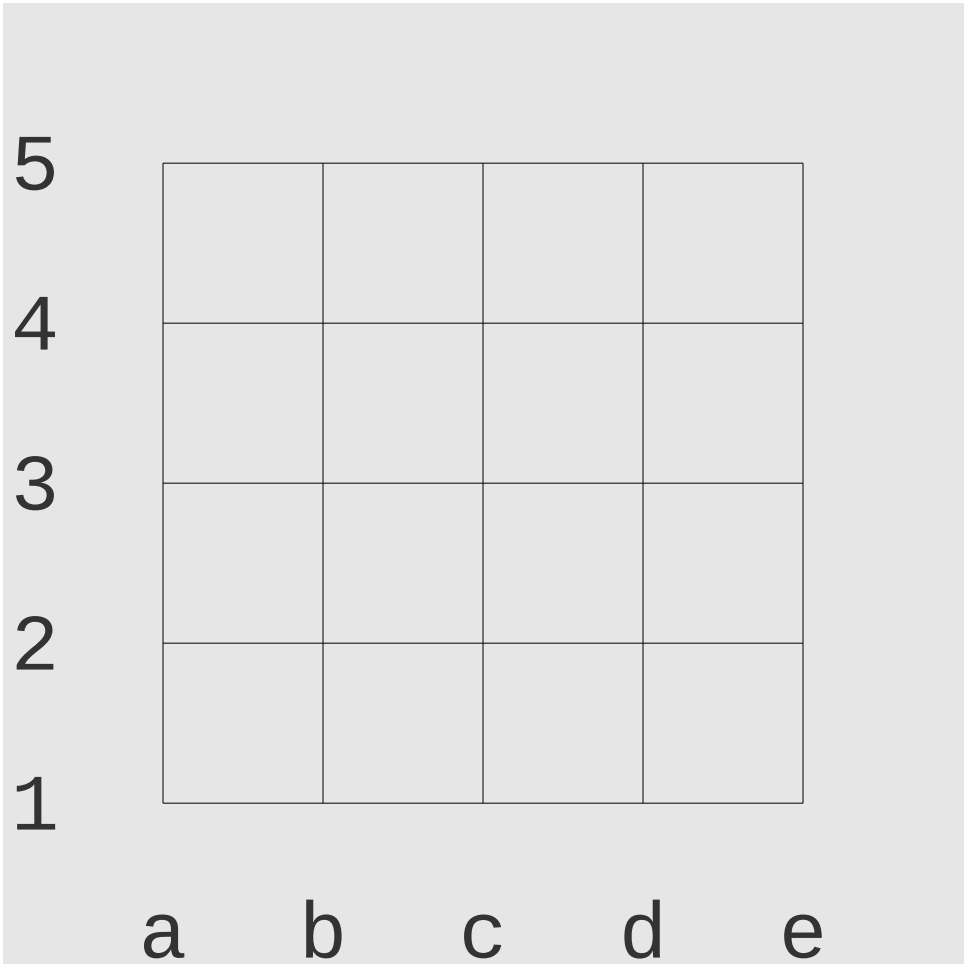
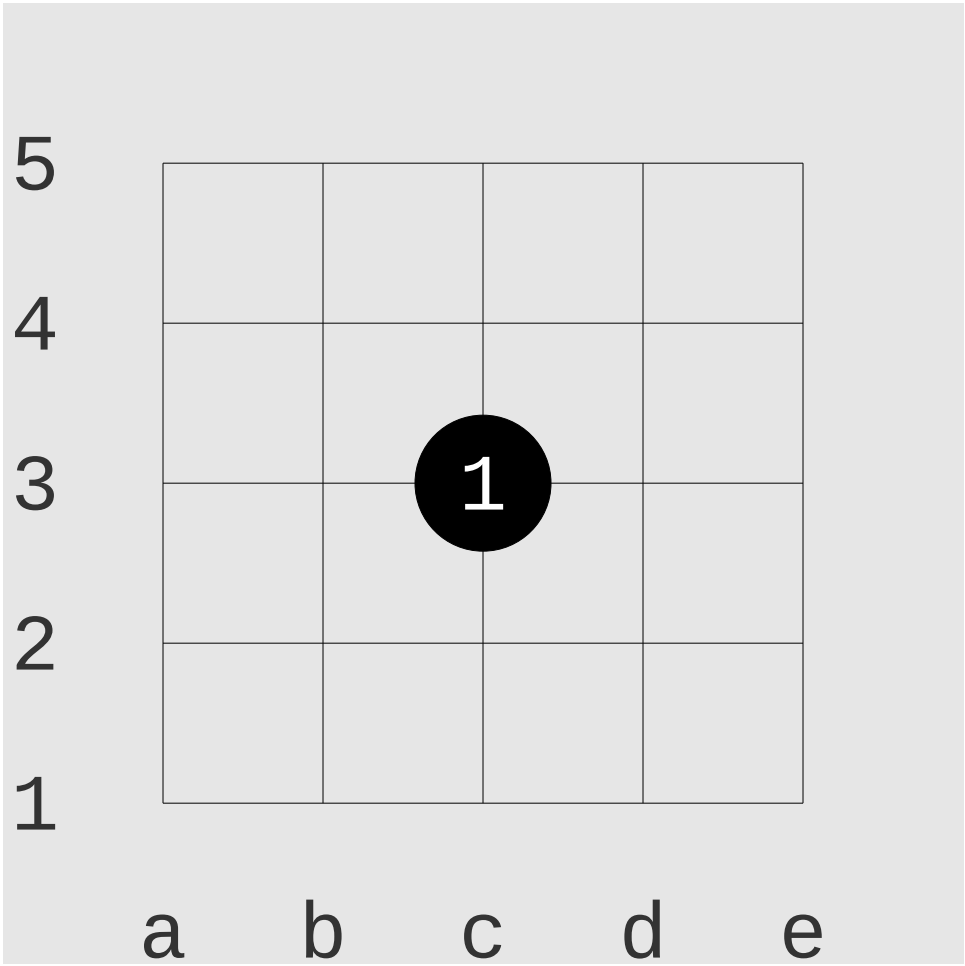
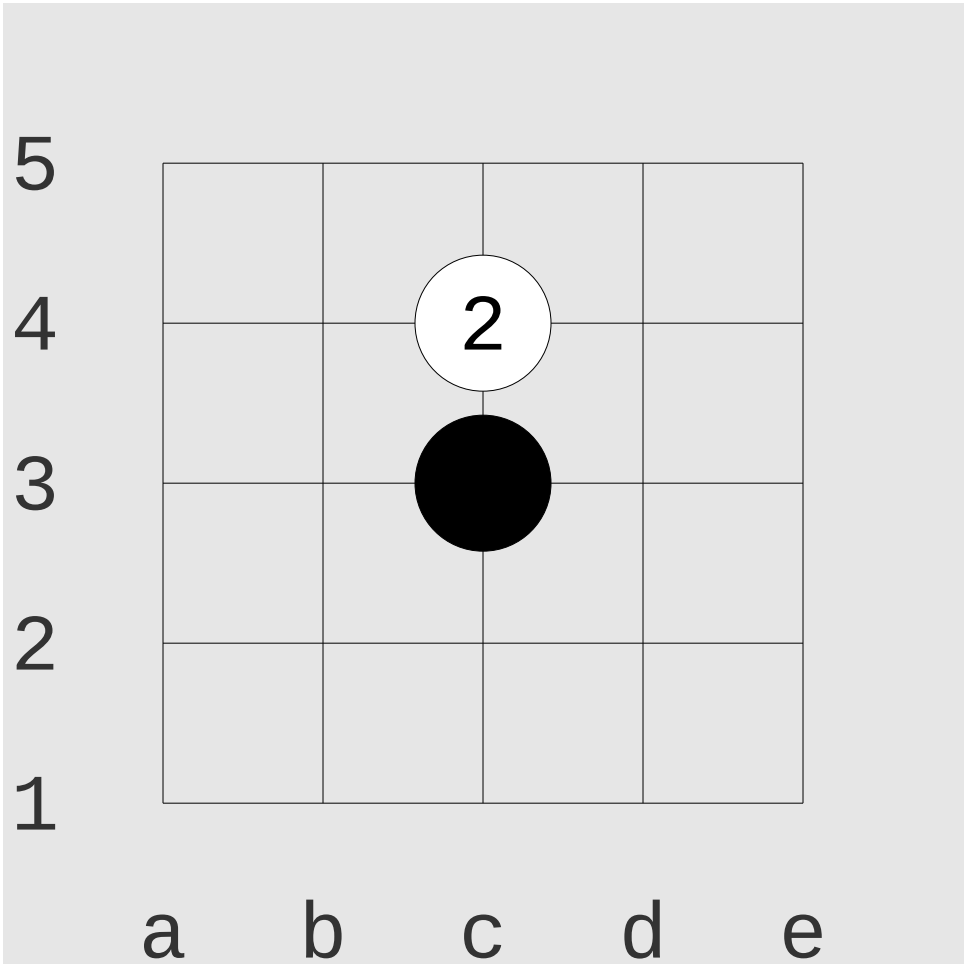
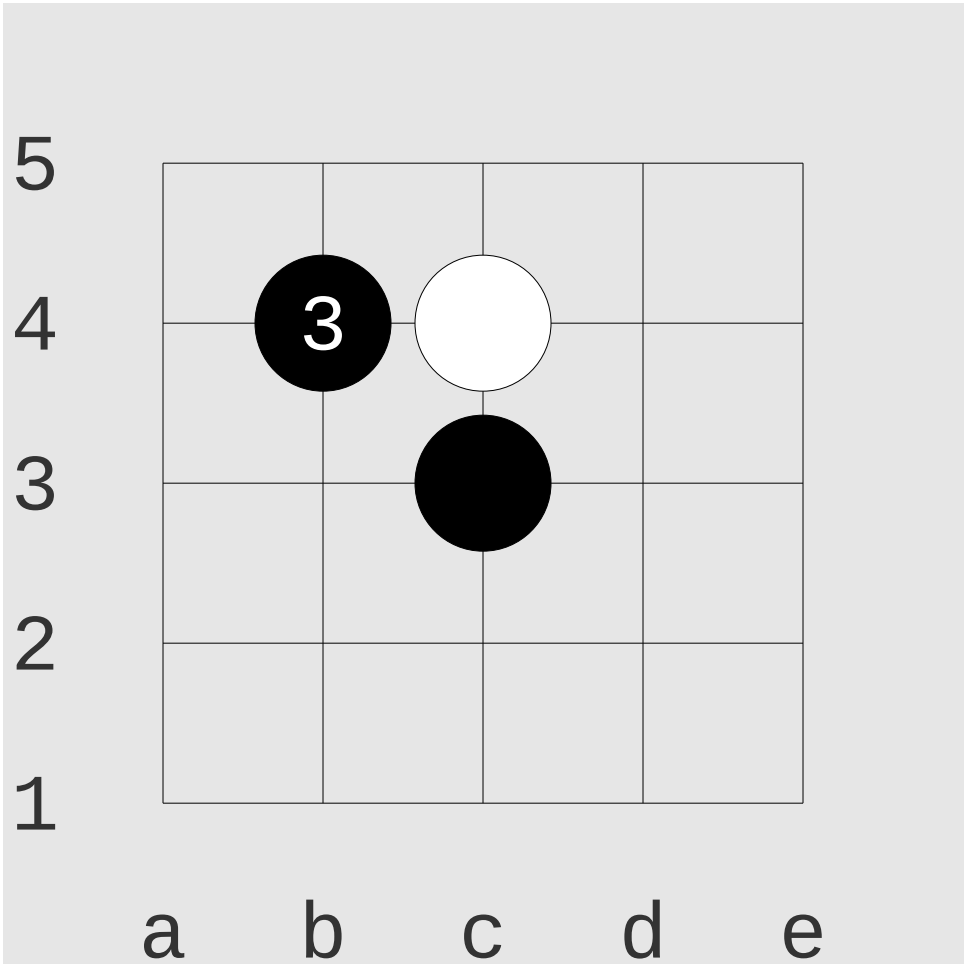
- https://tromp.github.io/go.html
- board (grid), points, players
- positions (blocks, liberties, legal)
- points reached by a point of other color
- alternate-turn, black first
- capturing (clearing)
- pass, move
- end of game
- score
- winner
- komi

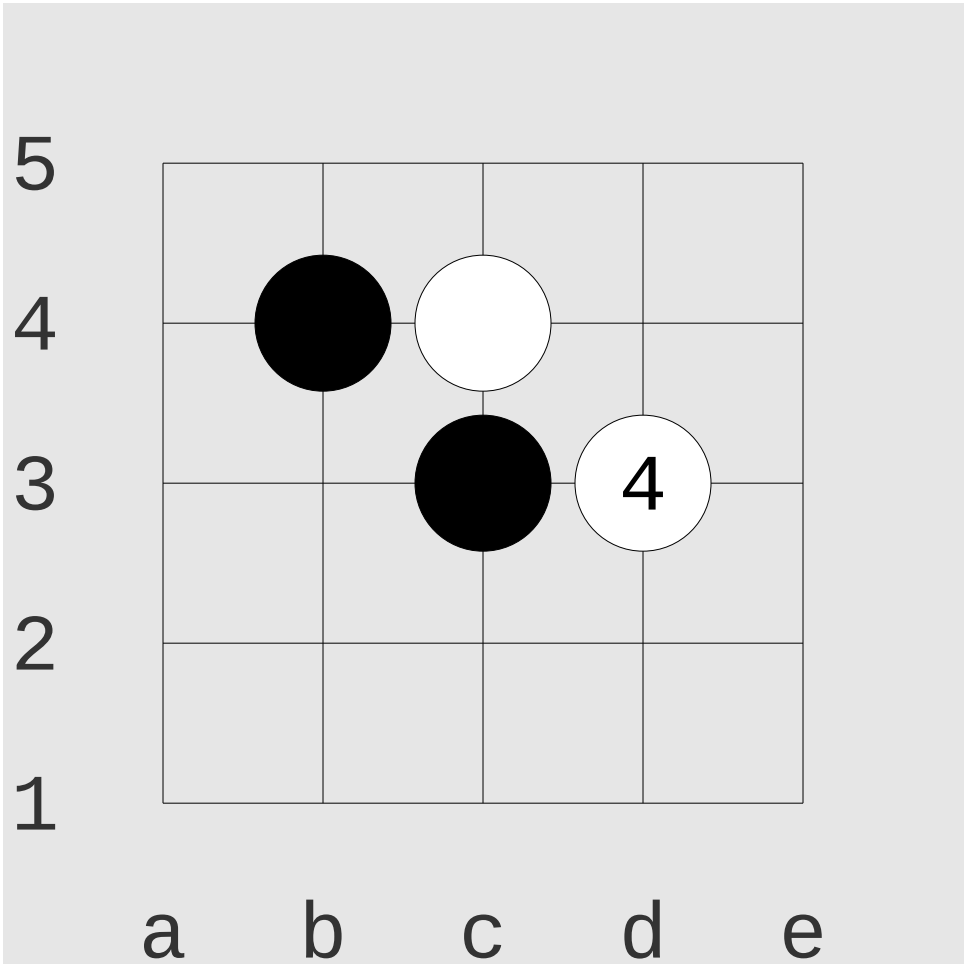


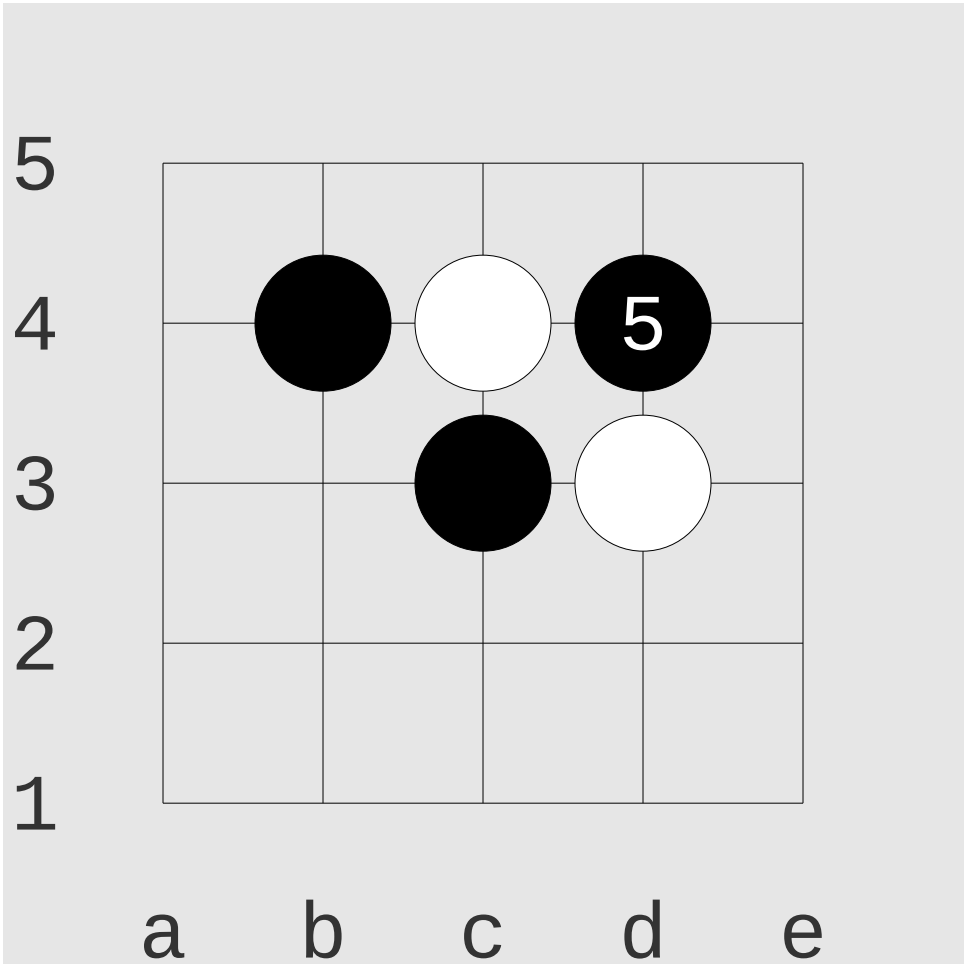
a 5×5 go game

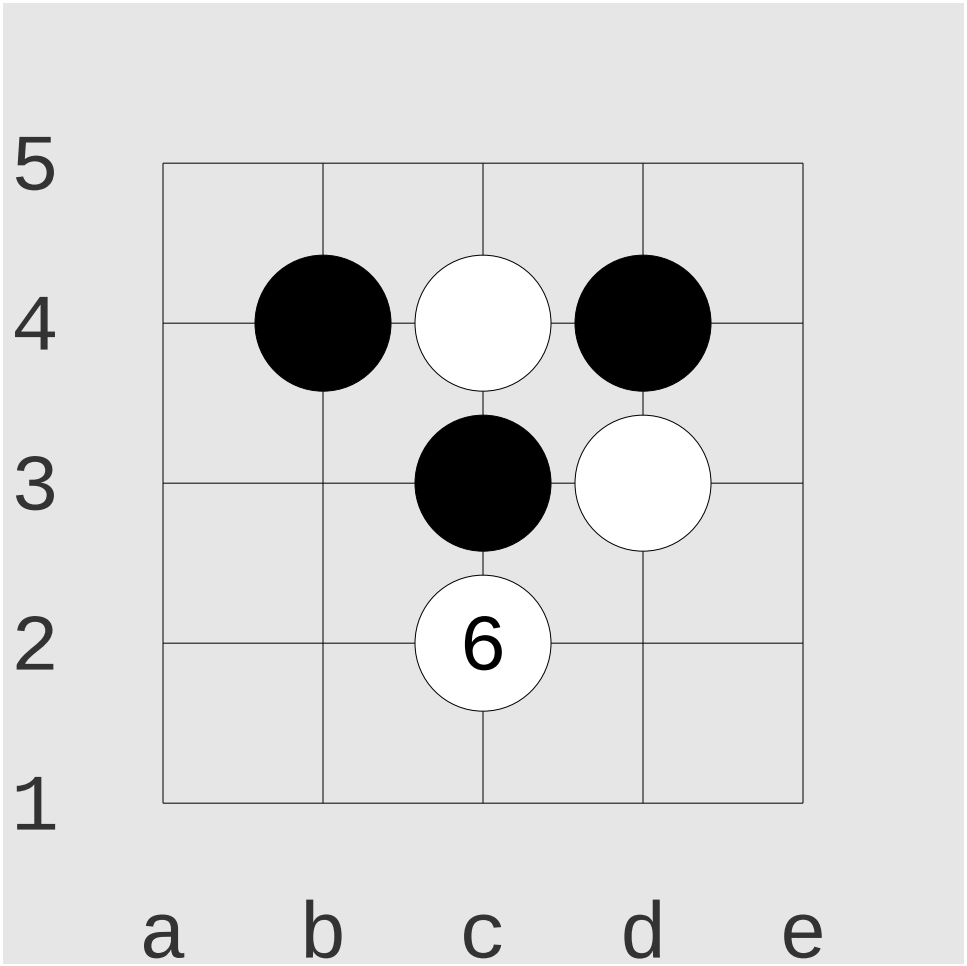

5
4
3
2
1
1
a
b
c
d
e

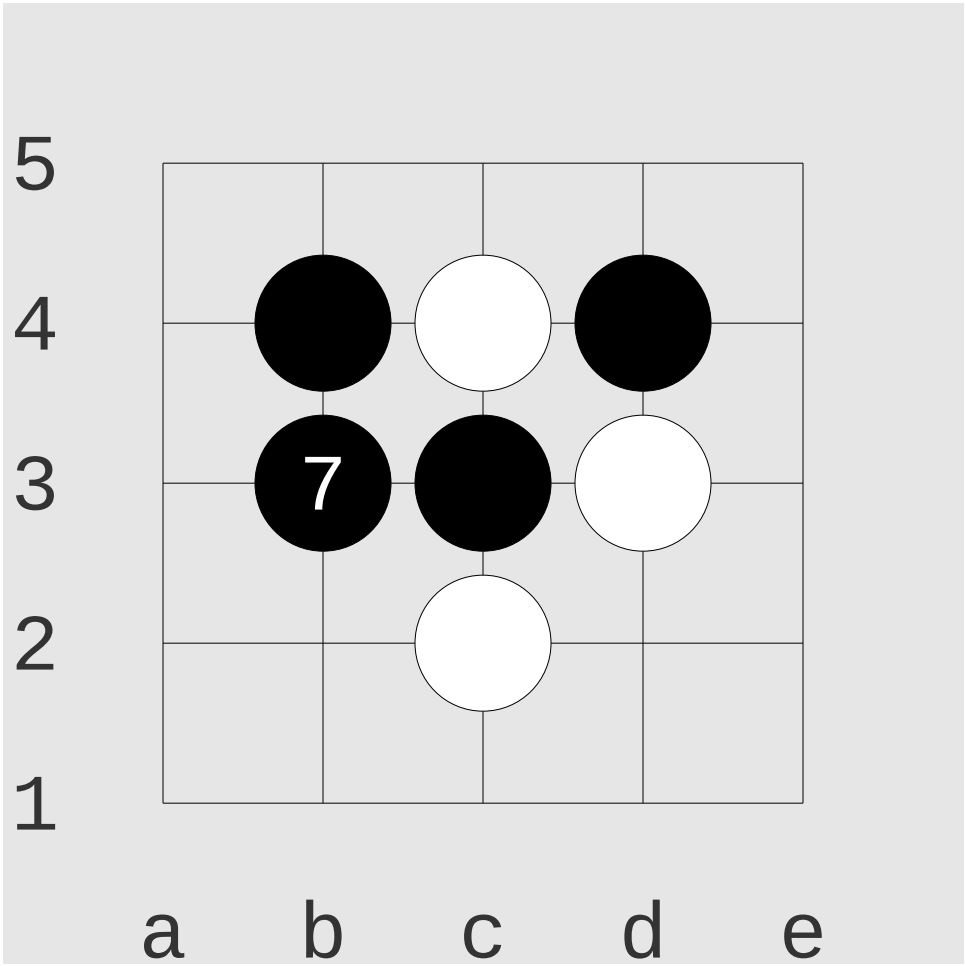

5
4
3
2
1
2
a
b
c
d
e

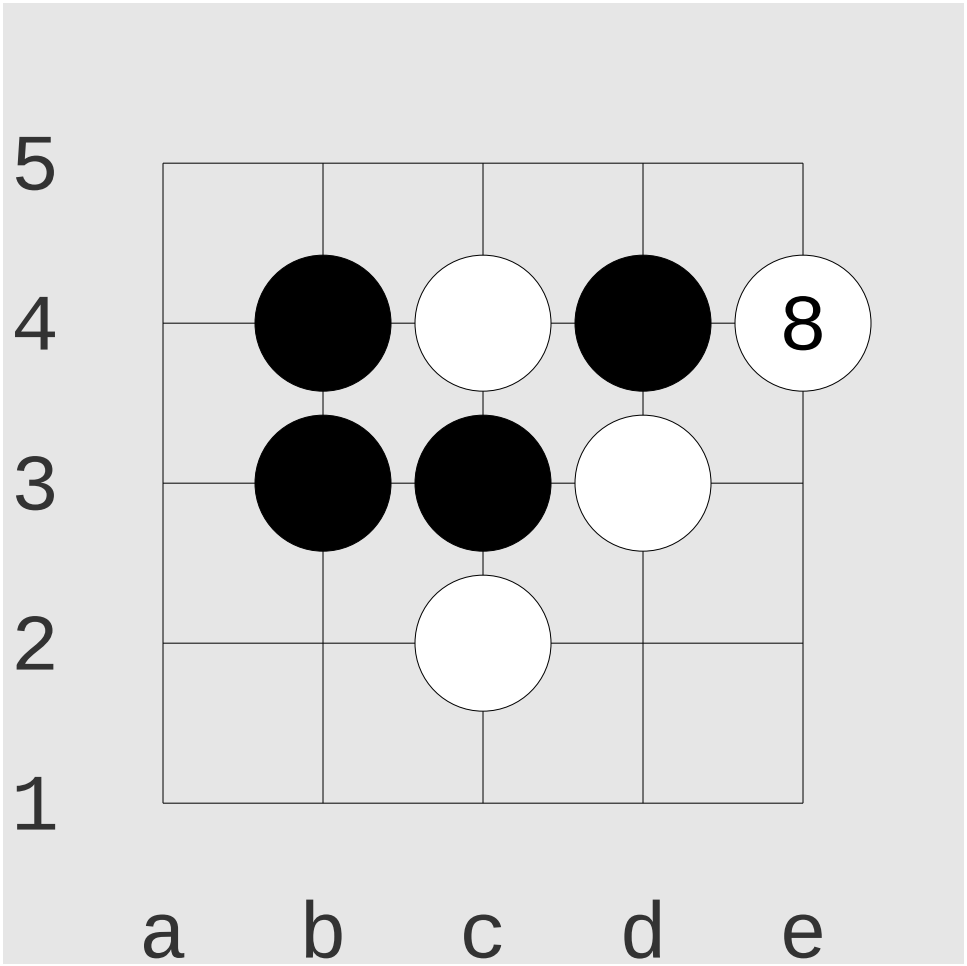

5
4
3
2
1
3
a
b
c
d
e

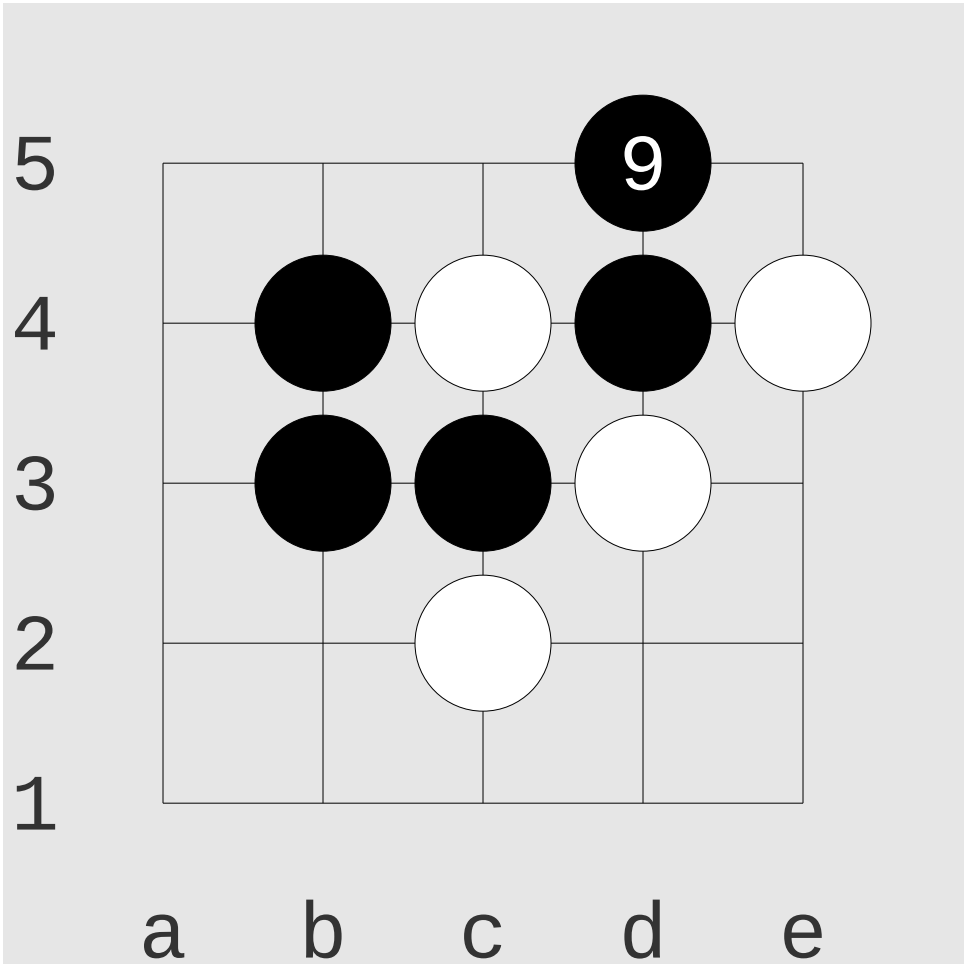

5
4
3
2
1
4
a
b
c
d
e

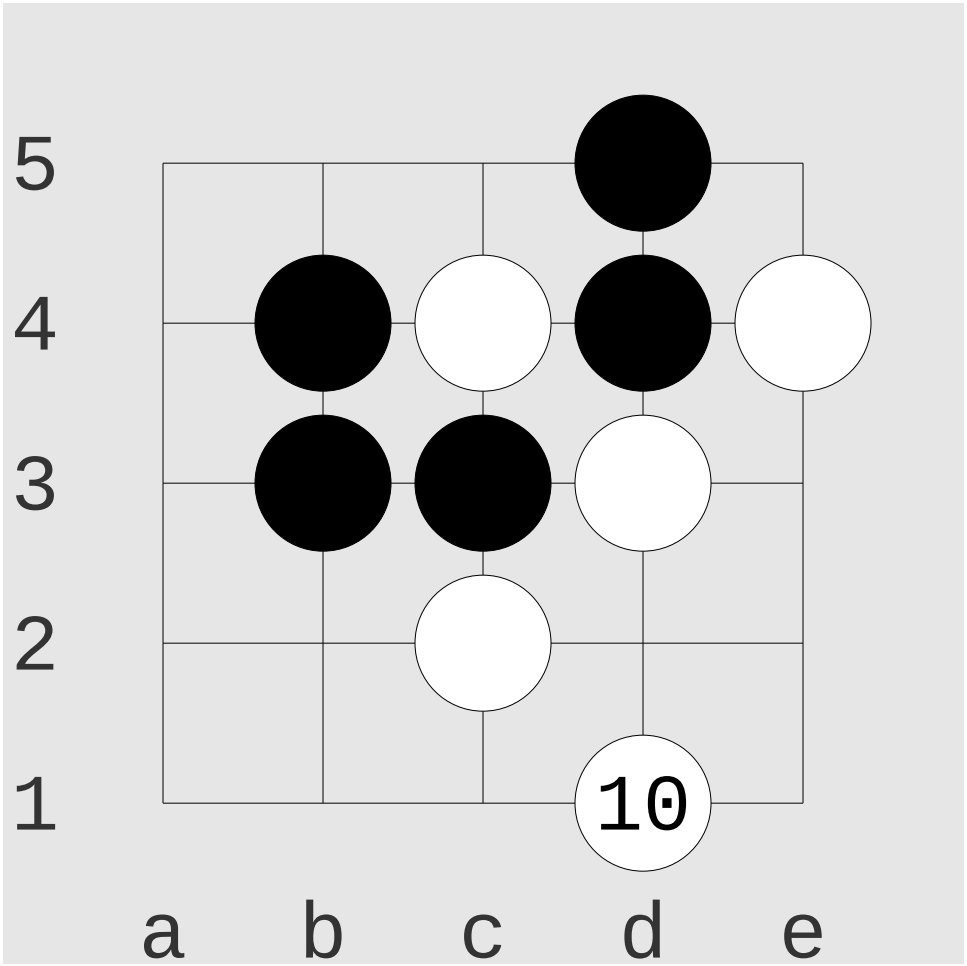

5
4
3
2
1
5
a
b
c
d
e

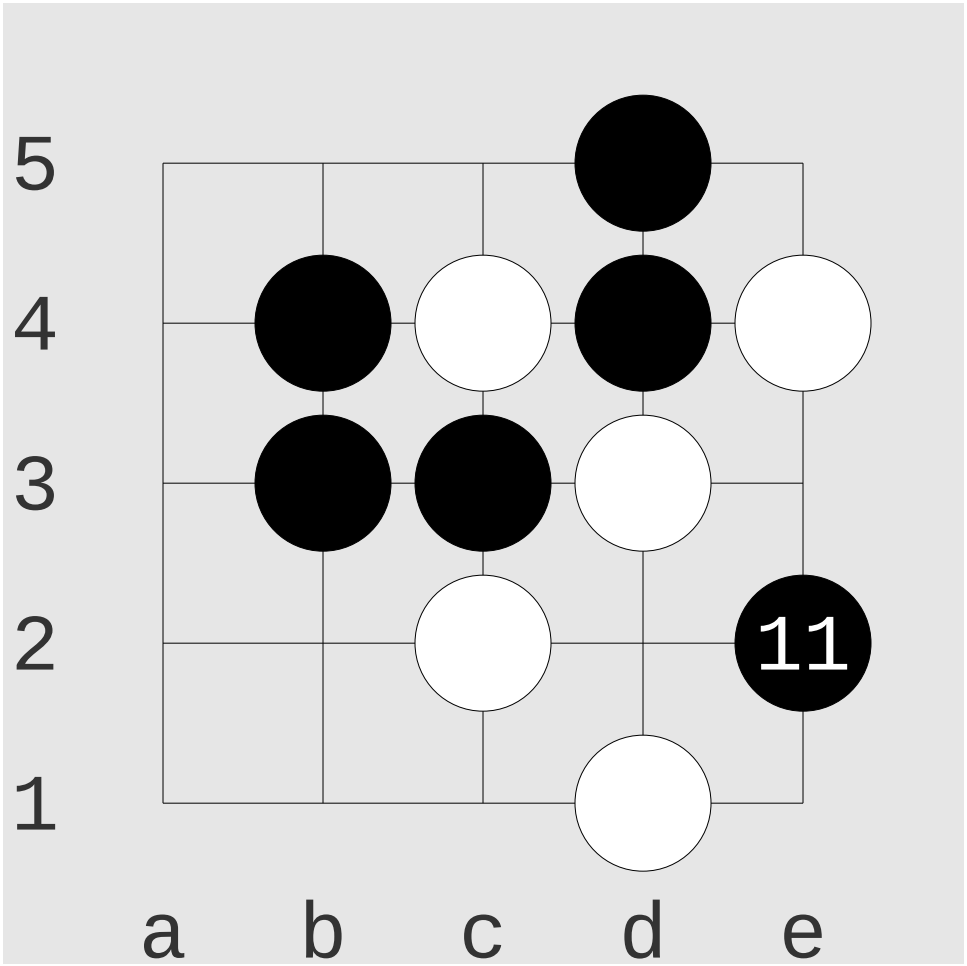

5
4
3
2
1
6
a
b
c
d
e

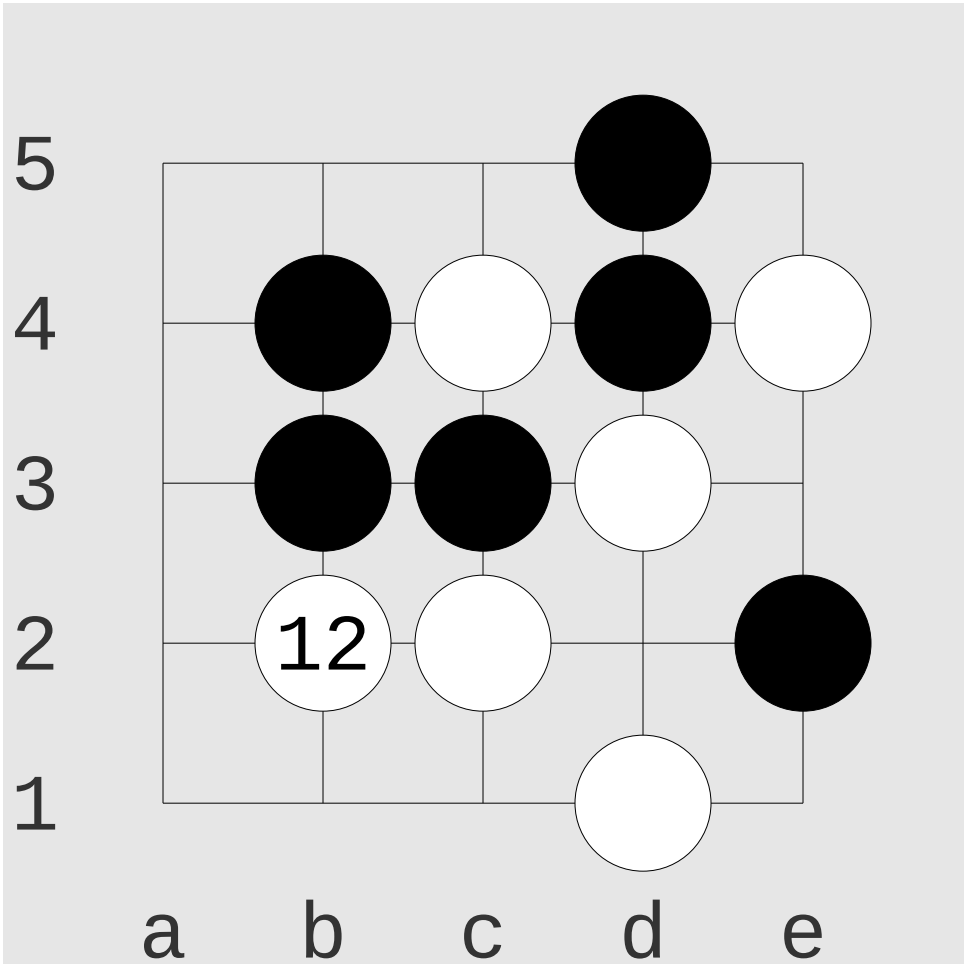

5
4
3
2
1
7
a
b
c
d
e

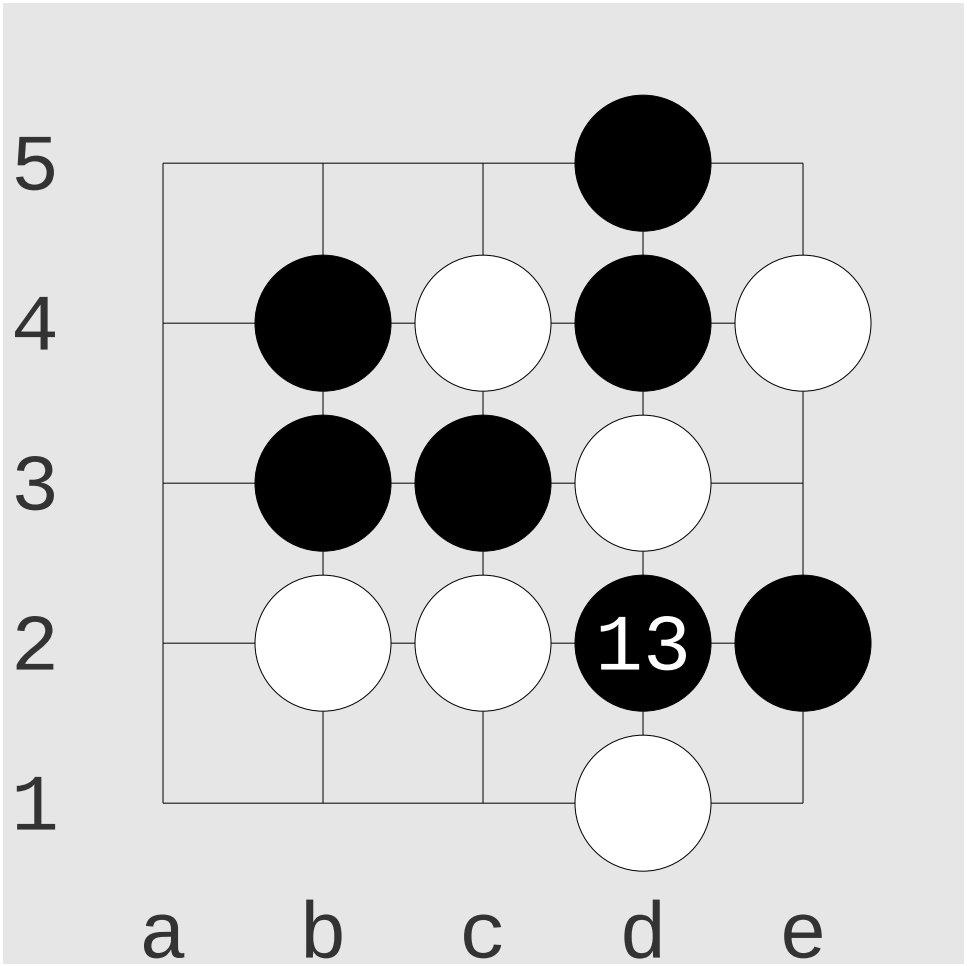

5
4
3
2
1
8
a
b
c
d
e

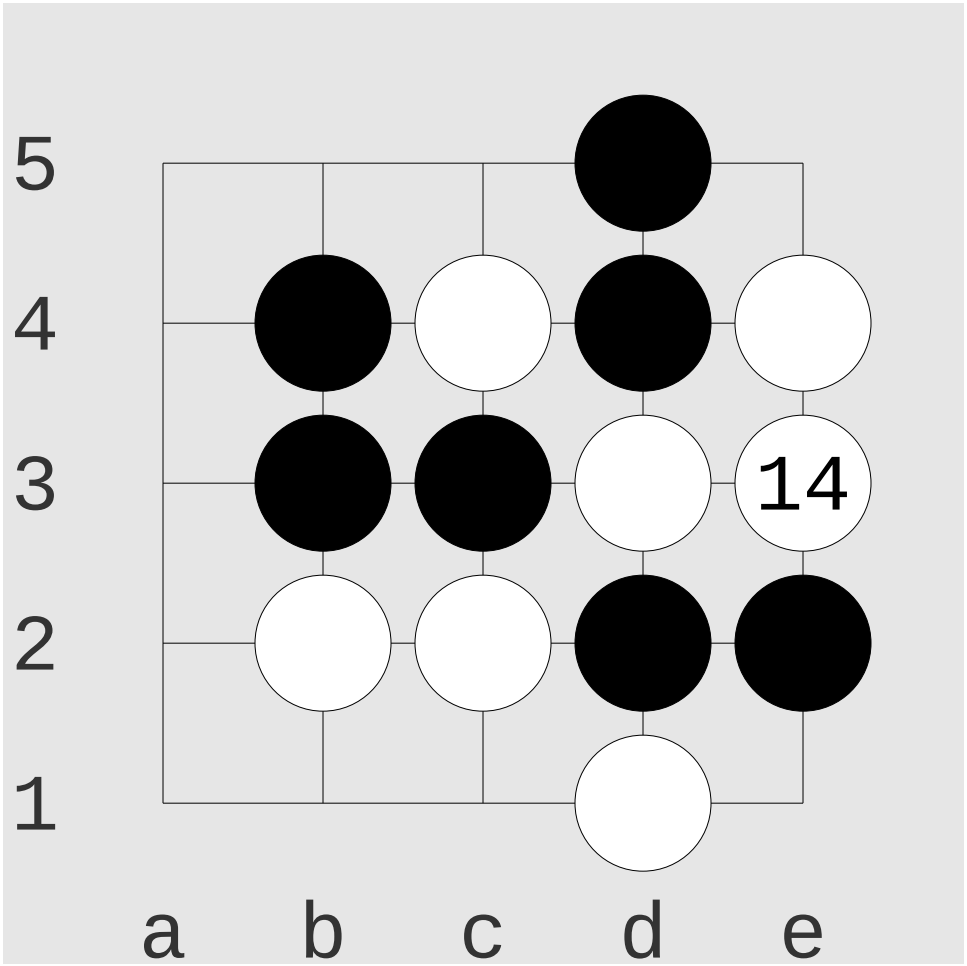

5
4
3
2
1
a
b
c
d
e
9

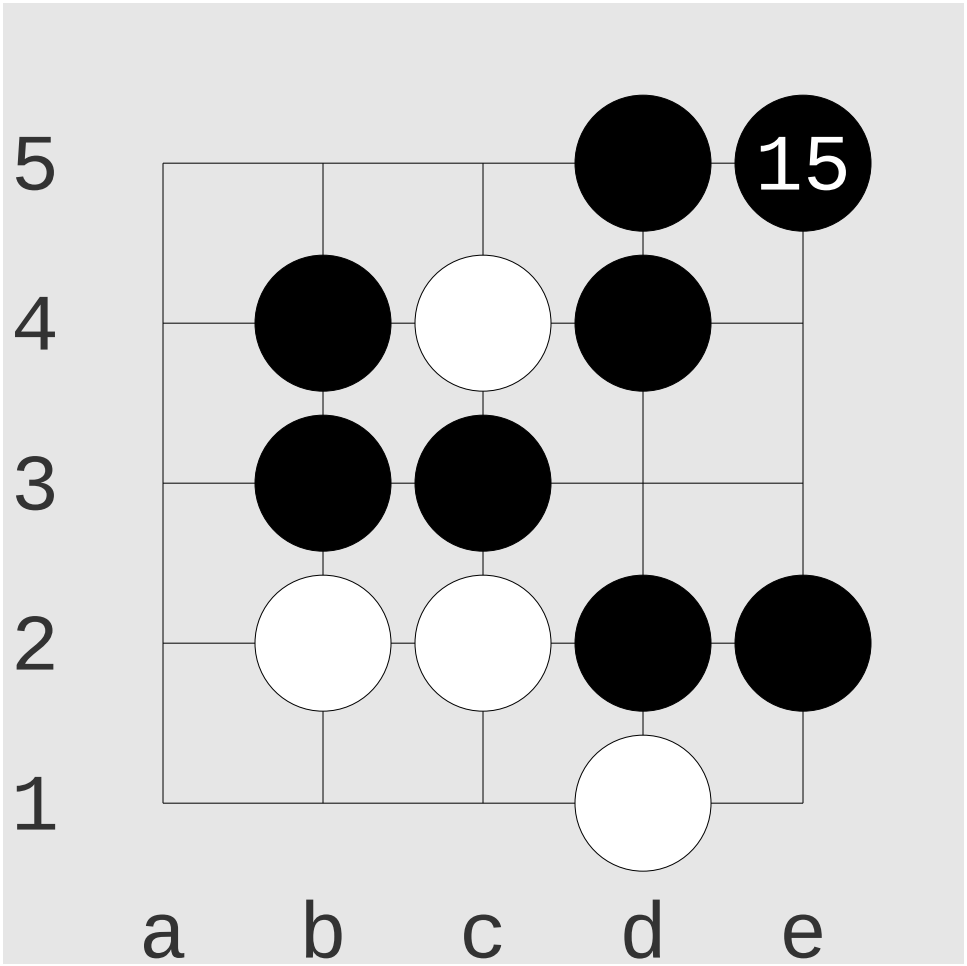

5
4
3
2
1
10
a
b
c
d
e

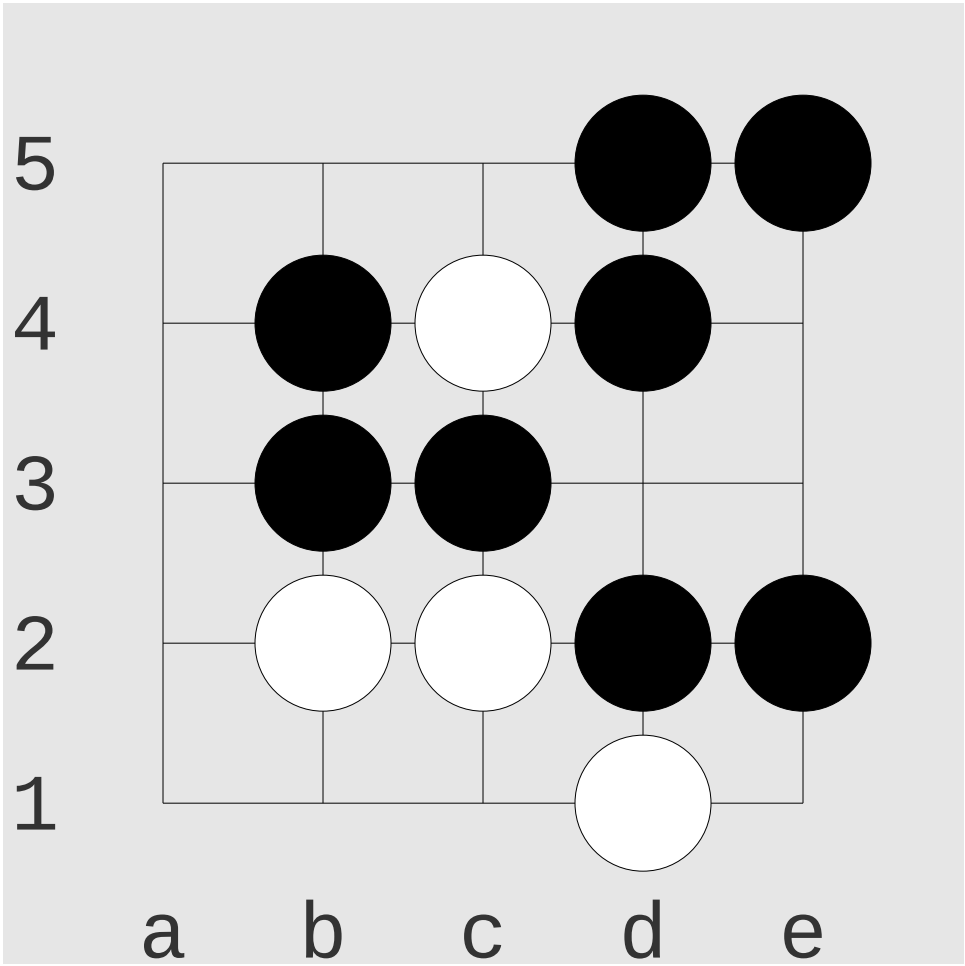

5
4
3
2
1
11
a
b
c
d
e


5
4
3
2
1
12
a
b
c
d
e

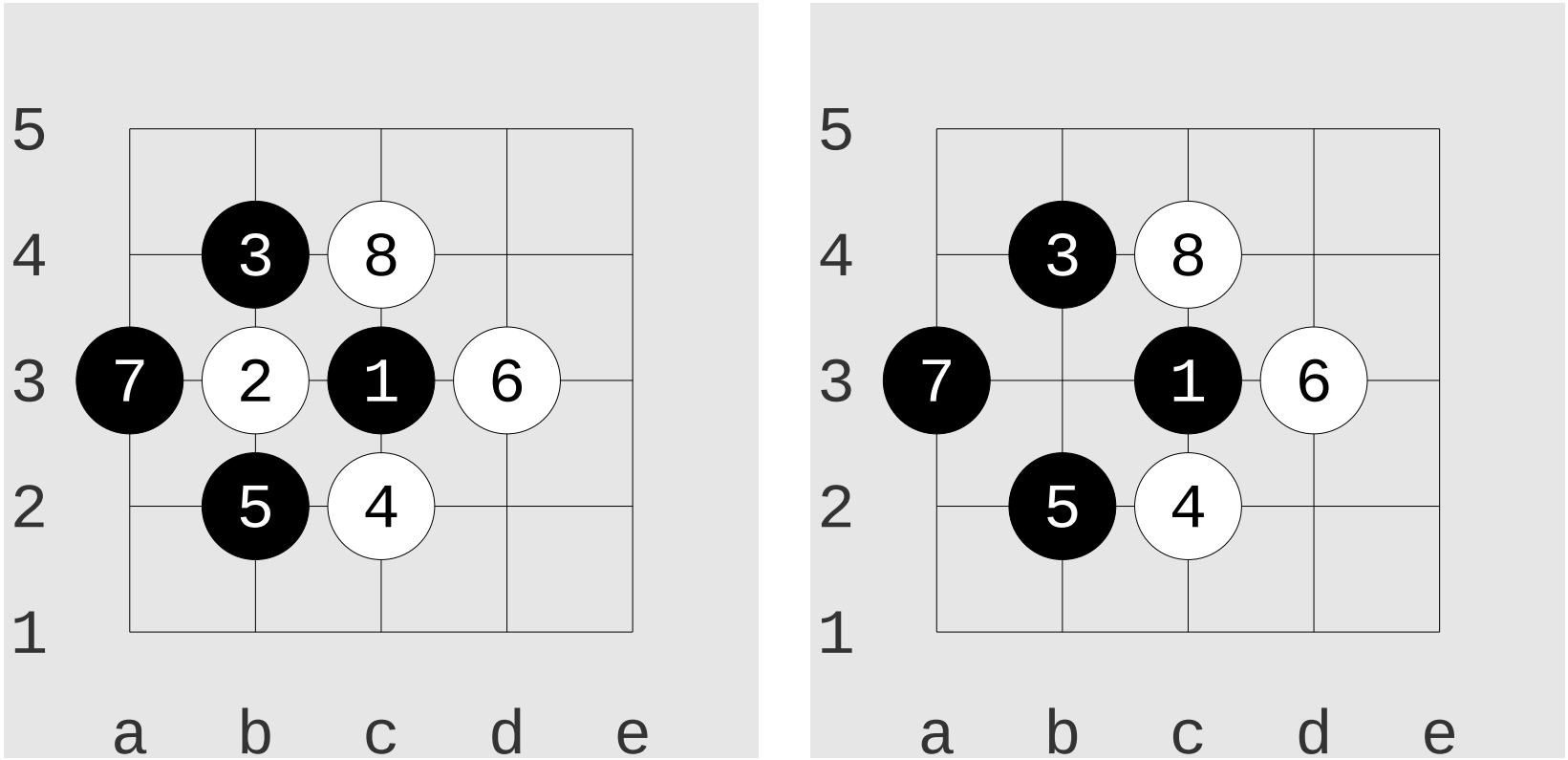

5
4
3
2
1
13
a
b
c
d
e


5
4
3
2
1
14
a
b
c
d
e

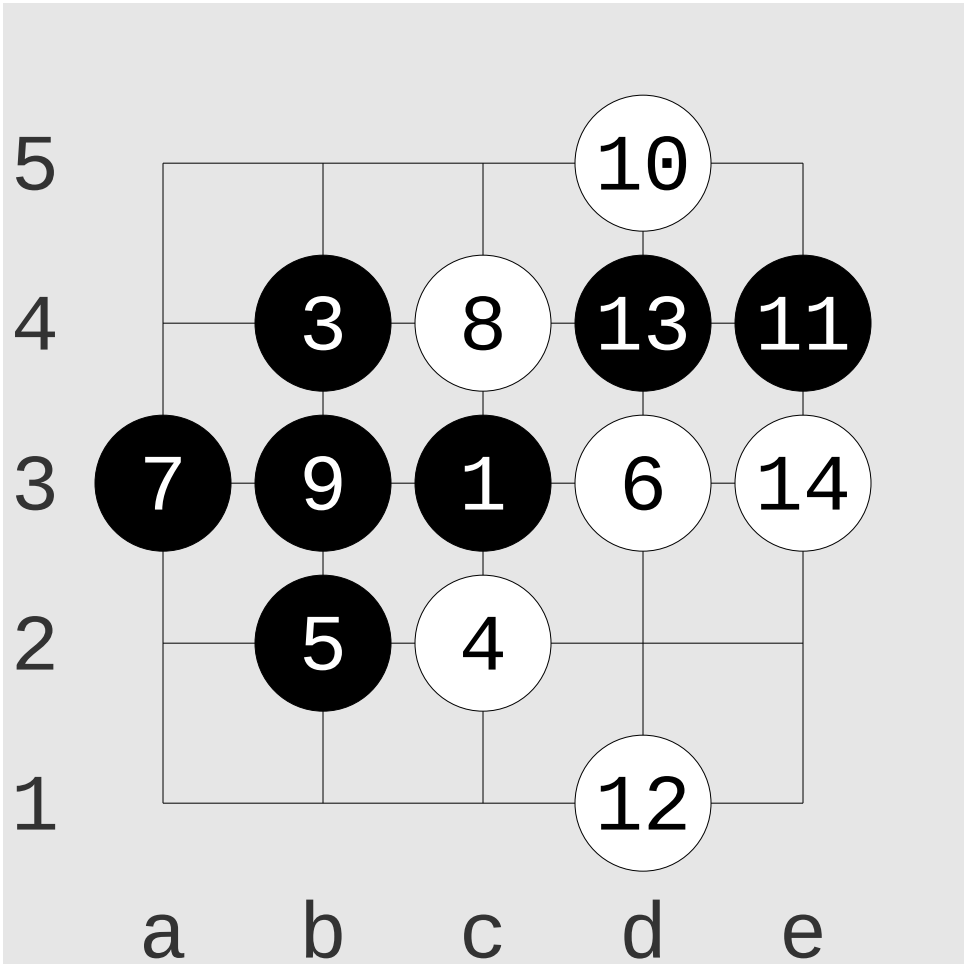

5
4
3
2
1
a
b
c
d
e
15



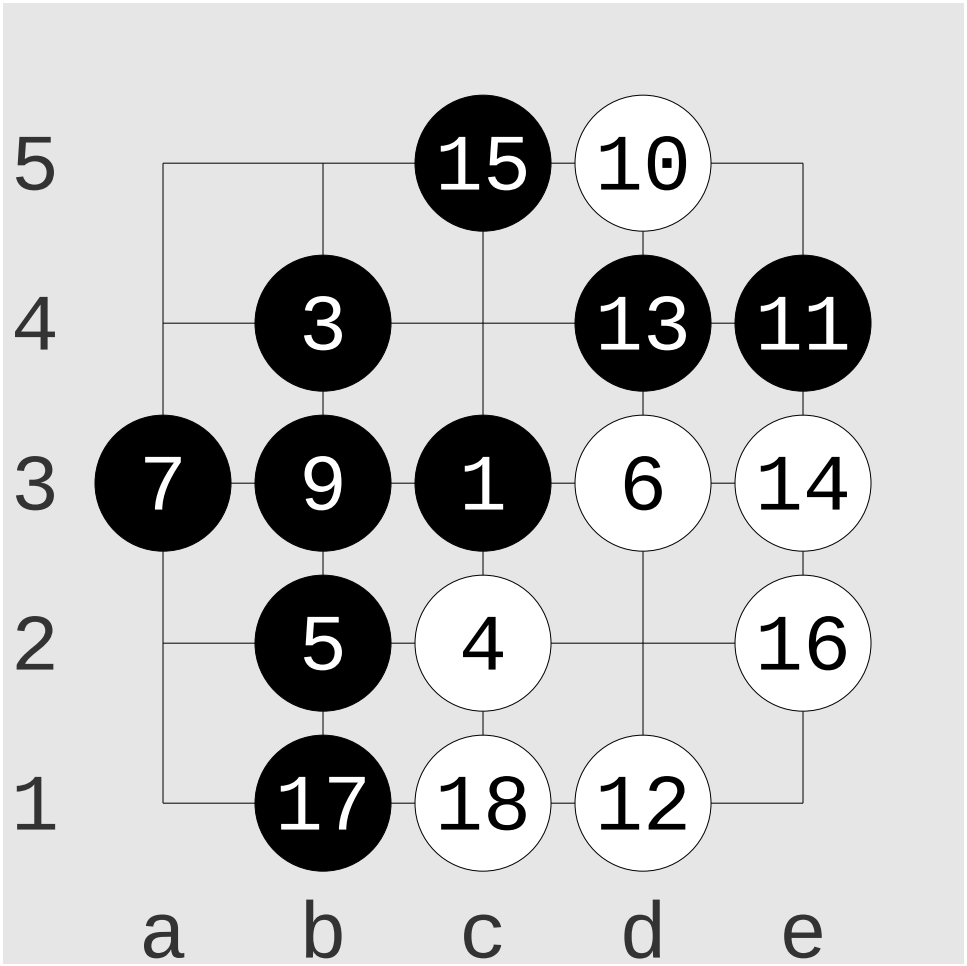
black wins 11 to 4



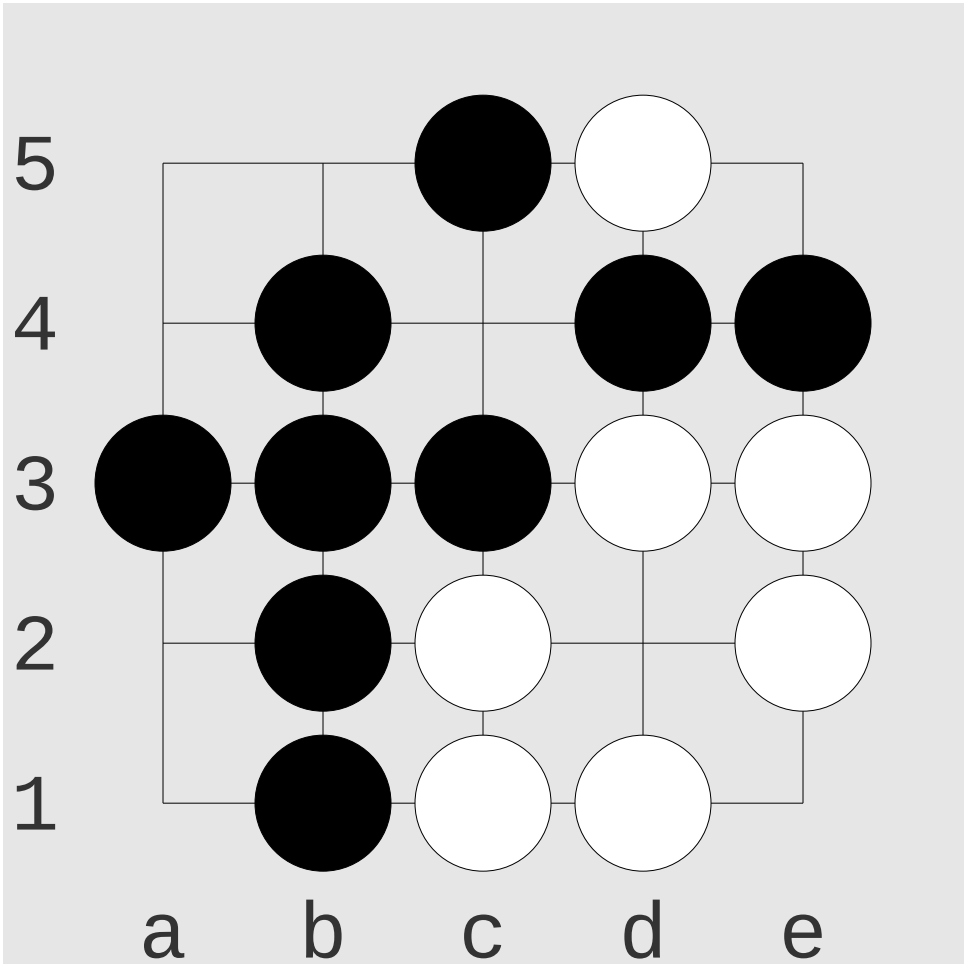
moves 1-8 (and position) from another go game



moves 9-14

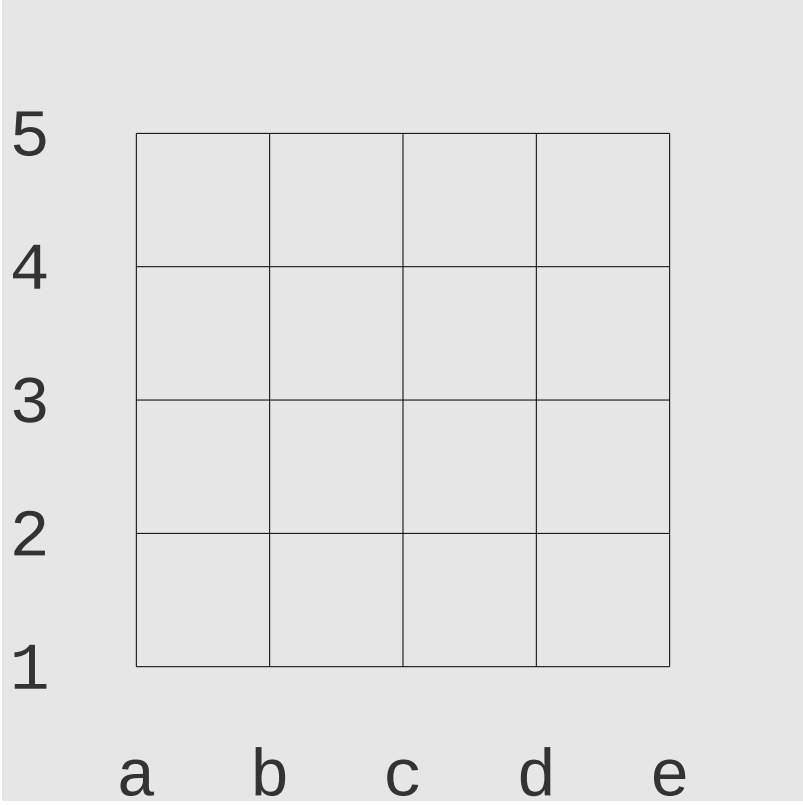


moves 15-18



black wins 15 to 9

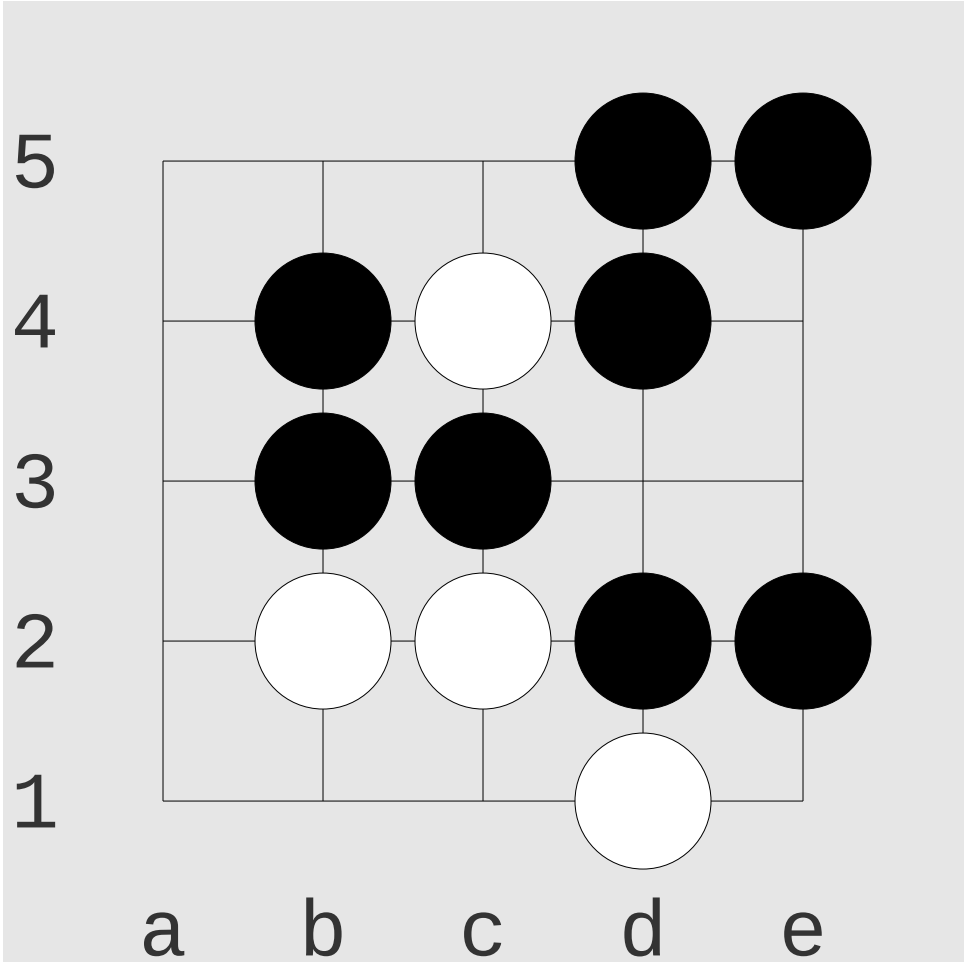
rule 1. ***board*** $m \times n$ **rectangular grid of** ***points*****,**
players **black and white**



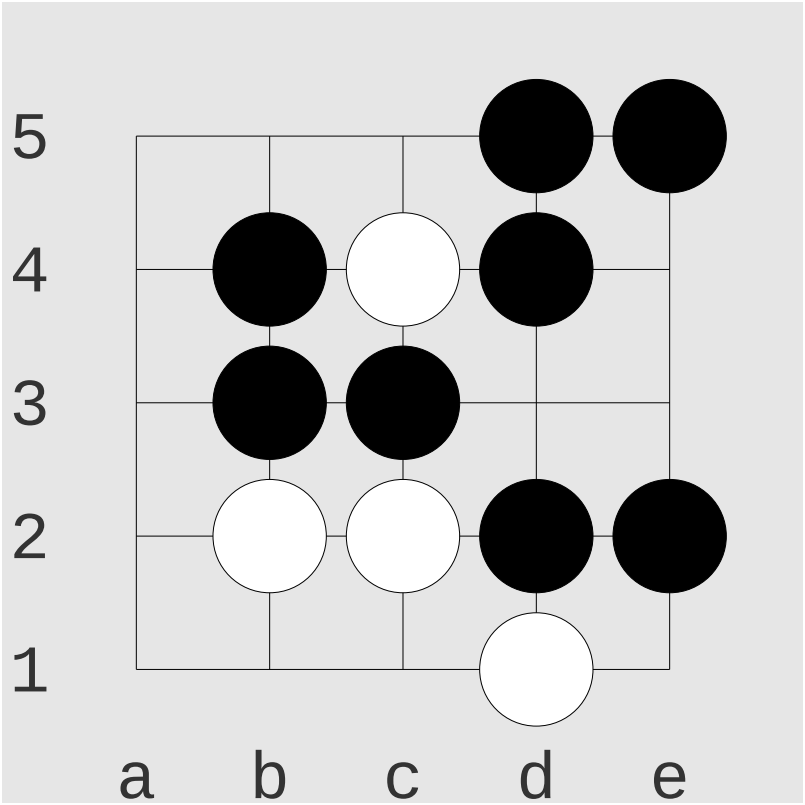
default board size 19×19

rule 2. each point can be colored black/white/empty

(*position* grid with each point colored)



rule 3. **point P *reaches* not-P-color C if exists P-color-path from P to a C-color point**

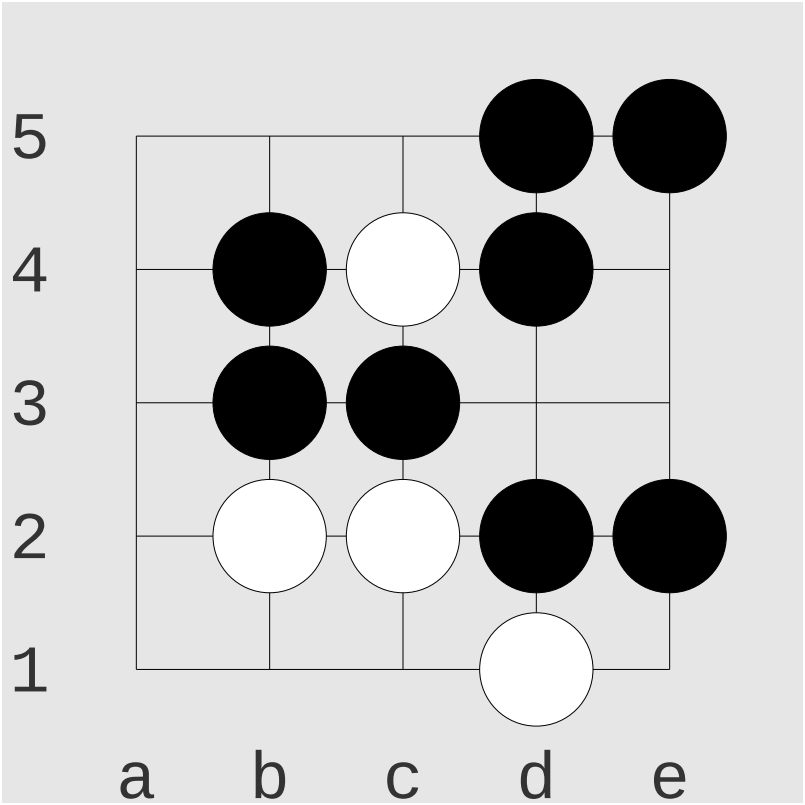


all B-points reach W,E **all W-points reach B,E**

3d 3e 4e reach B-only **other E-points reach B,W**

block **connected component of same-colored points**

block K liberties **all E-points that touch any point of K**



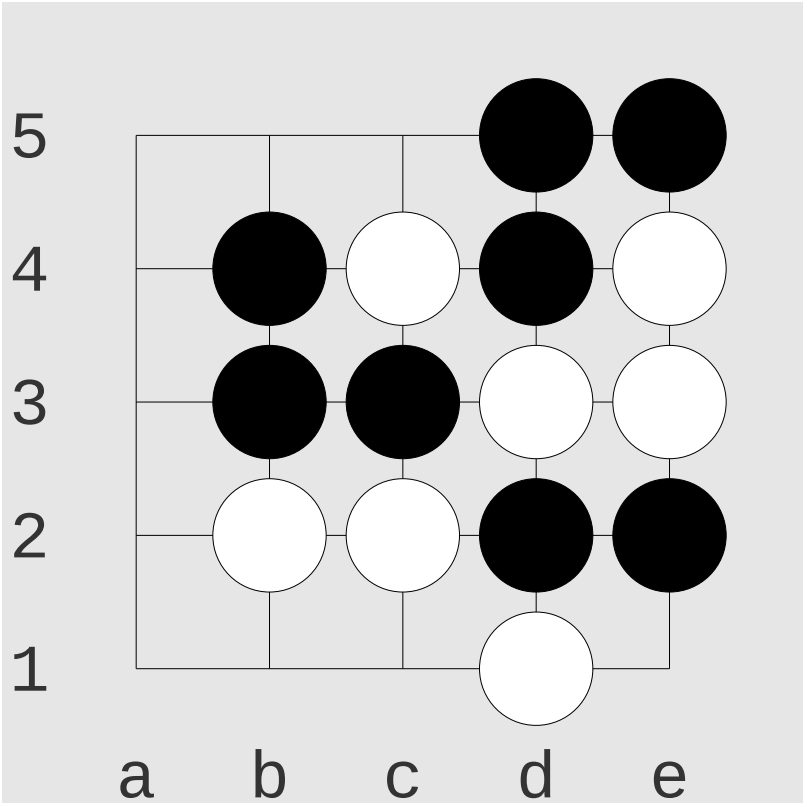
{b3,b4,c3} {d2,e2} {d4,d5,e5} {b2,c2} {c4} {d1}

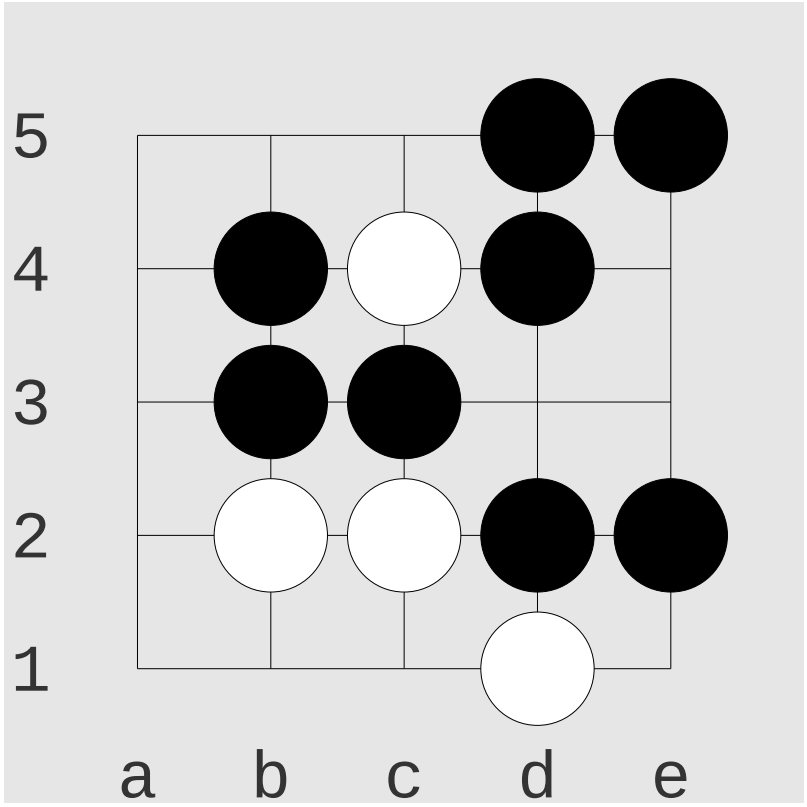
block {d4 d5 e5} liberties: {c5 d3 e4}

block {c4} liberty: {c5} **other 4 blocks?**

rule 4. ***clear color C***

erase all C-points that don't reach E

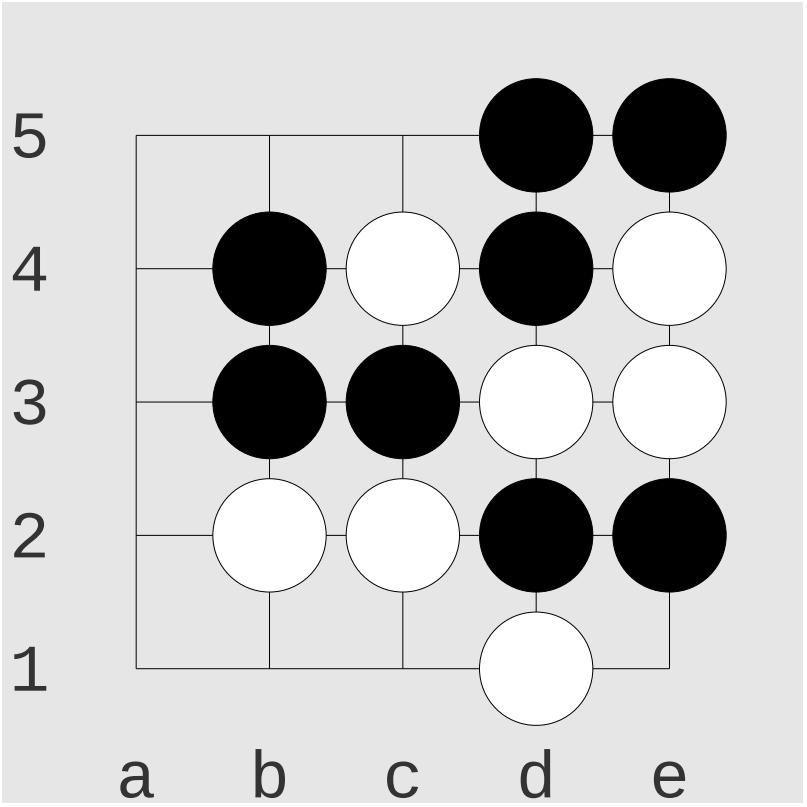


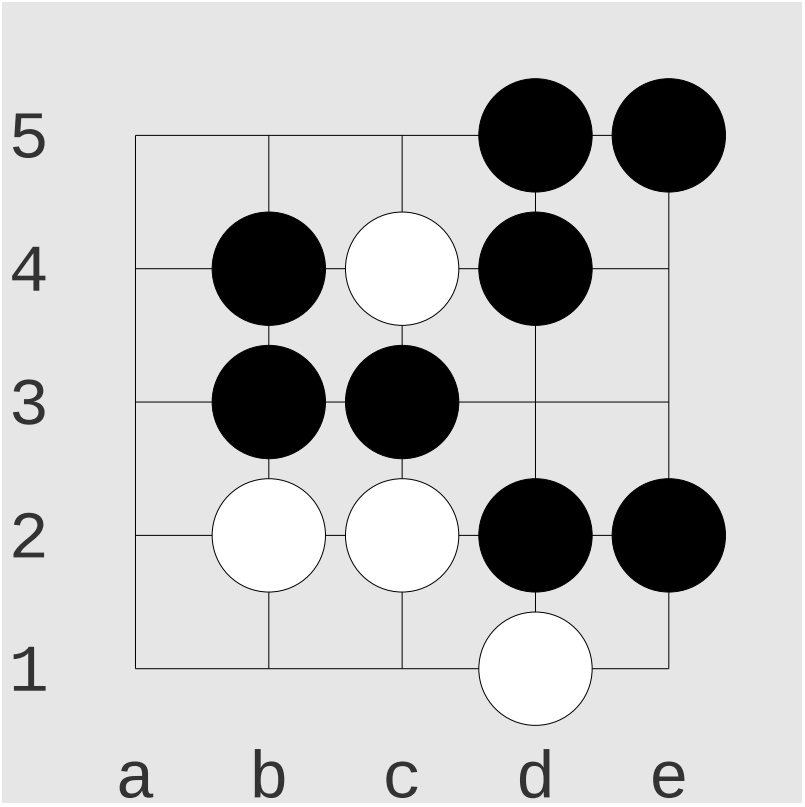


here, we cleared color W

a block is *captured* if it has no liberties

rule 4 (rephrased). remove all captured C-blocks

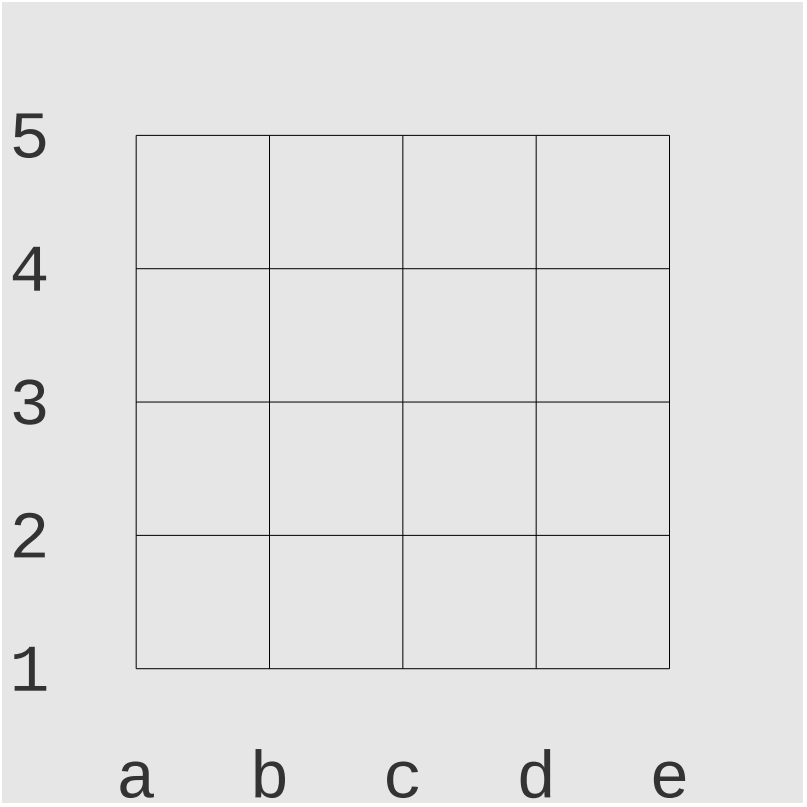


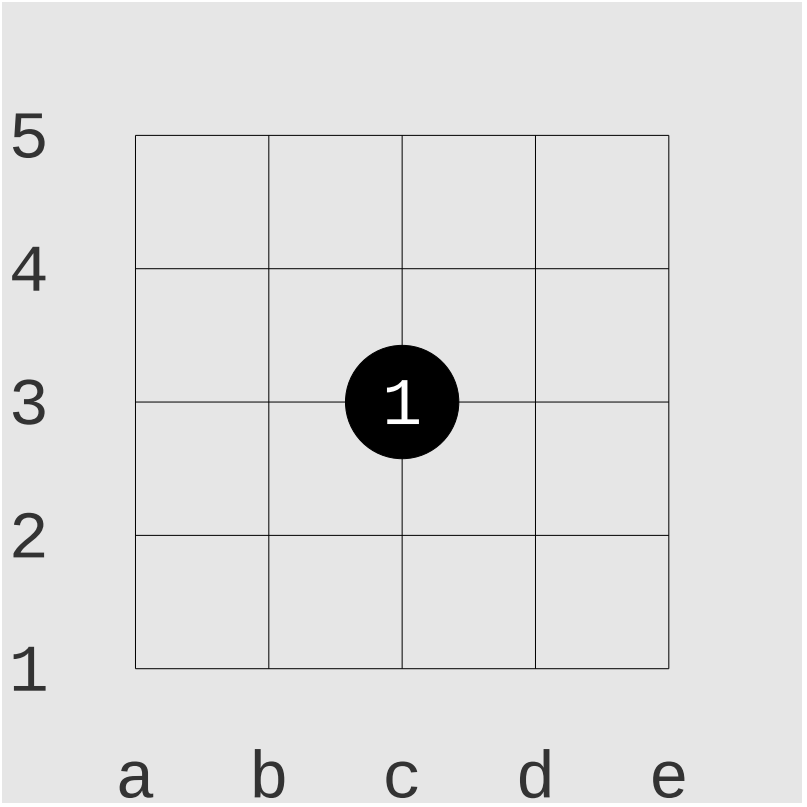


here, we removed all captured W-blocks

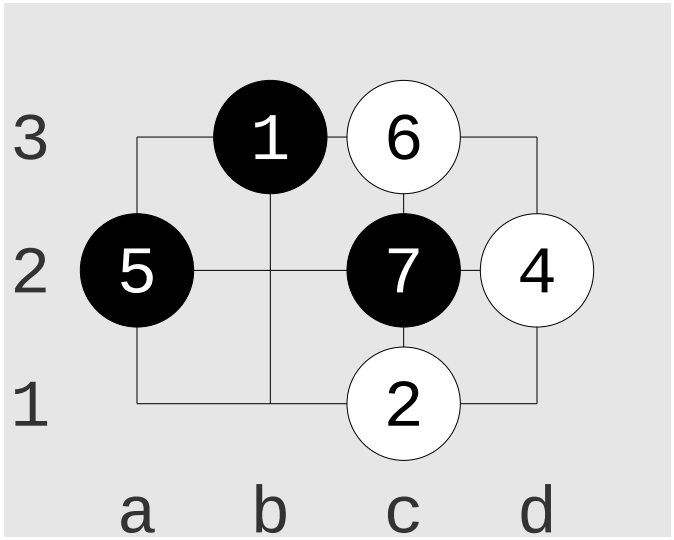
rule 5. **start from empty grid,**

players alternate turns, **black moves first**



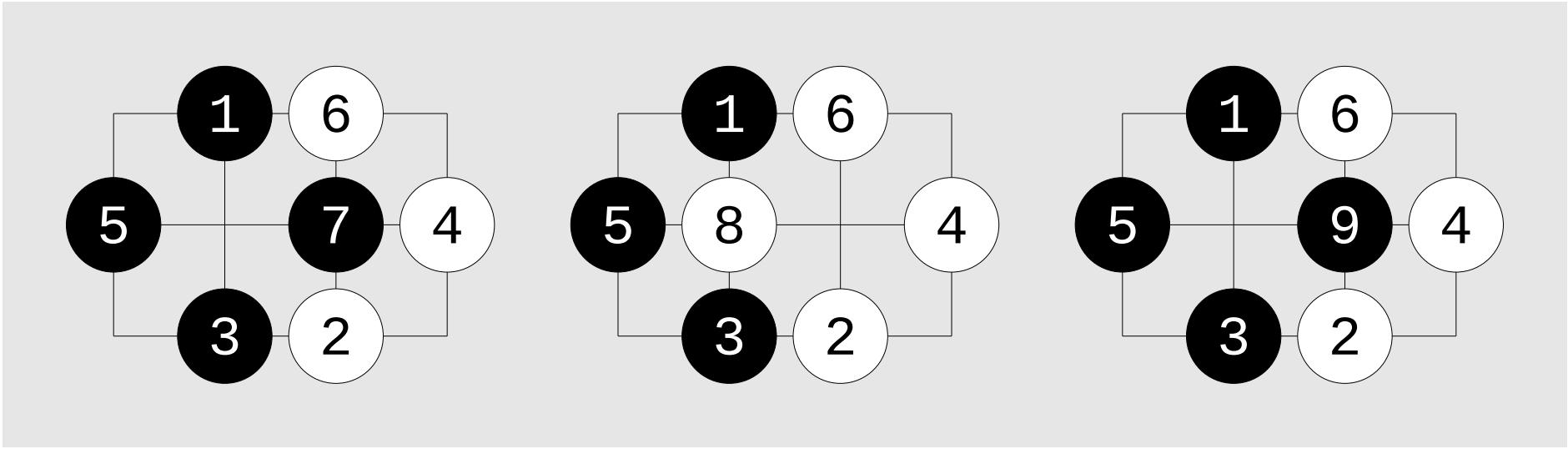


rule 6. a *turn* is either *pass* (do nothing) or
a move not making an earlier position (*superko*)



pass ? **1.B[b3] 2.W[c1] 3.B[pass] 4.W[d2] . . .**

rule 6. a *turn* is either *pass* (do nothing) or
a move not making an earlier position (*superko*)



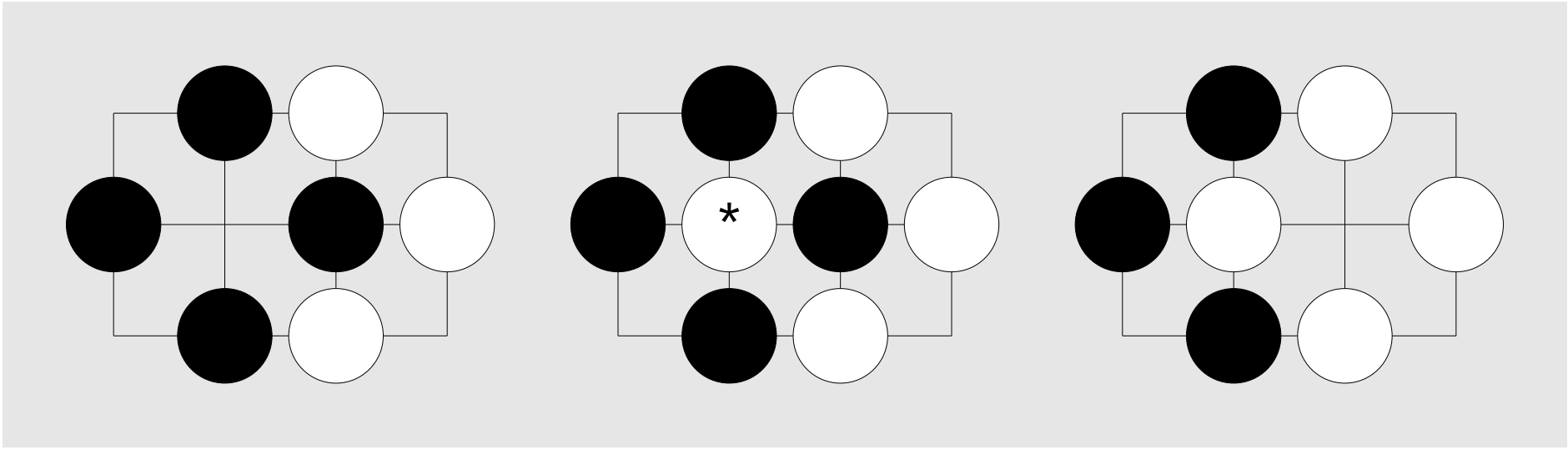
superko ? **move 9 illegal: superko**

rule 7. a *move by player P* is

(1) P-color an E-point

(2) clear color Q (opponent of P)

(3) clear color P



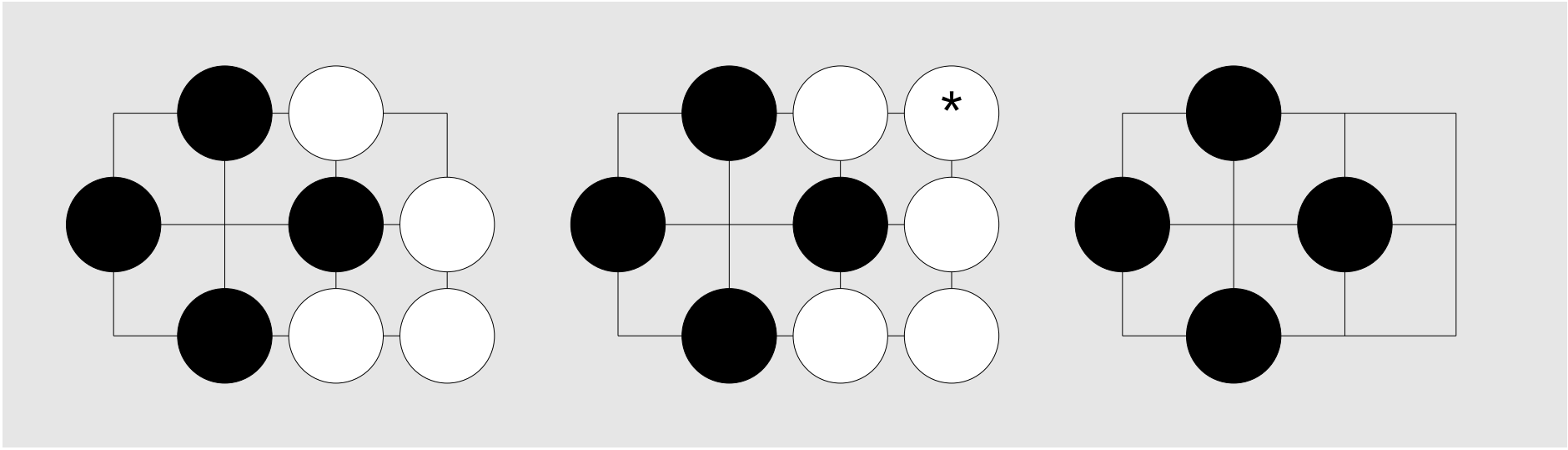
(2) is capture

rule 7. a ***move by player P*** **is**

(1) P-color an E-point

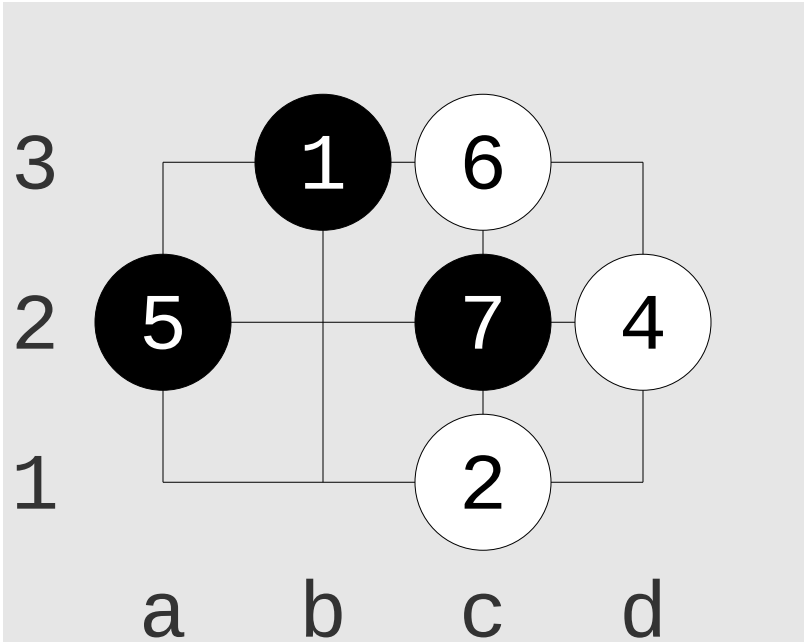
(2) clear color Q (opponent of P)

(3) clear color P



(3) is self-capture, allowed in logical rules of go,
but not in Chinese/Korean/Japanese/. . . rules

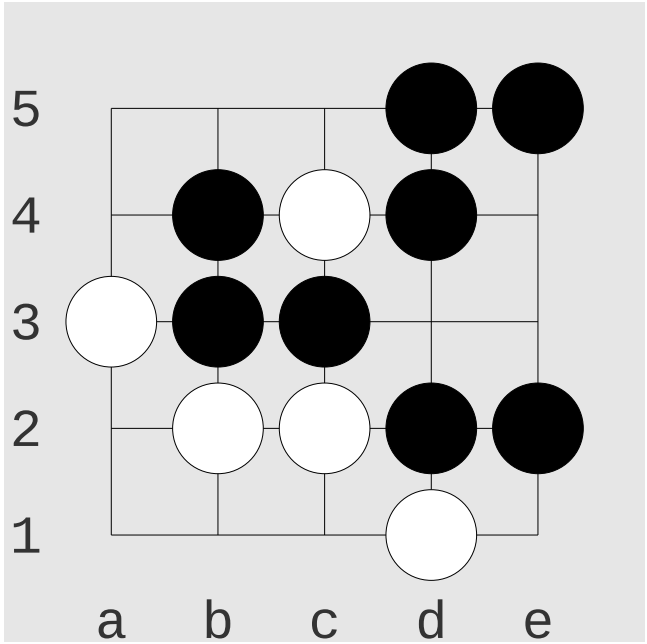
rule 8. **game ends after 2 consecutive passes**

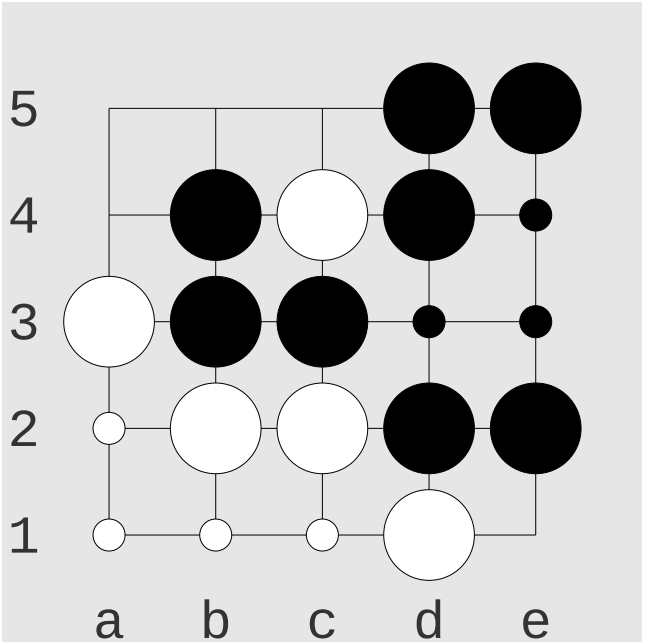


1. B[b3] 2. W[c1] 3. B[pass] 4. W[d2]
5. B[a2] 6. W[c3] 7. B[c2] 8. W[pass]
9. B[pass]

rule 9. ***score*** **of player P with color C:**

number C-points + number E-points that reach only C

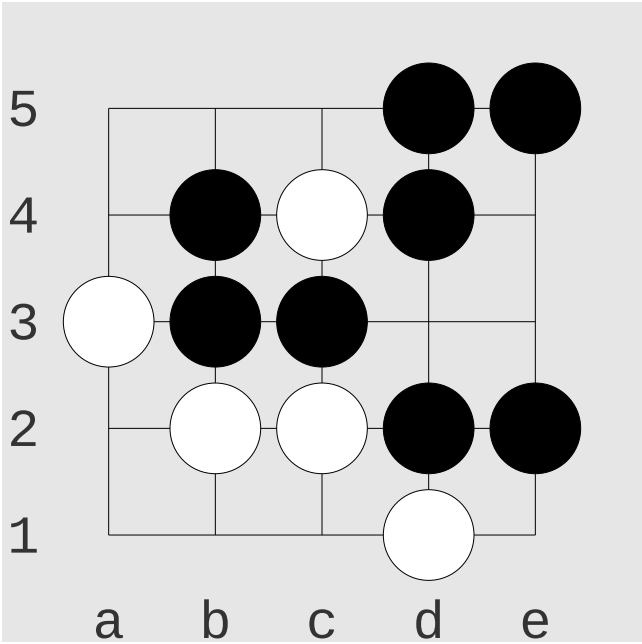




B-score 8 + 3 = 11

W-score 5 + 4 = 9

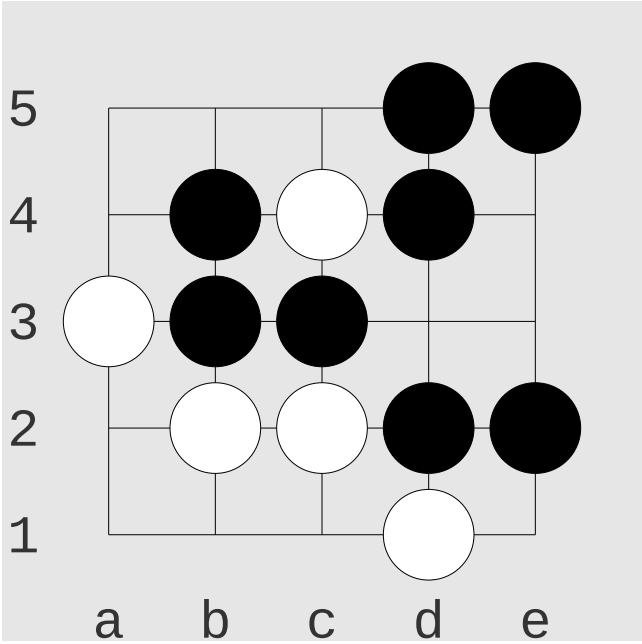
rule 10. ***winner*** **player with higher score at game end**



here, B wins by 2 (11 – 9)

rule 10. ***winner*** **player with higher score at game end**

by pre-game agreement, a fixed value *komi* (e.g. 6.5) will be added to W's final score



komi 0	**B wins by 2**	**(11 − 9)**
komi 6.5	**W wins by 4.5**	**(9 + 6.5 − 11)**

```
from go/go_helper.py

def makemove(self, where, color):
    assert (self.brd[where] == EMPTY), 'that point is not empty'
    self.brd = change_string(self.brd, where, color)

    cap = []
    for j in self.nbr_offsets:
      x = where + j
      if self.brd[x] == opponent(color):
        cap += self.captured(x, opponent(color))

    if (len(cap)>0):
      #print('removing captured group at', point_to_alphanum(where, self.C))
      for j in cap:
        self.brd = change_string(self.brd, j, EMPTY)
      return cap, True  # move ok sofar

    if self.captured(where, color):
      print('whoops, no liberty there: not allowed')
      self.brd = change_string(self.brd, where, EMPTY) # erase the attempted move
      return cap, False  # move not possible, point occupied
    return cap, True
```

the end (logical rules of go)