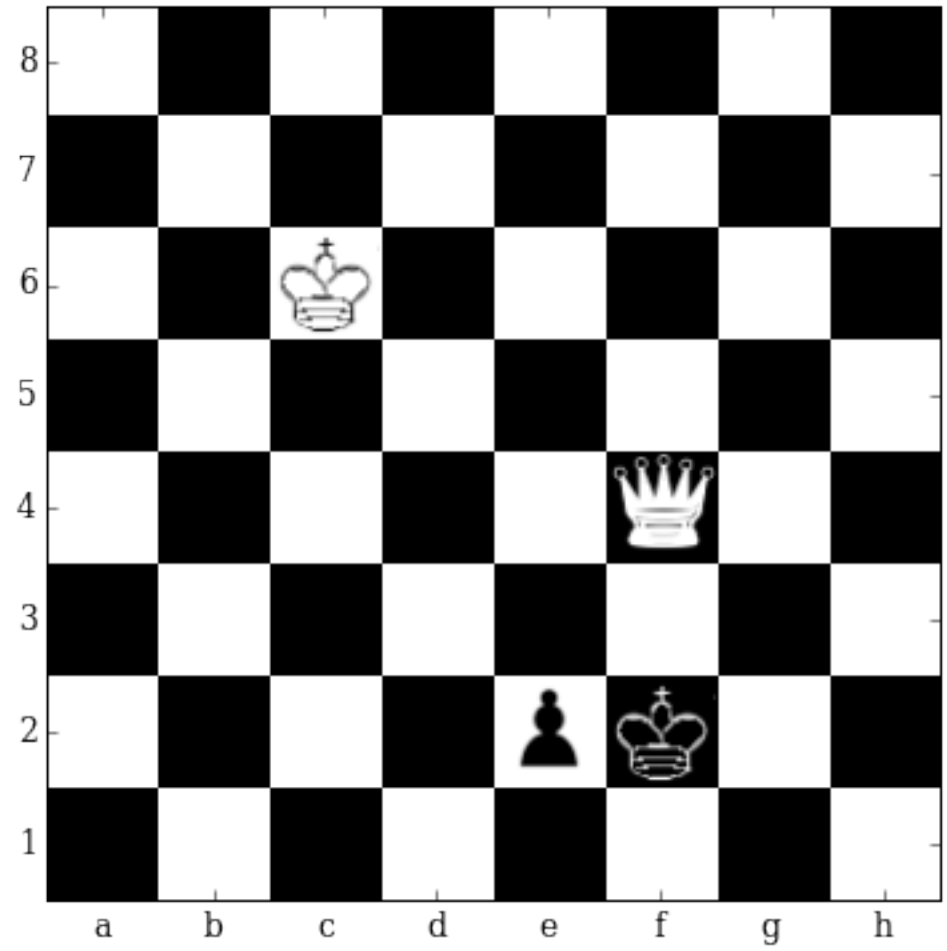
```
In [21]: '''
         Queen King vs King Pawn
         '''
         fen = '1Q6/8/2K5/8/8/8/4pk2/8 w - - 0 0'
         top_bottom_moves(fen)
```



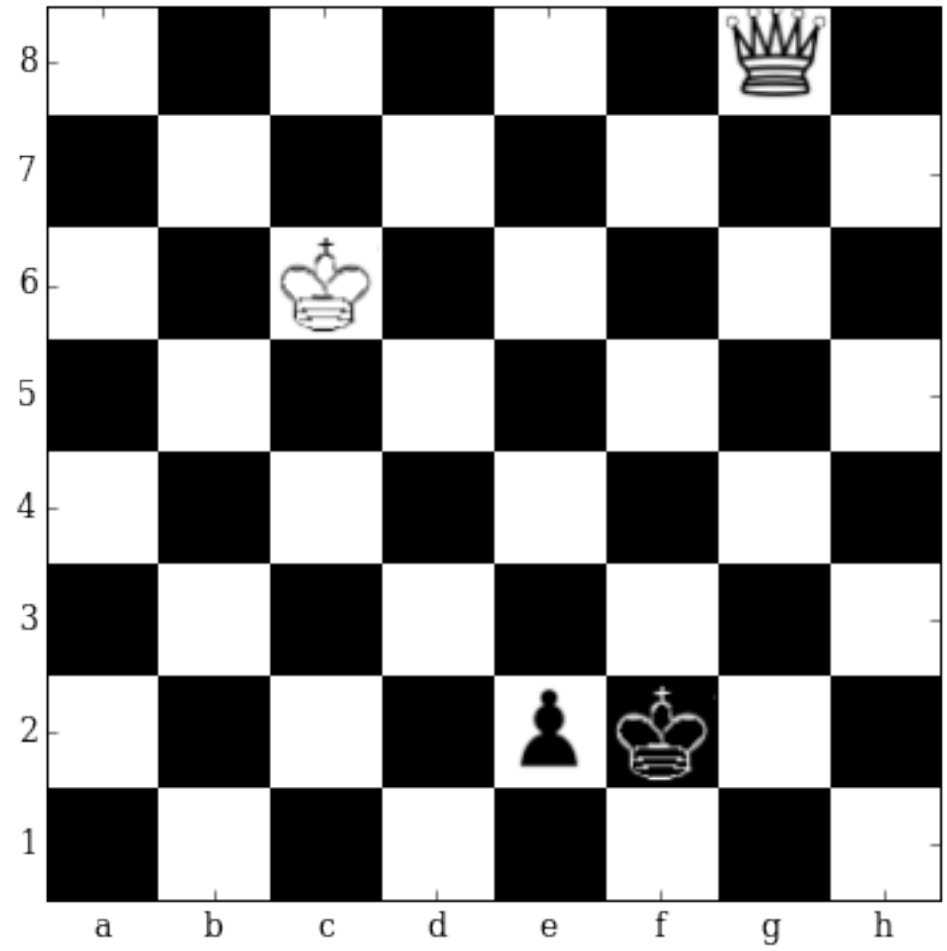

```
Current evaluation: 0.132124
Total 29 moves possible
TOP 5 Moves
```



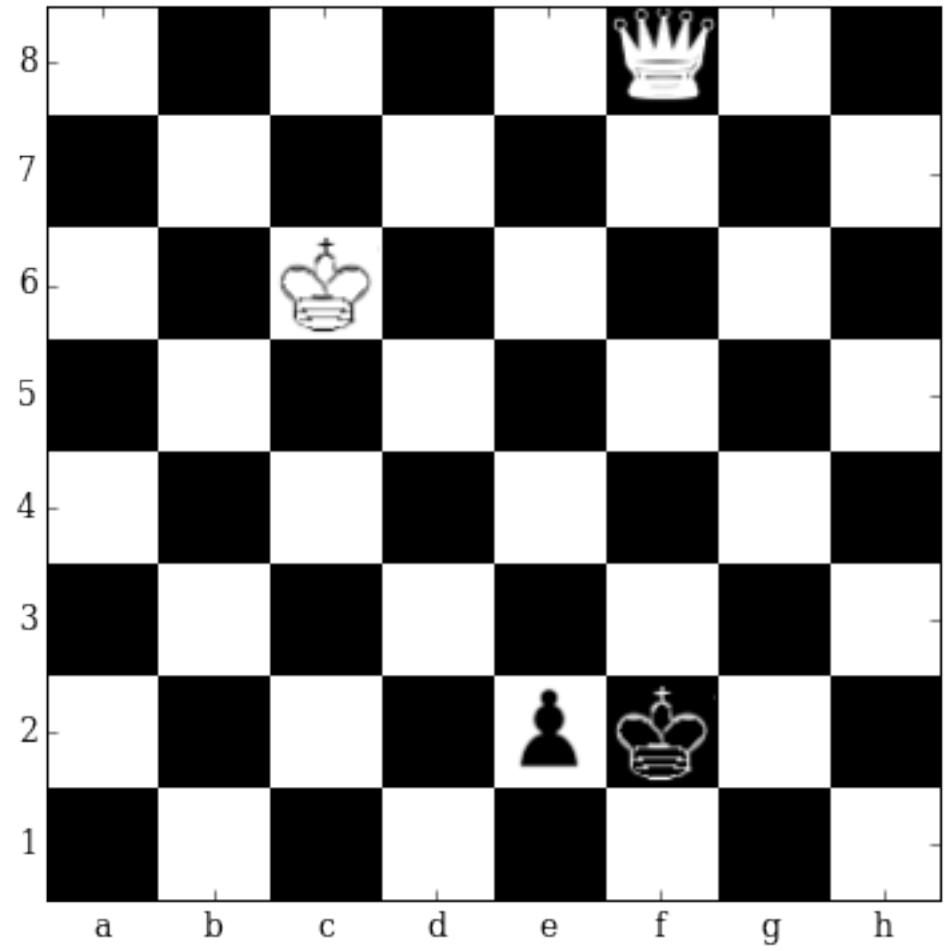

```
b8g3 0.232341885567
```



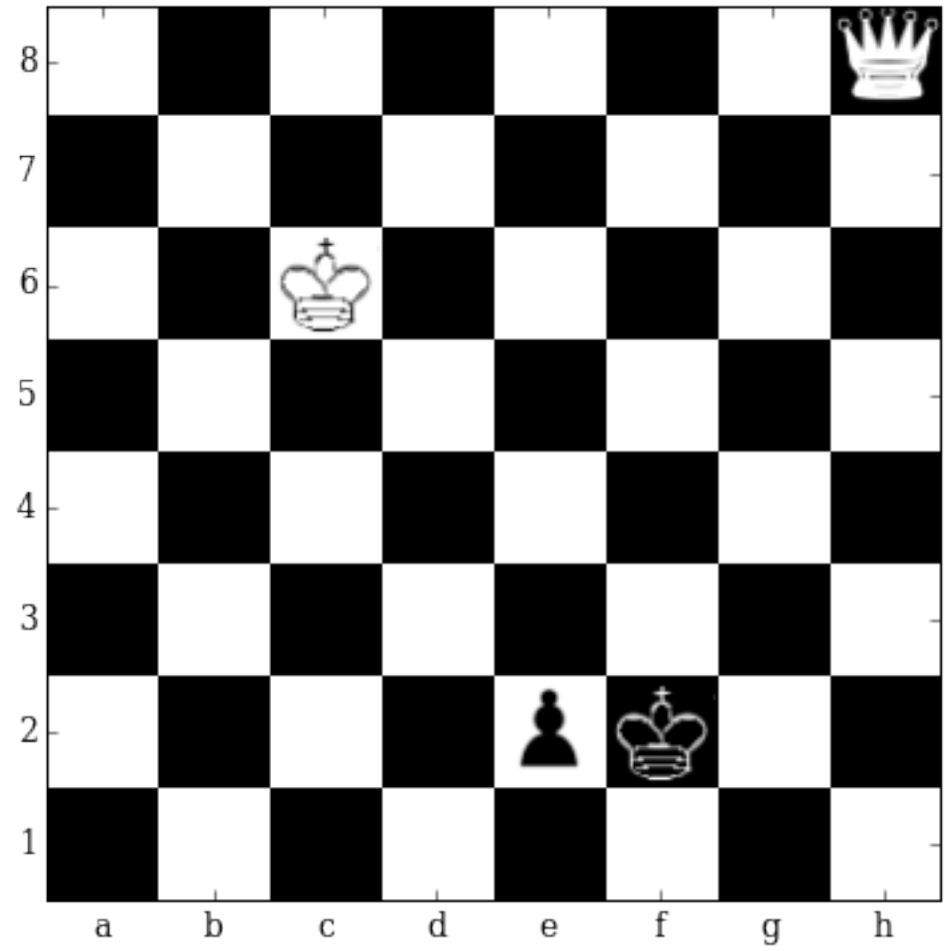

```
b8f4 0.220576122403
```



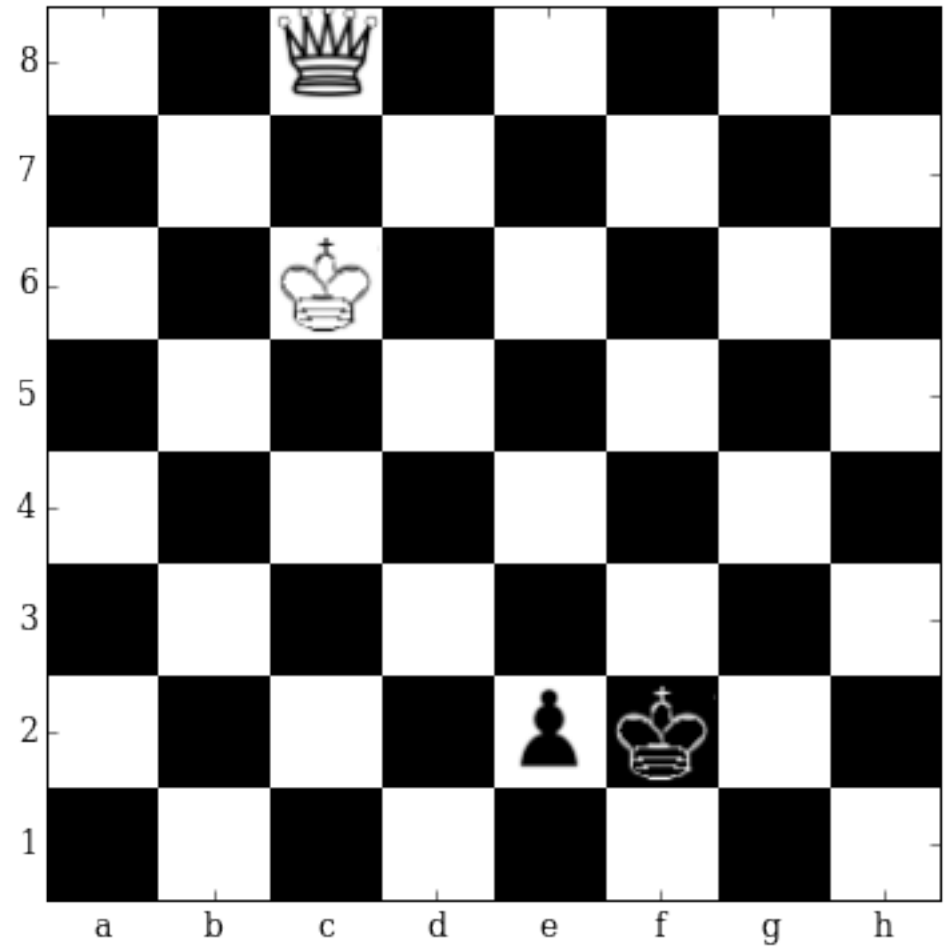

```
b8g8 0.212574183941
```



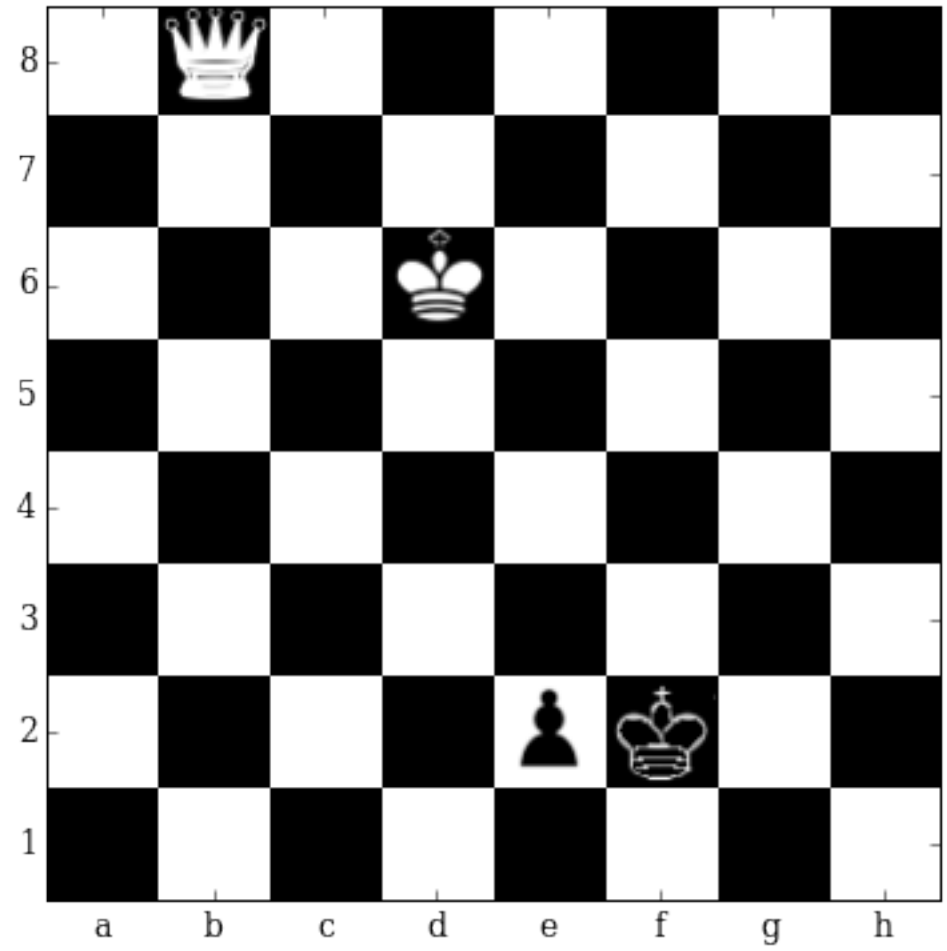

```
b8f8 0.208147808909
```



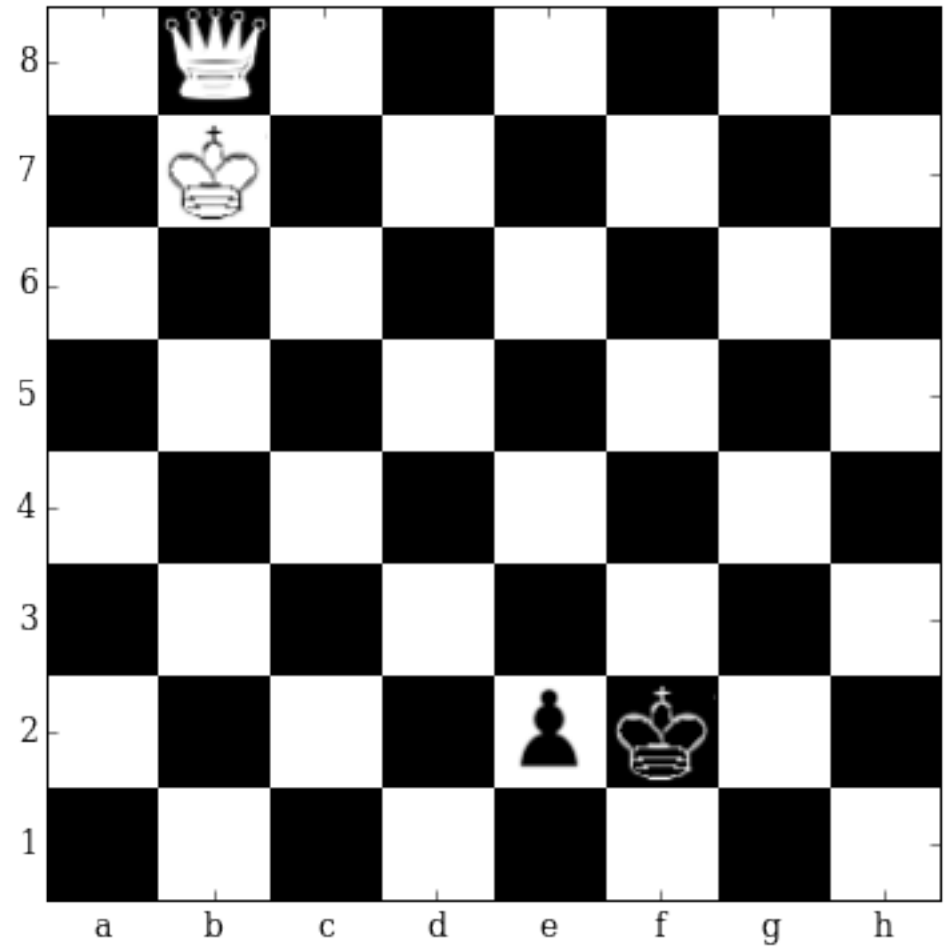

```
b8h8 0.205039218068
WORST 5 Moves
```



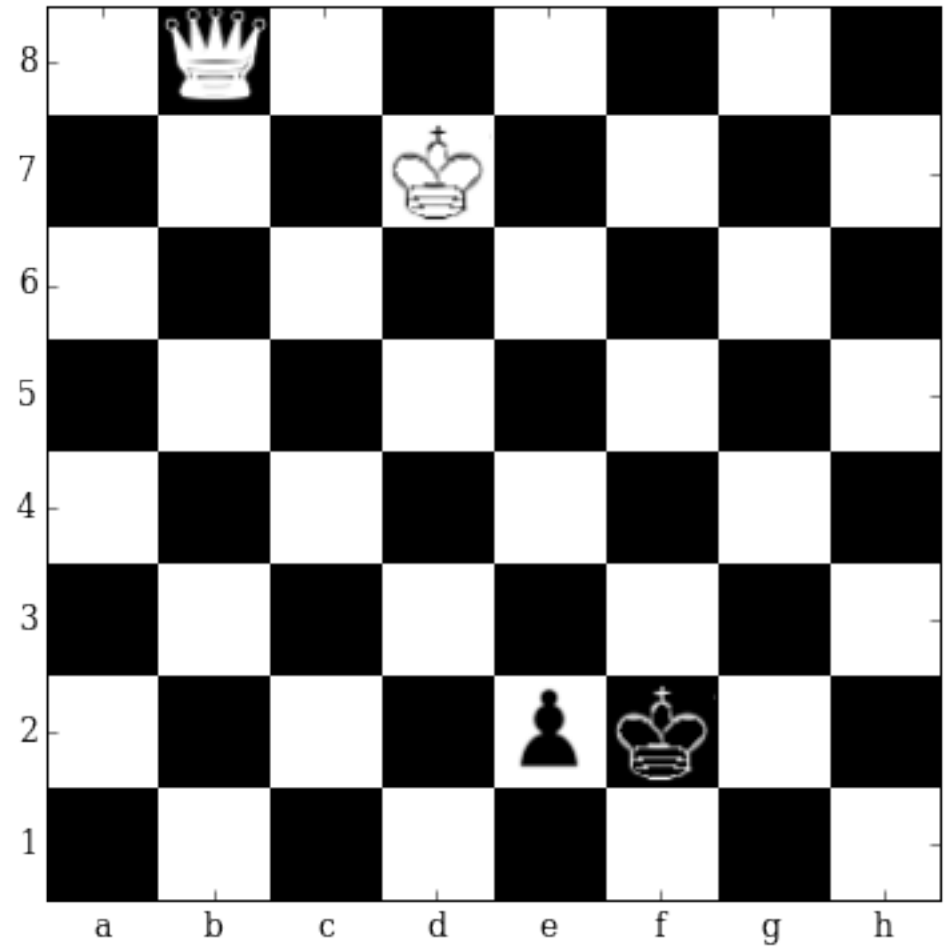

```
b8c8 0.131534919143
```



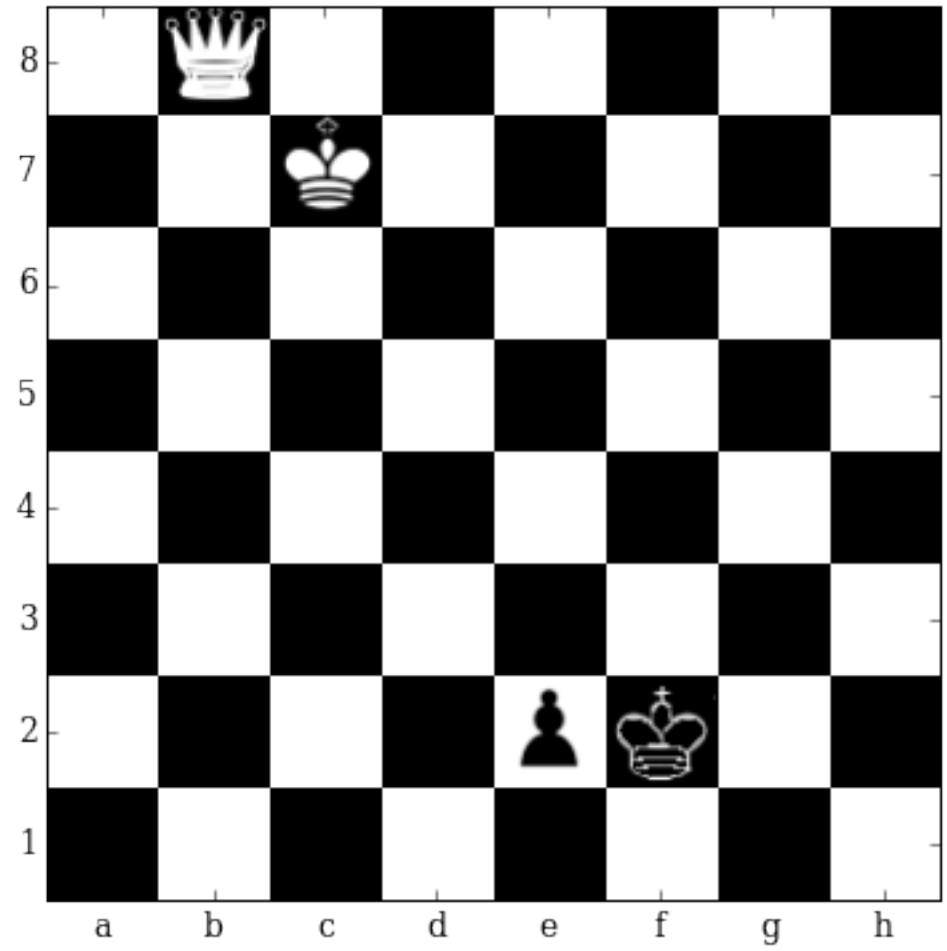

```
c6d6 0.129832103848
```




```
c6b7 0.121259056032
```

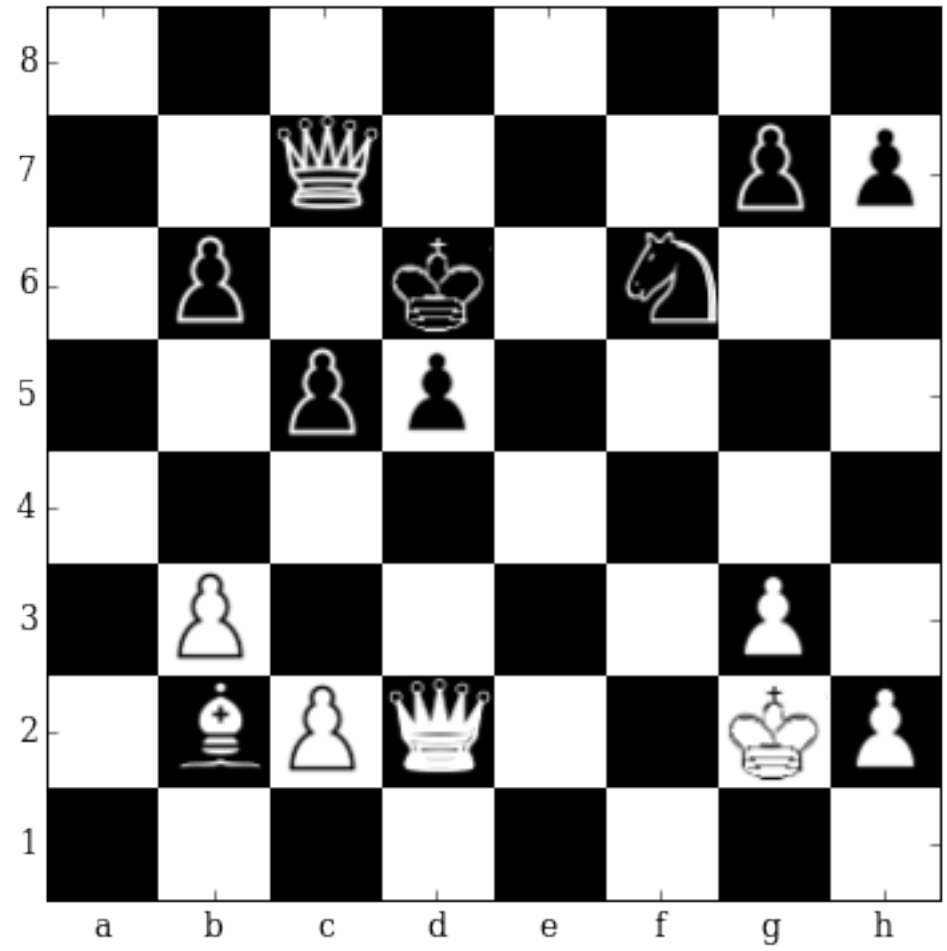



```
c6d7 0.109470196068
```

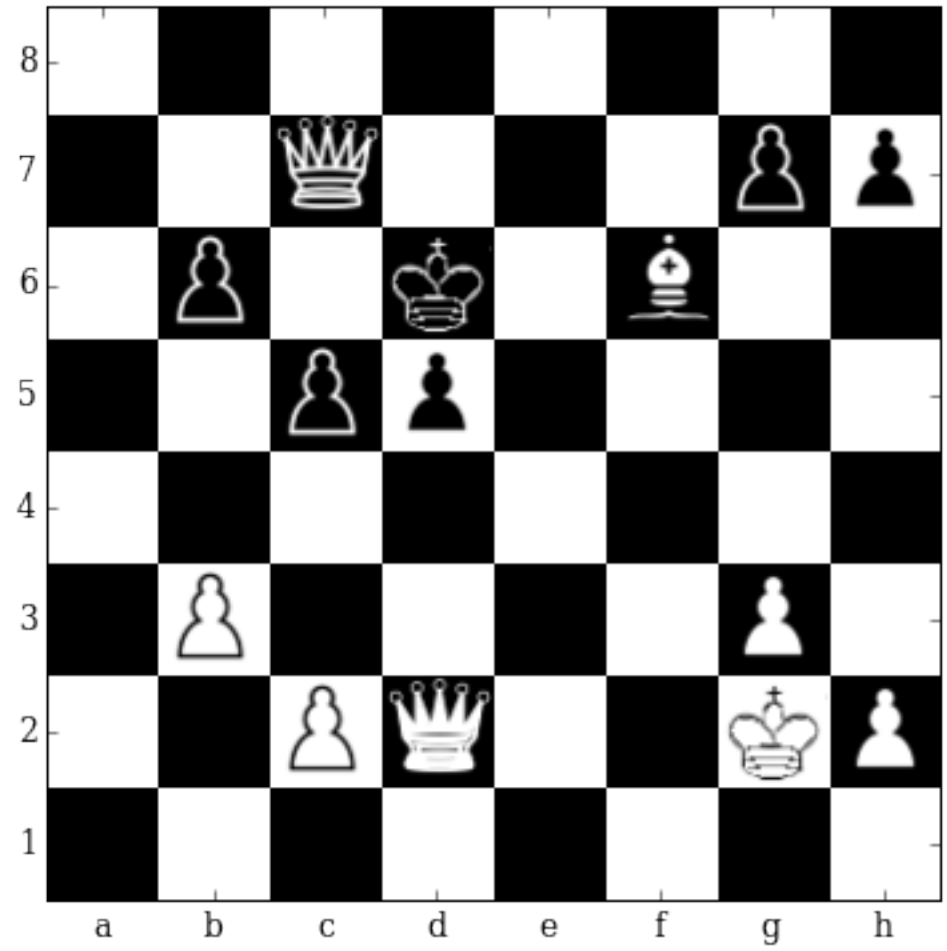



```
c6c7 0.105199843645
```

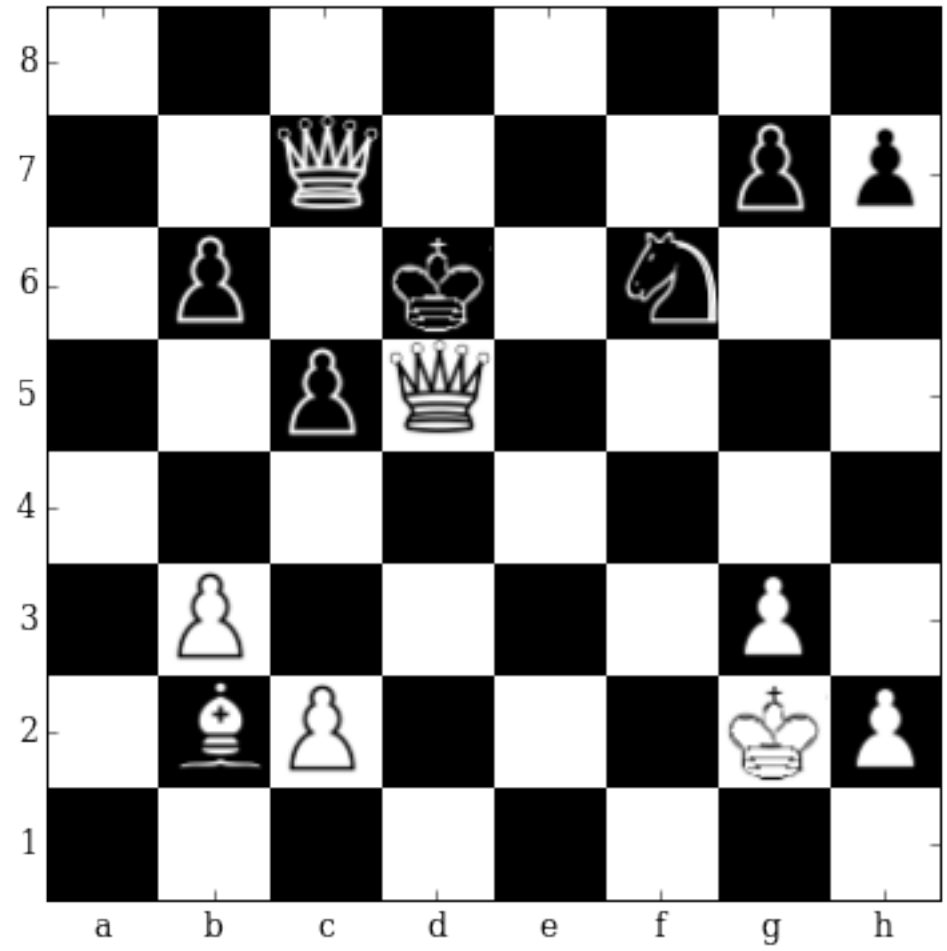
```
In [22]: fen='8/2q3pp/1p1k1n2/2pp4/8/1P4P1/1BPQ2KP/8 w - - 0 0'
         top_bottom_moves(fen)
```



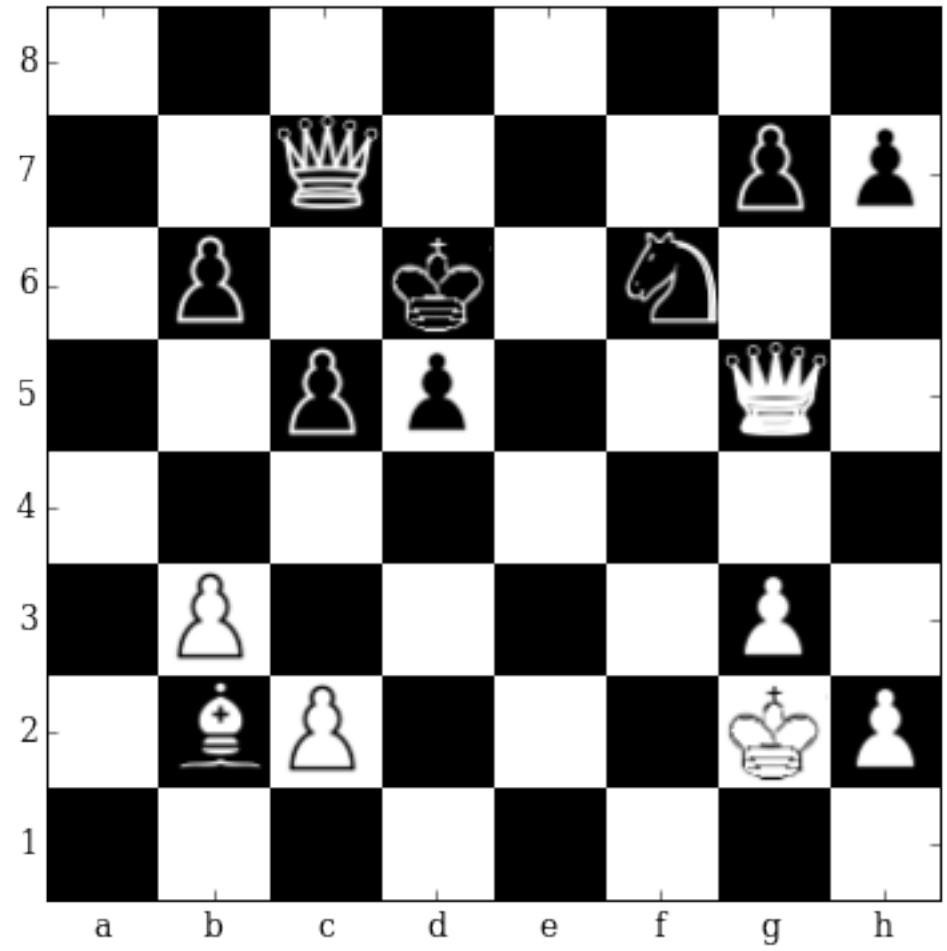

```
Current evaluation: -0.022609
Total 34 moves possible
TOP 5 Moves
```



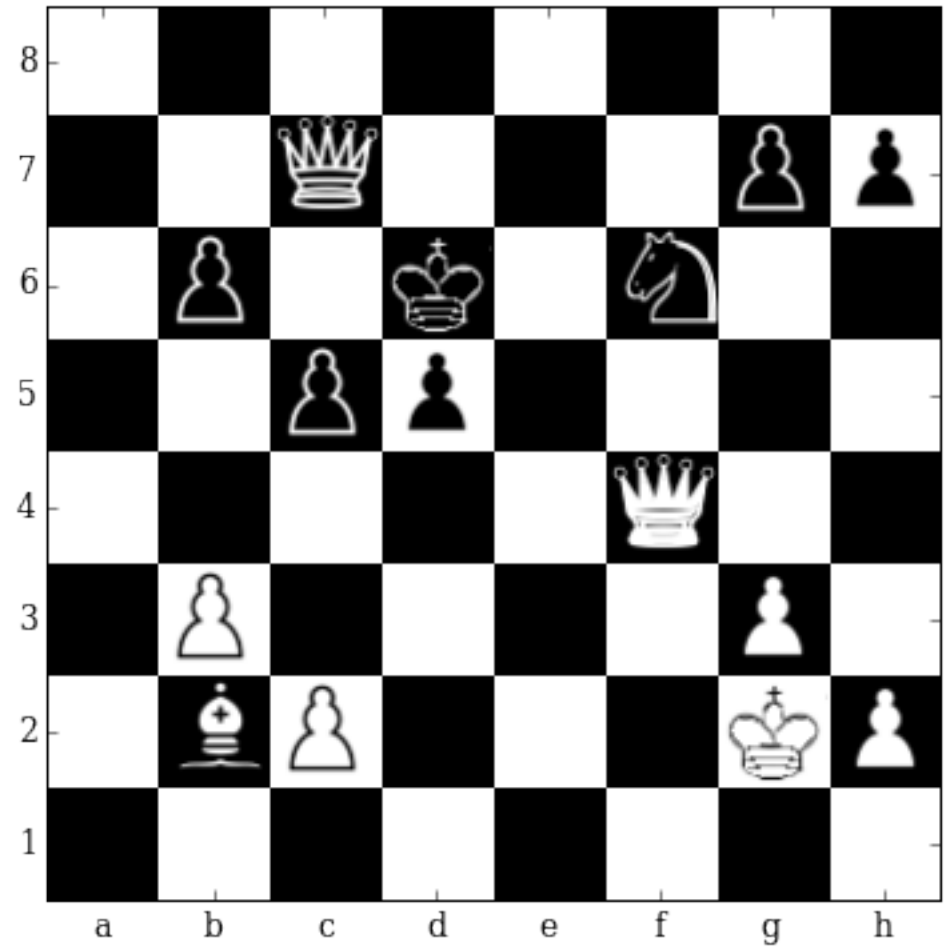
```
b2f6 0.0331075526774
```



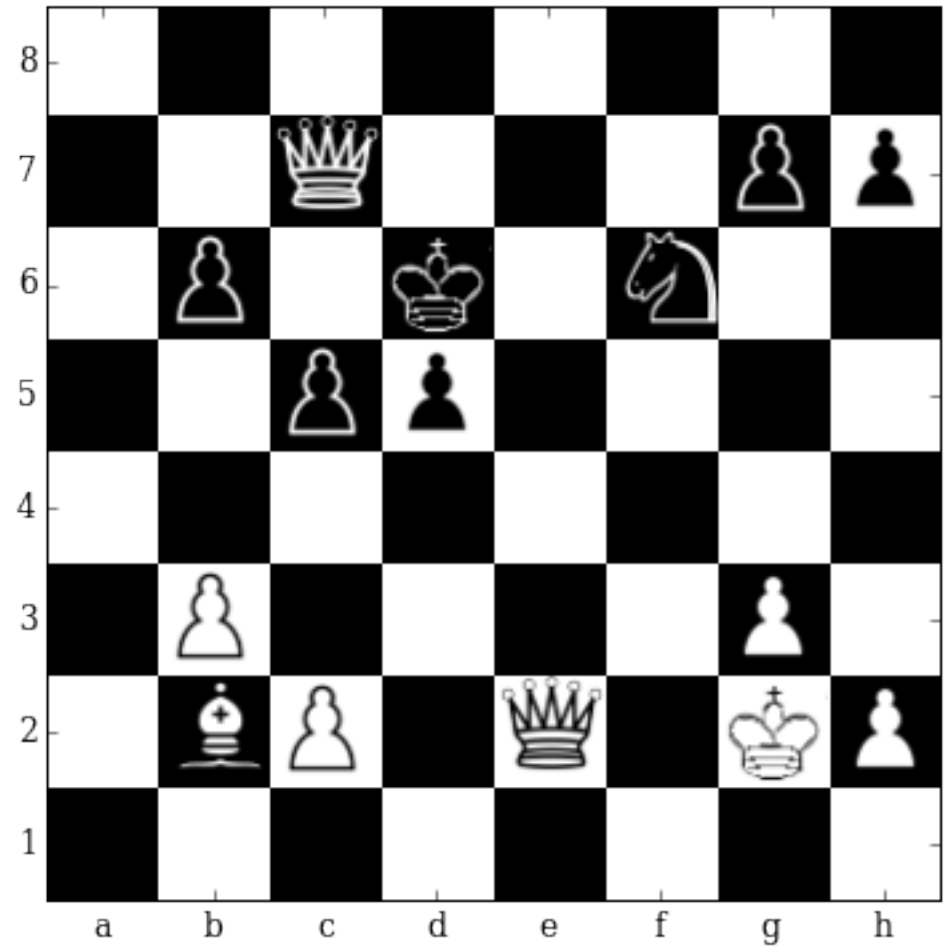

```
d2d5 0.0303655825555
```



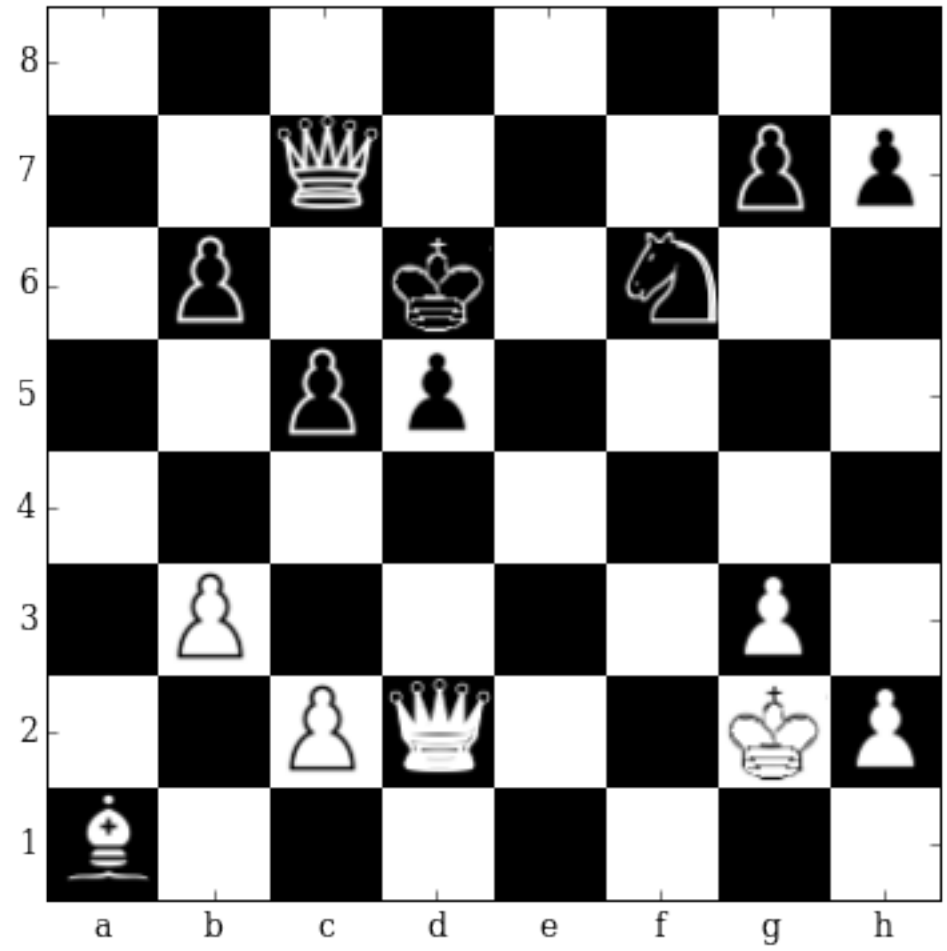

```
d2g5 0.0135372653604
```



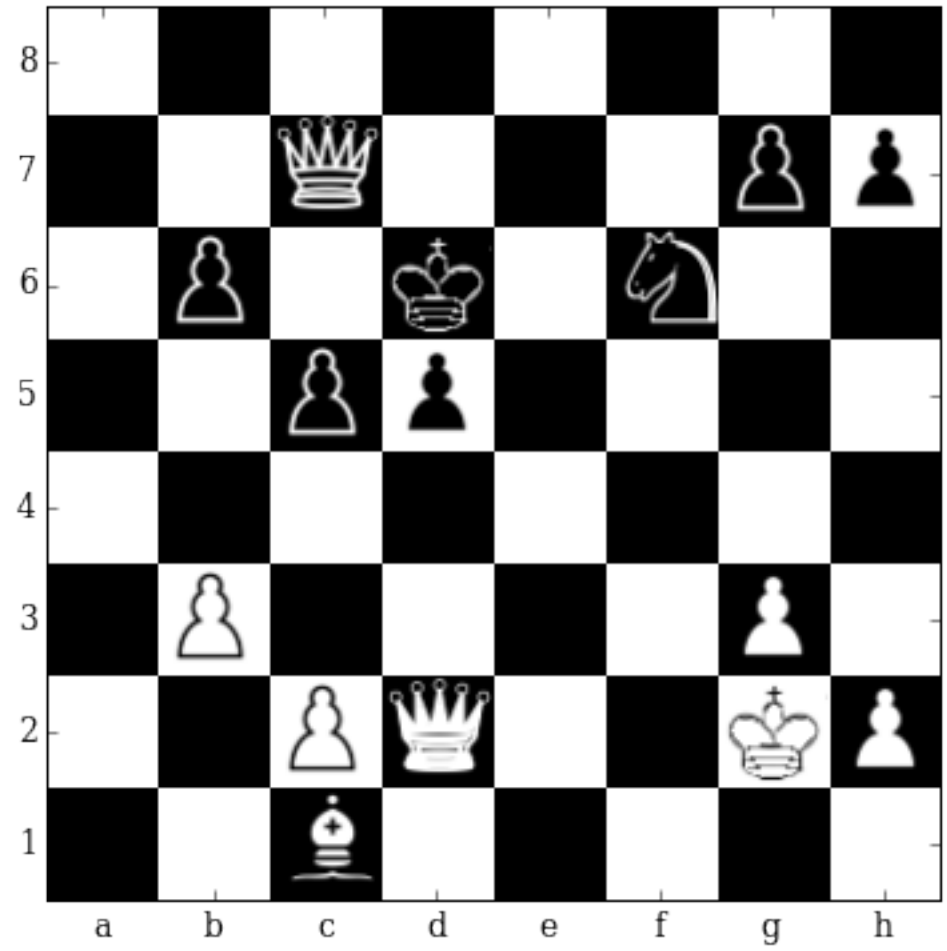

```
d2f4 0.0116871418431
```



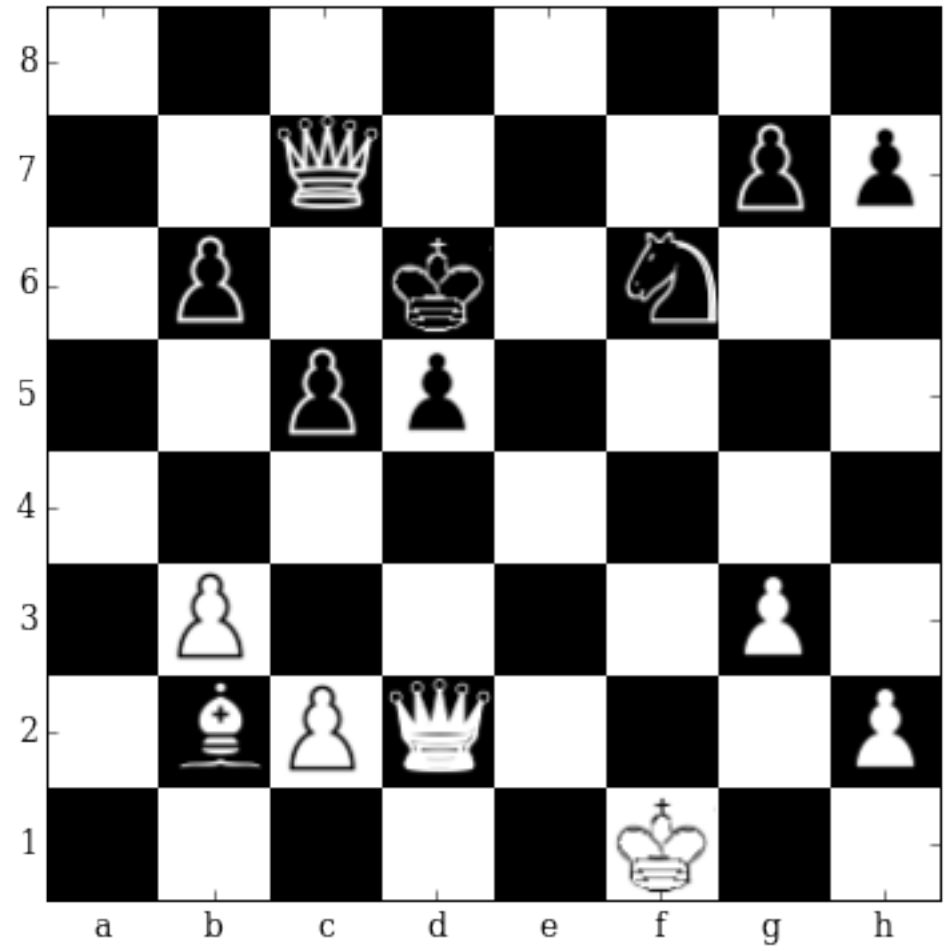

```
d2e2 0.00281526707113
WORST 5 Moves
```



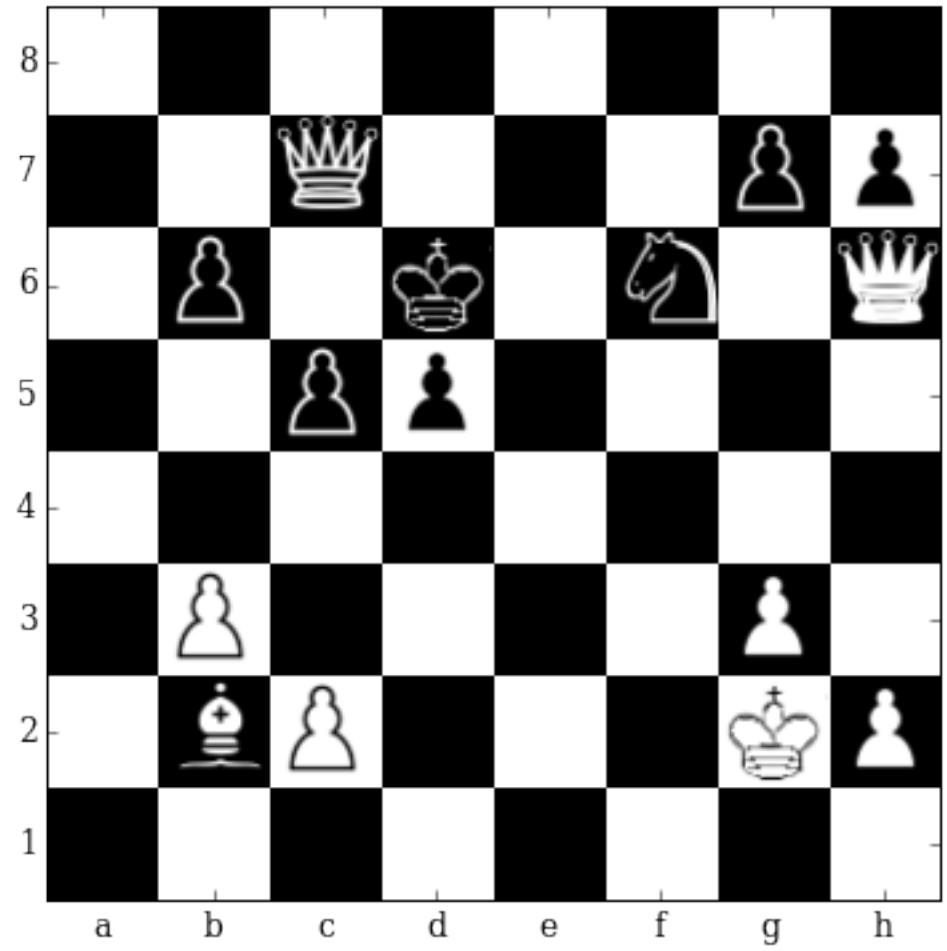

```
b2a1 -0.0309130325913
```



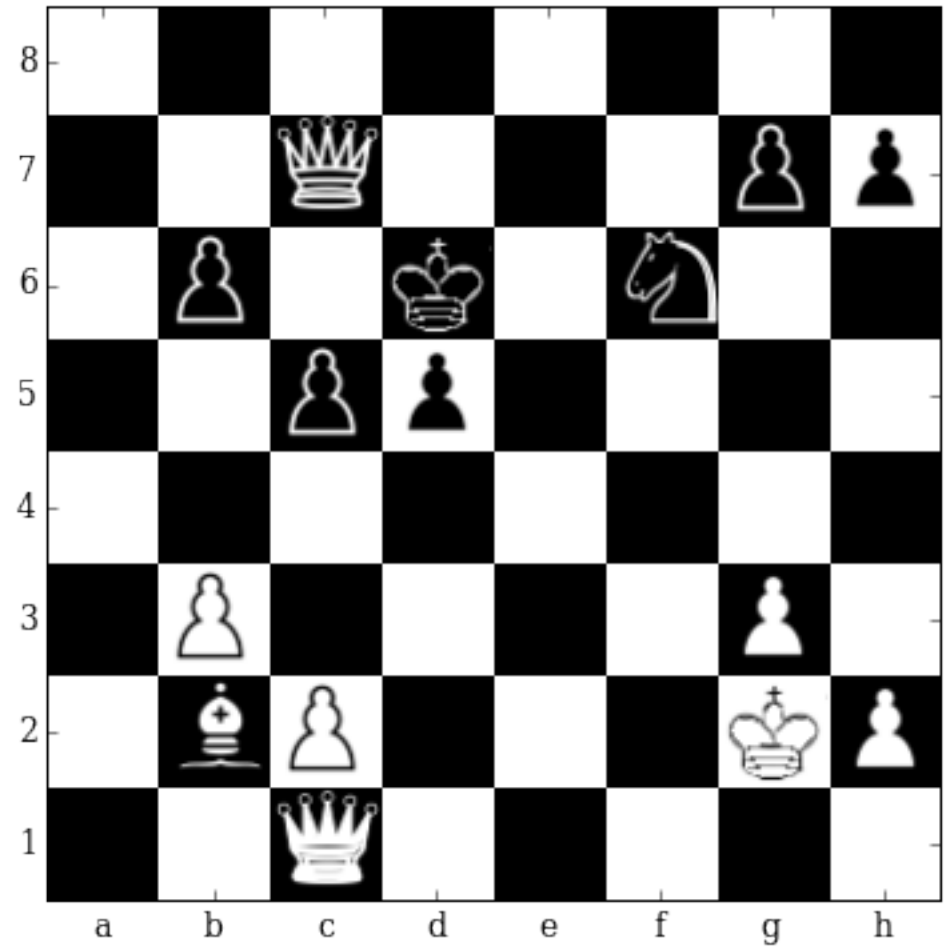

```
b2c1 -0.0322335064411
```




```
g2f1 -0.033534001559
```

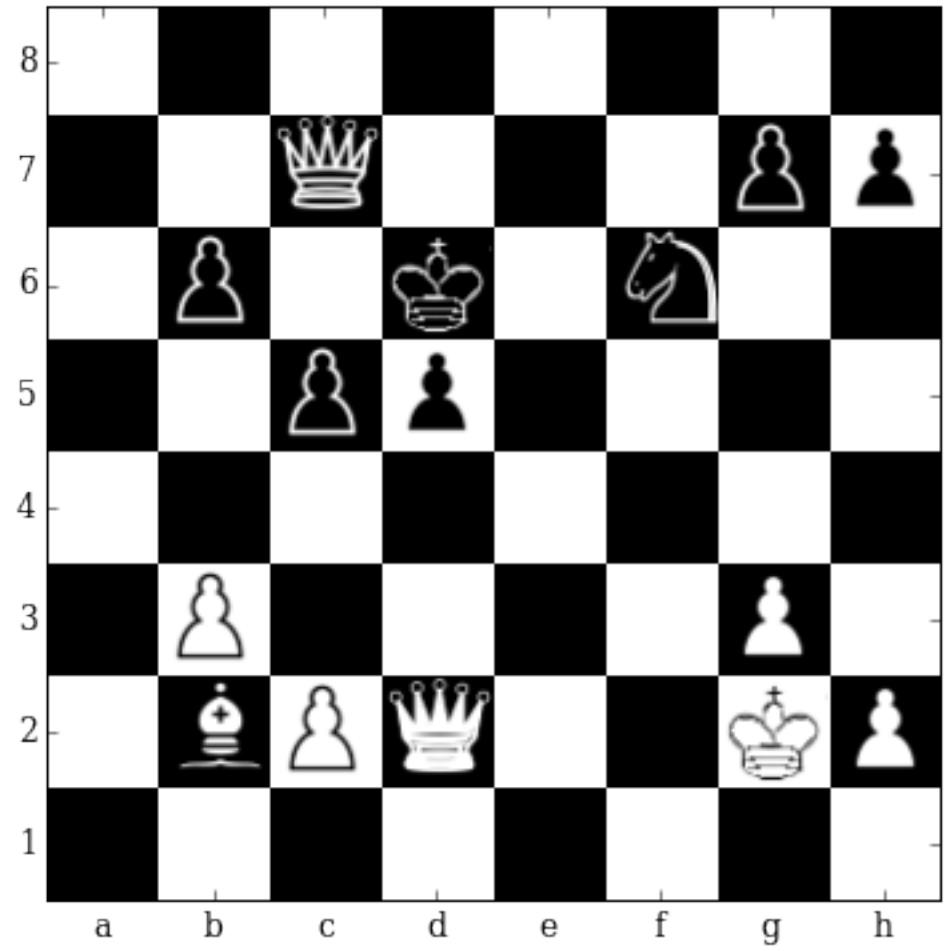



```
d2h6 -0.0337856337428
```

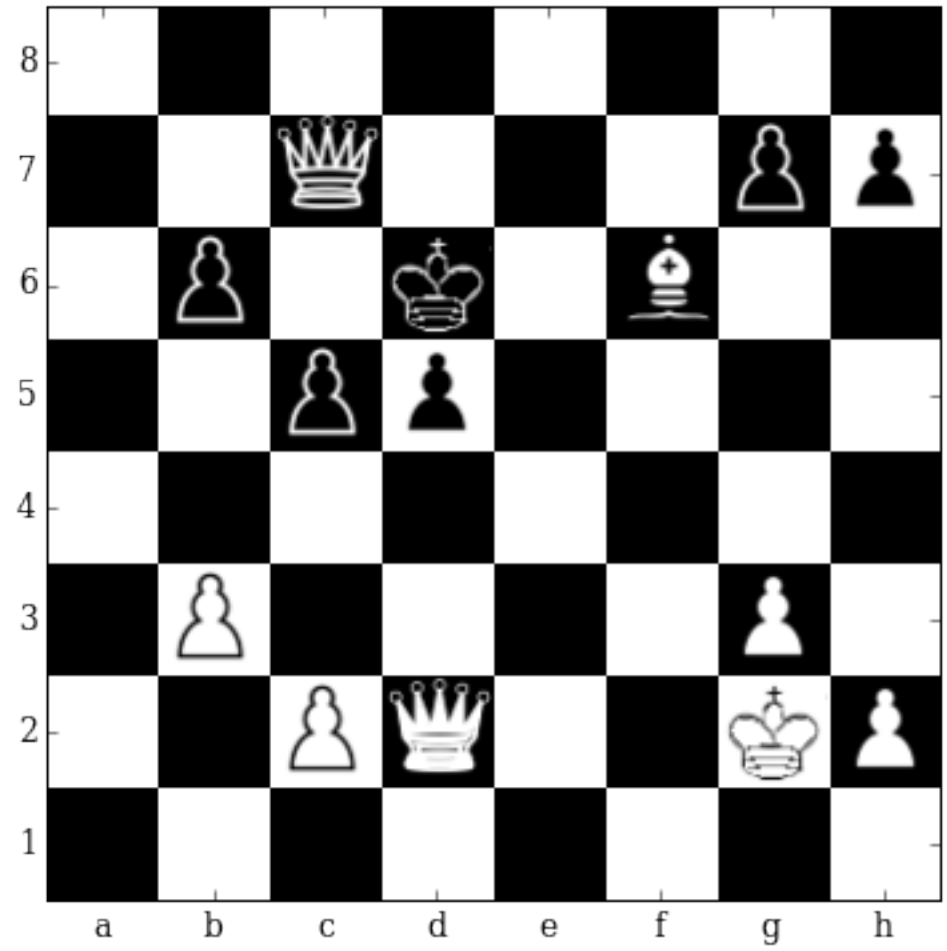



```
d2c1 -0.0525860488415
```

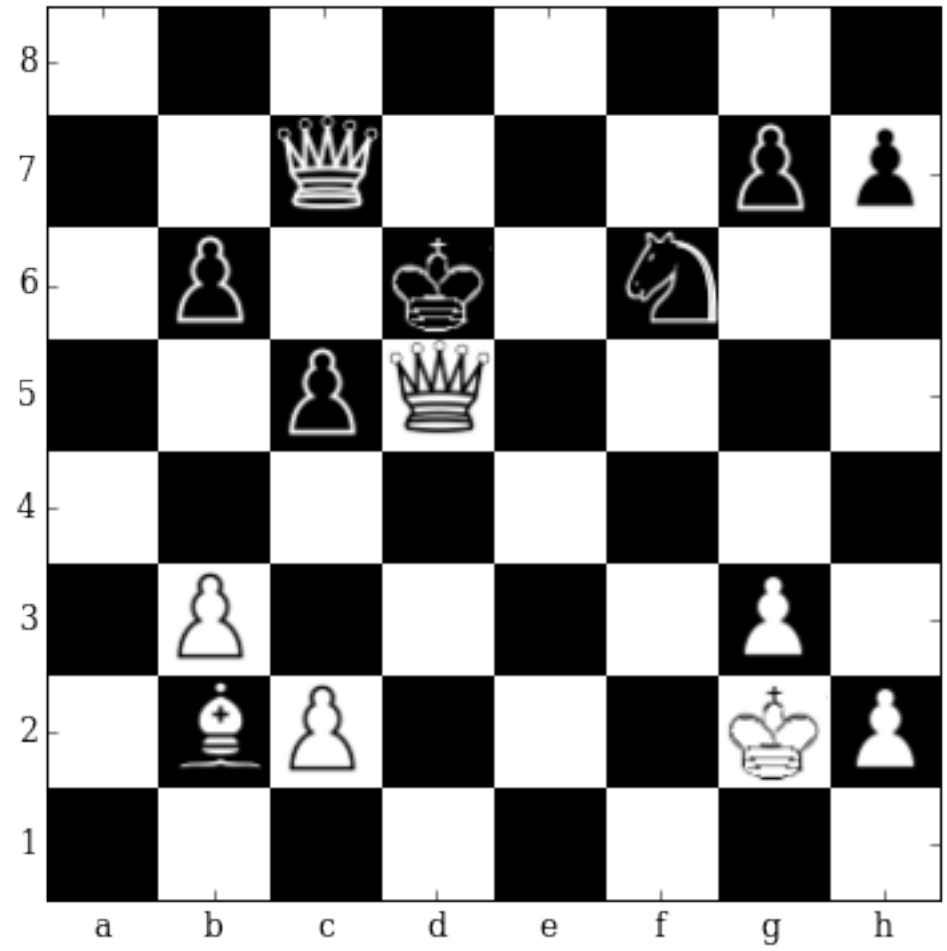
```
In [23]: top_bottom_moves(fen)
```



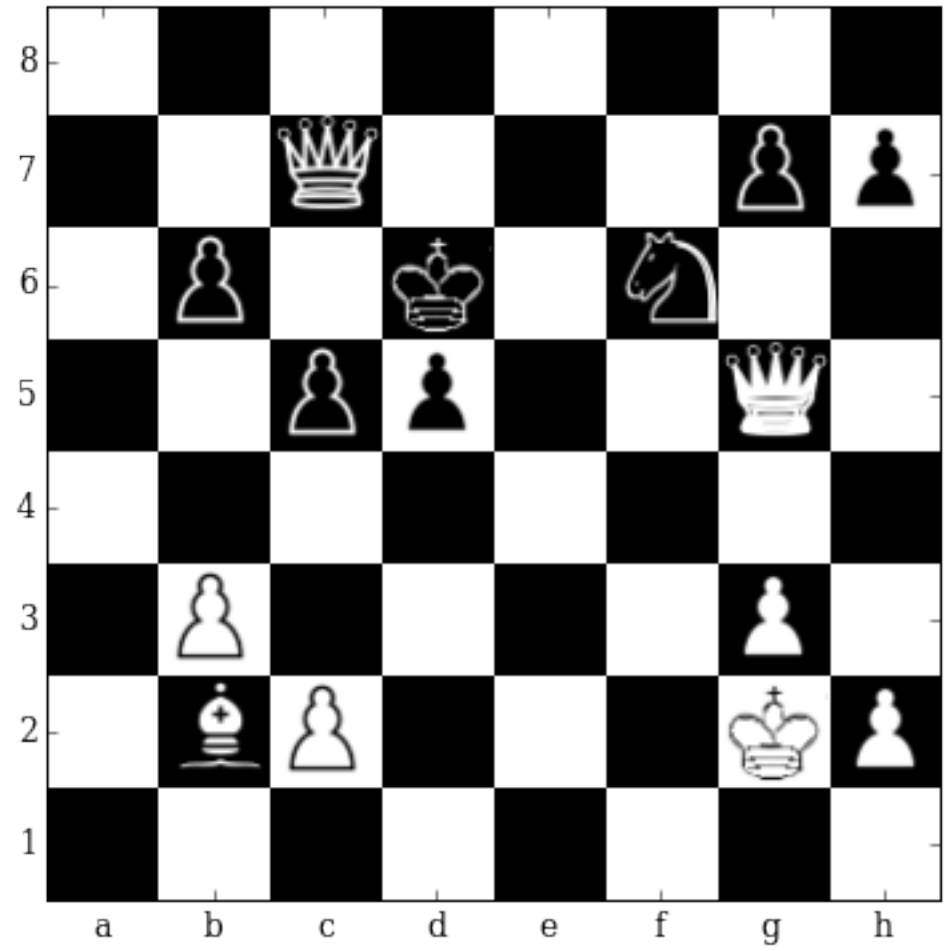

```
Current evaluation: -0.022609
Total 34 moves possible
TOP 5 Moves
```



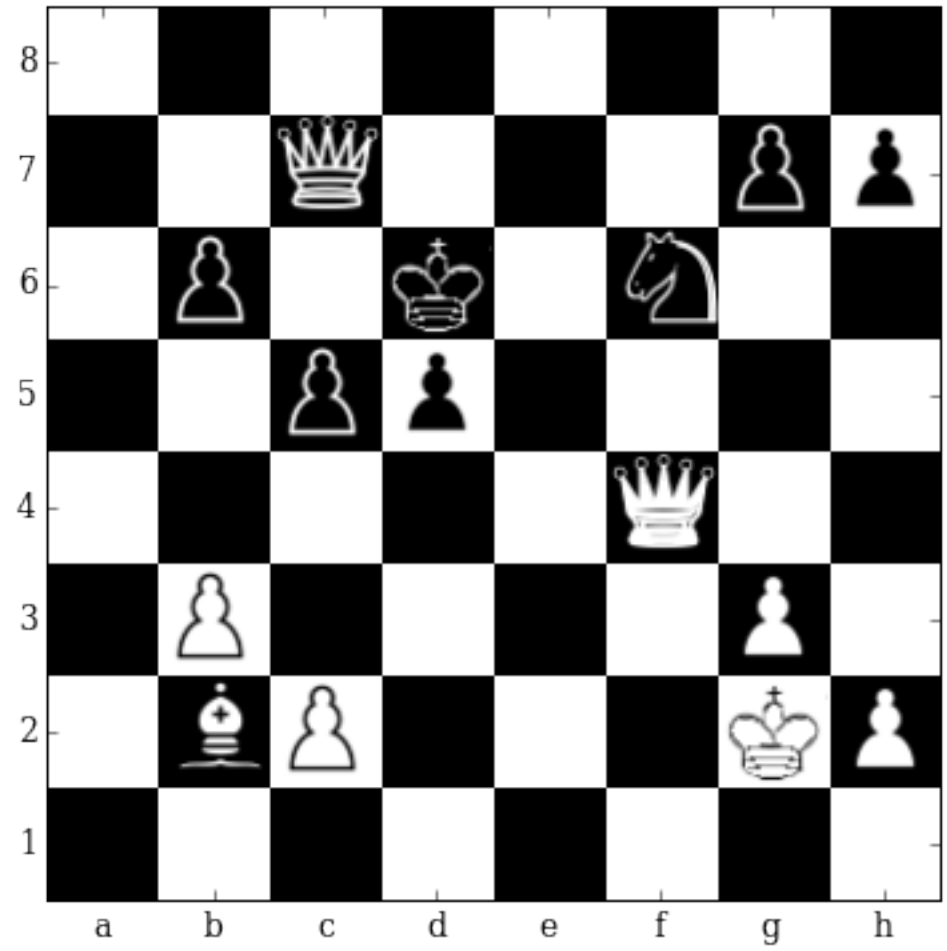

```
b2f6 0.0331075526774
```



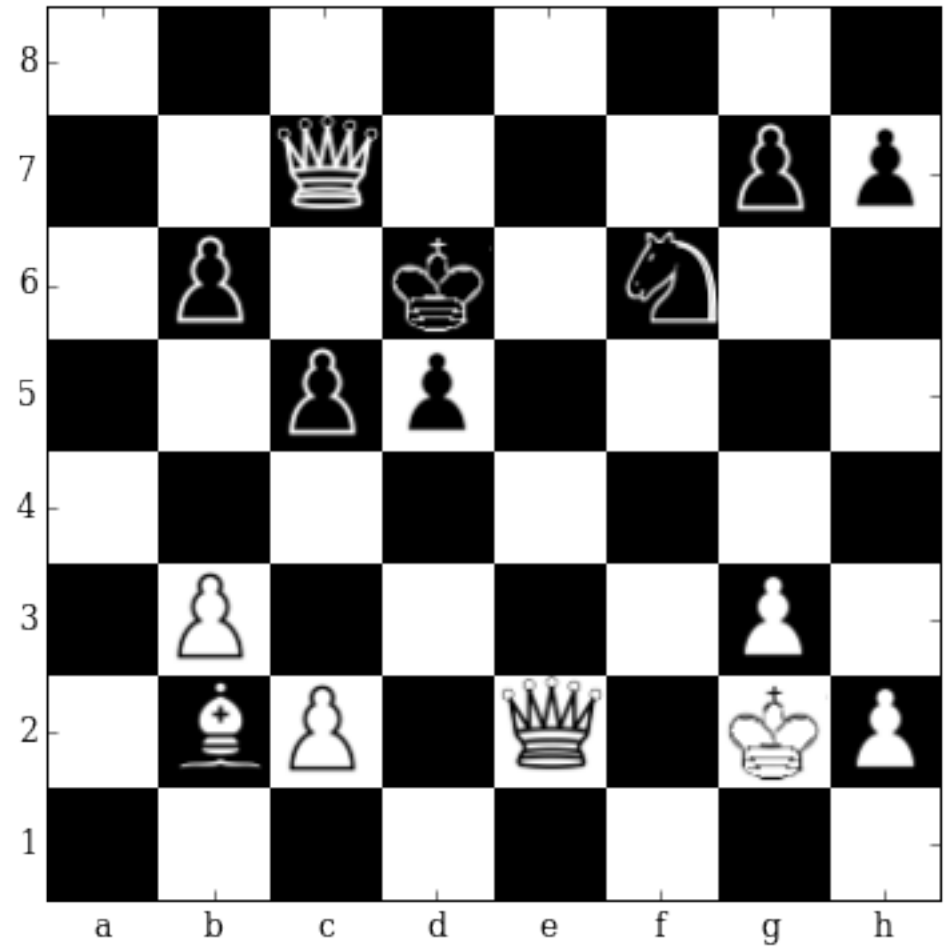

```
d2d5 0.030365575105
```



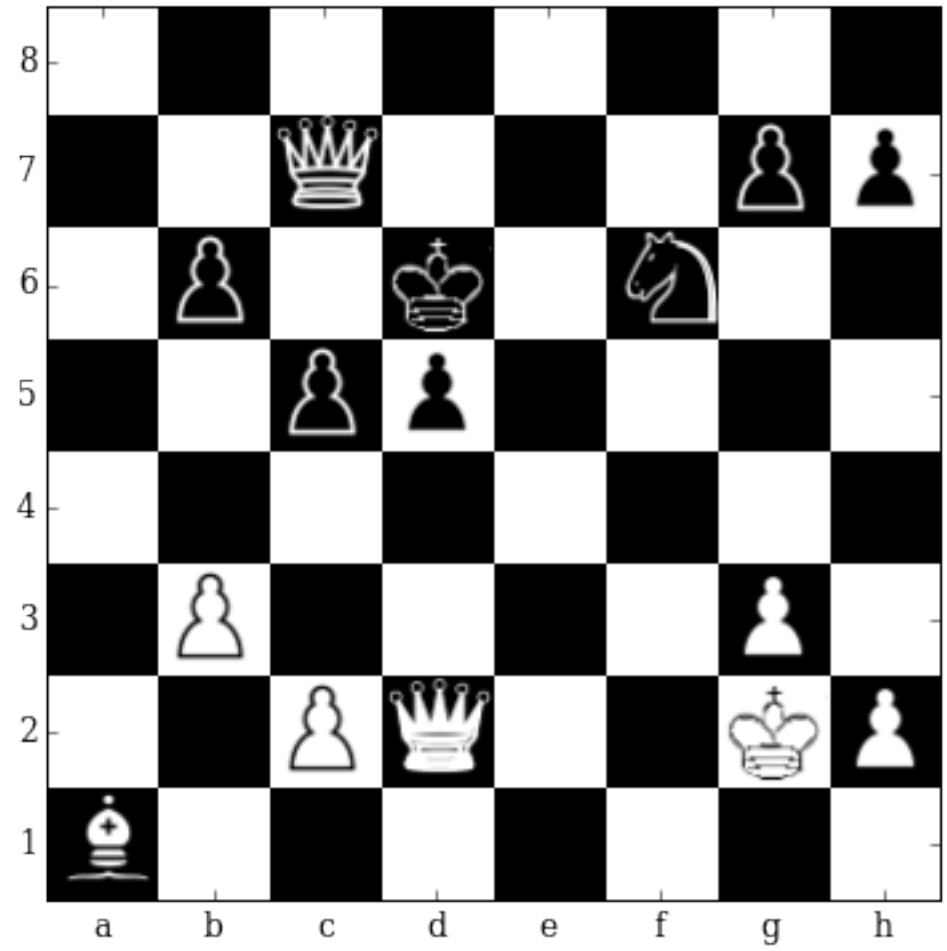
```
d2g5 0.0135372653604
```



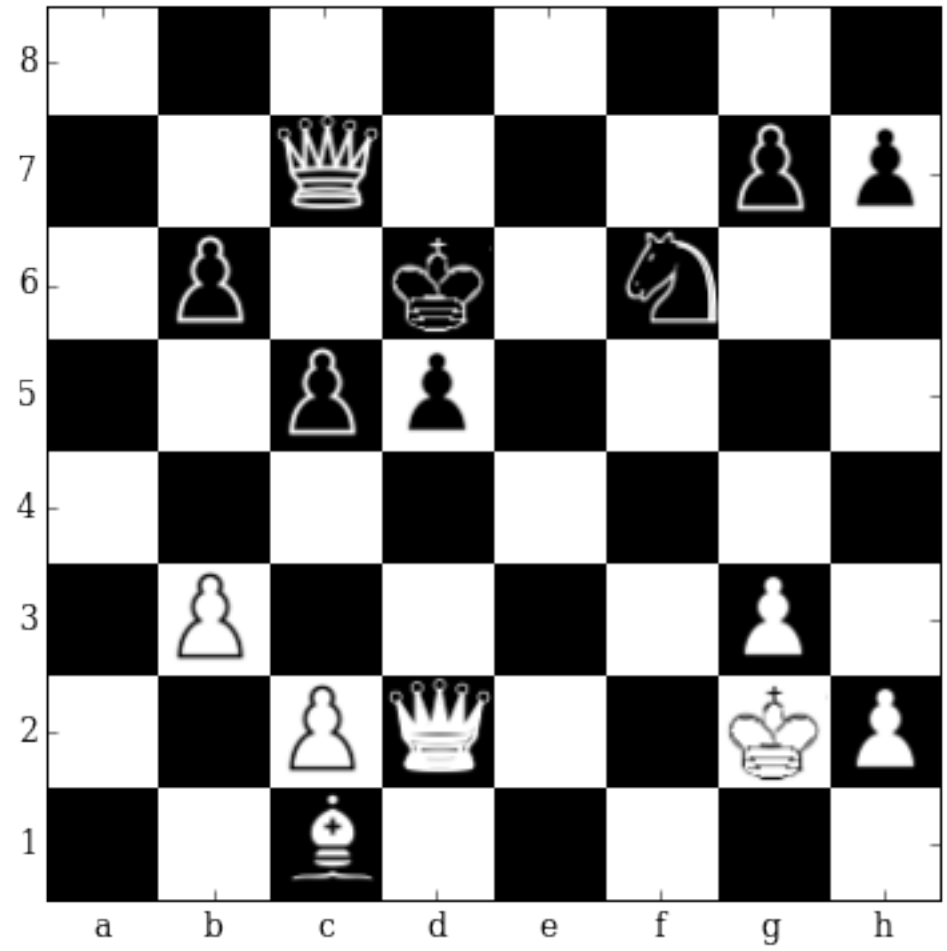

```
d2f4 0.0116871567443
```



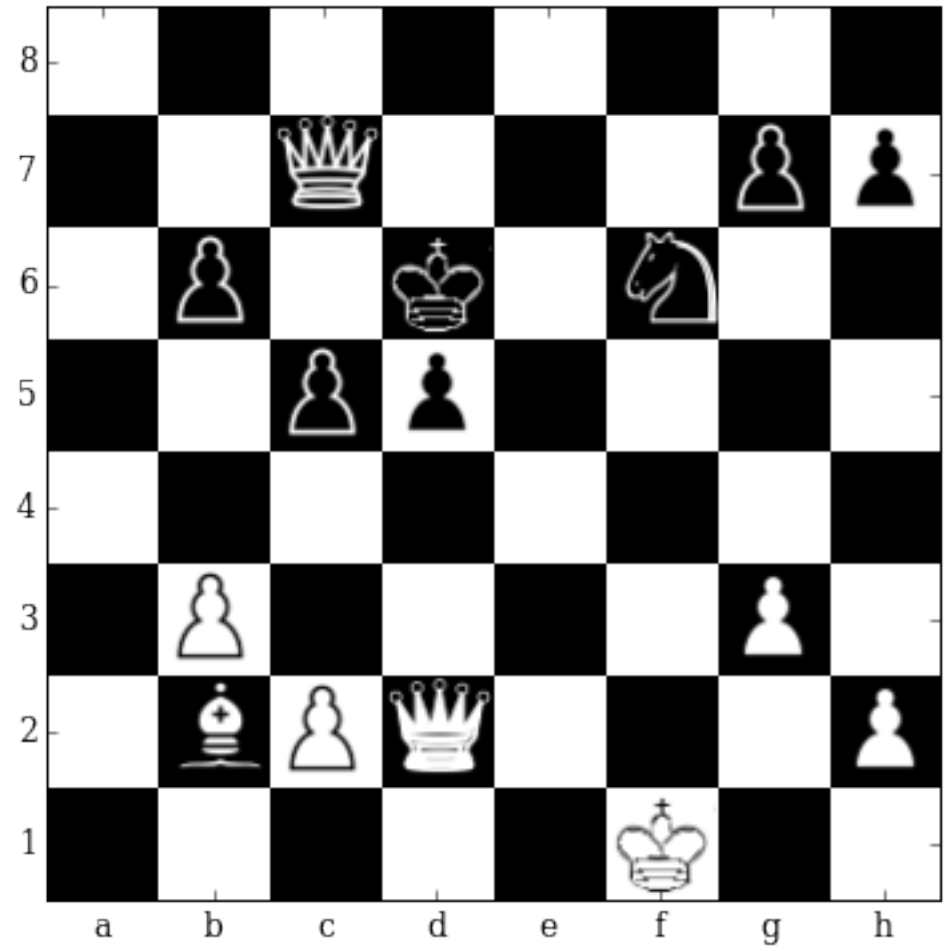

```
d2e2 0.00281526707113
WORST 5 Moves
```



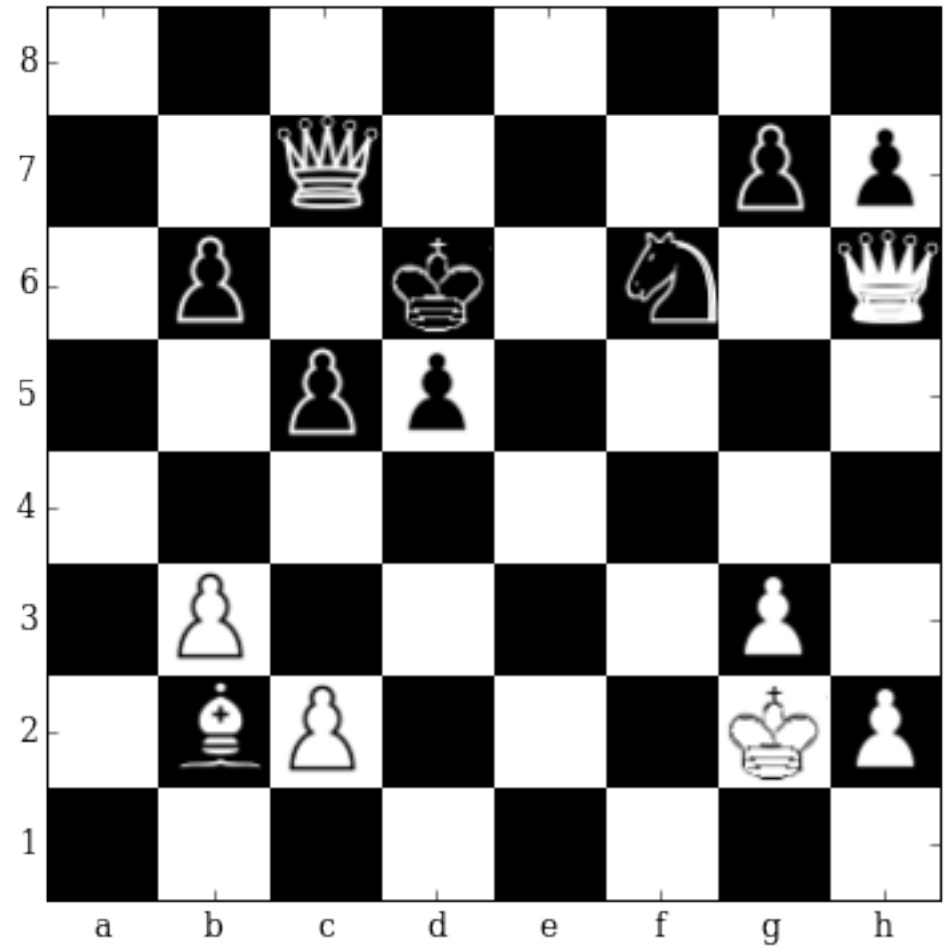

```
b2a1 -0.030913002789
```



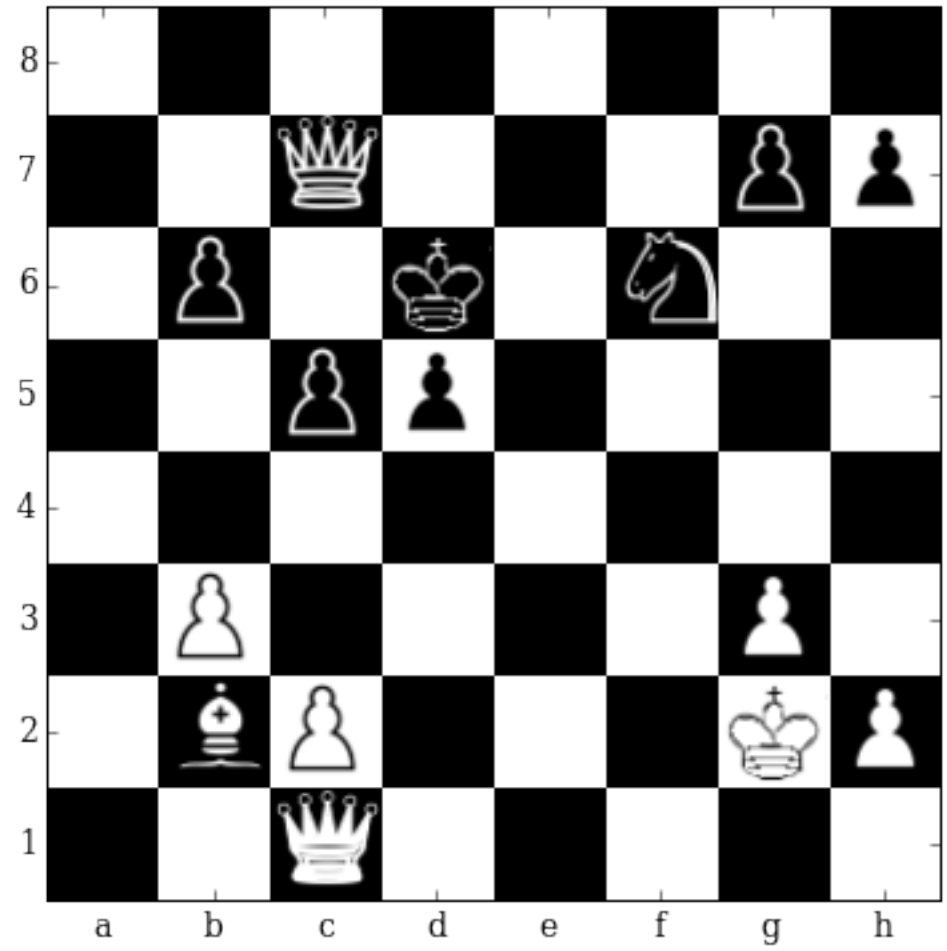

```
b2c1 -0.0322335138917
```



```
g2f1 -0.0335339568555
```

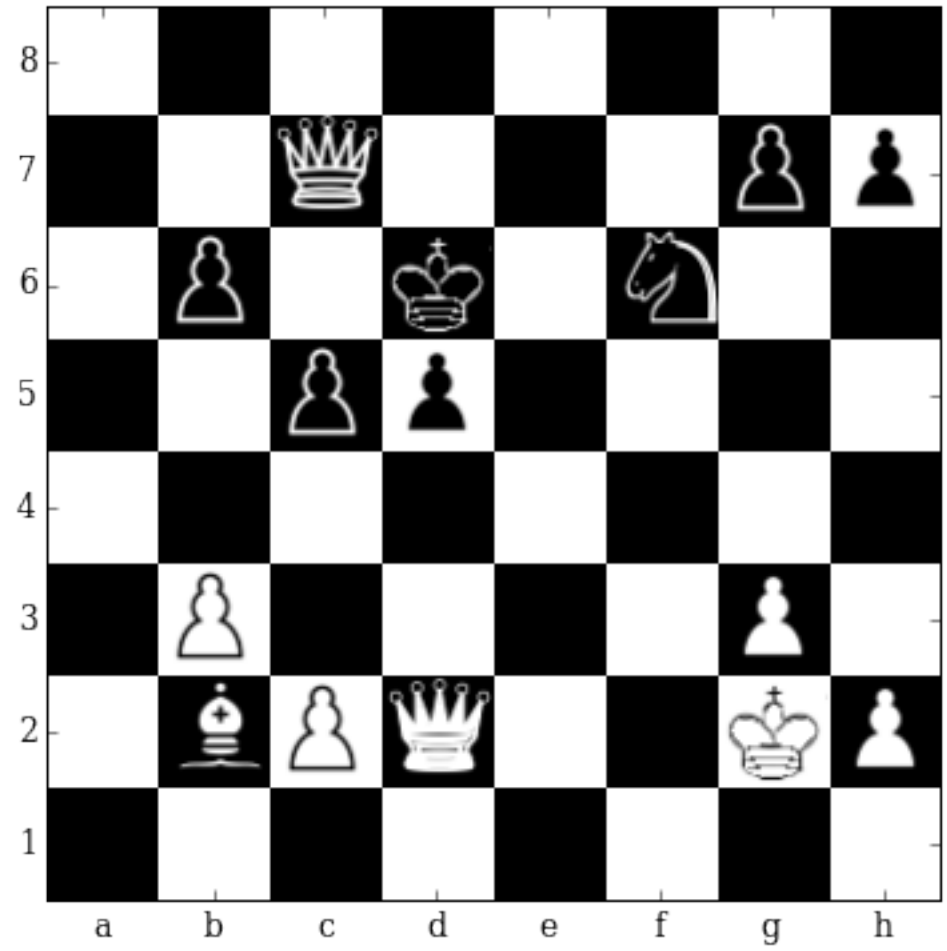



```
d2h6 -0.0337856262922
```

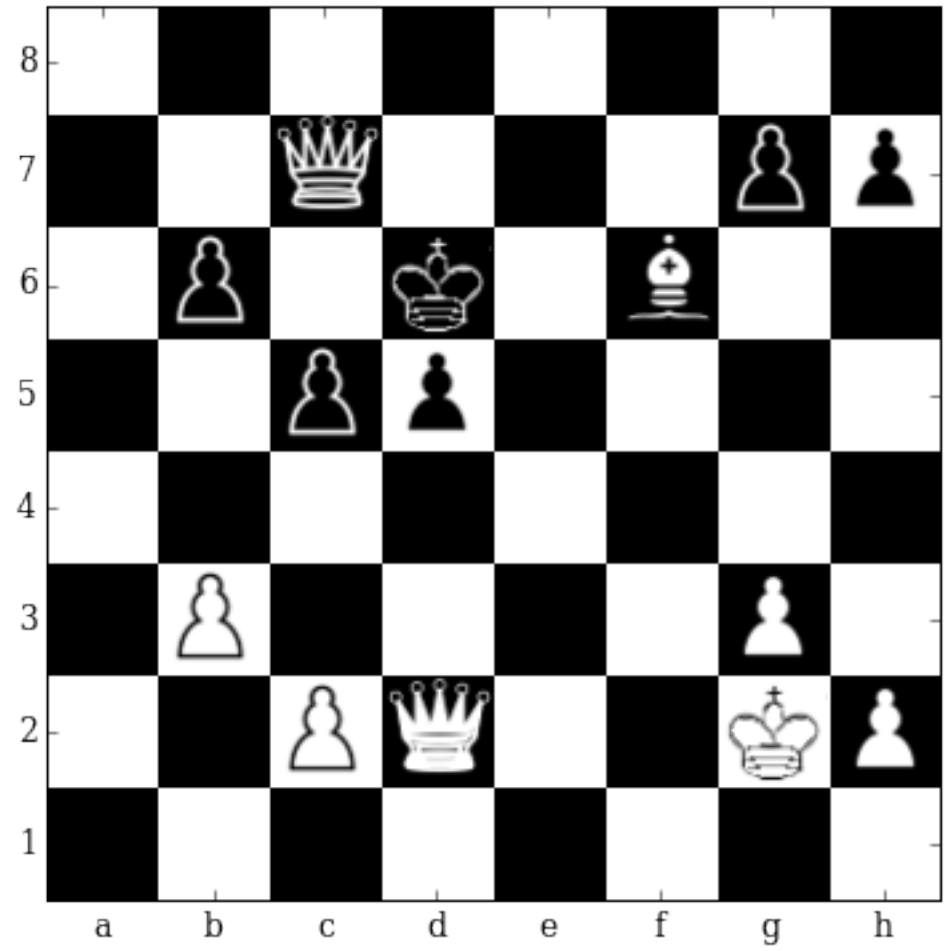



```
d2c1 -0.0525860711932
```

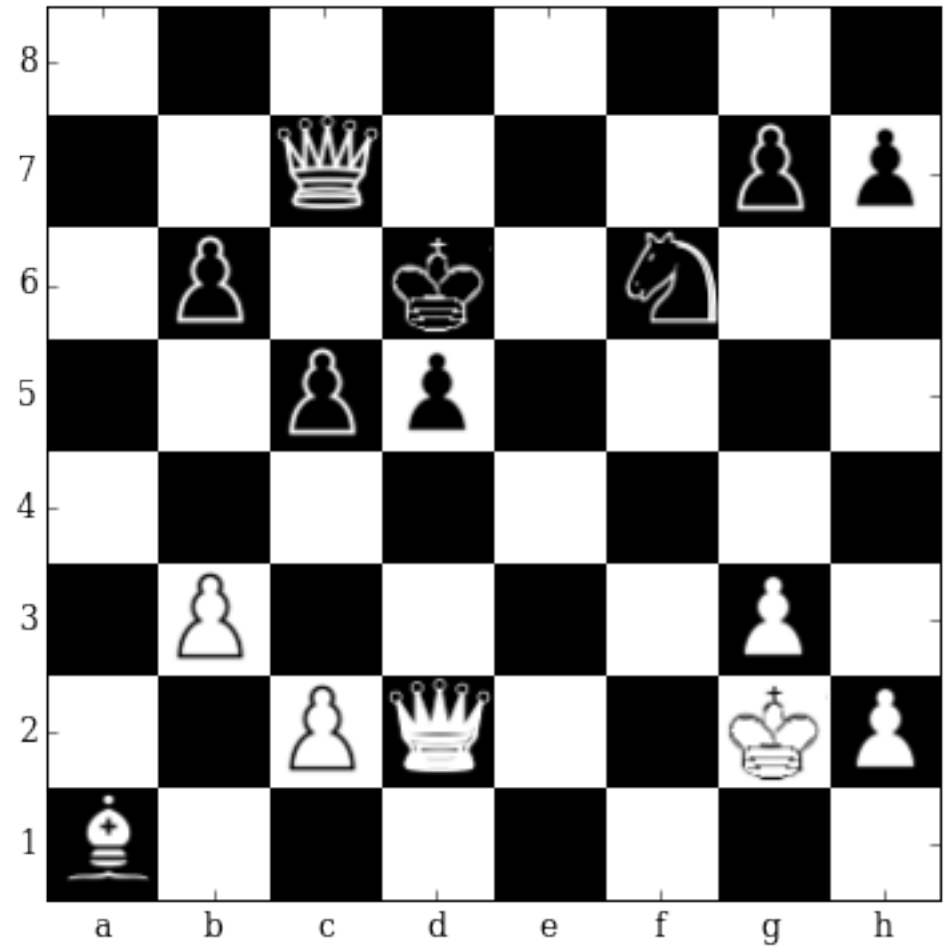
```
In [24]: fen='8/2q3pp/1p1k1n2/2pp4/8/1P4P1/1BPQ2KP/8 w - - 0 0'
         move = top_bottom_moves(fen, 1)
         pos = parseFEN(fen)
         print pos.board
         print evaluator.evaluate(np.rollaxis(convert_bitboard_to_image(pos.board), 2, 0))
         move= "c5e5"
         crdn = (sunfish.parse(move[0:2]), sunfish.parse(move[2:4]))
         print pos.move(crdn).rotate().board
         print evaluator.evaluate(np.rollaxis(convert_bitboard_to_image(pos.move(crdn).rotate().board),
         print pos.move(crdn).board
         print evaluator.evaluate(np.rollaxis(convert_bitboard_to_image(pos.move(crdn).board), 2, 0))
```



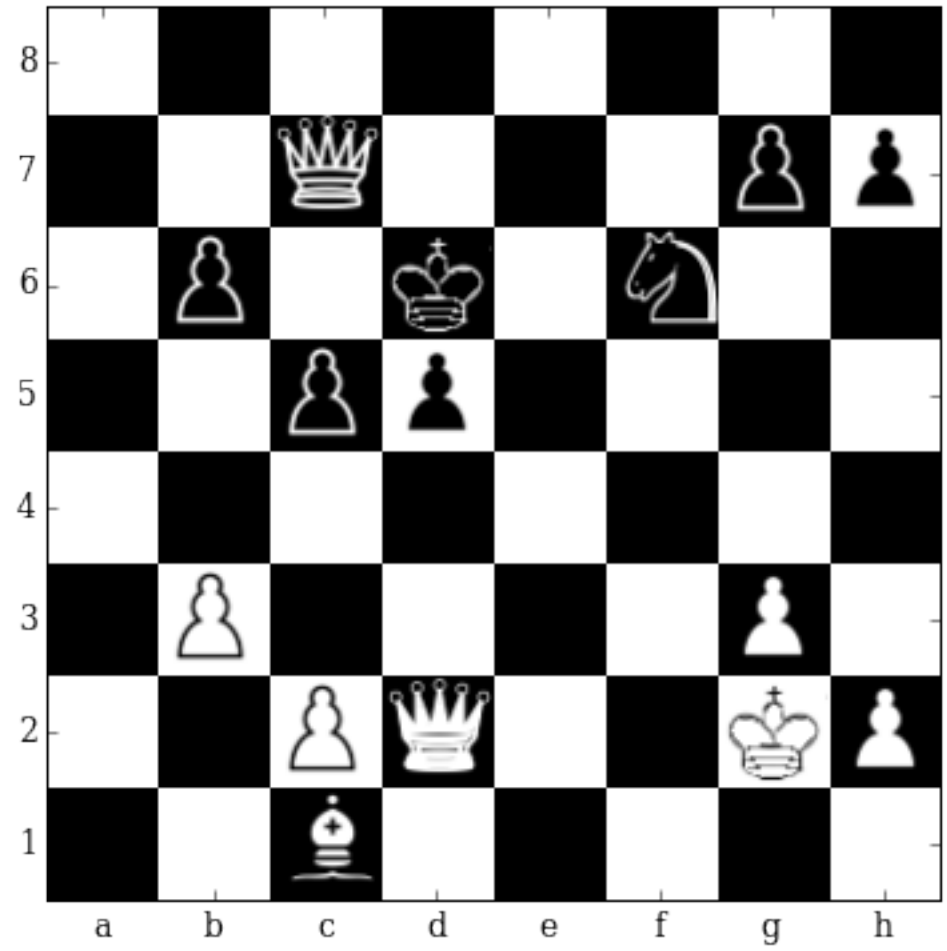

```
Current evaluation: -0.022609
Total 34 moves possible
TOP 5 Moves
```



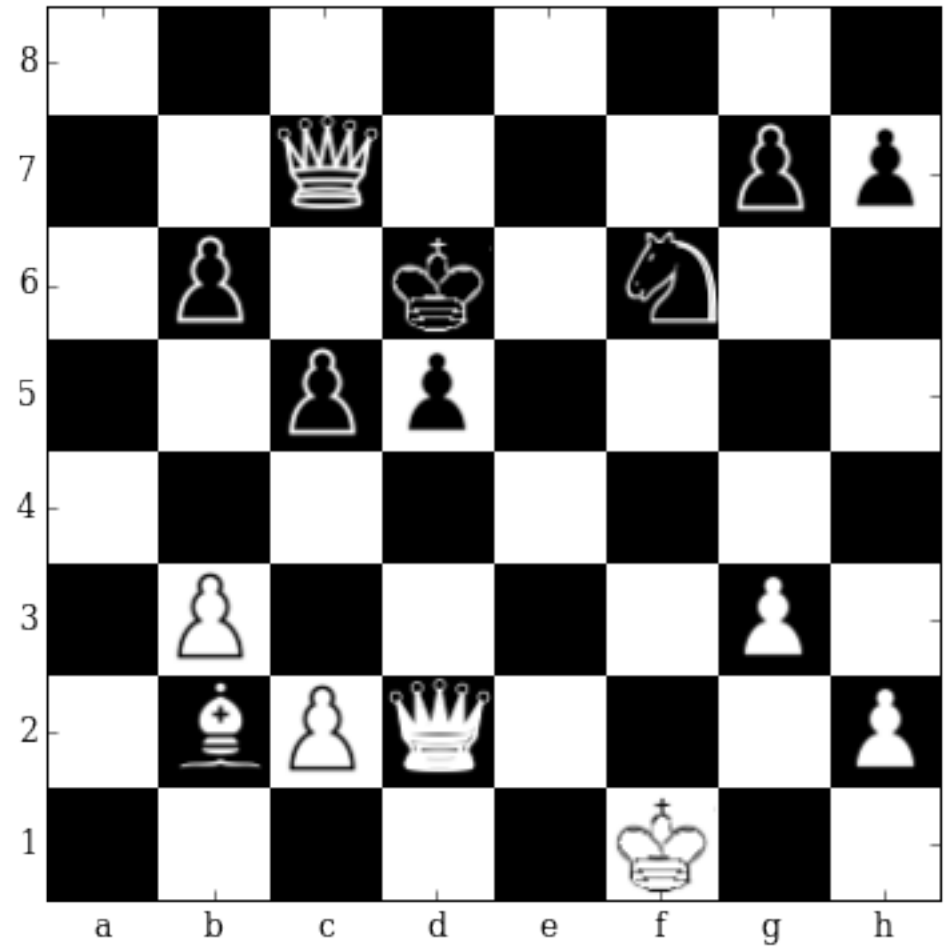

```
b2f6 0.0331075526774
WORST 5 Moves
```



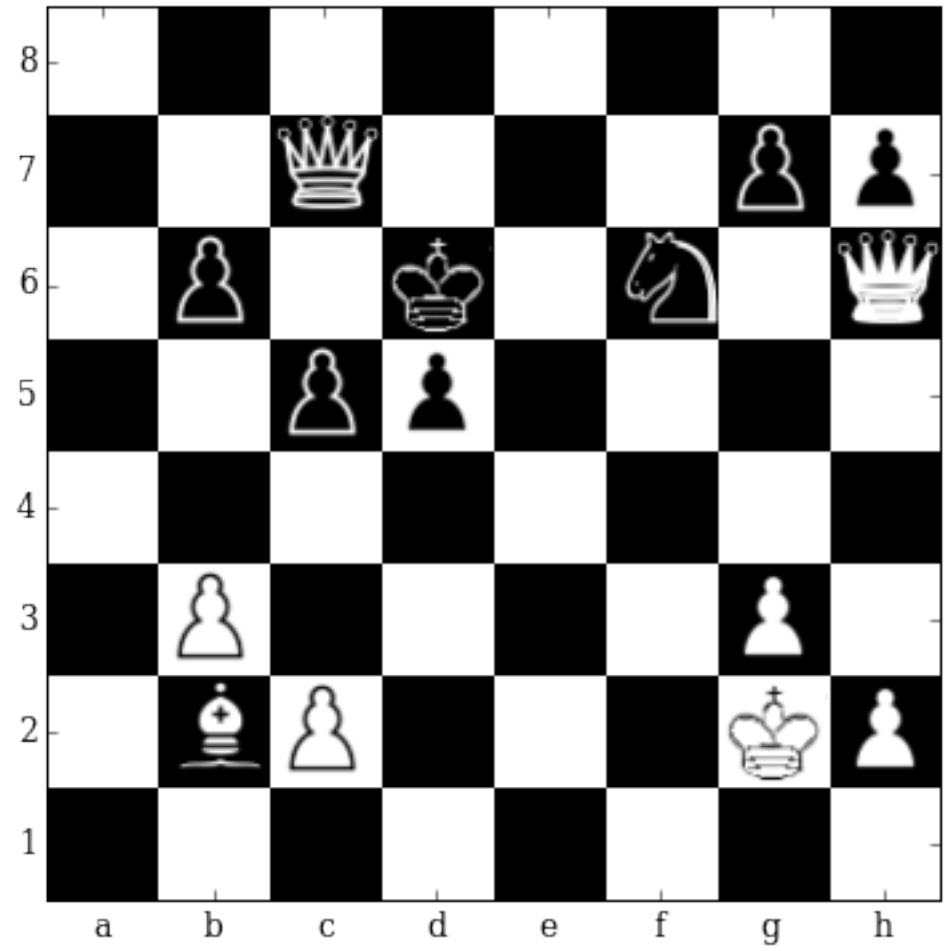

```
b2a1 -0.0309130176902
```



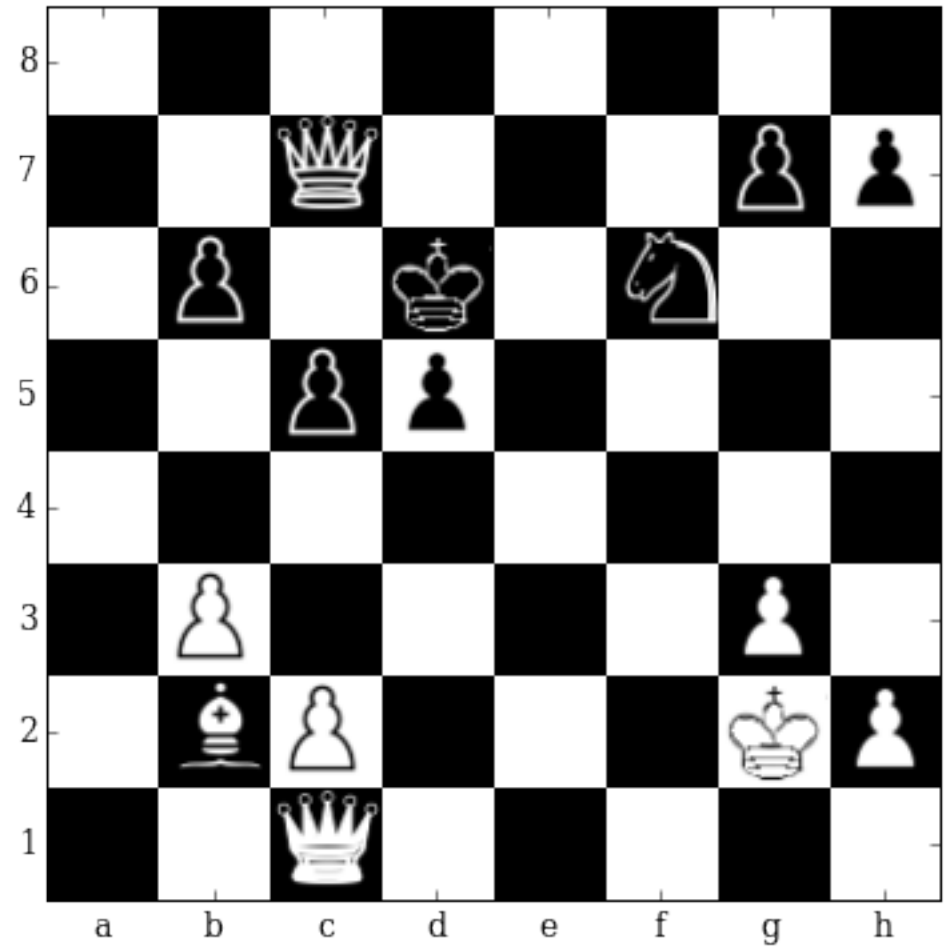

```
b2c1 -0.0322334840894
```



```
g2f1 -0.0335339941084
```




```
d2h6 -0.0337856411934
```




```
d2c1 -0.0525860711932

 ...
 ..q...pp
 .p.k.n..
 ..pp...
 ...
 .P...P.
 .BPQ..KP
 ...

-0.0226090531796
```

```
---------------------------------------------------------------------------

KeyError                                  Traceback (most recent call last)

<ipython-input-24-457a27f4848b> in <module>()
  6 move= "c5e5"
```

```
      7 crdn = (sunfish.parse(move[0:2]), sunfish.parse(move[2:4]))
----> 8 print pos.move(crdn).rotate().board
      9 print evaluator.evaluate(np.rollaxis(convert_bitboard_to_image(pos.move(crdn).rotate().board
     10 print pos.move(crdn).board


/data/sunfish/sunfish.pyc in move(self, move)
    180         board = self.board
    181         wc, bc, ep, kp = self.wc, self.bc, 0, 0
--> 182         score = self.score + self.value(move)
    183         # Actual move
    184         board = put(board, j, board[i])


/data/sunfish/sunfish.pyc in value(self, move)
    211         p, q = self.board[i], self.board[j]
    212         # Actual move
--> 213         score = pst[p][j] - pst[p][i]
    214         # Capture
    215         if q.islower():


KeyError: 'p'
```

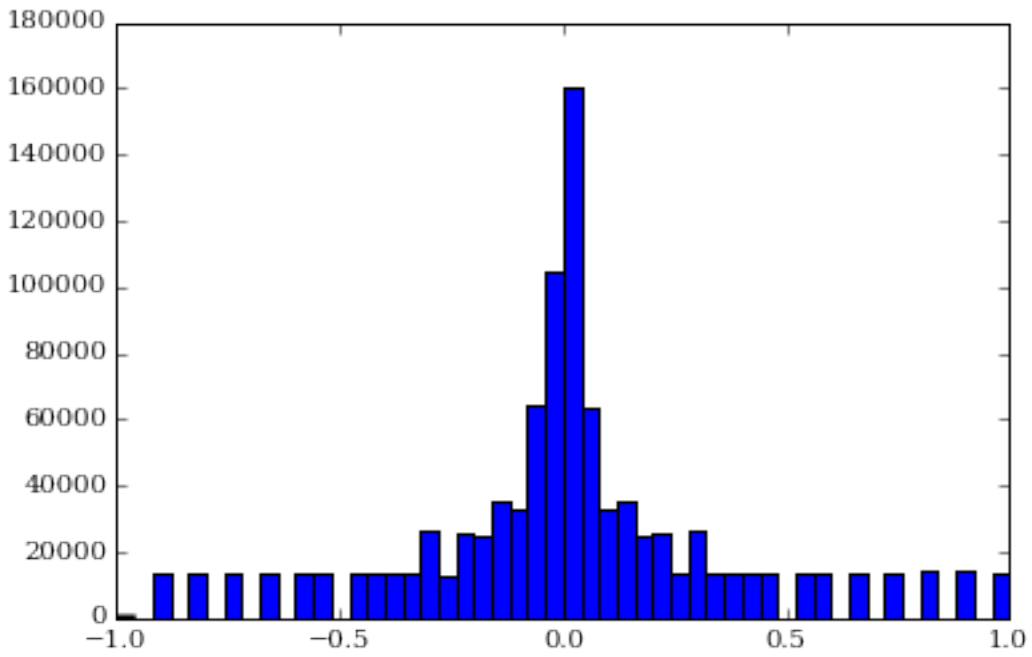
```
In [25]: import h5py as h5
         y = h5.File("/data/ConvChess/data/all_data_g09_pb_h5/piece.h5")
         y = y['label']
```

```
In [26]: y = y[0:1000000]
         print np.std(y)
         print np.var(y)
```

```
0.361267
0.130514
```

```
In [27]: #evaluator = CNN_evaluator('regression_models/model_CvC_g09.pkl')
```

```
In [30]: %matplotlib inline
         import matplotlib.cm as cm
         plt.hist(y, bins=50)
         plt.show()
```




```
In [29]: tables = np.load("/data/ConvChess/data/fics_regression_g07/X_80000_88000.npz")
         tables = tables['arr_0']
         print "Number of tables: %d"%len(tables)
         evaluations = evaluator.evaluate_batch(tables)
         tablevals = np.zeros(tables.shape[0])
         for i, table in enumerate(tables):
             tablevals[i] = im_to_tableval(table)
         print np.corrcoef(evaluations, tablevals)

Number of tables: 665727
[[ 1.          0.81938201]
 [ 0.81938201  1.        ]]

In [ ]:
```