1. Is the Nim position `(43, 7, 7, 43)` a winning or a losing position? Why?

   It is a losing position, because $\text{xorsum}(43, 7, 7, 43) = 0$. (Because $43 \oplus 43 = 0$ and $7 \oplus 7 = 0$, so $2 \oplus 2 \oplus 3 \oplus 3 = 0 \oplus 0 = 0$).

2. Is the Nim position `(43, 7, 7, 43, 107)` a winning or a losing position? Why?
   It is a winning position, because

$$\text{xorsum}(43, 7, 7, 43, 107) = \text{xorsum}(43, 7, 7, 43) \oplus 107 = 0 \oplus 107 = 107 \neq 0.$$

3. List all the winning moves from the Nim position `(11, 4, 2)`.

   - $\text{xorsum}(11, 4) = 15 > 2$, so no winning moves on the 2-stone pile.
   - $\text{xorsum}(11, 2) = 9 > 4$, so no winning moves on the 4-stone pile.
   - $\text{xorsum}(2, 4) = 6 \leq 11$: removing 5 stones from the 11-stone pile (leaving 6) is a winning move.

4. List all the winning moves from the Nim position `(45, 43, 7)`.

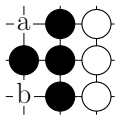
   - $\text{xorsum}(45, 43) = 6 \leq 7$, so remove 1 stone from the 7-stone pile (leaving 6) is a winning move.
   - $\text{xorsum}(45, 7) = 42 \leq 43$, so remove 1 stone from the 43-stone pile (leaving 42) is a winning move.
   - $\text{xorsum}(43, 7) = 44 \leq 45$, so remove 1 stone from the 45-stone pile (leaving 44) is a winning move.

5. Does the extra-stone argument apply to tic-tac-toe? That is, is it true that adding an X to a position (without changing whose turn it is) can never make X worse off? Why or why not?
   Yes, the extra-stone argument applies to tic-tac-toe. If an extra X appears on the board, it can make some X-lines possible that weren't possible before, which helps X; it can block some O-lines that would otherwise have been possible, which helps X; but it can never block an X-line that would otherwise have been possible.

6. Why does the strategy-stealing argument *not* prove that tic-tac-toe is a win for X the first player?
   We can use the strategy-stealing argument to prove that tic-tac-toe is not a win for O: Suppose for contradiction that O has a strategy that can guarantee a win. Then X can force a win by following O's strategy on their second turn, treating their first move as an extra stone. But then *both* players can force a win, which is a contradiction; so O must not have such a strategy.

   In Hex, showing that White does *not* have a winning strategy is sufficient to show that Black *does*, since Hex cannot end in a draw. However, tic-tac-toe can end in a draw, so the fact that O cannot force a win does not imply that X can.

7. Does the extra-stone argument apply to Go? That is, is it true that adding a Black stone to a position (without changing whose turn it is) can never make Black worse off? Why or why not?
   No, the extra-stone argument does not apply to Go. Consider the position at right; Black has won with a score of 6 to 3. But after an extra stone at either a or b, White can take Black's entire group.



8. Use a strategy-stealing argument to prove that White does not have a winning strategy from an empty Go board. Suppose for contradiction that White has a winning strategy. Then Black can simply Pass; if White also passes, then the game ends in a draw, and White was not able to force a win. Otherwise, White will make the first move, and Black can force a win by following White's winning strategy. But then both players can force a win, a contradiction, and hence White must not have a winning strategy after all.

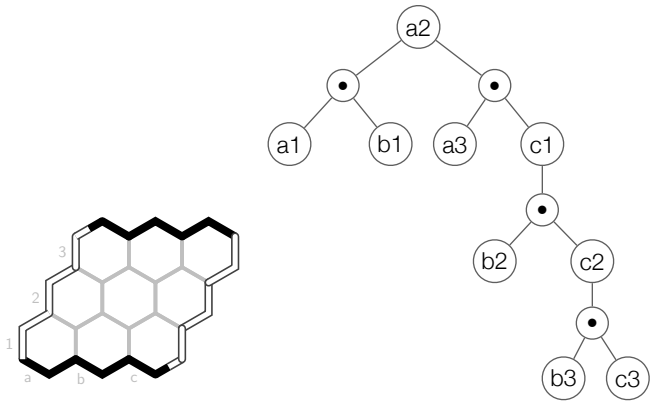
9. Consider the following pairing strategy for Black: $(\{a3, b3\}, \{b1, c1\})$

   (a) Where can Black play after `1.B[b2] 2.W[b3]`? Black must play $a3$.

   (b) Where can Black play after `1.B[b2] 2.W[a3]`? Black must play $b3$.

10. Consider the following AND/OR strategy on an empty 3×3 Hex board:

$$a2 \wedge (a1 \vee b1) \wedge (a3 \vee (c1 \wedge (b2 \vee (c2 \wedge (b3 \vee c3)))))$$



   (a) What must Black's first move be? Why? Black must play $a2$, because otherwise White could play $a2$, which would block the whole strategy (i.e., make the AND/OR expression false).

   (b) What are this strategy's *cell sets*?
   $\{a2, a1, a3\}$, $\{a2, a1, c1, b2\}$, $\{a2, a1, c1, c2, b3\}$, $\{a2, a1, c1, c2, c3\}$, $\{a2, b1, a3\}$, $\{a2, b1, c1, b2\}$, $\{a2, b1, c1, c2, b3\}$, $\{a2, b1, c1, c2, c3\}$

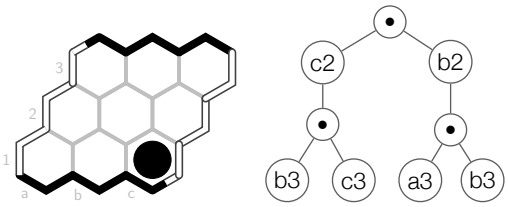
   (c) Is this a winning strategy for Black from this position? Why or why not?
   No. The cell set $\{a2, a1, c1, b2\}$ satisfies the strategy, but does not connect the two Black borders.

   (d) Where must Black play after `1.B[a2] 2.W[b1]`?
   $a1$; otherwise, White could play $a1$ on their next turn, which would make the left OR expression false, (block the left sub-strategy), which would make the top-level AND strategy false (blocked).

   (e) Where must Black play after `1.B[a2] 2.W[a3] 3.B[c1] 4.W[c3]`?
   Any of $b2$, $c2$, or $b3$. After $b2$ the right substrategy is satisfied, and there are two chances to satisfy the left substrategy, so White cannot block. After $c2$, there are still two ways for Black to satisfy the right substrategy and two ways to satisfy the left substrategy.

11. Consider the following AND/OR strategy after `1.B[c1]` on the 3×3 Hex board:

$$(c2 \wedge (b3 \vee c3)) \vee (b2 \wedge (a3 \vee b3))$$

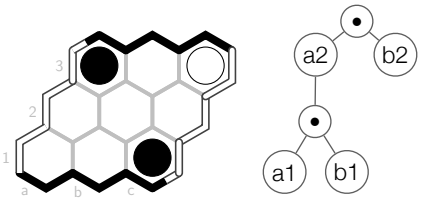


(a) What are this strategy's *cell sets*?
$\{c2, b3\}, \{c2, c3\}, \{b2, a3\}, \{b2, b3\}$

(b) Is this a winning strategy for Black from this position? Why or why not?
This strategy does *not* guarantee a win for Black, because although every cell set connects the two border, the cell sets for the two branches of the topmost OR are not disjoint; they both contain $b3$. So White can block both branches with a single move.

12. Consider the following AND/OR strategy after `1.B[c1] 2.W[c3] 3.B[a3]` on the 3×3 Hex board:

$$(a2 \wedge (a1 \vee b1)) \vee b2$$

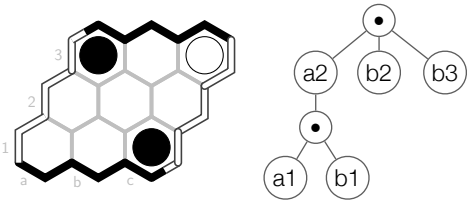


(a) What are this strategy's *cell sets*? $\{a2, a1\}, \{a2, b1\}, \{b2\}$

(b) Is this a winning strategy for Black from this position? Why or why not?
Yes; the cellsets of every branch of an OR expression are disjoint, and each cell set joins the top and bottom borders.

13. Consider the following AND/OR strategy after `1.B[c1] 2.W[c3] 3.B[a3]` on the 3×3 Hex board:

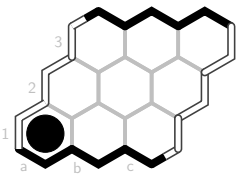
$$(a2 \wedge (a1 \vee b1)) \vee b2 \vee b3$$



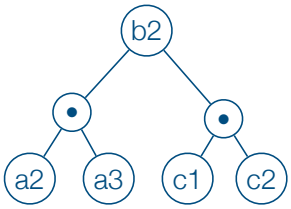
(a) What are this strategy's *cell sets*? $\{a2, a1\}, \{a2, b1\}, \{b2\}, \{b3\}$

(b) Is this a winning strategy for Black from this position? Why or why not?
No. One cellset ($\{b3\}$) fails to join the top and bottom borders.

14. Construct a winning AND/OR strategy for White from the position after Black plays on $a1$, or explain why this is impossible.

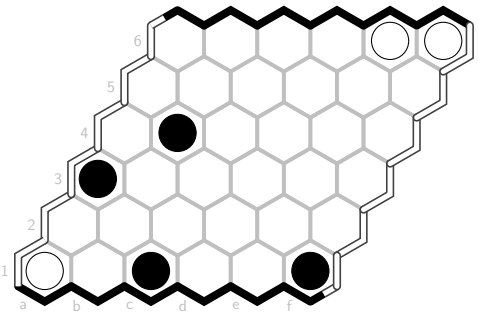


$b2 \wedge (a2 \vee a3) \wedge (c1 \vee c2)$



15. Consider the following 6×6 Hex position. For each pair of nodes below, indicate whether they are **F**ully connected, **S**emi-connected, or **N**ot virtually connected:

node 1	node 2	virtually connected?
$a3$	$b4$	Fully connected
$a3$	$c1$	Semi-connected
$c1$	$f1$	Semi-connected
$c1$	$b4$	Semi-connected
$b4$	$f1$	Not virtually connected



16. Consider the `go/tromp.py` implementation in the example code github.

    (a) How would you modify `go/tromp.py` to try pass moves last instead of first?

    Move the following lines to after the `for` loop in the `xab` function, and make an analogous change to the `oab` function:

```
s = score(black, white) if passed else oab(n + 1, black, white, alpha, beta, 1)
if (s > alpha):
    alpha = s
    if (alpha >= beta and CUT):
        if ngames < GSHOW: print(' CUT', alpha, beta)
        return alpha
```

    (b) Is `go/tromp.py` still able to solve 2×2 Go after this change? How long does it take?

    Yes, but it is dramatically slower. On my laptop the modified version takes 12.6 seconds instead of 0.122 seconds.

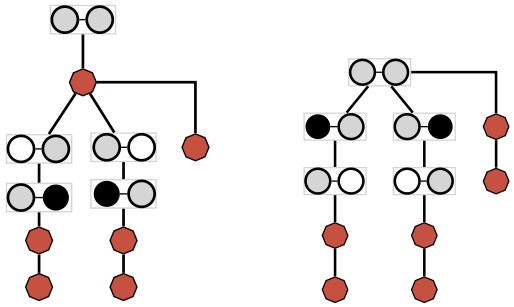
    (c) Why do `go/tromp.py` (and `go/tromp.c`) rely on bit manipulation instead of something more readable?

    Code that uses bit manipulations instead of equivalent, human-readable operations can be much faster if done correctly; on my laptop, the compiled `tromp.c` completes in 0.01 seconds

    (d) How would you modify `go/tromp.py` to solve 3×3 Go instead of 2×2 Go?

    The first step would be to update the environment to apply to 3×3 boards; e.g., the `visit(black, white)` function would need to read `h[black + 64 * white] = 1`, etc. The larger issue is that any straightforward modification would be intractably slow; indeed, some of the pruning that is already present (e.g., `owns`) would not apply to larger games.

17. (a) What is the minimax value of 1×2 Go? Prove your answer.
The minimax value is 0 (a draw). The easiest way to prove this is with two proof trees; the left proof tree shows that Black has a strategy that guarantees a score of at least 0 (so the minimax value ≥ 0), and the right proof tree shows that White has a strategy that guarantees a score of no more than 0 (so the minimax value is ≤ 0).



(b) What is the minimax value of 2×2 Go? You need not prove this answer.
We have seen in class (and from running `go/tromp.py`) that the minimax value of 2×2 Go is 1.

(c) What is the minimax value of 2×2 Go after the move `1.B[pass]`? Explain your answer.
We can use a strategy-stealing argument to show that the minimax value of this position is -1. We know from the previous answer that Black has a strategy that guarantees a value of 1. Furthermore, we know that this strategy must not pass on the first move (because if it did, then White could pass as well, giving a value of 0). So if Black passes first, White can simply follow Black's winning strategy, guaranteeing a win by at least 1, which gives a value of at most -1 to Black.