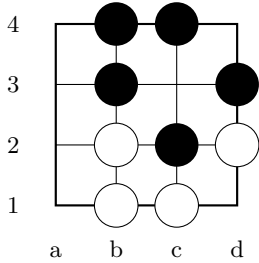


1. [5 points] Consider the following 4×4 Go position after 7 moves, with White stones at b1, b2, c1, d2 and Black stones at b3, b4, c2, c4, d3.



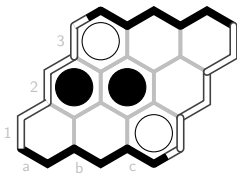
- (a) List the liberties for the block containing c4.
- (b) List the liberties for the block containing c2.
- (c) Suppose that White places a stone at c3. What stones, if any, does it capture?
- (d) Would it be legal for Black to reply by playing a stone at c2. Why or why not?
- (e) Would it be legal for Black to reply at d1 instead? Why or why not?

2. [5 points] The following line is from the go board initialization in `hexgo/stone_board.py`:

```
self.nbr_offset = ((-1,0), (0,1), (1,0), (0,-1))
```

- (a) Explain the purpose of this line.
- (b) Give the corresponding line for Hex board initialization with neighbour order below left, below right, right, above right, above left, left.

3. [5 points] Here are the `parent[x]` values after the position below has been created by the moves 1.B[a2], 2.W[a3], 3.B[b2], 4.W[c1]. Next, a black stone is played at a1. Recall that `Union(p,q)` sets the parent of the root of q's component to the root of p's component; assume *no* union-by-rank optimization. Below, show the new values of `parent[x]` after each union. TP, BM, LF, RT are the top, bottom, left, right borders respectively.



x	TP	BM	LF	RT	a1	b1	c1	a2	b2	c2	a3	b3	c3
current parent[x]	TP	BM	LF	RT	a1	b1	RT	a2	a2	c2	LF	b3	c3
after Union(a2,a1)	---	---	---	---	---	---	---	---	---	---	---	---	---
then after Union(BT,a1)	---	---	---	---	---	---	---	---	---	---	---	---	---

4. [5 points] Here is a portion of the `tromp_taylor_score` method in `go/go_helper.py`, with two lines missing. The method counts White territory into the variable `wt` and Black territory into the variable `bt`.

```
1 while (len(empty_points) > 0):
2     q = empty_points.pop()
3     for j in self.nbr_offsets:
4         x = j + q
5         b_nbr |= (self.brd[x] == BLACK)
6         w_nbr |= (self.brd[x] == WHITE)
7         if (self.brd[x] == EMPTY and
8             x not in empty_seen):
9             empty_seen.add(x)
10            empty_points.append(x)
11            territory += 1
12            if b_nbr and not w_nbr:
13                -----line_(A)-----
14            elif w_nbr and not b_nbr:
15                -----line_(B)-----
```

- (a) What should lines (A) and (B) be?
- (b) Explain the purpose of line 6.